Knowledge Creation via Data Analytics in a High Pressure Die Casting Operation

by

Adam E. Kopper

A Dissertation
Submitted to the Faculty in partial fulfillment of the requirements for the Degree of

Doctor of Philosophy

in

Materials Science and Engineering

at the

Worcester Polytechnic Institute

August 2020

APPROVED:

_____

Diran Apelian
Alcoa-Howmet Professor of Engineering
Thesis Advisor

_____

Brajendra Mishra
Kenneth G. Merriam Distinguished Professor of Mechanical Engineering
Director, Department of Materials and Manufacturing Engineering

Table of Contents

Abstract

Big Data is a term typically associated with large internet entities such as Facebook, eBay, and Google where every click, search, and upload builds an actionable dataset used to target advertisements and enhance the user experience. The Data Science realm classifies data as Big Data by the three V's: volume, velocity and variety. The high-pressure die-casting (HPDC) process is a commonly employed method of producing large volumes of cast components particularly in aluminum alloys. The automated pieces of equipment are interfaced such that any signal passed from one machine to another, or sensor input, is data which may be relevant to the output of the process. For each part cast, it is possible to record the input parameters: melt temperature, lock up tonnage, cycle time, plunger velocities, cavity fill time, intensification pressure, dwell time, and spray time to name a few. Outputs parameters are generally lacking across all parts, rather they are measured on an audit basis. Is die casting process data truly big data? In the scale of the internet giants and major banking and credit firms, no. However, in some respects to the three V's, it is. Certainly, in a die casting facility with multiple machines running production the velocity of data generation is high on the input side. Die casting data resides in spreadsheets, databases, images, and shift notes. Thus, variety of data is a consideration. Generating great volume, unfortunately, is often a challenge. It is posited that the same tools can be used to gain knowledge into the HPDC process as in other truly Big Data environments.

Production HPDC process data, generously donated by FCA Kokomo Casting Plant, covering one year of production across 12 die casting machines and 20 die cavities was used to assess the applicability of machine learning algorithms such as Random Forest, Support Vector Machine, and XGBoost for the prediction of casting quality and performance metrics. The challenges which arise from the characteristics of materials processing data, using HPDC as the exemplar, have been identified. The proper data preparation methods for machine learning have been described. Predictive modeling of part quality and mechanical properties of die-cast engine blocks has been performed with an emphasis on model evaluation and cross-validation.

Acknowledgements

Many thanks to God for multiple doors opened, abundant resources provided, and invaluable people placed in my path without whom I could never had accomplished this doctoral degree. For those of you who have been a part of this journey with me, I would like to express my most sincere gratitude.

In December of 2016 my friend and colleague, Ray Donahue, and I were at Logan Airport. The winter ACRC meetings at WPI had just concluded. He and I were talking about the potential of my embarking on a PhD program. He could tell I was not completely sure about it. Ray had his PhD, a long successful career, and countless industry accolades and awards. He said this to me, "The time is going to go by. Either, in three years, you will have a PhD, or you won't. It only depends what you decide." He went on to say, "I have no doubt you will finish. I know you; it's what you do." It was that simple. In those few words, Ray pushed aside all the what-ifs and worries that were clouding my view of myself and what I could accomplish. Ray was right. Ray passed away in October 2017 quite unexpectedly. There are many milestones along the path to a PhD that are worth celebrating like passing the qualifying exams, having your thesis proposal accepted, research epiphanies, and those presentations where you know you just nailed it. Ray and I did not have a chance to celebrate those, but he was there for the most difficult step, the first one. Thank you, Ray.

Professor Diran Apelian, thank you for your guidance, encouragement, wisdom, and friendship extending over many years. It has been an incredible adventure since we first met at the old CMI Tech Center. It gives me great comfort to know you remain in my corner throughout it all. You have shown me that the title "advisor" has no expiration date. At 45 years of age, I discovered that the title "student" does not either.

Rasika Karkare, I could not have wished for a better research partner. Revising these papers line-by-line over zoom was much more fun than it should have been. Professor Randy Paffenroth, thank you for your patience and assistance as I ventured into a new world of Data Science. My vocabulary is much different than it was when we first met. I catch myself cringing when the word "correlated" is thrown around so casually. That'll stick with me.

Professor Mishra, I always look forward to your ice breaker jokes at ACRC meetings. Professor Carl Soderhjelm, I value our friendship very much and I always enjoyed sharing a meal with you and Pamela when I am in town. Dr. Kevin Anderson thank you for being my advocate at Mercury Marine looking out for me and providing interesting opportunities in research and professional development.

I could not have begun my PhD program without the generous support of my employer, Mercury Marine. I especially want to thank Chris Drees and Mike Meyer for believing in me and approving my research proposal. Thank you to Jerry Cegielski for his unwavering support providing me time and funding. Conducting graduate level research from a remote location would be much more difficult without my on-site "graduate laboratory" to be a sounding board for ideas, coding assistance, proof reading, and encouragement. To Alex Monroe, I am so blessed to have you to exchange ideas with, to challenge me at times, and to share perspective. Thank you, David Blondheim and Brian Fruchter, for so much coding help when I started out.

Saving the dearest for last, thank you mom and dad for a lifetime of love and support. To my two sons, Nathan and Benjamin, whom I love so much. Never stop learning. Never stop growing. You make me so proud and I will always believe in you. My loving wife and best friend, Maria, this accomplishment is as much yours as it is mine. You have been so patient and supportive over the course of this program. Together we made it work.

Executive Summary

I. INTRODUCTION

Motivation

Big Data and Artificial Intelligence (AI) are terms often associated with relatively new internet entities such as Facebook, eBay, and Google. AI itself is not new, but its growth has been accelerated by this new business space. Social media sites like Facebook have users who voluntarily provide mountains of personal data. By collecting data from their users, Facebook gains knowledge on hobbies, dining preferences, travel preferences, posting trends, the popularity of any given topic across all users; the categories are literally endless. The most useful data can drive targeted advertising, site content, upgrades to the user experience, etc. Implementing these methodologies into their operations is not exactly a revolution to that industry because they had no prior way of doing business. Data is their business.

Established business sectors had to find ways to integrate this application of data into the existing systems in their organizations. Service industries such as banking, hotels, and shipping use Big Data extensively to reach their customers in a more personalized experience, while increasing their adaptability to new ways of how their customers want to shop for a mortgage, book a room, or pay for and print shipping labels.

Every interaction with a customer is a source of data. Through AI and machine learning, knowledge is generated about an individual, a gender, an age group, a socioeconomic group, a regional population, or any other illuminating segmentation. This knowledge is then utilized to make data-driven decisions that improve how a business operates to gain and keep delighted, loyal customers.

Materials processing operations generate extensive amounts of data. Perhaps not to the level of the internet giants, but enough high-dimensional samples to make analysis a challenge by traditional methods. In the same way these internet and service providers gather data about their customers, materials processors gather data about their product. What is not common is the application of data science to create knowledge. For the materials industry, new knowledge from the product data would be in the form of insight into the effects of process parameters, component designs, environment, materials composition, or any other area of interest. A higher level of monitor and control becomes justifiable when the data is being leveraged to improve the business. Imagine the benefit if these operations could predict part, or lot, quality by monitoring critical process inputs and running the data through a machine learning algorithm in near real time. Such a future would result in increased uptime, reduced operational costs, rapid response to production issues, and data driven confidence that the product made between quality checks is acceptable.

Current State of the Industry

An important materials processing method for casting near net-shape components today is high pressure die casting (HPDC). HPDC is the most utilized casting method for aluminum components by tonnage in the United States and widely used throughout world [1], [2]. Aluminum die-cast components are primarily employed where weight reduction and high annual production volumes are required, especially automotive applications. In terms of dollars, the North American Die Casting Association reported aluminum die castings to be over $8 billion in sales for 2019, while the American Foundry Society reports the entire aluminum foundry industry to be $9.67 billion [3]. In HPDC, we have a robust industry using state of the science technology making components critical to the daily lives of people all around the globe. As the manufacturing sector marches toward an Industry 4.0 future, HPDC and, indeed the entire foundry industry,

must be a part of it. A key piece in making this happen is the implementation of AI and creating knowledge from casting process data.

Modern foundries have the capability to capture a vast amount of process data on a daily basis [4]. These include molten metal preparation details, casting process parameters, simulation results, part geometry, Non-Destructive Evaluation (NDE) data, etc. The first obstacle to using this data is cultural and centers on organization for analysis. Data fusion for machine learning is more difficult when the data is stored within operational silos (*Figure 1*). The type of data and collection methods used by isolated departments within the same facility have evolved over years. Methods range from high-tech automatic uploading to a cloud database to handwritten records in a logbook. This creates challenges for combining the various sources into a cohesive dataset as the collection frequency and identifiers often differ. Communication among stakeholders through the entire process is critical to identify which, how, and how often data should be collected to give the best description of the system to be modeled. Integrated data is the prerequisite for performing machine learning, and it is a lost opportunity for the foundry industry if no effort is made to compile, fuse, and analyze these data to better understand the process factors influencing the quality of the castings.

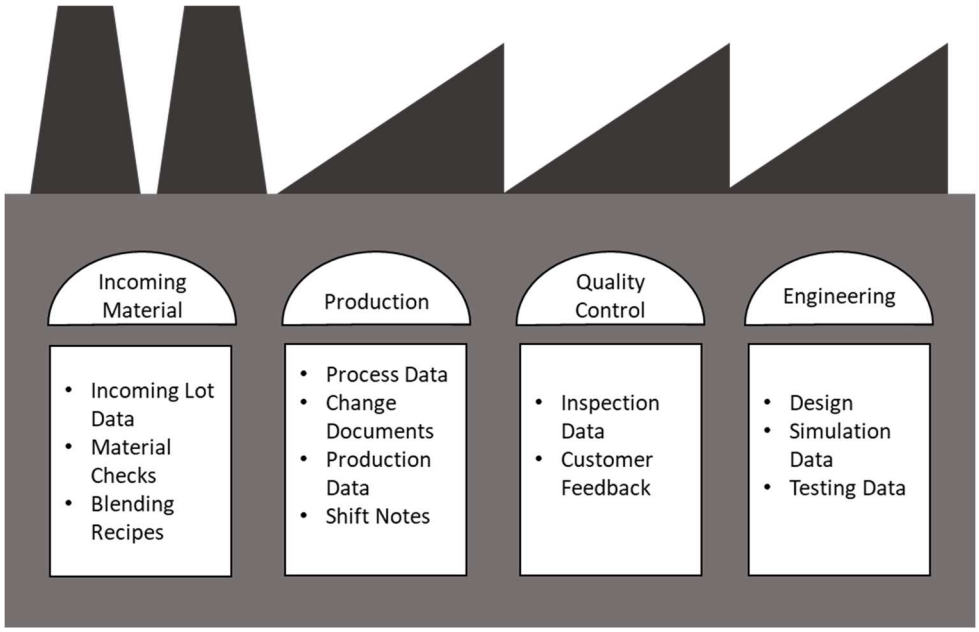


Figure 1. *Departmental data silos are a challenge to implementing machine learning in many materials manufacturing operations.*

Resources are tight in casting plants. Between the development of new products and the attention demanded by the most difficult castings, there is little time left to dedicate to the analysis of the parts that are running very well day in and day out. However, it is a missed opportunity to merely count the easy parts as blessings. The data from these castings hold the key to solving the issues with trouble castings. It is the data from the good parts that will right the ship when a normally well-behaved process beings making scrap parts. By collecting data, creating knowledge, and applying that knowledge, operations can put up data-determined guardrails to keep the process in control. When resources are tight, we need new tools to watch over that for which there is no person available to do. Machine learning can be that set of tools for the foundry and for the metal processing industry in general.

Industry 4.0

The fourth industrial revolution that ushered the Internet of Things (IoT) and the Internet of Services (IoS) has come to be known as Industry 4.0. At the Hannover Messe in 2011, Germany launched a project called *"Industrie 4.0"* designed to fully digitize manufacturing. The larger vision of Industry 4.0 is the digital transformation of manufacturing to integrate connected factories within industry, decentralized and self-optimizing systems and the digital supply chain in the information-driven cyber-physical environment of the fourth industrial revolution [5], [6], *Figure 2*.

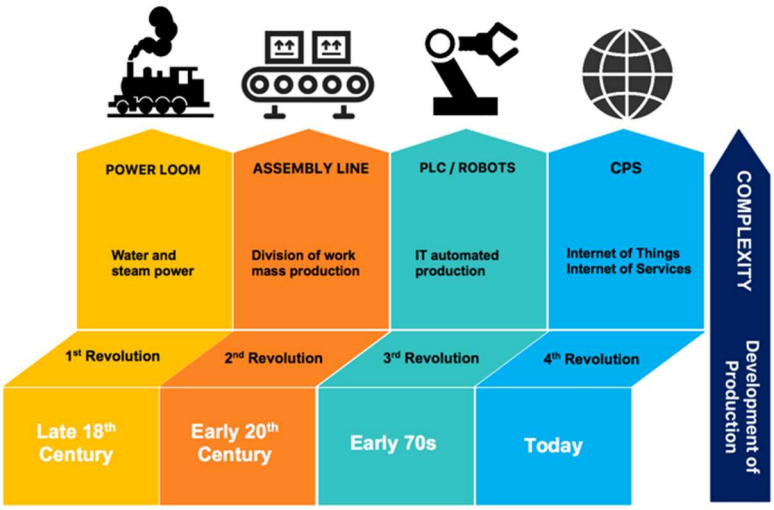


Figure 2. *Chronology and characterization of the four Industrial Revolutions* [5].

The initial goals of Industry 4.0 have been automation, manufacturing process improvement and productivity optimization. The more advanced goals are innovation and the transition to new business models and revenue sources using information technologies and services as cornerstones. These developments will transform manufacturing plants into smart factories. Three keystone digital technologies will enable the transformation to smart factories: (i) connectivity, which implies executing industrial IoT to collect data from various segments of the plant; (ii) intelligent automation which includes advanced robotics, machine vision, digital twins, distributed control; and (iii) cloud-scale data management and analytics (AI and Machine Learning) [7].

Problem Statement

Traditional research into the effects of process parameters on the quality of cast products are structured to investigate a wide range of input settings. To investigate the effect of a parameter, e.g. intensification pressure, a traditional study evaluates a limited number of samples cast at three levels: no intensification pressure, an intermediate amount of pressure, and the maximum safely attainable with the machine and set-up used. The purpose is to observe a difference in the output, e.g. a casting, which can be measured and analyzed from one set of parameters to the next. From a study such as this, we, as an industry, learn that porosity is reduced as intensification pressure is increased. These studies serve as a compass to direct which parameters should be controlled and where one should begin developing their process. On the other hand, in production environments, only one optimized set of parameters is used which has some natural variation associated with it. The objective in production is to observe no difference from one casting to the next; to make the same quality part cycle after cycle. However, it is true that a small percentage of parts are not of acceptable quality and are salvaged or scrapped. The root cause for some process scrap is not easily

determined from our current understanding of process parameter effects. Further complicating root cause detection is the relatively few examples of non-conforming product from which to gather data.

Few samples may be required to discern the porosity or mechanical property impact of a given input variable over a wide range in values. Finding a similar relationship over a much narrower range, due to natural variation of the equipment and process, requires many more samples. Machine learning accuracy thrives on large datasets. Additionally, it may be that previously dismissed input parameters become highly important in tightly controlled processes, their effect being overshadowed by large experimental changes in pressure or velocity. Machine learning algorithms have the capacity to look at high dimension datasets with many features so that all collected input data can be examined. The question we are asking is: *Given enough data, can machine learning algorithms uncover new insight into process parameters or interactions of process parameters which are important in predicting casting quality or performance metrics from production process data?*

The industry needs tools which can recognize patterns that are too nuanced for humans to see. The foundations of machine learning lie in statistical pattern recognition [8], [9]. A key capability of machine learning algorithms is their ability to uncover patterns and relationships between inputs and outputs for high-dimensional datasets [10]. Casting process data is high-dimensional, having many inputs: temperatures, velocities, pressures, timers, and chemical composition. There are opportunities to add more dimensions [4], however, having too many adds confusing noise to the data. There is a need to begin exploring data the industry is currently collecting and determining which parameters matter and which are missing. This research is aims to do that working with HPDC process data and to inspire casting operations to begin creating knowledge from their own process data.

Machine Learning in HPDC - *Literature Review*

A comprehensive literature review can be found in ***Appendix A*** of this volume. There, one will find an overview of AI, a detailed description of the HPDC process, and a review of the effects various process parameters have on casting quality, porosity, and tensile properties. A summary highlighting the main concepts introduced follows.

HPDC is rich for data mining. Potentially useful data can be pulled from the controllers of each piece of equipment in the modern integrated work cell. Blondheim estimates that there are over 300,000 data which can be captured for each cycle [4]. If one includes thermal imaging data of the die cavity and the individual data points which make up the shot trace, this number explodes to over 2M input variable data per cycle. A reasonable estimate for an annual volume on one die casting machine is 100,000 cycles. That would equate to two-hundred billion data points per machine per year. Clearly, amassing features is not the challenge. Learning which features are most important and collecting enough observations to be sure of it is where the difficulty resides. Researchers in this space are turning to AI seeking insight into this problem.

Early applications of machine learning to HPDC center on the application of Neural Networks to predict virtual process outputs. Rai et al. used supervised learning by creating datasets with process simulation software and then teaching a Neural Network to predict cavity fill time, solidification time, and porosity based on the process inputs of melt and die temperature and slow and fast shot velocities [26]. They found that the results of the Neural Network model compared well to those generated by commercially available finite element mesh-based simulation software but did so in near real time. Similarly, Yarlagadda et al. predicted fill time from the melt temperature, die temperature, injection pressure, and casting weight with a Neural Network trained via process simulation software and went a step further by including domain expertise from casting specialists [27]. Their predictions matched very closely to actual production die castings.

Simulation software packages are built utilizing assumptions which generate useful direction in building die casting tooling and choosing initial process settings. During process development, parameters are tuned more finely to optimize part quality. This tuning is done based on domain expertise and the results of actual castings. It is reasonable to expect a Neural Network to find the rules the simulation software is using and make very similar predictions. The next step is to apply the algorithms to serial production castings and determine which input variables are driving quality or mechanical performance metrics and direct the process engineer how to tune the process for best results.

The leap between the computationally trained algorithms and algorithms trained on observational data from casting operations may seem daunting. There are many variables which are not monitored or controlled on the factory floor (ambient environment, die temperature, cooling water flow rate). These features may not be included in the simulation, are held constant, or provided as an output. In a controlled experiment where 413-alloy aluminum was cast into simple cylindrical geometry under three levels of squeeze (intensification) pressure, die preheat temperature, and molten metal temperature, Soundararajan et al. were able to train and test a Neural Network predicting the ultimate tensile strength (UTS) and yield strength (YS) of extracted tensile bars with a correlation coefficient of 0.95 and 0.96 respectively [28]. In their work, the selected settings represent a wider range of process inputs than one might encounter on a fully developed production process. Predicting the UTS variation of each sample accounting for small variations as seen in production processes is a more difficult problem. This type of research lays the foundation from which the industry can build and develop algorithms which predict the UTS of serial production castings with low variation in input parameters.

In the literature, it is generally assumed that the process operates as consistently as possible. Several cycles, perhaps 5 to 10, are run to achieve a thermal steady state before collecting samples for investigation. The number of samples collected for analysis tends to be small, less than 50. The industry has gained much from these studies, but there are some potentially significant parameters which cannot be accounted for in lab-scale or development-cell scale operations. In the late 1990's Balasubramaniam applied statistical analysis to 27 casting variables from manufacturing production and found that higher intensification pressure rise time and lower cycle time were key inputs which improved the part density [29]. Interestingly, this study was unique in identifying these inputs as important. Die casting is a thermal process and time is an important factor that is often overlooked or simply held as constant as possible, but rarely measured and reported. The impact of variation in overall cycle time or timers for specific segments of the cycle are not published. Time impacts the die temperature. Running shorter cycle times will put more heat into the die raising the die temperature. But overall cycle time is not the whole picture. Increasing cycle time by increasing the dwell time (the time between casting and part ejection) will also put more heat into the die. Thus, it depends not only on if time is changing but when time is changing. This highlights the need for more high-dimensional parametric research. Our research sought to uncover potentially important features which have not been the subject of prior process parameter studies.

Introduction to Artificial Intelligence (AI), Machine Learning (ML) and Neural Networks (NN)

The following section contains material from the first journal article submitted for publication from this research. It can be read in its entirely in ***Appendix B – Machine Learning Pathway for Harnessing Knowledge and Data in Material Processing*****.** The paper serves as a primer for the metalcasting industry to start thinking about the future and the smart foundries of Industry 4.0. Explanation is given as to how artificial intelligence, machine learning, and deep learning are integral to realizing this future. A case study illustrates the challenges of foundry data and the methodology of a machine learning application. Finally, perspective is provided on value of AI with a caution to the danger of incorrect application.

Publication Details:

Ning Sun, Adam Kopper, Rasika Karkare, Randy C. Paffenroth, and Diran Apelian, ***"Machine Learning Pathway for Harnessing Knowledge and Data in Material Processing"***, International Journal of Metalcasting, Accepted for publication, July 2020.

A summary highlighting the key points are given below.

Artificial Intelligence

Artificial Intelligence is the use of computer algorithms intended to mimic tasks commonly performed by humans. Algorithms for image recognition, health analytics, natural language processing, and self-driving vehicles are all examples of AI that have transformed industries that affect our daily lives [11]. AI clearly has a role to play in advanced manufacturing where there are myriad tasks that could be automated by algorithms such as defect detection, process optimization, and new materials development, to name but a few [6].

Machine Learning

It is generally agreed that both machine learning and deep learning are forms of artificial intelligence rather than something entirely unique. The term machine learning, in this text, represents the family of methods which use statistical and probability models trained on historical data to make predictions about new observations. Common methods which fall under this umbrella include linear regression, decision trees, k-means clustering, Apriori algorithm, and Support Vector Machines (SVM) [12]–[19]. While packages and commands readily exist to facilitate using such algorithms, these methods are not black box functions shrouded in mystery. Many of them rely on using mathematical distances to determine how various observations are alike and what outcome should be expected if trained on relevant samples where that information is known.

Deep Learning

Deep learning utilizes the same data preparation strategies and similar functions with which to make predictions as machine learning [20]. Mostly, what makes two different is how feature engineering is performed (*Figure 3*) [21]. Feature engineering is where the data scientist relies on domain expertise to engineer the model inputs to make a higher performing model. In machine learning, this is performed manually. Deep learning utilizes hidden layers comprised of nodes which automatically assign weights to variables as the algorithm learns more about the data [22]–[24]. In this way, the deep learning algorithms are more of a black box than their machine learning kin. By using the training data to generate the weights automatically, deep learning algorithms can be more accurate than a human would otherwise be. As deep learning algorithms add additional complexity (i.e. increase the number of hidden layers or nodes per layer), it is critical that large datasets be used to train them. If not, the resulting model will not generalize well and, thereby, perform poorly on new data.

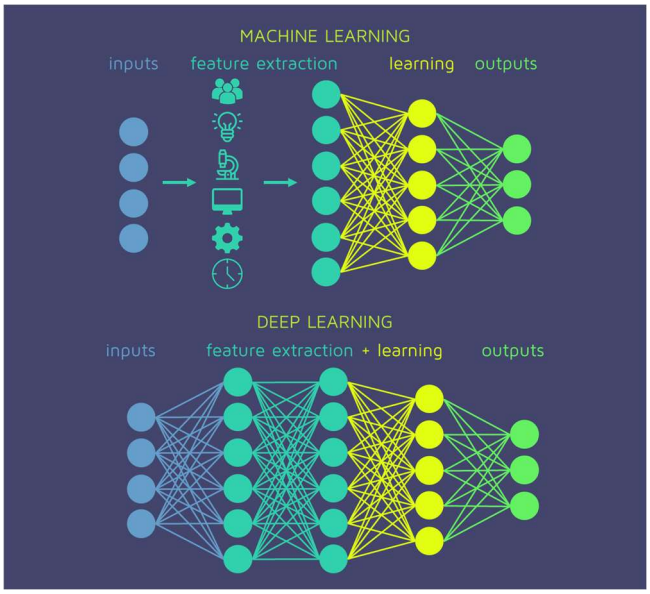


Figure 3. *The difference between machine learning and deep learning is how the feature engineering is performed* [21]. *Traditional machine learning utilizes subject matter expert input to the model while deep learning employs automatic processes.*

Challenges of Foundry Datasets

It turns out that analyzing materials processing data is not straightforward [25]. As materials processing companies bring their data to the data science community to find answers, new insight into how the data is traditionally collected and the challenges which are created thereby are brought to light. Let us look at three of them: most metalcasting data are not Big Data, heterogenous data sources resulting in missing inputs and outputs, and an imbalance in output data class where high quality samples far outweigh unacceptable samples. In this research, we work to navigate these challenges in the application of pre-preprocessing techniques and selecting appropriate algorithms that suit this type of data. No small part of the value of this thesis is to provide metal casters a foundation to build from, identifying pitfalls, and demonstrating the correct methodology with which they can begin examining their own data.

II. APPROACH and METHODOLOGY

This section summarizes ***Appendix C – Approach and Methodology*** of this thesis. There, a description of the HPDC data used for this project including a table of the features evaluated can be examined. The bulk of ***Appendix C*** provides detailed explanations and illustrations of the tasks performed in the machine learning research conducted and submitted for publication during this thesis project. After each example, the key components of the Python code are listed with helpful references for the reader.

The published literature contains many papers which report observations of various process inputs on mechanical properties and porosity. Forward focused HPDC facilities do a good job of capturing many of these data for each casting they produce. As an industry, we believe that we are collecting the correct data. The literature confirms the importance and die casters document and demonstrate process control to their customers by this data. The hypothesis that this work aims to test is that die casters collect the correct input

information and, given a large enough dataset, quality and performance properties can be predicted from that data.

High Pressure Die Casting Data

Universally, die casters are not to the level where every potential important variable is captured, and has been for years, such that large datasets are commonplace. There is also the challenge of accessibility to the data for analysis. Leaders in the industry recognize the importance of taking the first steps in bringing machine learning into die casting. The Aluminum Casting Research Center (ACRC) at WPI is an industry-university consortium where a cross-section of the aluminum casting industry including alloy producers, casters, industry suppliers, and end users meet and sponsor pre-competitive fundamental research [30]. FCA, a major automobile manufacturer with a large die casting operation and longtime member of the ACRC, partnered with this research team to provide a calendar year worth of HPDC process data, alloy chemistry checks, and mechanical property testing data. The size of each dataset is given in Table I. The details of the datasets with respect to which inputs and outputs are available and descriptions of each are given in Tables C-I and C-II at the end of ***Appendix C.***

Table I. Size details of the FCA datasets.

Dataset Name	**Raw Dataset (Rows x Columns)**
HPDC Process	956,986 x 109
Alloy Composition	980 x 17
Tensile Testing	1,634 x 14

The cycle summary data is historically the best, and often the only, information available to troubleshoot the process and make intuitive, experience-based predictions regarding quality of castings. For this reason, it is also the most easily and widely stored data. Additional data from the cycle are collected and appended to this information prior to uploading into long-term storage. The HPDC process data can be thought of as a spreadsheet with each row representing an individual casting and each column containing a piece of information about that casting. Similarly, in the mechanical property dataset, each row represents a tensile bar and the columns contain the input and output variables associated with each bar. The chemistry dataset has rows which represent each check and columns containing the amount of each elemental constituent in the melt at that time. This description is rather straightforward; however, visualization is difficult. The raw HPDC dataset has 109 columns. Humans are finite beings and, as such, have no ability to visualize what is happening in 109 dimensions. Fortunately, machines can perform these tasks on our behalf via machine learning algorithms that analyze high-dimensional data.

Modern HPDC equipment is more interconnected than ever to facilitate data organization and collection in the die casting cell. Platforms now exists for storing and accessing large amounts of data with which to train machine learning models. The need largely remains within the die casting industry to begin taking advantage of this reality and start investigating how to process data and train algorithms to create knowledge for data-driven decision making.

Data Science Approach

Data science projects are more intricate than collecting data and plugging it into an algorithm. The answers one gets from the algorithm will be misleading without following the required steps and understanding how each impacts the results. There are steps one must take in an iterative process to generate reliable

predictions and actionable results. An overview is given below with more detail on the methods used and references in the subsections that follow. The main building blocks of a data science project are initial data exploration, pre-processing to prepare the data for use in algorithms, running the algorithms, evaluating the results, iterating as necessary, and communicating the results.

Data exploration is quite simply looking at the data. The strategies of how to pre-process the data for machine learning are developed by first examining the data. Production data is messy. Missing values, erroneous sensor readings, duplicated entries, typos, format changes in the source files, etc. must be sorted out before one can engage in meaningful analysis. Considering the FCA HPDC process data set with over 950,000 observations and 109 variables, one cannot simply scroll through and hope to catch these issues by eye. Running summaries of the data, examining the data class, and locating *NaN* values are a few of the tasks to accomplish in this step. Fortunately, there are simple commands to execute which reveal issues in the data. These are standard procedure cleaning methods.

However, some advanced techniques were implemented. In the data cleaning step, we encounter two of the key challenges with foundry data: heterogeneous data and class imbalance. Both are popular domains for data science research today. In this work, heterogeneous input data was handled with naïve mean or median imputation, or sample removal, due to relatively few instances in the large HPDC process dataset. For classification modeling, imbalance between good parts and process scrap leads to predictive results which err toward classifying the minority class, process scrap, as good parts. To address class imbalance, we implemented the oversampling methods Synthetic Minority Oversampling TEchnique (SMOTE) and Boderline SMOTE [31]–[33]. These are data augmentation tools which increase the population of the minority class to train better models. New, synthetic samples are generated by selecting an example of the minority class, finding its nearest neighbors (k-neighbors = 5 is the default), and drawing a line between the example and one of its neighbors at random (*Figure 4*). The new sample is created along the connection line. Some of the disadvantages of SMOTE are lessened in Borderline SMOTE. While SMOTE generates new samples from all the minority class samples, Borderline SMOTE is a selective technique which models those along the boundaries which separate one class from another. By focusing along the border, we are creating more training data where the samples from one class resemble those of another class. The objective is to reduce the probability of classifying process scrap as a good part.

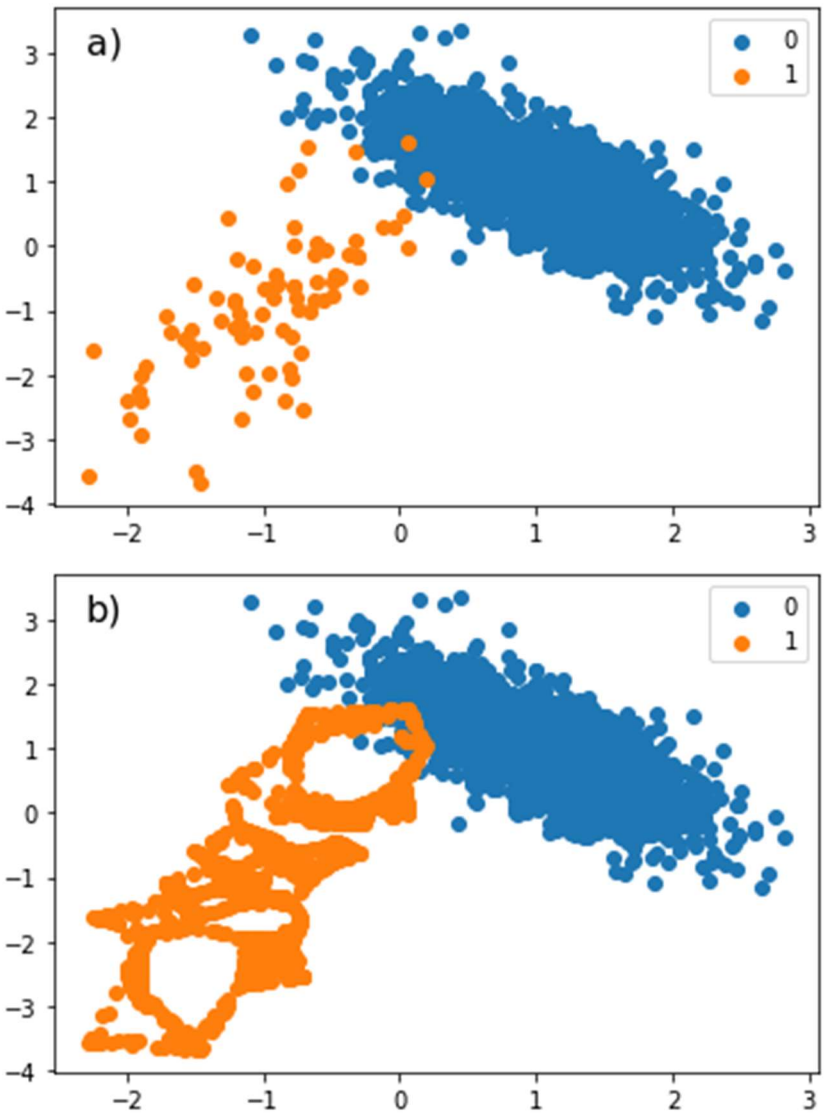


Figure 4. *An illustration of SMOTE application showing a) the unbalanced data prior to synthetic data generation and b) the balanced dataset after SMOTE. In this classification example, many synthetic data are generated far from the border of the two class clusters which is less useful for making predictions between classes.*

Other methods performed in the pre-processing step are covered further in ***Appendix C.*** These include the discretization of categorical variables, checking for correlation between variables, and data standardization. Data standardization is a crucial step which is, perhaps, easy to overlook if one does not know how to properly execute machine learning. The data collected in HPDC contains a wide range in scale. Also, different equipment manufacturers may capture data in only English or metric units. In round figures, intensification pressure of 10,000 psi, melt temperature of 1300 F/ 704 C, cycle time of 150 seconds, biscuit size of 2 inches, and an iron content of 0.60% are a few examples which show the range of scale is in orders of magnitude. If left in this format, the intensification pressure would register as highly significant and outweigh any influence the iron content would show simply because the numbers are larger. This is because many machine learning algorithms rely on a mathematical distance calculation to determine the similarity between two samples. Bringing the columns of data into the same scale makes these comparisons uniform and meaningful. While standardization is not necessary for every algorithm, it is typically a case of either

being essential to the algorithm (i.e. K-Nearest Neighbors [12]) or the algorithm is not hurt by it (i.e. Random Forest [34], [35]) . The method employed throughout this work is the Z-transformation [36], [37] (*Equation 1*). K-Nearest Neighbors (KNN) is a supervised learning method highly sensitive to distances [12]. Thus, benefits from standardizing data are readily shown in *Figure 5* where KNN was applied to the *toydata* dataset (see ***Appendix C***) with and without Z-transformation.

$$Z_{i,j} = \frac{X_{i,j} - \mu_j}{\sigma_j} \qquad \textit{Eq. 1}$$

Where:

- $Z_{i,j}$ is the Z-transform value in the *i*th row of the *j*th column
- $X_{i,j}$ is the original value in the *i*th row of the *j*th column
- μ_j is the mean of the original values in the *j*th column
- σ_j is the standard deviation of the original values in the *j*th column

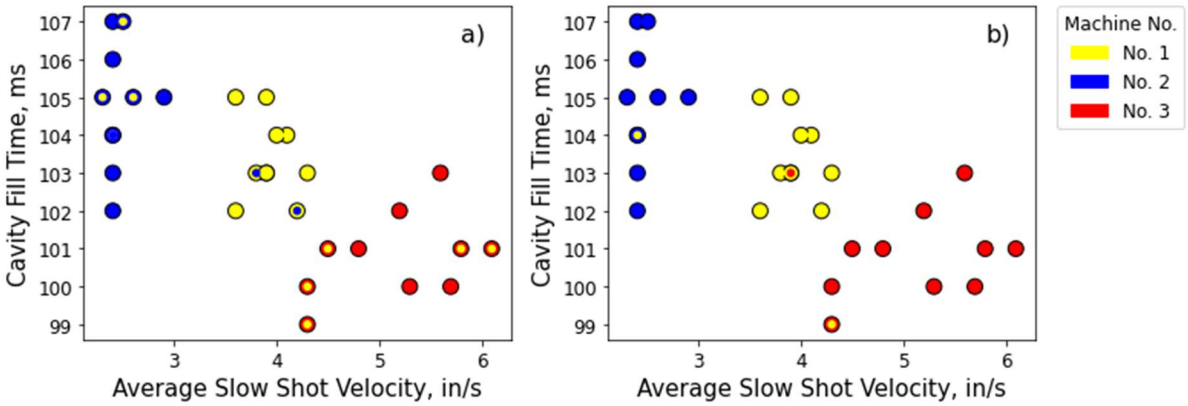


Figure 5. *K-Nearest Neighbor classification of machine identifier a) without Z-transform and b) with Z-transform. Each point represents the true and predicted machine class. Misclassified samples display as bi-colored points. The axes were chosen to show separation between three different machines. With the Z-transform, the model performed much better, classifying at a 90% accuracy.*

The final pre-processing task to cover in this summary is dimension reduction. Machine learning algorithms are adept at working in high dimensions, but there are negatives associated with too many features. The curse of dimensionality refers to how a given dataset becomes sparser as it is projected into higher and higher dimensions [38], [39]. Thus, more samples are required as dimensionality increases. Increasing dimension also increases the amount of noise in the dataset. An important method for dimension reduction implemented in this work is the Principal Component Analysis (PCA). PCA is an unsupervised dimension reduction technique [40]–[42]. The goal of PCA is to determine linear combinations of the input variables, called principal components (PCs), which capture the most variation in the dataset while minimizing the error when the dataset is reconstructed from the PCs. In doing so, a high-dimensional dataset can be condensed into a smaller number of PCs. PCA enables visualization of high-dimensional datasets in two or three dimensions. *Figure 6* shows a two-dimensional scatter plot of PCs 1 and 2 which were derived form a sample dataset of ten dimensions (see *toydata,* ***Appendix C*****).** As one reads this figure, it is important to recognize that this is a simple scatter plot. PC1 and PC2 are not functions of one another. The main disadvantage of PCA is that it is limited to linear principal components.

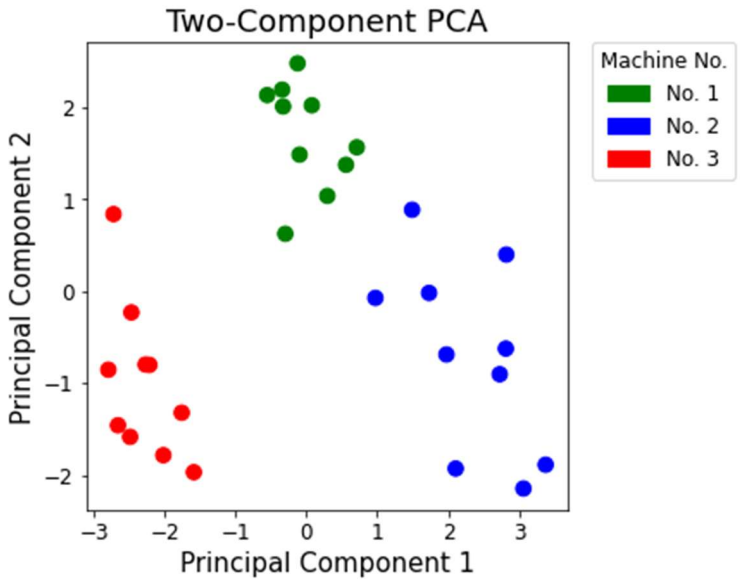


Figure 6. *Two-component PCA. On the toydata dataset, we see a separation of the castings into clusters by machine number.*

Moving on to the algorithms themselves, it is important to consider the amount of data one has available and complexity of the algorithms selected with respect to the bias-variance trade-off [43], [44]. Bias is error in the model driven by the underlying assumptions in the algorithm. Variance refers to the error in the model due to its sensitivity to noise in the training dataset. Understanding these two phenomena is essential to remedy underfitting and overfitting conditions in the model performance. *Figure 7* shows how models can be too complex (high variance) or too simple (high bias) and thus overfit or underfit the training data respectively. Either will result in poor performance on the testing data or new production data fed into the model. There is a desirable sweet spot where the model is general enough to make reliable predictions on the training data which translate to the performance on new data such as the middle plot of *Figure 7*. Choosing the correct algorithm and properly tuning it to work with the type of data being collected is how one arrives at the best performing model. Minimizing the predictive error between the training and testing datasets is the target for determining the best model.

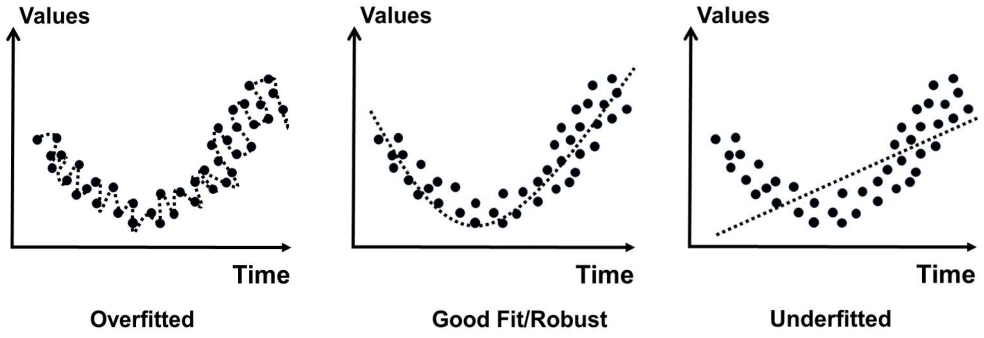


Figure 7. *The phenomenon of underfitting and overfitting is seen in this figure* [45]. *We want a model that is optimal for the kind of data and application that we are exploring. For example, a good fit is illustrated in the center plot. The plots on the right and left show underfitting and overfitting respectively and should be avoided.*

The primary machine learning algorithm used in this work is Random Forest [34], [35]. This method was used extensively in ***Appendix D - Predicting Quality of Cylinder Block Castings via Supervised Learning Method*** and proved to be the best traditional machine learning algorithm investigated when paired with PCA in ***Appendix E - Model Selection and Evaluation for Machine Learning: Deep Learning in Materials Processing.*** In ***Appendix E***, Random Forest was examined alongside SVM and XGBoost [46], [47] in the machine learning category. It was outperformed by a shallow Neural Network which was reported by my research colleague Rasika Karkare in the same article. ***Appendix D*** focuses on Random Forest to classify part quality and ultimate tensile strength, while ***Appendix E*** applies it to a regression problem to predict ultimate tensile strength values.

Random Forest is selected because it works well with high-dimensional data, is robust to non-linear data, has low bias, and variance is reduced through bagging [34]. To combat misclassification, Random Forest uses the results from hundreds, or thousands, of tree estimators to make predictions. Random Forest is an ensemble learning, or prediction by committee, approach where the observations are randomly broken into subsets and built into trees split on a random subset of the features. The predictions of many trees built from the training data are compiled to make a final prediction for each observation. In a classification model, group voting among the trees is conducted to determine the predicted class. For regression models, the final reported value is calculated the by average of the predictions for each observation from all the trees. The result of the ensemble can be better than what any one of the trees would determine on its own. A detailed explanation of how the tree estimators are built is given in ***Appendix C***.

In ***Appendix E,*** XGBoost was chosen to evaluate a more recent adaptation of Random Forest which, in addition to bagging, uses boosting to reduce bias by training the subsequent model on the errors of its predecessor. Bagging reduces overfitting while boosting improves accuracy at the cost of possible overfitting [48], [49]. SVM was chosen for its ability to determine non-linear decision functions via the *kernel trick.* The kernel trick maps the input data into a higher dimensional feature space where the data is linearly separable resulting in non-linear boundaries between the input data [50], [51]. These methods are compared to a Neural Network which is effective for handling nonlinearity, tolerant of noise, utilizes advanced learning methods, and generalizes well.

Finally, this summary would be remiss without mentioning training, testing, and cross-validation. This description is placed at the end of the approach because it is vital for evaluation of model performance, but it arches over the process more broadly. Holding out a testing dataset prior to performing the machine learning step is how the trained model is evaluated. This is data that is new to the trained model, so the predictive error on the testing data is what one could expect on newly generated data from the process. Recognize that the train/test split can influence the model. For that reason, cross-validation is conducted to determine how different splits of the data affect the model performance metrics. K-folds is a common method of cross-validation [52]. In K-folds, the user sets the number of folds and the model is run that many times on the training data. Each time, a different segment of the training population is set aside as the test data and run through a model created on the balance of the training data for that fold.

The performance of the algorithm can be measured in many ways. Mean absolute error (MAE, *Equation 2*) and mean squared error (*Equation 3*) values can be used to score regression algorithms [53], [54]. Accuracy, precision, recall, and f1-scores (*Equations 4-7*) are often chosen to evaluate classifiers [55]. Regardless of the algorithm, it is common for the error on the training data to be less than the test data. When the difference between the two is large, the model is said to be overfit to the training data. Data scientists are keenly aware of over-fitting because such a model does not generalize. The model shows amazing accuracy on the training data, however, when fed new data, the predictions of the algorithm are unreliable.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right| \qquad \textit{Eq. 2}$$

Where

- MAE is the mean absolute error
- n is the number of samples in the dataset
- Y_i is the actual value of the output
- $\hat{Y}_i$ is the predicted value of the output

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \qquad Eq.\ 3$$

Where

- MSE is the mean squared error

$$Accuracy = (TP + TN) / (TP + TN + FP + FN) \qquad Eq.\ 4$$
$$(Precision)_i = TP_i / (TP_i + FP_i) \qquad Eq.\ 5$$
$$(Recall)_i = TP_i / (TP_i + FP_j) \qquad Eq.\ 6$$
$$(f1\_score)_i = 2 * ((Precision_i * Recall_i)/(Precision_i + Recall_i)) \qquad Eq.\ 7$$

Where

- *TP* is the number of true positives
- *TN* is the number of true negatives
- *FP* is the number of false positives
- *FN* is the number of false negatives

III. RESULTS and DISCUSSION

A. Predicting Quality of Cylinder Block Castings via Supervised Learning Method

The following section contains material from the second journal article to be submitted for publication from this research. It can be read in its entirely in ***Appendix D - Predicting Quality of Cylinder Block Castings via Supervised Learning Method.*** The paper is directed toward the HPDC industry and a scaled down version is submitted for the 2020 NADCA Congress and Tabletop which will be held virtually due to the COVID-19 pandemic. An introduction section is given to highlight the need for this research and the challenges associated with HPDC data. HPDC process data is analyzed via supervised machine learning methods to successfully model the prediction of good parts and process scrap as determined by the die casting machine (DCM). Additionally, the prediction of ultimate tensile strength via a classification method of extracted tensile bars is performed and the important features identified are discussed. Supervised learning is found to be a useful tool for materials processing applications.

Publication Details:

Adam Kopper and Diran Apelian, ***"Predicting Quality of Cylinder Block Castings via Supervised Learning",*** submitted to Intl. J. of Metal Casting.

The work is summarized, and highlights are given below.

Approach

At the center of the HPDC work cell is the die casting machine (DCM). Ancillary equipment fills out the cell to execute tasks of metal delivery, die preparation, casting removal and trimming the part of excess material like runners and overflows. The DCM can be programmed to identify a casting as being good, scrap, or a warm-up shot based on the parameters which created the casting. This is accomplished by setting upper and lower control limits (UCL and LCL) for key variables as determined by the manufacturing engineer. The DCM is using a series of Boolean checks (Is parameter *n* between LCL_n and UCL_n?), all of which must be TRUE for a part to be good [56]. If a part is cast with too low of intensification pressure, for example, the machine will identify the casting as scrap and send a signal to the operator or a robot to place the part into the scrap hopper or set it aside for inspection. Parts cast within the prescribed limits are labeled good parts and further processed as normal.

The FCA HPDC process dataset was used for this study. Removing highly correlated columns, noisy features, and those with no variation brought the number of columns down from 109 to 83 columns of input/output variable data. The data is collected from 12 HPDC work cells and 20 die casting tools. Periodically, the production castings are destructively evaluated for mechanical property testing via a tensile test. In this dataset, there are 1494 observations for which both the HPDC process variables and the mechanical property data are collected into 159 columns. The blocks are cast in E380 aluminum alloy [57] and subjected to T5 heat treatment post castings. For a specific application, a subset of the blocks receives an additional 24-hour natural age prior to T5.

Two classification models are used in this study, Decision Tree and Random Forest. The Decision Tree classifier is a supervised machine learning method used to build a predictive model for a given process output by sorting the castings into classes at various nodes using an input variable as the sorting criteria [13]. This input variable is chosen by the algorithm because sorting by it provides the greatest information gain to the model. Random Forest classifiers use many Decision Trees together to train the model [34]. Both methods are examined in this study. The splits in the trees are determined by Gini index (*Equation 8*), which is a measure of the purity of the resulting nodes by making a split [58]. The Gini index varies between zero and one. A Gini of zero represents a pure node where all the observations are of the same class. A high Gini value means that the various classes are mixed and there is a high probability that a new observation would be misclassified.

$$Gini = 1 - \sum_{i=1}^{n} P_i^2 \qquad \textit{Eq. 8}$$

Where:

- $Gini$ is the Gini index
- n is the number of classes
- P_i is the probability of finding each class in the node

When developing the models, the datasets are split into training and test subsets. Model performance metrics include accuracy, precision, recall, *f1-score* (*Equations 4-7*) [55]. Accuracy is percentage of correctly classified observations. Precision is the proportion of predicted positives which are correct. Recall is the proportion of actual positives correctly classified as positive. *f1-score* is the harmonic mean of the precision and recall and is useful for unbalanced datasets, such as the one in this study, where there are many more good parts than process scrap.

There are two objectives of this research: The first is to use machine learning via a classification model to predict the quality label assigned by the DCM: good part, process scrap, and warm-up. The second is to determine which other classifications can be determined by this method; specifically, the presence of discontinuities (i.e. porosity) in a tensile bar machined from the casting was examined. The first objective evaluates machine learning on a large dataset where it is known that the correct information has been

captured to classify the observations. The second uses a small dataset where there may be parameters which influence the classification that are not captured in the dataset.

Results

DCM Quality Label Classification

A successful classification model evaluates the die casting process data and the known class assigned to the part in the training dataset and determines what the rules are such that the model will accurately assign testing samples to the correct class. The three labels are: good part, process scrap, and warm-up. Initial Decision Tree models run on the FCA HPDC process dataset had little difficulty identifying warm-up shots. This is because warm-up shots typically have different process settings from production shots utilizing low shot velocity and minimal intensification pressure. The challenge is separating the good parts from the process scrap. For the remainder of this exercise the warm-up shots have been utilized in a calculation as a process input which designates the number of shots performed since the last warm-up shot was made. This value is included to serve as a directional proxy for die temperature. Since this column equals zero for all warm-up shots, the prediction of warm-up shots by the model becomes automatic. Thus, the warm-up shots are removed from the dataset.

Summaries of the model performance will be shown via *confusion matrix* (Table II). The matrix rows track the actual known classifications of the test population and the columns correspond to the classifications of the test population as predicted by the model. A perfect model would have zero *FN* and *FP* predictions.

Table II. Interpretation of the confusion matrix. A perfect model would have zero *FN* and *FP* predictions. For this study, positives are good parts and negatives are process scrap.

		Predicted Value	
		Positive	**Negative**
Actual Value	**Positive**	*TP*	*FN*
	Negative	*FP*	*TN*

Both Decision Tree and Random Forest performed reasonably well on the testing data which is made up of 174,869 rows of data which the models had not seen before, especially predicting good parts. Although, manufacturing operations make many good parts, it is predicting the process scrap which is of the greatest value. The results showed too many *FP* results: 1,577 for the Decision Tree and 1,613 for the Random Forest. False Positives must be minimized, as these would have a negative impact on downstream operations. Unbalanced data is a challenge for modeling production manufacturing data. Since each split in each tree is done without the consideration of any other splits, the best Gini split may sweep many process scrap samples into a node which overwhelmingly consists of good parts. This results in misclassification when the node is a leaf. Fortunately, there are methods to working with unbalanced data.

To reduce the amount of FP, the issue of data imbalance is addressed by generating more process scrap data by which to train the model. The simplest way to do this is to reproduce samples from the process scrap class, but this provides no new information to the algorithm. A better method, which does provide new information to the model via the creation of new minority class samples, is called Synthetic Minority Oversampling TEchnique (SMOTE) [31], [32]. SMOTE creates each new minority class sample by selecting an example of the minority class, finding its nearest neighbors, and drawing a line between the example and one of its neighbors at random. The new sample is created along the connection line. This is done repeatedly until the minority class balances out the majority class.

To investigate, the training data was oversampled using SMOTE and new Decision Tree and Random Forest models were trained. Predictions were made on the same testing data using the new models. It is

important that SMOTE be applied to the training data only, and not the testing data. This way the testing data is faithful to the process. The results are shown in Table III and Table IV for the Decision Tree and Random Forest respectively. By balancing out the process scrap with the good parts, the new models are more adept at recognizing process scrap and *FP* are reduced.

Table III. Confusion matrix showing the performance of the Decision Tree with SMOTE classifier model for part quality.

Decision Tree w/ SMOTE – Part Quality		**Predicted Value**	
		Good Part	**Process Scrap**
Actual Value	**Good Part**	164,716	3,282
	Process Scrap	924	5,947

Table IV. Confusion matrix showing the performance of the Random Forest w/ SMOTE classifier model for part quality.

Random Forest w/ SMOTE – Part Quality		**Predicted Value**	
		Good Part	**Process Scrap**
Actual Value	**Good Part**	165,443	2,555
	Process Scrap	1038	5,833

Comparing Tables III and IV, it is difficult to see which model is best suited for our data. Both exhibit false positives and false negatives. To determine the better performing model, it is useful to use scoring metrics. These measures are tabulated for both models below (Table V). The metrics associated with the minority class (process scrap) are more telling for model performance. The models perform quite similarly, however, the Random Forest is the better model due to the higher *f1-score* for the process scrap class. The results between the testing and training datasets are nearly the same, therefore, neither model is overfitting to the training data.

Table V. Key scoring metrics for the part quality Decision Tree and Random Forest classifiers with SMOTE training data. Mean values are reported from 5-fold cross validation.

	Decision Tree w/ SMOTE		**Random Forest w/ SMOTE**	
	Training Data	Test Data	Training Data	Test Data
Model Accuracy	98.81 %	98.70 %	98.40 %	98.66 %
Precision	98.76 %	98.64 %	98.41 %	98.63 %
Recall	98.81 %	98.70 %	98.40 %	98.66 %
f1-Score	98.76 %	98.64 %	98.40 %	98.55 %
f1-Score (Process Scrap)	97 %	74 %	99 %	76 %

A useful summary for the process engineer can be pulled from the model called *feature importance* [59]. Understanding the influence of each variable on the model helps the engineer determine which variables to monitor more accurately, perform designs of experiment around, and where to invest in process control measures for best results. Feature importance of the Random Forest with SMOTE model are given in Table VI below. The list of 83 variables was truncated at values greater than 0.02.

Table VI. Feature importance for the part quality Random Forest classifier with SMOTE generated training data.

Part Quality Random Forest w/ SMOTE	
Feature Name	**Importance**
Time Between Cycles	0.2830
Biscuit Length	0.0868
Final Intensifier Pressure	0.0641
Plunger position at the end of shot	0.0505
Cycle Time	0.0480
Average Intermediate Shot Velocity	0.0426
Cavity Fill Time	0.0392
Average Fast Shot Velocity	0.0381
Shots Since Last Warm-up Shot	0.0314
Intensification Velocity Rise Time	0.0239
Dwell Time	0.0239
Intensification Stroke	0.0215

Feature importance can also be used to assist in feature selection for creating more efficient models which take less time to run and perform better when noisy features are removed. Ultimately, the final set of features is based on trial and error and the preferred performance metric. The same Random Forest model set-up was run using only the top 12 features shown in Table VI. Dropping the low importance input variables minimally reduces predictive power, and overfitting to the training data is still avoided (Table VII).

Table VII. Scoring metrics for the part quality Random Forest with SMOTE classifier models using the top 12 features. Mean values are reported from 5-fold cross validation.

	Random Forest Classifier	
	Training Data	**Test Data**
Model Accuracy	98.81 %	98.55 %
Precision	98.82 %	98.48 %
Recall	98.81 %	98.55 %
f1-Score	98.81 %	98.45 %
f1-Score (Process Scrap)	99 %	79 %

Breaking the data down into unique combinations of DCM number and die cavity number yielded interesting results. It was observed that, when subsets representing each combination of DCM number and die cavity number were run across the general part quality Random Forest classifier, the metrics of the predictions varied. This suggests that each DCM and cavity combination is to some degree a unique process. The details of this result are shown in ***Appendix D.***

The example of predicting part quality assigned by the DCM is a straightforward example where the dataset is very large and contains all the information available to the DCM for labeling parts good or process scrap. Many materials processing problems are more difficult due to the challenges of small datasets. Next, we turn our attention to how well Random Forest classification modeling can be applied to predicting porosity in castings using the process data by which they were made.

Porosity Classification

HPDC process input data is used by manufacturing operations as a real-time quality check. Thus, it is of interest to test if these data can be analyzed further to predict levels of porosity in good parts. For the cast component of this study, production castings are selected for destructive mechanical property testing via testing tensile bars extracted from the casting itself. Ultimate tensile strength, yield strength, tensile strain (elongation), and hardness data are collected for the purpose of quality assessment [60]. In most HPDC products, the location of the tensile bars is limited to the few heavy areas of the casting which can accommodate their geometry. Intensification pressure is applied during solidification to compress gas porosity and feed shrinkage; however, once the gates freeze, pressure is no longer transmitted to the last areas to solidify (i.e. heavy walled sections). In the long freezing range aluminum alloys commonly utilized in HPDC, like 380-alloy, the resulting shrinkage is often microshrinkage which is difficult to detect via NDE methods such as digital X-ray. Thus, the presence of porosity is a characteristic of HPDC which must be controlled and not necessarily an indication of a poor casting. Discontinuities do impact the measured mechanical properties resulting in additional work and cost to reproduce the test. It has been shown that mechanical properties are dependent on the amount of porosity in the area of fracture [61]–[64]. Making a connection between mechanical properties and porosity is of interest to die casters because, in many applications, the presence of porosity can result in scrap due to uncovered porosity after machining or loss of pressure tightness or leaking. Finding which process inputs contribute to porosity in a mature process is challenging for humans to solve. It is also a difficult problem to model because all the castings in the new dataset are classified as good parts, so the difference between any given input variable from one observation to the next is likely small.

Binary classification by the noted presence of a discontinuity on the tensile bar fracture surface proved to be a poor target for prediction. Cáceres' research shows that a binary classification for porosity is not adequate since the amount of porosity affects the mechanical properties [61]. Whether or not the porosity was observed by the tester in the tensile bar has no bearing on how the bar performed. *Figure 8* shows an empirical cumulative distribution function for the bars with and without observed discontinuities. The curve for the data with observed discontinuity is shifted to lower UTS values. There is considerable overlap which supports the assertion that microshrinkage porosity is often undetectable via visual inspection.

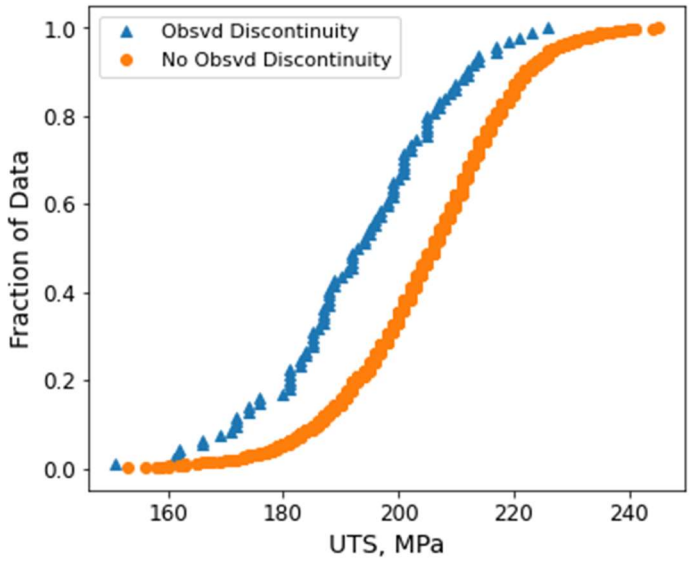


Figure 8. *Empirical cumulative distribution functions for the UTS of tensile bars with and without observed discontinuities on the facture surface by visual inspection.*

A classification model based on a UTS value has two benefits: undetected porosity gets captured in the lower performing bars and the two classes can be set up to be more balanced. Two classes were selected: UTS < 205 MPa and UTS ≥ 205 MPa. There is no assignment of "good" versus "bad" implied in selecting the ranges. The value of 205 MPa is chosen as it is the median value of the tensile bar dataset. Importantly, 80% of the bars with observed discontinuities exhibited less than 205 MPa of UTS as well. A Random Forest classifier was used to predict which UTS class each bar in the test dataset would fall into using the HPDC process input data. Table VIII shows the Random Forest classifier results cross validated over 10 iterations. This model is slightly overfitting to the training data as there is more of a difference between the training and testing metrics than we saw in the DCM quality label example. One of the better performing models is shown in Table IX.

Table VIII. Key scoring metrics for the Random Forest classifier model predicting UTS over or under 205 MPa. The support of the test dataset is: 138 UTS < 205 MPa samples and 161 UTS ≥ 205 MPa samples.

	Random Forest Classifier: UTS over/under 205 MPa	
	Training Data	**Test Data**
Model Accuracy	60.62 %	56.87 %
Precision (weighted)	60.76 %	57.13 %
Recall (weighted)	60.62 %	56.87 %
f1-Score (weighted)	60.57 %	56.37 %

Table IX. Confusion matrix of the Random Forest classifier for UTS tensile bars above and below 205 MPa using HPDC process inputs only.

Random Forest Model – UTS		**Predicted Value**	
		> 205 MPa	**< 205 MPa**
Actual Value	**> 205 MPa**	98	63
	< 205 MPa	47	91

If die casting operations examine their data in this way, there is benefit gained even from imperfect models. Referring to Table IX, the test dataset consists of 299 samples of which 161 were of the higher UTS class. This amounts to 53.8% high UTS samples. This model suggests that there are operating conditions where high UTS bars can be expected. If those conditions are employed, one would find that 98 of 145 are high UTS bars, or 67.6%. The parameters which rise to the top of the feature importance list in Table X are worthy of study since splitting on their value has the largest impact on UTS prediction.

Table X. Average feature importance calculated over ten iterations of the Random Forest classifier.

Feature Name	Importance	Feature Name	Importance
Ejection Forward Time	0.0747	Total Tie Bar Tonnage	0.0379
Spray Robot Time	0.0505	Final Intensifier Pressure	0.0357
Die Close Tank Level	0.0498	Avg Head Pressure during Intermediate Shot	0.0349
Avg Head Pressure during Fast Shot	0.0466	Extract Robot Cycle Time	0.0348
Shot Count Since Last Warm Up Shot	0.0414	Cycle Time	0.0347
Die Close Time	0.0403	Die Opening Time	0.0329
Intensification Pressure Rise Time	0.0396	Vacuum Pressure during Shot	0.0321
Average Fast Shot Velocity	0.0386	Ladle Pour Time	0.0332
Avg Head Pressure during Slow Shot	0.0383	Intensification Stroke	0.0304

Conclusions

- Supervised learning performed better on the larger HPDC process dataset. The complete population has 874,344 observations and the DCM makes quality determinations based on this data, so the right data is collected. The result is a good model.
- Oversampling using SMOTE is effective for teaching the model to better predict the minority class.
- The Random Forest classifier outperforms a single Decision Tree by reducing variance. The ability to differentiate good parts from process scrap improve when focusing on unique combinations of machine and cavity number as stand-alone processes.
- For predicting porosity, UTS has been shown to be a better output for predictive modeling than relying on porosity observation alone. Microshrinkage porosity can easily be missed by the unaided eye, but its effect is apparent in the UTS measured.
- A key difference between the DCM part label problem and the porosity prediction problem is the size of the dataset available to the model. The smaller tensile bar dataset is impacted by overfitting issues that the larger dataset avoids.

B. Model Selection and Evaluation for Machine Learning: Deep Learning in Materials Processing

The following section contains material from the third journal article to be submitted for publication from this research. It can be read in its entirely in in ***Appendix E – Model Selection and Evaluation for Mahcine Learning: Deep Learning in Materials Processing.*** The paper is directed more generally toward the material processing industry with a focus on proper evaluation of model performance and the bias-variance trade-off. An introduction section is given to highlight the usefulness of predicting UTS citing its connection to porosity in the casting which is of high interest to foundries. Predictions of UTS based on HPDC process data are made via traditional machine learning methods (Random Forest, Support Vector Machine, and XGBoost) and a shallow Neural Network. Algorithm predictions are compared to an alternative of predicting the mean from historical data. All the machine learning algorithms and the Neural Network outperform the historical mean prediction with the Neural Network achieving the lowest mean absolute error in its predictions.

Publication Details:

Adam Kopper, Rasika Karkare, Randy C. Paffenroth, and Diran Apelian, ***"Model Selection and Evaluation for Machine Learning: Deep Learning in Materials Processing",*** Integrating Materials and Manufacturing Innovation, submitted 7/10/2020.

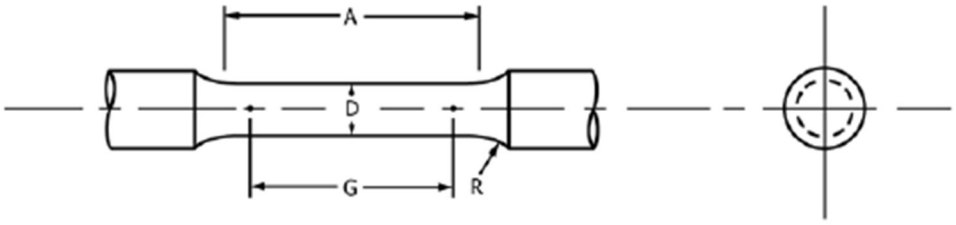
The work is summarized, and highlights are given below.

Approach

It is paramount to understand the type of data HPDC operations generate, and the machine learning and deep learning methods that are best suited for analysis. One is often introduced to the terms machine learning, deep learning, and Neural Network as buzz words used interchangeably in marketing or general audience publications. All of these are subsets of artificial intelligence and defining where machine learning ends and deep learning begins is somewhat blurry. Perhaps it is best to look at these as a continuum of complexity. Machine learning algorithms reside on the lower end of the complexity spectrum making use of linear and other low-order functions [12]. While deep learning is at the other end employing layers of mathematical transformations and activation functions for creating models [65]. The most suitable method depends on the data available.

This study was conducted to compare the performance of various machine learning and deep learning methods in predicting the UTS of tensile bars excised from engine block castings. The mean absolute error of the algorithm is used to score the methods. Furthermore, an explanation of the importance of bias-variance trade-off is given to provide context for the results [43], [44].

When designing castings, minimum mechanical properties may be specified by the designer which are required for the final product. Process and alloy selection are largely driven by these requirements [66]. Testing mechanical properties such as UTS, YS, and elongation requires destructive methods which can only be conducted on an audit basis. Tensile testing of test bars extracted from the cast part itself, or cast alongside the part, is the most employed method to measure these properties [67]. The tensile bars come from four different locations in the engine block and are machined to a 0.350 inch (9 mm) diameter sub-sized geometry based on ASTM B 557 (*Figure 9*) [60]. The dataset captures the UTS, YS, and the tensile strain. The 0.2% offset method is used to calculate the YS [68]. The tensile strain is measured with the extensometer over the course of the test and is reported as a percentage. Included in the dataset is a notes column which is text mined for mentions of fracture location and the presence of observed discontinuities such as porosity or an inclusion [13]. Each bar is classified accordingly. Tests with no indication of a discontinuity are classified as *unknown*, rather than to assume none were present.



G – Gage Length	1.400 +/- .005 (35.5 +/- 0.1)	**R – Radius (min)**	0.25 (6.35)
D – Diameter	0.350 +/- .007 (9.0 +/- 0.2)	**A – Reduced Section Length (min)**	1.650 (41.9)

Figure 9. *Tensile bar geometry per ASTM B557* [60, p. 55].
Dimensions in inches (mm).

Tensile test data were examined to determine which output to target for prediction. Like most production manufacturing data, there is noise in the data that can be difficult to filter out with certainty as the actual tensile bars are not typically retained and were not available for this study. Statistical analysis via Welch's t-test is performed to detect significant shifts in the mean value of UTS and tensile strain from one population to another [69]. Location of bar extraction, presence of observed discontinuities, and the heat treatment were analyzed. The results of Welch's t-test confirmed that the mean UTS value is statistically different based on the presence of defects and heat treatment used. UTS was selected over Quality Index (QI) [70], [71] and tensile strain for its sensitivity to the presence of observed anomalies in the tensile bars. The literature has shown that UTS is sensitive to the presence of such casting features in tensile bars [72] [73]. This connection of UTS to porosity is very useful to die casting producers, since quality issues in die casting are largely porosity related [74]. Preliminary modeling efforts confirmed that UTS was showing less error in the model performance as compared to prediction of tensile strain and QI.

Bias-Variance Trade-Off

When choosing an algorithm, the two most important considerations are size of the available data and bias-variance trade-off [43], [44]. This dataset of 1494 tensile tests are exceedingly large when compared to typical mechanical property studies. However, in the world of data science, this is not Big Data. The amount of data available is a limiting factor in the complexity of the model. *Figure 10* shows a performance comparison of the models as data size increases.

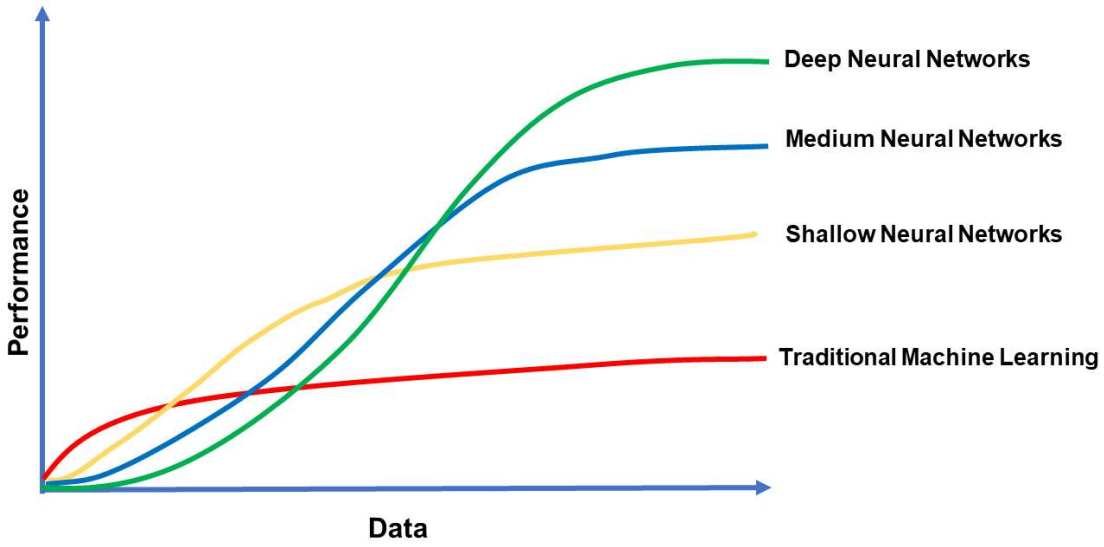


Figure 10. *Performance comparison of Neural Network models with traditional machine learning models as training data size increases. On smaller datasets, traditional algorithms outperform deep learning models however, as the amount of data increases, deep learning models perform better.*

For smaller datasets, one would pick traditional algorithms as compared to deep learning models. However, as the quantity of data increases, deep learning models perform better because traditional algorithms reach a saturation point and do not improve any further whereas deep learning models performance keeps increasing with training data size [75].

Understanding the bias-variance trade-off is essential in deciding which algorithms to select for a particular dataset and application. In *Figure 11*, the X-axis shows model complexity and the Y-axis is predictive error. As model complexity increases, variance increases and bias decreases. An increase in the variance causes the model to overfit to the training data and it fails to generalize on new data. The left side of the plot shows a high bias but low variance region. This implies that the model is too simple and, hence, it is highly biased. It fails to learn the complexity of the data. The ideal point is where bias and variance intersect, as shown by the optimum model complexity in *Figure 11* [43].

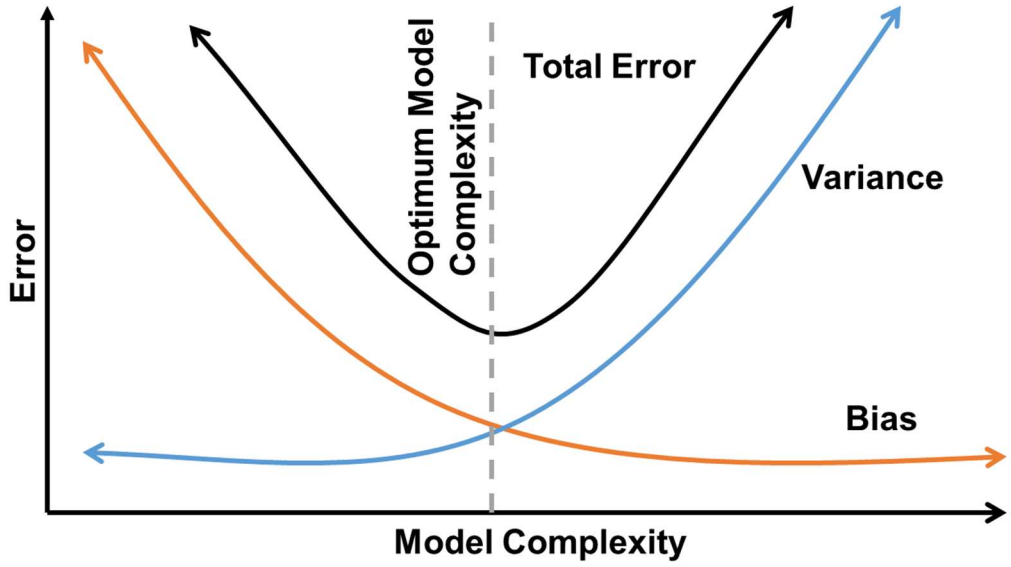


Figure 11. *Bias-variance trade-off* [43], [44] *shows how error changes as the complexity of the model increases. The region on the right is that of high variance and low bias whereas the region on the left is that of high bias and low variance. These regions are where the model overfits or underfits the training data and should be avoided. The optimal model complexity is where variance and bias intersect, and one should utilize algorithms in this region.*

Figure 7 shows the phenomena of overfitting and underfitting. The *Overfitted* graphic in *Figure 7* depicts a model from the region in *Figure 11* where variance is high. Such a model will fail to generalize because it is too specific to the training data. The *Underfitted* graphic from *Figure 7* shows an example from the region in *Figure 11* where bias is high. Here, the model fails to learn enough complexity in the dataset and underfits [44], [45]. The *Good Fit*/*Robust* graphic in *Figure 7* shows the optimum model which corresponds to the intersection of bias and variance in the bias variance trade-off and gives a robust fit to the data.

Mean absolute error (MAE) values are reported to score the algorithm (*Equation 2*). It is common for some overfitting to the training data to exist in the model, so the error on the training data tends to be less than the test data. The goal of a robust model is to minimize the difference in error between the training data and testing data results.

The data was properly pre-processed before machine learning. Methods such as discretization, standardization, and dimension reduction via PCA were performed. For machine learning algorithms, Random Forest, SVM, and XGBoost were chosen. These methods are compared to a Neural Network deep learning algorithm.

Results

Machine Learning Regression Results

The process of adjusting the controlling parameters within the algorithm is called tuning [76]. The chosen method of tuning selected for these models is Grid Search Cross-Validation (GSCV) [77]. In GSCV, multiple parameters can be tuned at once optimizing the model with respect to the target metric rather than each parameter at a time. The goal of tuning is to minimize the difference between the training and testing

data results. *Figure 12* was generated using the default parameters of the algorithm while *Figure 13* shows the improvement realized from tuning. In the Random Forest and the XGBoost the difference between the training and testing error decreases. The model becomes more general. The tuned SVM is not far from where the default parameters started.

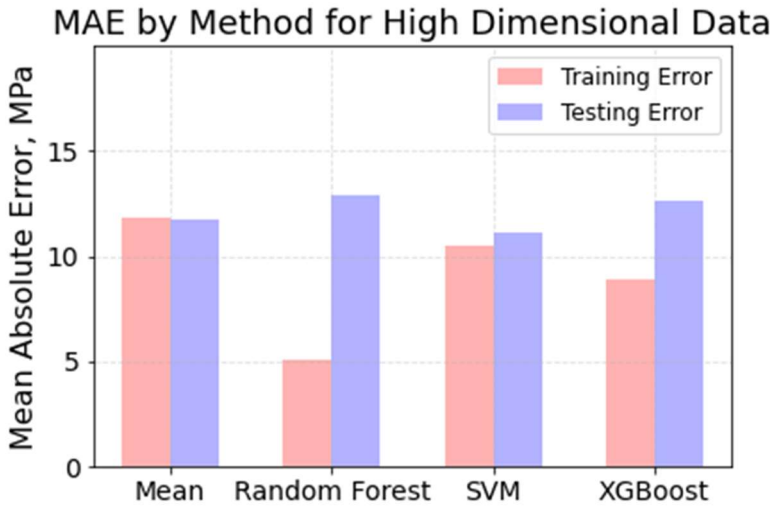


Figure 12. *MAE in UTS prediction results for the high dimension dataset using default settings. Both the Random Forest and the XGBoost are showing significant overfitting to the training data.*

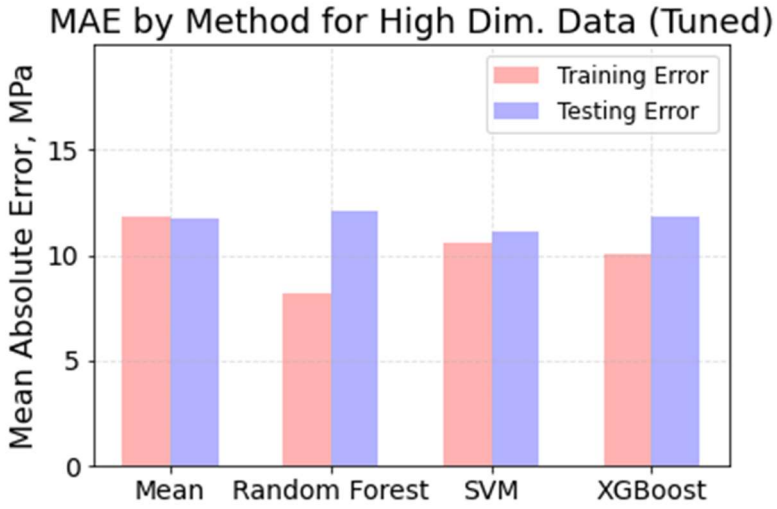


Figure 13. *MAE in UTS prediction results for the high dimensional dataset using tuned parameters. Compared to the default algorithm results in Figure 7, the amount of overfit in the Random Forest and XGBoost is lessened. The SVM improvement is imperceptible in the graph.*

Figure 13 represents the algorithm performances on the full dataset containing all process input columns. Dimension reduction techniques were applied to the data to reduce noise of marginal features and further reduce the gap between the testing and training error. The Random Forest and XGBoost algorithms have

an output called *feature importance* that shows which parameters have the most influence in training the model. The top 15 features from the tuned high dimensional Random Forest are shown in *Figure 14*.

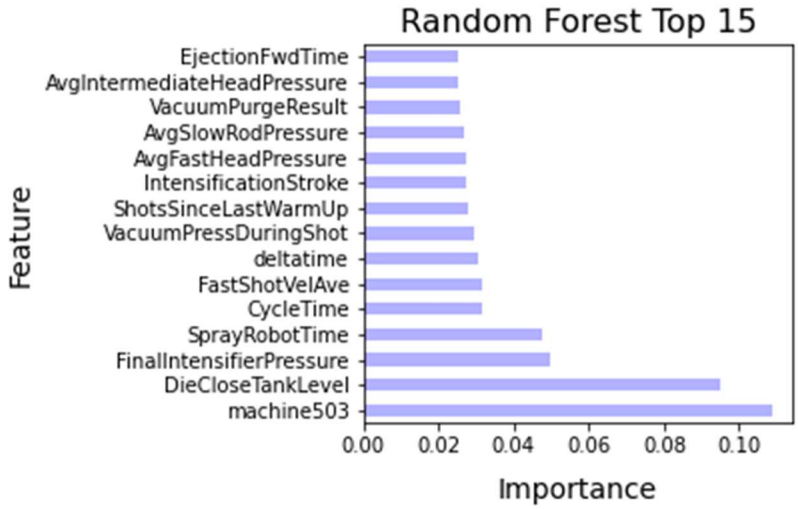


Figure 15. *Feature importance generated from the tuned Random Forest regressor. The top 15 features are shown.*

Machine 503 is connected to the heat treat schedule including the natural age step which resulted in a statistically significant higher UTS than the standard heat treatment (***Appendix D***). The Random Forest was able to identify that as being important. The die close tank level variable refers to the fluid level in the hydraulic tank. It is showing up as important because of a highly positive correlation to Machine 503 of 0.82. Beyond these two, the important features uncovered by the high dimensional tuned Random Forest look much like the parameters one finds in the literature when investigating the impact of process settings on mechanical properties or defects [78]–[82]. Based on this observation, a new feature selected dataset, "LitRev Features", was evaluated. The selected features are: Machine 503, average slow shot velocity, average fast shot velocity, average intermediate shot velocity, cycle time, intensification pressure, intensification pressure rise time, melt temperature, robot spray time (a proxy for amount of time the die was open between shots), and vacuum pressure during the shot. The predictive performance is displayed in *Figure 15*. The Random Forest improved significantly from the full feature set iteration in *Figure 13*.

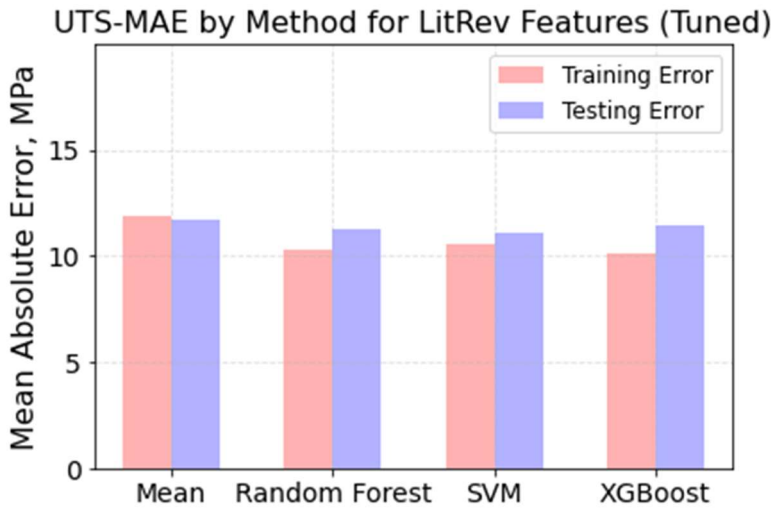


Figure 15. *Machine learning results on the feature selected dataset using important features from the literature. Small reductions in the training and testing error were found in all three algorithms with the most improvement in the XGBoost test data.*

Additionally, a different dimension reduction method, PCA, was applied to the high dimensional dataset. The number of principal components to explain 85% of the variation in the original dataset is 27. Each principal component is a linear combination of the original 77 dimensions, thus none of the inputs are completely dropped from the analysis as they are in feature selection. The PCA transformed data can be run through the same machine learning algorithms as the original data and the same tuning methods are employed. In *Figure 16*, the PCA Random Forest demonstrates the best performance overall in terms of UTS MAE and the degree of overfit.

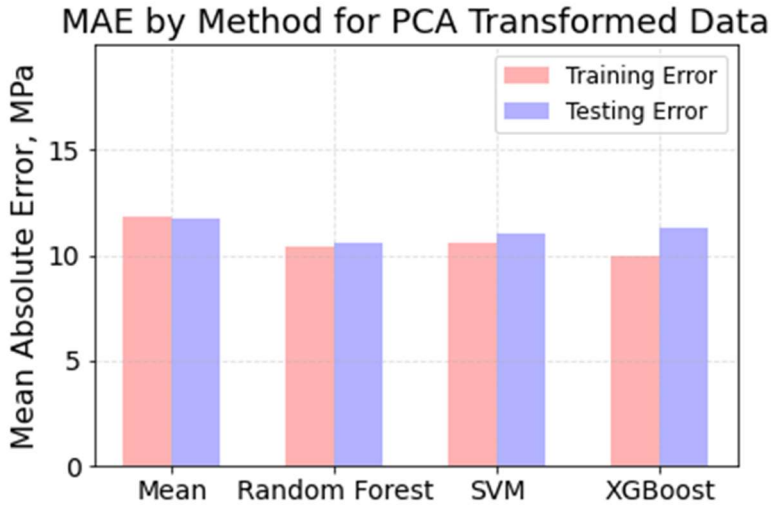


Figure 16. *Machine learning results on the PCA transformed dataset using top 27 principal components which explain 85% of the variation in the high-dimensional dataset. The Random Forest applied to the PCA dataset is the best machine learning performance in this study.*

Neural Network Regression Results

Deep learning based Neural Networks have proven useful for advanced analytics of big manufacturing datasets [65]. In this section, we will show results of a Neural Network model for predicting the UTS and a comparison of the Neural Network with traditional state-of-the-art machine learning models namely, the Random Forest, SVM, and XGBoost for the same dataset as shown above.

Selecting the right combination of parameters for the Neural Network is critical for optimizing the target metric and reducing the overfitting phenomenon [83]. We choose the parameters of the network in a way such that the model generalizes and does well on data that it has not seen during training. Success is measured by reducing both the MAE of the model and the difference between the training and testing MAE. The parameters tuned in the optimization of the Neural Network are learning rate, batch size, number of hidden layers, and the number of nodes within the hidden layer. *Figures 17 and 18* show the how the number of hidden layers and the number of nodes affect MAE. Similar plots for the other parameters examined are included in the full text in ***Appendix E.***

The learning rate parameter should be chosen in a way such that it is low enough that the model is able to reach the minimum error solution, while at the same time, it should be high enough such that the model does not take excessive time to converge [84]. A learning rate of 0.001 was the best performer.

Smaller batch sizes were found to give lower error as compared to higher batch sizes. The difference in the MAE is more significant as the batch size is increased above 128. We train the model using batches instead of training the entire data at once in order to make it computationally efficient and have other desirable properties such as avoiding local minima [83], [84]. We use a batch size of one for this analysis.

Figure 17 shows a comparison of the MAE using different number of hidden layers [85]. Using one hidden layer not only gives the best performance in terms of the MAE value but also gives the lowest difference between the training and the testing errors as compared to using a higher number of hidden layers.

Hidden layers of a Neural Network are comprised of nodes, which are the basic units of a Neural Network. The hidden layer is where the learning of the data takes place which includes learning important features of the dataset; also, obtaining a compressed representation of the data. Contrast this with machine learning where this step is accomplished by human input during pre-processing. The complexity of the model increases as the number of hidden layers is increased.

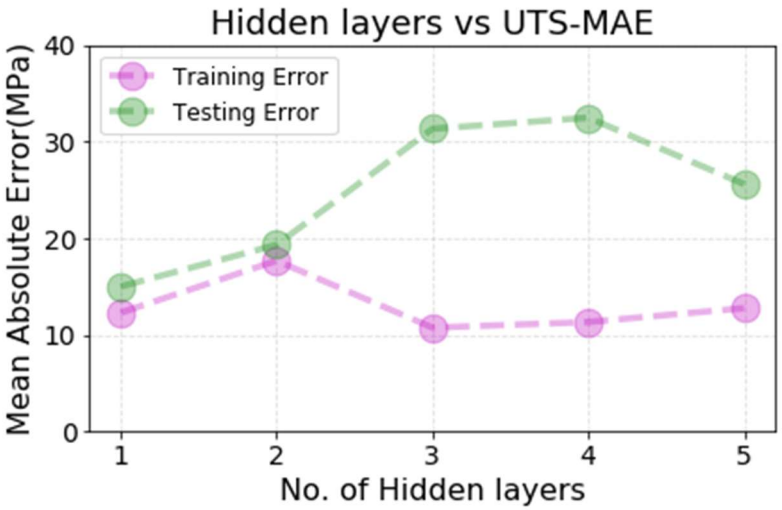


Figure 17. *A comparison of MAE as the number of hidden layers changes. The MAE value is lowest for one hidden layer as compared to higher number of hidden layers. Using one hidden layer optimizes the performance in terms of the metric itself and reduces overfitting.*

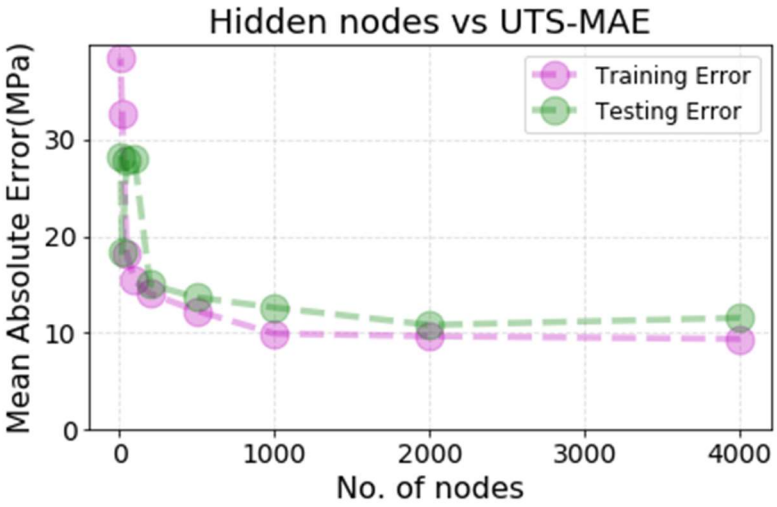


Figure 18. *Evaluation of MAE values in terms of number of nodes in the hidden layer. A higher number of nodes in the hidden layer performs better than fewer nodes.*

Figure 18 shows a comparison in terms of MAE with number of nodes in the hidden layer. A larger number of nodes gives lower errors as compared to lesser nodes in the hidden layer for this dataset. The difference between the training and testing errors is also low which shows that the model would generalize better on unseen data.

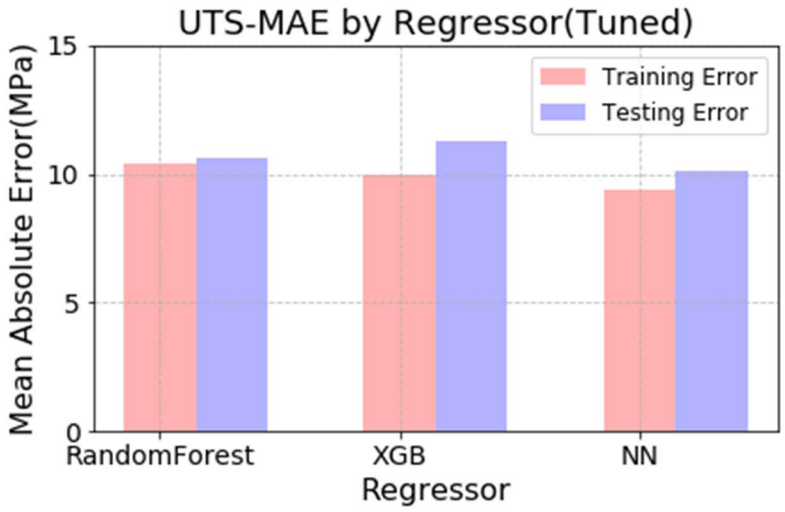


Figure 19. *Comparison of the optimized Neural Network model with optimized state-of-the-art traditional algorithms namely, Random Forest and XGBoost* [75]. *The Neural Network model gives the best performance in terms of the training as well as testing errors as compared to the traditional algorithms.*

Figure 19 shows a comparison of the optimized Neural Network model using the best combination of parameters with traditional machine learning algorithms namely, Random Forest and XGBoost. The Neural Network gives the best performance in terms of MAE as compared to the other two models on this dataset.

Figures 16 and 19 illustrate that by using either traditional machine learning methods or a Neural Network we can reduce the error in predicting UTS below that of predicting the mean value. Our results demonstrate the importance of understanding the relationship between algorithm complexity and the predictive error on a particular dataset. In the context of *Figure 10*, the tensile dataset fits in the area where the traditional machine learning and shallow Neural Networks cross. It is crucial to appreciate the bias-variance trade-off for this relationship, so that we select the appropriate algorithm, with optimal parameters, to improve the predictive performance.

Conclusions

- Machine learning and Neural Network regression models utilizing HPDC process data as inputs can improve the predictability of UTS above that of predicting the mean from prior tests. It is reasoned that the predictive power can be improved by increasing the number of rows and adding new input data columns.
- Principal component analysis is an effective dimension reduction technique to reduce complexity and overfitting of a dataset. A Random Forest of a PCA transformed dataset was the top performing machine learning method in this study.
- To optimize the models, parameter tuning must be performed with the objective of minimizing the error in the model predictions as well as the difference between training data and testing data errors.
- It can be seen that given the right combination of parameters for a Neural Network such as learning rate, batch size and number of hidden layers, the predictive performance of a Neural Network can be optimized not only in terms of the error metric but also in terms of obtaining a robust model fit for a given dataset without overfitting or underfitting.

- Selecting the correct models to use for the data being considered requires an understanding of the bias-variance trade-off such that a balance is struck between algorithm complexity and size of the dataset in question.

IV. CONCLUSIONS and IMPACT

Early project focus group meetings in the ACRC Big Data project brought to light that many in the materials processing industry, specifically foundries, were somewhat familiar with machine learning but not actively using it for mining knowledge from the data they currently collect. The needs of the industry were identified to be a platform for collection and fusion of data from throughout the operation, skill in the tools with which to work on large datasets and perform machine learning analysis, knowledge of how to evaluate the various available algorithms, and explore new data which may be helpful in making better predictive models.

Data Collection and Fusion

The platforms for data collection and organization exist. However, operationally, a key step is implementing an identification strategy that is common to all areas of the operation. Upfront commonization in data identification will save time in merging data from different operational departments later. Disciplined measures include tracking parts by serial numbers or lot codes, standard datetime formats, and common units of measure.

From there, it is a matter of connecting the DCM, and other hubs of data generation, to a server and make the data accessible to those who need it. For modern HPDC, the work cell is integrated enough that all data collection can be passed through the DCM computer. From the DCM, it can be uploaded to the cloud after each cycle. As an operation seeks to collect more and more data, there will be barriers to be overcome. There are only so many input slots in a programmable logic controller (PLC). Once those are full, more can be added. Then there is the issue of how much data a PLC can process. For example, adding an array of 10 thermocouples to a die casting tool to collect the temperature before and after the cycle is not an issue for most PLCs to handle. However, when collecting timeseries data such as the die temperature at 60Hz sampling frequency from those same 10 thermocouples over the 100 seconds between die close and die open, the data jumps from 20 to 60,000 data points per cycle for die temperature. At this point, new hardware may be required such as data acquisition (DAQ) devices which are designed to handle larger data demands.

The details of data storage are not in the scope of this work, but storage and access are critical pieces of successfully implementing a data strategy for machine learning. Before the advent of data warehousing in cloud-based data lakes, companies owned, maintained, and upgraded servers which came at a significant cost. With cloud-based data warehousing, a company rents space on a massive server network so that local storage does not get overwhelmed with old data. Local servers are tasked with short term access for recent data and once data reaches a certain age it gets moved up to long term cloud-based storage. The data is stored in a database which can be accessed and downloaded.

Developing a Machine Learning Skillset

Learning any new skill requires a significant time investment. Becoming capable at implementing machine learning will require foundries to develop talent, or bring in new talent familiar with key machine learning programming languages, especially Python [86], [87] and R [88]. The majority of the machine learning work in this thesis was conducted in Python 3. With Python, one can import libraries which facilitate the

various activities involved in a machine learning project. Key libraries used in this project are presented in ***Appendix C – Approach and Methodology.***

AI algorithms will run the data they are fed if it is of the proper type for the model. To avoid being misled one must follow the required steps and understand how each impacts the results. The main building blocks of a data science project are initial data exploration, pre-processing to prepare the data for use in algorithms, running the algorithms, evaluating the results, and iterating as necessary. This thesis serves as an assist to metal casters providing a method for machine learning using the type of data the industry generates and applying some useful algorithms. Most importantly, this work educates those new to machine learning on the topics of model metrics, the bias-variance trade-off, and cross-validation to evaluate the results of the models they train.

Once the integration and storage of data is set within an operation and some bedrock coding is in place, the ability to conduct further feature engineering, perform new analyses, try different algorithms, and add new data is very straightforward.

HPDC Data Conclusions

Data from the HPDC process was made available from FCA for two papers published from this effort. FCA runs a very large, by industry standards, HPDC facility in Kokomo, Indiana with state of the science capital equipment. The data routinely collected by this forward-thinking operation is much like that in my own experience in the industry at Mercury Marine and prior employers who ran larger HPDC operations. Modern HPDC machines collect much of the data and make it available. In production data, the performance of the machine follows closely to the process settings with some amount of natural variation. These data include average shot velocity through the different stages of the shot profile, head/rod pressure and intensification data. Based on the issues which arise at a given company, or for a specific part, additional data is collected to gain understanding and find variation in the process. The data provided by FCA is listed at the end of ***Appendix C.***

The results of the articles in ***Appendix D and Appendix E*** suggest that the standard HPDC cycle summary data does not contain the silver bullet parameter upon which the amount of porosity or ultimate tensile strength depend. In our regression modeling work, we compared traditional machine learning algorithm predictions to making predictions based on the mean UTS value of the training data and found that the error was lessened when employing the algorithms. The data at our disposal is largely what the literature has reported as being significant in affecting porosity and resulting mechanical properties in die castings (***Appendix A***). Notwithstanding, the amount of error reduction was small. Is the literature wrong? Not likely. Rather, most published studies investigating the effect of parameter X on porosity examine a wide range of X settings. Our research is aimed at a process where that level of optimization has already been performed. However, there is still variation in the mechanical properties of test bars excised from the castings. After application of feature engineering, feature selection, PCA dimension reduction, various algorithms, and parameter tuning only a small improvement was realized. This suggests that important input variables are not captured in the dataset.

Therefore, we must rethink about what other parameters can be included to increase our predictive power. Especially, consider parameters moving within a wide band because accurate control is difficult or costly. Perhaps there are parameters the HPDC industry has never considered measuring or controlling. It has been stated that one cannot control anything unless one has measures; the question is *which measures?* The question begs itself: *are we measuring the correct parameters?* One of the indirect key results of AI is the realization that perhaps what we have been measuring in the past is not appropriate, and that there are other key parameters that we should be capturing.

Metallurgists know that alloying elements are important with respect to porosity because the weight percentage of alloying elements impacts the solidus temperature of the aluminum alloy [66]. Copper, for example has a large effect on porosity in aluminum when increased over the range of zero to four weight percent [82], [89], [90]. Alloy composition was explored in this thesis, however, the variation in the elemental composition data was minimal (Table XI). Most rows required imputation to complete the dataset because chemistry is measured once per shift. Matching the cast time stamp of the part selected for mechanical testing to the time the furnace was evaluated is rarely close. With the imputed data, elemental weight percentages never came up as being significant to the algorithms because of the low variation. If a facility suspects that their composition is varying significantly, then data collection by real time method such as LIBS would be valuable [91].

Table XI. Average and standard deviation values of 930 alloy composition checks during the timeframe the engine block casting data were collected. Low variation in elemental composition results in individual element weight percent being unimportant in the machine learning algorithms.

	Cu	**Si**	**Fe**	**Mn**	**Mg**	**Zn**	**Ni**	**Pb**	**Sn**	**Ti**	**Cr**
Average (wt%)	3.38	8.45	0.89	0.22	0.26	1.36	0.08	0.04	0.02	0.05	0.07
Std Dev	0.08	0.27	0.04	0.04	0.03	0.27	0.02	0.01	0.02	0.01	0.01

Reflecting on other inputs which were not included in the data for this study, the temperature of the die cavity where the metal is solidified into its final shape is perhaps the most influential input parameter that is to a large extent passively controlled. Most die casters rely on a condition of *steady state*, which is a somewhat nebulous combination of cycle time, dwell time, die spray application parameters, cooling water temperature and flow rate, the melt temperature and amount of metal delivered during each shot, alloy chemistry, and the ambient environment in the factory. The effects of die temperature on porosity, misconceptions on steady state, and its influence on porosity are expanded on in Section V – Recommendations for Future Work.

Impact

A major impact of this research is fueling conversations, getting the industry thinking about data, talking about data, and using their data. Professional industry societies like American Foundry Society (AFS), North American Die Casting Association (NADCA), and The Minerals, Metals, and Materials Society (TMS) are creating forums through conferences and webinars to educate the industry on the application of machine learning to materials processing. I have been honored to participate and share my research through these organizations.

This project has garnered industry attention by way of presentations at national conferences like the 2018 AFS Metalcasting Congress and 2018 AFS Aluminum Division Specialty Conference. Further propagation occurred in presentations for the ASM Milwaukee Chapter in November 2018 and twice yearly at WPI's Advanced Casting Research Center from 2017 through 2020. Industry has responded. The American Foundry Society (AFS) scheduled their first Foundry 4.0 Conference for June 8-10, 2020. Due to COVID-19 pandemic, the conference has been postponed until spring 2021. This conference will bring in experts from around the industry to discuss what the foundry of the future will look like and machine learning will be an important pillar of the program.

Universities and professional societies are critical in the distribution of publications and knowledge. WPI via the ACRC, AFS, and NADCA provide excellent opportunities for their member organizations to learn about the future of the metalcasting industry through papers, trade magazines, conferences, courses, and collaborative research. It is now in the hands of the foundries themselves to be intentional about data

collection and organization. Metalcasting operations must identify promising talent in the industry and support their continuing education as Mercury Marine has chosen to do.

The following papers and industry presentations are products of this research:

1. Adam Kopper, Ning Sun, Diran Apelian, "Creating Knowledge from Big Data in Metal Casting Operations", Presented at 122nd Metalcasting Congress, April 3-5, 2018, Fort Worth, TX, AFS.
2. Adam Kopper, "Influence of Process Variables on Mechanical Properties of High-Pressure Die Castings", Presented at AFS Aluminum Cast Conference 2018, November 5-7, 2018, Knoxville, TN, AFS.
3. Adam Kopper, "Data Analytics Opportunities in Production Foundry Operations", Presented at ASM Milwaukee Chapter Meeting, November 19, 2018, Milwaukee, WI.
4. Ning Sun, Adam Kopper, Rasika Karkare, Randy C. Paffenroth, Diran Apelian, "Machine Learning Pathway for Harnessing Knowledge and Data in Material Processing", International Journal of Metalcasting, Accepted for publication, July 2020.
5. Adam Kopper, Rasika Karkare, Randy C. Paffenroth, and Diran Apelian, "Model Selection and Evaluation for Machine Learning: Deep Learning in Materials Processing", J. of Integrating Materials and Manufacturing Innovation, submitted 7/10/2020.
6. Adam E. Kopper, Diran Apelian, "Predicting Quality of Cylinder Block Castings via Supervised Learning Method", International Journal of Metalcasting, Submitted July 2020.
7. Adam Kopper, "Bringing Artificial Intelligence to Your Materials Organization", TMS, July 22, 2020. Webinar.
8. Adam Kopper, "Casting Quality Prediction via Supervised Machine Learning", To be Presented at AFS Foundry 4.0 Conference, June 8-9, 2021

V. RECOMMENDATIONS for FUTURE WORK

Machine learning in manufacturing is in its infancy which makes the potential for future work seemingly boundless. There are many aspects of manufacturing to which machine learning can be applied near to the scope of this work. Similarly, one can think of studies far from the field of the present one, but no less interesting and transformative. In this section, we cover some of the data believed to increase the performance of algorithms in predicting UTS and porosity as well as interesting food for thought on other applications of machine learning. Owing to the large datasets required to start generating meaningful results, it is important to start collecting data now in these areas.

In the previous section, the temperature of the die cavity was proposed to be an important piece of missing information. At start-up, the die cavity steel increases in temperature as castings are made until the process reaches steady state. Often this concept of steady state has more to do with how the parts look than actual temperature readings from the cavity. Experience and process development history establish the criterion for steady state, which is usually an established number of cycles, rather than a temperature value.

Die cavities are plumbed and cooled with water lines to obtain favorable solidification patterns in the parts and keep the dies from getting too hot that the cycle time is impacted. Water lines are typically on or off at the valve, not cycled and timed like one might find in a low-pressure permanent mold casting operation. As a result, the die cavity is a heat sink with its temperature a function of the heat put in by the casting and the heat removed by the water lines, convection from air movement, and conducted into other die components. None of these heat removal operations are actively controlled with respect to the die temperature. Thus, timing changes within the process due to delays small and large, affect the temperature of the die. This is supported by the high feature importance of robot spray time, ejection time, and shot count since last warm-up shot in the present research each of which can serve as a pseudo-proxy for die temperature.

The importance of die temperature as a boundary condition for solidification processing and the potential for variation suggest that die temperature be studied further. Miller demonstrated in a 1-D model how many cycles it takes to attain a quasi-steady state [92]. This work challenges the common notions of steady state, as the actual number of cycles to reach a quasi-steady state are often much larger than five. Other studies, especially modeling based investigations, have shown that die temperature is a high impact parameter on castings and the dies themselves [93]–[96]. Die temperature was found to be relevant to making predictions via application of machine learning and Neural Networks [26], [27]. Mercury Marine has implemented thermal imaging technology for die temperature measurement and has demonstrated promise in anomaly detection such as active warm-up shot detection, die spray issues, and water line issues [95]. The data collected has not been used to actively control the temperature of the cavity steel. For these reasons, a robust and reliable method of collecting the die temperature is the next source of data to drive predictive modeling forward. Knowing how and what to measure will lead the industry toward active control of die cavity temperature [97], [98].

To support this claim that die temperature is an influential parameter, three process simulations were run using MAGMASOFT 5.4 [99]. The subject of the simulation is a balance shaft housing for an outboard motor application. In each simulation, six warm-up cycles are used. Conventional practice would agree we are at steady state after these six cycles and the next castings are kept as good parts. Metal temperature in the furnace was set to 677 ˚C (1250 ˚F) and the process parameters such as velocity, cycle time, intensification pressure are held constant across all simulations. The parameter under investigation is the starting temperature of the die cavity. For the three simulations the starting temperature of the cavity was set to 149 ˚C (300 ˚F), 232 ˚C (450 ˚F), and 315 ˚C (600 ˚F). Temperature data was collected in the runner and in an overflow as shown in *Figure 20*. The timeseries temperature data is presented in *Figures 21 and 22* for the runner and overflow thermocouples respectively. These results confirm the importance of monitoring and controlling die temperature. After six warm-up cycles the die is generally thought to be at steady state, but the simulation tells a different story. The starting temperature matters to what the die temperature will be when the operation starts considering parts to be of good quality.

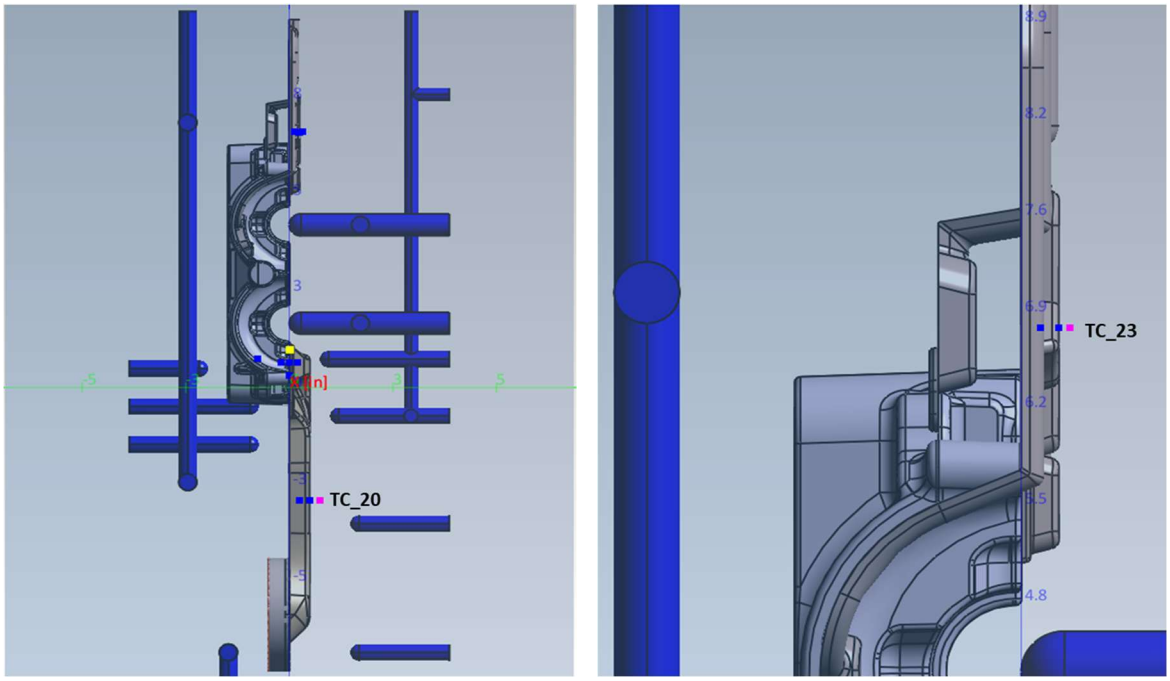


Figure 20. *Thermocouple locations where temperature data is collected during the process simulations. a) TC_20 is in the runner as shown on the left and b) TC_23 is in the overflow where the metal exits the cavity in the righthand image.*

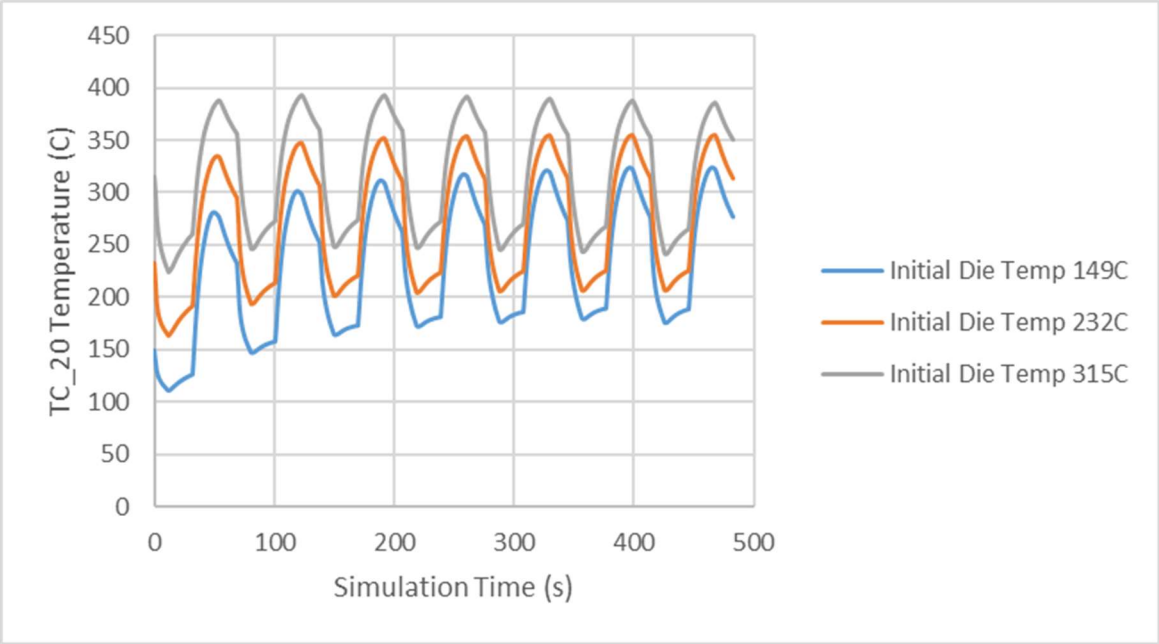


Figure 21. *Runner thermocouple temperature over six warm-up shots with varying starting temperatures 149 ˚C, 232 ˚C, and 315 ˚C. By differing starting temperature, the resulting "steady state" temperature of the die is not the same.*

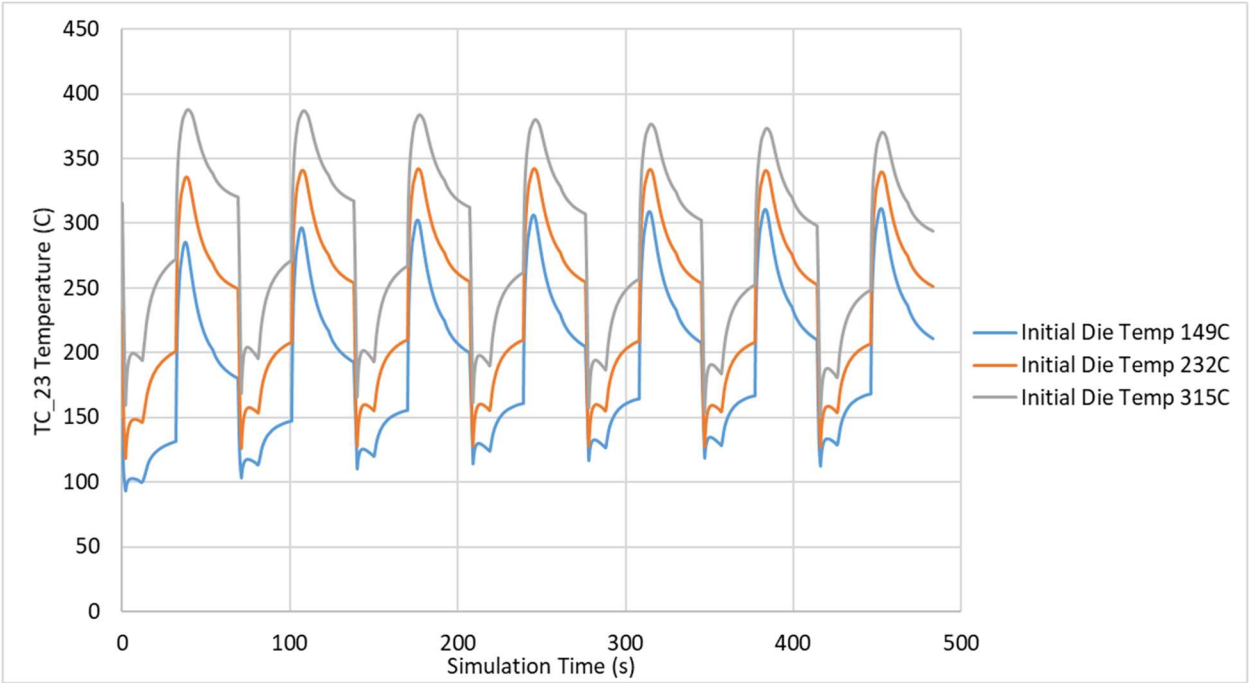


Figure 22. *Overflow thermocouple temperature over six warm-up shots with varying starting temperatures 149 ˚C, 232 ˚C, and 315 ˚C. By differing starting temperature, the resulting "steady state" temperature of the die is not the same.*

Our simulation predicts the die temperature, but the significance of that is measured by whether or not the internal soundness of the part is affected. Bulk porosity is an output of the simulation software, and Table XII has the predicted bulk porosity for the last cycle of the three models. The percent increase in porosity as starting die temperatures drop is enough to suggest that this is a process parameter that warrants more attention.

Table XII. Effect of initial die temperature on the bulk porosity in the balance shaft housing simulation after six cycles.

Initial Die Temperature (˚C)	Bulk Porosity (%)	Porosity Increase from 315 ˚C model
315 (600 F)	0.413	--
232 (450 F)	0.516	+ 25%
149 (300 F)	0.568	+ 38 %

Additionally, environmental data such as the ambient temperature and relative humidity in the plant were shown in this work to be important variables in determining porosity severity in a sand foundry operation (***Appendix B***). These may well be important in HPDC porosity too.

An adjacent application of machine learning which would be of high interest to foundries is X-ray porosity recognition and interpretation. Reading digital X-ray is a method performed by humans which is subjective and difficult to maintain a standard. It starts with being able to identify porosity from everything else in the X-ray image. Can a machine learn how a human determines what porosity looks like? Porosity recognition agnostic to the casting of which the X-ray was taken would be highly valuable to the foundry industry. The next step would be to determine if an area of porosity captured on an X-ray is anomalous when compared to the standard deviation about a mean X-ray image.

VI. REFERENCES

[1] S. P. Udvardy, "2018 State of the Die Casting Industry," *Cast. Eng.*, no. January 2019, pp. 10–15, 2019.

[2] A. Spada, "Revitalization of North American Metalcasting," 2012, Accessed: May 24, 2020. [Online]. Available: https://www.diecasting.org/docs/statistics/North_America.pdf.

[3] J. Folk, "U.S. Aluminum Casting Industry - 2019," *Cast. Eng.*, no. July 2019, pp. 16–19, Jun. 2019.

[4] D. Blondheim, "Artificial Intelligence, Machine Learning, and Data Analytics: Understanding the Concepts to Find Value in Die Casting Data," presented at the 2020 NADCA Executive Conference, Clearwater Beach, FL, Feb. 25, 2020.

[5] "Industry 4.0: the fourth industrial revolution- guide to Industrie 4.0." https://www.i-scoop.eu/industry-4-0/ (accessed May 26, 2020).

[6] K.-D. Thoben, S. Wiesner, T. Wuest, BIBA – Bremer Institut für Produktion und Logistik GmbH, the University of Bremen, Faculty of Production Engineering, University of Bremen, Bremen, Germany, and Industrial and Management Systems Engineering, "'Industrie 4.0' and Smart Manufacturing – A Review of Research Issues and Application Examples," *Int. J. Autom. Technol.*, vol. 11, no. 1, pp. 4–16, Jan. 2017, doi: 10.20965/ijat.2017.p0004.

[7] Capgemini Consulting Group, "Industry_4.0_-The_Capgemini_Consulting_V.pdf." Capgemini, 2014, [Online]. Available: https://www.capgemini.com/consulting/wp-content/uploads/sites/30/2017/07/capgemini-consulting-industrie-4.0_0_0.pdf.

[8] S. Weiss and I. Kapouleas, *An Empirical Comparison of Pattern Recognition, Neural Nets, and Machine Learning Classification Methods.*, vol. 1. 1989, p. 787.

[9] A. K. Jain, R. P. W. Duin, and Jianchang Mao, "Statistical pattern recognition: a review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 4–37, Jan. 2000, doi: 10.1109/34.824819.
[10] J. Fan and R. Li, "Statistical challenges with high dimensionality: feature selection in knowledge discovery," in *Proceedings of the International Congress of Mathematicians Madrid, August 22–30, 2006*, M. Sanz-Solé, J. Soria, J. L. Varona, and J. Verdera, Eds. Zuerich, Switzerland: European Mathematical Society Publishing House, 2007, pp. 595–622.
[11] L. Hauser, "Internet Encyclopedia of Philosophy," *Artificial Intelligence*. https://www.iep.utm.edu/art-inte/ (accessed May 26, 2020).
[12] J. Friedman, R. Tibshirani, and T. Hastie, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.
[13] D. Dietrich, B. Heller, and B. Yang, *Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data*, 1st ed. Wiley, 2015.
[14] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl, "Constrained K-means Clustering with Background Knowledge," in *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001, pp. 577–584, Accessed: Jun. 29, 2020. [Online]. Available: https://www.cs.cmu.edu/~./dgovinda/pdf/icml-2001.pdf.
[15] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern Recognit.*, vol. 36, no. 2, pp. 451–461, Feb. 2003, doi: 10.1016/S0031-3203(02)00060-2.
[16] M. Al-Maolegi and B. Arkok, "An Improved Apriori Algorithm for Association Rules," *ArXiv14033948 Cs*, Mar. 2014, Accessed: Jun. 29, 2020. [Online]. Available: http://arxiv.org/abs/1403.3948.
[17] Yanbin Ye and Chia-Chu Chiang, "A Parallel Apriori Algorithm for Frequent Itemsets Mining," in *Fourth International Conference on Software Engineering Research, Management and Applications (SERA'06)*, Aug. 2006, pp. 87–94, doi: 10.1109/SERA.2006.6.
[18] Y. Zhu and Y. Zhang, "The Study on Some Problems of Support Vector Classifier," *Comput. Eng. Appl.*, no. 13, 2003, [Online]. Available: http://en.cnki.com.cn/Article_en/CJFDTotal-JSGG200313011.htm.
[19] M. Peixeiro, "The Complete Guide to Support Vector Machine (SVM)," *Towards Data Science*, Jul. 29, 2019. https://towardsdatascience.com/the-complete-guide-to-support-vector-machine-svm-f1a820d8af0b.
[20] L. Deng and D. Yu, "Deep Learning: Methods and Applications," *Found. Trends Signal Process.*, vol. 7, no. 3–4, pp. 197–387, Jun. 2014, doi: 10.1561/2000000039.
[21] Aporras, "What is the difference between Deep Learning and Machine Learning?," *QuantDare*, Jan. 08, 2019. quantdare.com/what-is-the-difference-between-deep-learning-and-machine-learning/ (accessed Jun. 02, 2020).
[22] Y. Jiang *et al.*, "Expert Feature-Engineering vs. Deep Neural Networks: Which Is Better for Sensor-Free Affect Detection?," in *Artificial Intelligence in Education*, Cham, 2018, pp. 198–211, doi: 10.1007/978-3-319-93843-1_15.
[23] F. Nargesian, H. Samulowitz, U. Khurana, E. B. Khalil, and D. Turaga, "Learning Feature Engineering for Classification," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, Melbourne, Australia, Aug. 2017, pp. 2529–2535, doi: 10.24963/ijcai.2017/352.
[24] W. Long, Z. Lu, and L. Cui, "Deep learning-based feature engineering for stock price movement prediction," *Knowl.-Based Syst.*, vol. 164, pp. 163–173, Jan. 2019, doi: 10.1016/j.knosys.2018.10.034.
[25] T. Prucha, "From the Editor: AI Needs CSI: Common Sense Input," *Int. J. Met.*, vol. 12, no. 3, pp. 425–426, Jul. 2018, doi: 10.1007/s40962-018-0235-2.
[26] J. K. Rai, A. M. Lajimi, and P. Xirouchakis, "An intelligent system for predicting HPDC process variables in interactive environment," *J. Mater. Process. Technol.*, vol. 203, no. 1–3, pp. 72–79, Jul. 2008, doi: 10.1016/j.jmatprotec.2007.10.011.

[27] P. K. D. V. Yarlagadda and E. Cheng Wei Chiang, “A neural network system for the prediction of process parameters in pressure die casting,” *J. Mater. Process. Technol.*, vol. 89–90, pp. 583–590, May 1999, doi: 10.1016/S0924-0136(99)00071-0.
[28] R. Soundararajan, A. Ramesh, S. Sivasankaran, and A. Sathishkumar, “Modeling and Analysis of Mechanical Properties of Aluminium Alloy (A413) Processed through Squeeze Casting Route Using Artificial Neural Network Model and Statistical Technique,” *Adv. Mater. Sci. Eng.*, vol. 2015, pp. 1–16, 2015, doi: 10.1155/2015/714762.
[29] S. Balasubramaniam and R. Shivpuri, “Improving the Quality in Die Casting Production Using Statistical Analysis Procedures,” *NADCA Trans. T99-071*, 1999, [Online]. Available: http://www.diecasting.org/transactions/T199-071.
[30] “ACRC - Advanced Casting Research Center.” https://wp.wpi.edu/acrc/ (accessed Jun. 02, 2020).
[31] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002, doi: 10.1613/jair.953.
[32] R. Blagus and L. Lusa, “SMOTE for high-dimensional class-imbalanced data,” *BMC Bioinformatics*, vol. 14, no. 1, p. 106, Dec. 2013, doi: 10.1186/1471-2105-14-106.
[33] H. Han, W.-Y. Wang, and B.-H. Mao, “Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning,” in *Advances in Intelligent Computing*, vol. 3644, D.-S. Huang, X.-P. Zhang, and G.-B. Huang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 878–887.
[34] L. Breiman, “Random Forests,” *Mach. Learn.*, vol. 45, no. October 2001, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.
[35] G. Drakos, “Random Forest Regressor explained in depth,” *GDCoder*, Jun. 04, 2019. https://gdcoder.com/random-forest-regressor-explained-in-depth/ (accessed Jul. 12, 2020).
[36] “Z-Transform,” *Wolfram MathWorld*. https://mathworld.wolfram.com/Z-Transform.html (accessed May 26, 2020).
[37] T. Wuest, D. Weimer, C. Irgens, and K.-D. Thoben, “Machine learning in manufacturing: advantages, challenges, and applications,” *Prod. Manuf. Res.*, vol. 4, no. 1, pp. 23–45, Jan. 2016, doi: 10.1080/21693277.2016.1192517.
[38] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.
[39] F. Y. Kuo and I. H. Sloan, “Lifting the Curse of Dimensionality,” *Not. AMS*, vol. 52, no. 11, pp. 1320–1329, 2005.
[40] H. Abdi and L. J. Williams, “Principal component analysis: Principal component analysis,” *Wiley Interdiscip. Rev. Comput. Stat.*, vol. 2, no. 4, pp. 433–459, Jul. 2010, doi: 10.1002/wics.101.
[41] S. Wold, K. Esbensen, and P. Geladi, “Principal Component Analysis,” *Chemom. Intell. Lab. Syst.*, vol. 2, pp. 37–52, 1987, doi: 10.1016/0169-7439(87)80084-9.
[42] C. Eckart and G. Young, “The approximation of one matrix by another of lower rank,” *Psychometrika*, vol. 1, no. 3, pp. 211–218, Sep. 1936, doi: 10.1007/BF02288367.
[43] E. Briscoe and J. Feldman, “Conceptual complexity and the bias/variance tradeoff,” *Cognition*, vol. 118, no. 1, pp. 2–16, Jan. 2011, doi: 10.1016/j.cognition.2010.10.004.
[44] “Bias-Variance Tradeoff in Machine Learning,” *AI Pool*, Oct. 20, 2019. https://ai-pool.com/a/s/bias-variance-tradeoff-in-machine-learning (accessed Jun. 02, 2020).
[45] A. Bhande, “What is underfitting and overfitting in machine learning and how to deal with it,” *medium.com*, Mar. 11, 2018. https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-howto-deal-with-it-6803a989c76 (accessed Jun. 08, 2020).
[46] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 785–794, Aug. 2016, doi: 10.1145/2939672.2939785.
[47] K. Song, F. Yan, T. Ding, L. Gao, and S. Lu, “A steel property optimization model based on the XGBoost algorithm and improved PSO,” *Comput. Mater. Sci.*, vol. 174, p. 109472, Mar. 2020, doi: 10.1016/j.commatsci.2019.109472.

[48] T. G. Dietterich, “An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization,” *Mach. Learn.*, vol. 40, no. August 2000, pp. 139–157, 2000, doi: https://doi.org/10.1023/A:1007607513941.
[49] A. Vezhnevets and O. Barinova, “Avoiding Boosting Overfitting by Removing Confusing Samples,” in *Machine Learning: ECML 2007*, Berlin, Heidelberg, 2007, pp. 430–441.
[50] A. Apsemidis, S. Psarakis, and J. M. Moguerza, “A review of machine learning kernel methods in statistical process monitoring,” *Comput. Ind. Eng.*, vol. 142, Apr. 2020, doi: 10.1016/j.cie.2020.106376.
[51] M. Hofmann, “Support Vector Machines — Kernels and the Kernel Trick.” 2006, Accessed: Jul. 05, 2020. [Online]. Available: https://cogsys.uni-bamberg.de/teaching/ss06/hs_svm/slides/SVM_Seminarbericht_Hofmann.pdf.
[52] M. Sanjay, “Why and how to Cross Validate a Model?,” *Towards Data Science*, Nov. 12, 2018. towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f (accessed Jun. 04, 2020).
[53] Cort J. Willmott and Kenji Matsuura, “Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance,” *Clim. Res.*, vol. 30, no. 1, pp. 79–82, 2005.
[54] T. Chai and R. R. Draxler, “Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature,” *Geosci. Model Dev.*, vol. 7, no. 3, pp. 1247–1250, Jun. 2014, doi: 10.5194/gmd-7-1247-2014.
[55] J. Davis and M. Goadrich, “The Relationship between Precision-Recall and ROC Curves,” in *Proceedings of the 23rd International Conference on Machine Learning*, New York, NY, USA, 2006, pp. 233–240, doi: 10.1145/1143844.1143874.
[56] J. I. Moore and P. J. Van Huis, “US4493362.pdf,” 4493362, Jan. 15, 1985.
[57] The Aluminum Association, *Designations and Chemical Composition Limits for Aluminum Alloys in the Form of Castings and Ingot*, October 2018. Arlington, VA: The Aluminum Association, 2018.
[58] R. I. Lerman and S. Yitzhaki, “A note on the calculation and interpretation of the Gini index,” *Econ. Lett.*, vol. 15, pp. 363–368, 1984, doi: 10.1016/0165-1765(84)90126-5.
[59] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, “Permutation importance: a corrected feature importance measure,” *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, Apr. 2010, doi: 10.1093/bioinformatics/btq134.
[60] ASTM International, “ASTM B 557-15, Test Methods for Tension Testing Wrought and Cast Aluminum- and Magnesium-Alloy Products.” ASTM International, doi: 10.1520/B0557-15.
[61] C. H. Cáceres, “On the effect of macroporosity on the tensile properties of the Al-7%Si-0.4%Mg casting alloy,” *Scr. Metall. Mater.*, vol. 32, no. 11, pp. 1851–1856, Jun. 1995, doi: 10.1016/0956-716X(95)00031-P.
[62] M. K. Surappa, E. Blank, and J. C. Jaquet, “EFFECT OF MACRO-POROSITY ON THE STRENGI~ AND DUCTILITY OF CAST,” vol. 20, no. 9, p. 6.
[63] C. D. Lee and K. S. Shin, “Constitutive prediction of the defect susceptibility of tensile properties to microporosity variation in A356 aluminum alloy,” *Mater. Sci. Eng. A*, vol. 599, pp. 223–232, Apr. 2014, doi: 10.1016/j.msea.2014.01.091.
[64] C. D. Lee, “Effects of microporosity on tensile properties of A356 aluminum alloy,” *Mater. Sci. Eng. A*, vol. 464, no. 1–2, pp. 249–254, Aug. 2007, doi: 10.1016/j.msea.2007.01.130.
[65] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, “Deep learning for smart manufacturing: Methods and applications,” *J. Manuf. Syst.*, vol. 48, pp. 144–156, Jul. 2018, doi: 10.1016/j.jmsy.2018.01.003.
[66] D. Twarog, D. Apelian, and A. Luo, *High Integrity Casting of Lightweight Components*, Publication #307. NADCA, 2016.
[67] J. G. Kaufman and E. L. Rooy, *Aluminum Alloy Castings Properties, Processes, and Applications*, 1st ed. ASM, 2004.

[68] W. D. Callister, *Materials Science and Engineering An Introduction*, 3rd ed. Wiley, 1994.
[69] L. M. Surhone, M. T. Timpleton, and S. F. Marseken, *Welch's T Test*. VDM Publishing, 2010.
[70] M. Drouzy, S. Jacob, and M. Richard, "Interpretation of Tensile Results by Means of Quality Index and Probable Yield Strength," *AFS Int. Cast Met. J.*, no. June 1980, pp. 43–50, 1980.
[71] S. Jacob, "Quality Index in Prediction of Properties of Aluminum Castings - A Review," *AFS Trans.*, vol. 108, pp. 811–818, 2000.
[72] M. K. Surappa, E. Blank, and J. C. Jaquet, "EFFECT OF MACRO-POROSITY ON THE STRENGTH AND DUCTILITY OF CAST," *Scr. Metall.*, vol. 20, no. 9, pp. 1281–1286, 1986, doi: 10.1016/0036-9748(86)90049-9.
[73] C. H. Caceres and B. I. Selling, "Casting defects and the tensile properties of an Al-Si-Mg alloy," *Mater. Sci. Eng. A*, vol. 220, pp. 109–116, 1996, doi: 10.1016/S0921-5093(96)10433-0.
[74] D. L. Twarog, "State of the Die Casting Industry," *Cast. Eng.*, no. January, pp. 16–25, 2011.
[75] A. Oppermann, "Artificial Intelligence vs. Machine Learning vs. Deep Learning," *Towards Data Science*, Oct. 29, 2019. https://towardsdatascience.com/artificial-intelligence-vs-machine-learning-vs-deep-learning2210ba8cc4ac (accessed Jun. 08, 2020).
[76] R. G. Mantovani, T. Horváth, R. Cerri, J. Vanschoren, and A. C. P. L. F. d. Carvalho, "Hyper-Parameter Tuning of a Decision Tree Induction Algorithm," in *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*, Oct. 2016, pp. 37–42, doi: 10.1109/BRACIS.2016.018.
[77] D. Krstajic, L. J. Buturovic, D. E. Leahy, and S. Thomas, "Cross-validation pitfalls when selecting and assessing regression and classification models," *J. Cheminformatics*, vol. 6, no. 1, p. 10, Dec. 2014, doi: 10.1186/1758-2946-6-10.
[78] L. Garber and A. B. Draper, "The Effects of Process Variables on the Internal Quality of Aluminum Die Castings," *NADCA Trans. T79-022*, 1979, [Online]. Available: http://www.diecasting.org/archive/transactions/T79-022.
[79] B. M. Asquith, "The Use of Process Monitoring to Minimize Scrap in the Die Casting Process," *NADCA Trans. T97-063*, 1997, Accessed: May 25, 2020. [Online]. Available: http://www.diecasting.org/archive/transactions/T97-063.pdf.
[80] S. L. dos Santos, R. A. Antunes, and S. F. Santos, "Influence of injection temperature and pressure on the microstructure, mechanical and corrosion properties of a AlSiCu alloy processed by HPDC," *Mater. Des.*, vol. 88, pp. 1071–1081, Dec. 2015, doi: 10.1016/j.matdes.2015.09.095.
[81] H. Cao, M. Hao, C. Shen, and P. Liang, "The influence of different vacuum degree on the porosity and mechanical properties of aluminum die casting," *Vacuum*, vol. 146, pp. 278–281, Dec. 2017, doi: 10.1016/j.vacuum.2017.09.048.
[82] I. Outmani, L. Fouilland-Paille, J. Isselin, and M. El Mansori, "Effect of Si, Cu and processing parameters on Al-Si-Cu HPDC castings," *J. Mater. Process. Technol.*, vol. 249, pp. 559–569, Nov. 2017, doi: 10.1016/j.jmatprotec.2017.06.043.
[83] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1 -- learning rate, batch size, momentum, and weight decay," *ArXiv180309820 Cs Stat*, Apr. 2018, Accessed: Jun. 08, 2020. [Online]. Available: http://arxiv.org/abs/1803.09820.
[84] Wen Jin, Zhao Jia Li, Luo Si Wei, and Han Zhen, "The improvements of BP neural network learning algorithm," in *WCC 2000 - ICSP 2000. 2000 5th International Conference on Signal Processing Proceedings. 16th World Computer Congress 2000*, Beijing, China, 2000, vol. 3, pp. 1647–1649, doi: 10.1109/ICOSP.2000.893417.
[85] J. de Villiers and E. Barnard, "Backpropagation neural nets with one and two hidden layers," *IEEE Trans. Neural Netw.*, vol. 4, no. 1, pp. 136–141, Jan. 1993, doi: 10.1109/72.182704.
[86] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.
[87] "Welcome to Python.org," *Python.org*. https://www.python.org/ (accessed Jul. 12, 2020).
[88] R Core Team, *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing, 2013.

[89] A. Fabrizi, S. Ferraro, and G. Timelli, “The influence of Sr, Mg and Cu addition on the microstructural properties of a secondary AlSi9Cu3(Fe) die casting alloy,” *Mater. Charact.*, vol. 85, pp. 13–25, Nov. 2013, doi: 10.1016/j.matchar.2013.08.012.
[90] S. G. Shabestari and H. Moemeni, “Effect of copper and solidification conditions on the microstructure and mechanical properties of Al–Si–Mg alloys,” *J. Mater. Process. Technol.*, vol. 153–154, pp. 193–198, Nov. 2004, doi: 10.1016/j.jmatprotec.2004.04.302.
[91] S. W. Hudson, J. Craparo, R. De Saro, and D. Apelian, “Applications of Laser-Induced Breakdown Spectroscopy (LIBS) in Molten Metal Processing,” *Metall. Mater. Trans. B*, vol. 48, no. 5, pp. 2731–2742, Oct. 2017, doi: 10.1007/s11663-017-1032-7.
[92] R. A. Miller, “Multi-time Scale Systems and Quasi Equilibrium,” *NADCA Trans. T16-082*, 2016, [Online]. Available: https://www.diecasting.org/archive/transactions/T16-082.pdf.
[93] S. Shahane, N. Aluru, P. Ferreira, S. G. Kapoor, and S. P. Vanka, “Optimization of solidification in die casting using numerical simulations and machine learning,” *J. Manuf. Process.*, vol. 51, pp. 130–141, Mar. 2020, doi: 10.1016/j.jmapro.2020.01.016.
[94] W. Sequeira, S. Sikorski, and M. Brown, “Application of Simulation As A Front-End Design Tool In Die Cast Product Development And For The Optimization Of The Die Casting Process,” *NADCA Trans. T02-012*, 2002, [Online]. Available: https://www.diecasting.org/archive/transactions/T02-012.pdf.
[95] D. Blondheim, “Unsupervised Machine Learning and Statistical Anomaly Detection Applied to Thermal Images,” *NADCA Trans. T18-071*, 2018, [Online]. Available: http://www.diecasting.org/transactions/T18-071.
[96] B. Kosec, G. Kosec, and M. Soković, “Case of temperature field and failure analysis of die-casting die,” *J. Achiev. Mater. Manuf. Eng.*, vol. 20, pp. 471–474, 2007, doi: 10.1.1.526.5501.
[97] W. Bishenden and R. Bhola, “Die Temperature Control,” *NADCA Trans. T99-051*, 1999, [Online]. Available: https://www.diecasting.org/archive/transactions/T99-051.pdf.
[98] D. Schwam, “Additive Manufacturing of Cores with Conformal Cooling Lines,” *NADCA Trans. T16-041*, 2016, [Online]. Available: http://www.diecasting.org/transactions/T16-041.
[99] *MAGMASOFT*. MAGMA.

Appendix A – Literature Review

I. Overview of Artificial Intelligence

The primary objective of the research is to delve into the nexus of materials processing and machine learning to begin our understanding of the challenges unique to the materials processing field. It is important to recognize that machine learning may be a new buzz word in industry and media, but the roots of machine learning trace back to Allied Forces code breaking during World War II and Alan Turing. The Turing test: where a remote human interrogator must distinguish between a computer and a human based on responses to a series of questions posed by the interrogator was proposed in 1950 as a method for determining when a computer, or an artificial intelligence, is thinking [1]. The mathematics was maturing, but the computing power of the mid-century was prohibitively expensive and only capable of executing commands [2], [3]. It could not store data. In 1956, the Dartmouth Summer Research Project on Artificial Intelligence is credited as being the kick-off of artificial intelligence and is, in fact, the event at which the term "artificial intelligence" was first introduced [4]. In 1965, Gordon E. Moore authored Moore's Law which states that every couple of years we can expect our computers be twice as fast and cost half as much [5] (*Figure 1*).

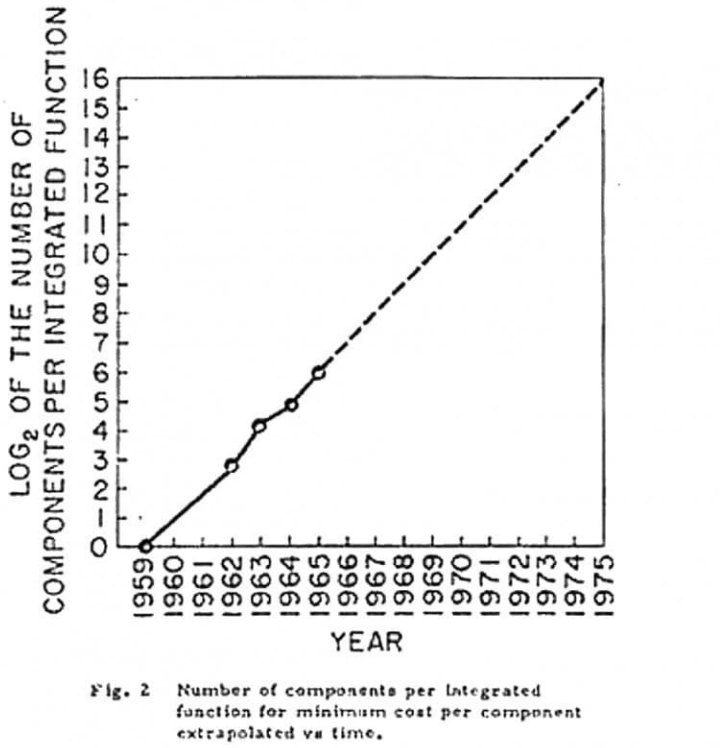


Figure 1. *Moore's original 1965 chart predicting future computing power increases.*

As computing power increased, popular culture anticipated the dawn of an era where machines would become sentient. Science fiction novels such as Asimov's I, Robot (1950) and Heinlein's The Moon is a Harsh Mistress (1966); are two classics in which artificial intelligence is presented as both fantastic and dangerous to its creator. Moviegoers of the 1980's were captivated by the possibilities of AI in classic films like WarGames (1983) and The Terminator (1984) while at the same time terrified at how our human ingenuity can be turned on us. These entertaining warning signs have not dissuaded human progress toward artificial intelligence, rather they served as inspiration to the curious and industrial minds who awaited the tools to make fantasy into reality.

In the scope of human progress, the wait has been brief. Since the above, the internet was created, and the first website launched in 1991. Data has since become an endless resource to be mined and monetized through targeted advertisements popping up before the eyes of the casual web surfer offering the right product at the right time based on data gleaned from internet viewing and purchasing history. Artificial intelligence would go on to notch key victories in man versus machine publicity events. In 1997, Deep Blue defeated chess grand champion Gary Kasparov and Watson tallied a win for AI on the TV gameshow *Jeopardy!* in 2011 [6]–[9]. Both stand as milestones in the progress of computers paralleling humans as simply machines taking in, storing, and processing data.

Behind the headlines, advances in very practical uses of machine learning continued and, today, the use of machine learning and deep learning is ubiquitous. The greatest source of data is people; all of us. Algorithms that learn about us, store that information as data, and make decisions are the brains behind Amazon's Alexa™ and Apple's Siri®. Texting is a commonplace method of communication that performs AI before our eyes, yet many take this marvel for granted. We even get frustrated and curse an autocorrect feature which is an amazing feat of machine learning on its own. When you send a text, you are offered options for your next word to speed the process of communication. One of those options is often exactly the word you were about to type. This is because the algorithm has been taught how humans communicate in various languages. Moreover, it is learning how the specific owner of the smartphone communicates. You can witness learning when you text something unique such as an uncommon name or a nonsense word that only you and your brother would understand. The first time, you are offered similar options which are familiar to the AI and, probably, you get autocorrected. Force the issue, and your phone learns that your friend does spell her name that way, and, in context, will offer that spelling after the first couple of letters are entered. How has artificial intelligence become so integrated into our lives without many even noticing? The answer is that the economics have changed [10], [11]. Today, we are reaping the benefits of Moore's Law in computing speed and data storage and the growth continues [12]. An updated chart of Moore's Law is presented in *Figure 2*.

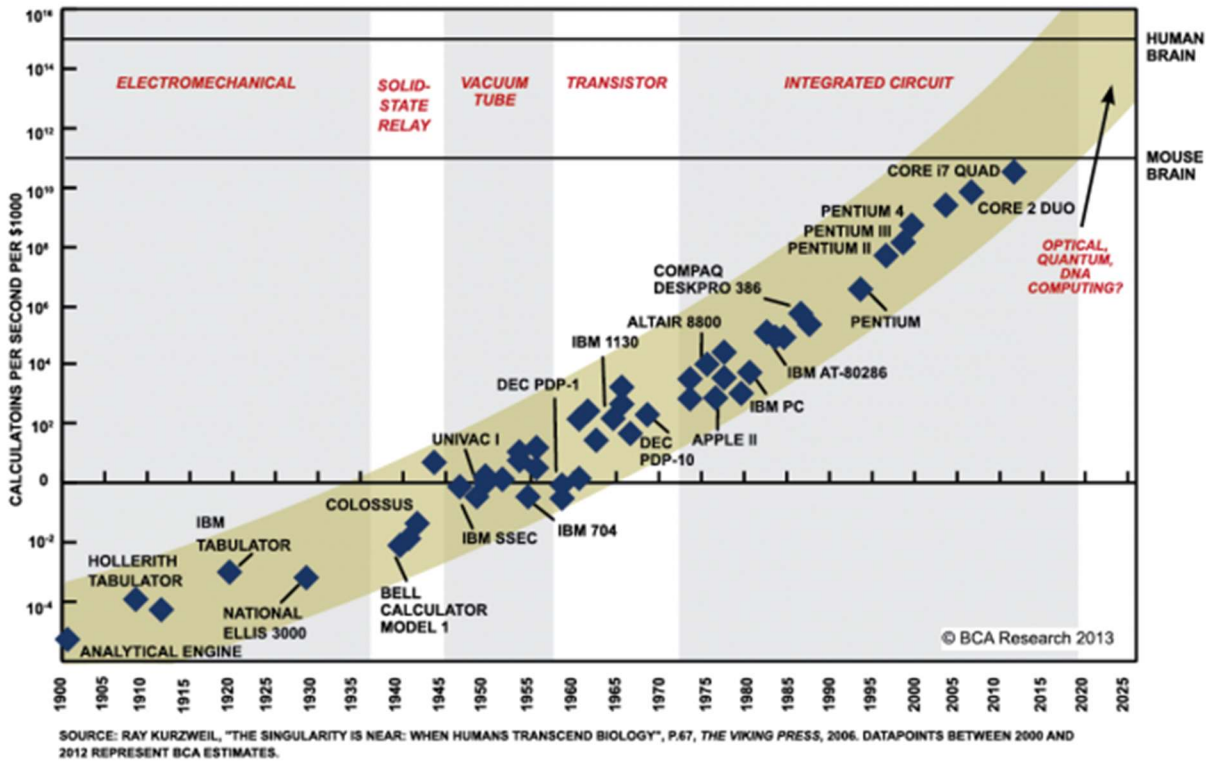


Figure 2. *Timeline of computing speed growth [12].*

Cloud data storage is allowing practically limitless amounts of data to be stored by companies without the need to install and maintain server capacity. Simply rent storage space and the provider houses and upgrades the equipment. The cost of memory has plummeted since Moore's landmark paper and today a megabyte of memory is pennies. The same amount of data, inflation adjusted, would have cost $5M in 1965 (*Figure 3*) [13].

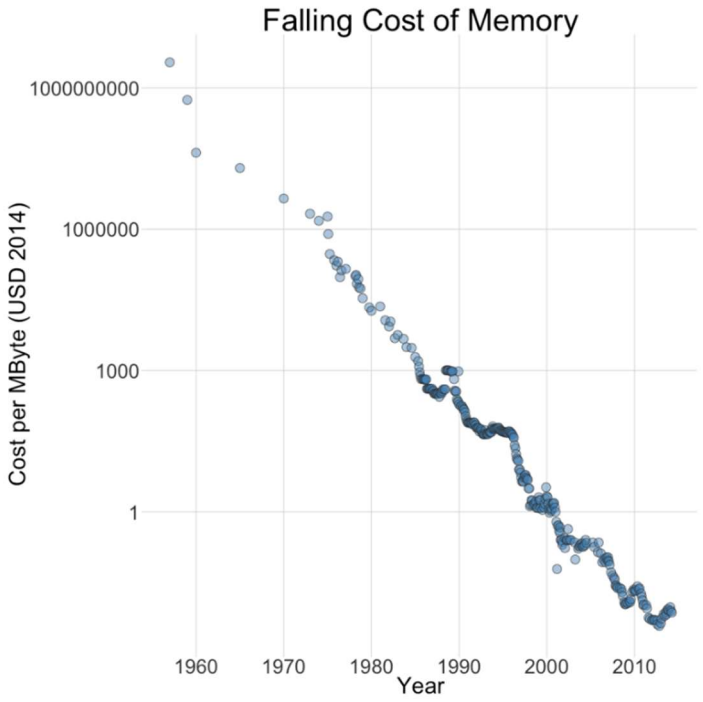


Figure 3. *Cost of memory has dropped sharply over the last 60 years [13].*

As computational power and data storage per dollar increase, applications for AI will grow and improve. This is not a fad, rather it is a transformation. The field of Data Science is growing and the job market for this skillset is expected to grow by over 27% from 2016 to 2026 [14]. As recently as 2015, there were virtually no undergraduate majors offered in Data Science. In 2020, there are over 60 universities offering a undergraduate Data Science major [15].

While marketing applications such as targeted ads and coupons, credit card fraud detection, and streaming media service recommendations are the average person's daily interaction with machine learning, the development of next wave applications is well underway [16], [17]. AI is driving object detection and sign recognition for autonomous vehicles [18]–[20]. The medical field is using predictive modeling in disease diagnosis [21] which is a game changing technology for rural areas and developing nations where doctors are few and collaboration is limited. Facial recognition is going beyond finding individuals and national security applications to emotion detection from facial pattern recognition [22], [23]. Once successfully introduced in a business function, artificial intelligence is not easily supplanted. Humans constantly seek productivity gains at home and at work. Businesses create value when they act on data driven decisions. Efficiency gains are realized when human resources can focus on executing value-creating action while algorithms do the laundry of crunching data 24 hours a day, 7 seven days a week to provide near real-time direction.

Terminology

In this document, thus far, the terms *artificial intelligence* and *machine learning* have been used interchangeably. Even now, as the field of artificial intelligence expands from the research groups and universities into the media and to the public, the language has not fully settled. Media outlets intermix

these words so as not to be repetitive or to create pizzazz, and it can be confusing to the reader of this work. *Figure 4* illustrates how common terms are related.

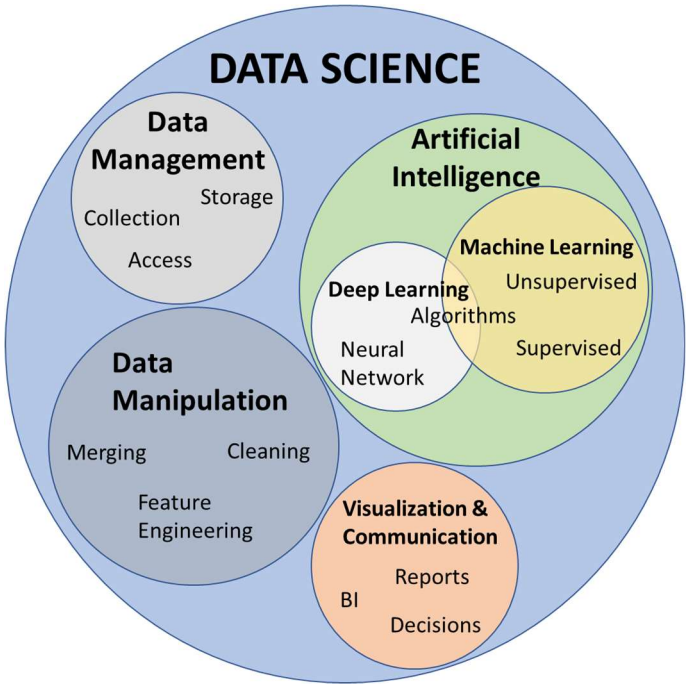


Figure 4. *Organization of common data science terminology.*

For clarity, I will use the following glossary of terms through this text.

Algorithms: Any list of instructions which describe how to perform a task. In data science, these are functions programmed in coding language which describe how a computer is to perform a task. For our purposes, algorithm refers to a specific machine learning method or approach.

Artificial Intelligence (AI): The general application of employing machines to analyze data in the aim of performing tasks or solving problems which require humanlike decision making.

Data Management: The collection, storage, and organization of data.

Data Pre-processing: Manipulation of data in preparation of performing analysis on the data. This includes merging, cleaning, normalizing, and feature engineering.

Data Science: An all-encompassing term for the tools involved in data management, pre-processing, analysis, and communication.

Deep Learning: A sub-set of machine learning which makes use of neural network learning algorithms to perform predictive analyses. Deep learning algorithms often perform feature engineering functions autonomously.

Feature Engineering: The application on domain expertise to the inputs of the dataset. Feature engineering investigates what new information can be gained from the existing input data, application of weight to inputs based on experience or physical laws, imputation strategy for missing values, and performing mathematical operators like logarithms to specific inputs. Feature engineering is not applied to output data.

Machine Learning (ML): A sub-set of artificial intelligence in which algorithms learn through statistical data analysis to make logic predictions. The teaching can be *supervised*, where the output of the training data is known, or *unsupervised*, where the output of the training data is not known. Machine learning algorithms rely on feature engineering by the user prior to executing the algorithm.

Neural Network: Neural networks are algorithms which run the input data through one or more hidden layers to determine the output. The hidden layers are made of neurons which incorporate weights, bias, and an activation function to arrive at an output which can be fed into the next hidden layer if there are more than one.

Observation: An observation is synonymous with the word sample or individual. In a data frame, these terms describe a row of data which consists of all the inputs and outputs associated with one unique object of interest. In this project, each observation represents a unique cast component.

Standardizing: Bringing various input data columns into the same scale so that they can be analyzed by machine learning algorithms which rely on mathematical distances to determine likeness.

Variables: Also referred to as features, inputs and outputs, or X's and Y's; variables represent the columns of a data frame. These are the process parameters which are used to train models for making predictions about new observations.

II. Industry 4.0

The fourth industrial revolution that ushered the Internet of Things (IoT) and the Internet of Services (IoS) has come to be known as Industry 4.0. At the Hannover Messe in 2011, Germany launched a project called *"Industrie 4.0"* designed to fully digitize manufacturing. The larger vision of Industry 4.0 is the digital transformation of manufacturing, leveraging advanced technologies and innovation accelerators in the convergence of IT (Information Technology) and OT (Operational Technology). The purpose is to integrate connected factories within industry, decentralized and self-optimizing systems and the digital supply chain in the information-driven cyber-physical environment of the fourth industrial revolution [16], [24]. The evolution toward Industry 4.0 is given in *Figure 5*.

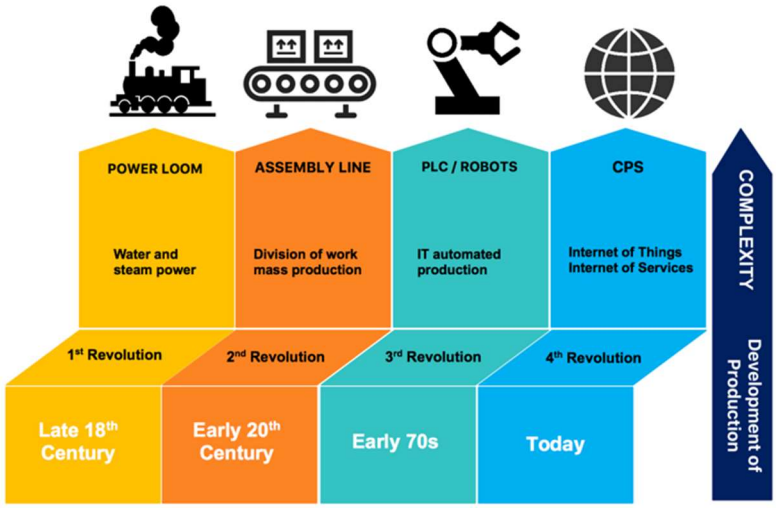


Figure 5. *Chronology and characterization of the four Industrial Revolutions* [24].

What does this mean for the foundry industry? Industry 4.0 is creating disruptive innovation in business models and revenue sources transforming manufacturing plants into smart factories or foundries. Smart foundries of the future will utilize Cyber-Physical Systems, digital interplay between the physical process and its virtual representation, to assess current production activity, optimize and adjust in real time, to ensure the best quality, efficiency, and production scheduling [25], [26]. IOT is the connection of, and communication between, every capable device and sensor in a system thereby breaking down the departmental and intra-factory information silos which hinder free data exchange. Extending beyond the four walls of the foundry to the subsequent machining and assembly processes downstream opens the possibilities of production scheduling adjustments and communication of specific quality issues instantly fed upstream to the foundry, even directly to the casting work cell. Through the proper application of AI and machine learning to Big Data analytics, the casting process can be fully mapped out and modeled to create knowledge of which combinations of process parameters make goods parts. Given downstream operational feedback and the current state of the process, the work cell can adjust its process accordingly to a move into favorable operating window without human interaction. Three keystone digital technologies will enable the transformation to smart factories: (i) connectivity, which implies executing industrial IoT to collect data from various segments of the plant; (ii) intelligent automation which includes advanced robotics, machine vision, digital twins, distributed control; and (iii) cloud-scale data management and analytics (AI and Machine Learning) [17].

Machine Learning in Brief

Avoiding semantical arguments on where the lines are drawn between machine learning and deep learning, it is generally agreed that both are forms of artificial intelligence rather than something entirely unique. Both are useful in the analysis of materials processing data as well. The term machine learning, in this text, represents the family of methods which use statistical and probability models trained on historical data to make predictions about new observations. Common methods which fall under this umbrella include linear regression, decision trees, k-means clustering, Apriori algorithm, and Support Vector Machines (SVM) [27]–[34]. While packages and commands readily exist to facilitate using such algorithms, these methods

are not black box functions shrouded in mystery. Many of them rely on using mathematical distances to determine how various observations are alike and what outcome should be expected if that information is known.

In materials processing, where the input data often greatly exceeds the output data, unsupervised methods such as k-means clustering are powerful tools allowing the user to group observations into clusters of likeness. Using k-means is a good way to detect anomalies in the process which may be associated with quality issues in the product or equipment performance issues which would be difficult to detect otherwise [35]. When the output data is known, classification and regression models can be created using decision trees or SVM for example. Thus, one can envision models which predict general attributes in a classification model or specific values of a given output in regression model. The details of the specific machine learning methods employed in this research are covered in ***Appendix C – Approach and Methodology*** section of this text.

Deep Learning in Brief

Deep learning utilizes the same data preparation strategies and similar functions with which to make predictions as machine learning [36]. Mostly, what makes two different is in the feature engineering (*Figure 6*) [37]. Feature engineering is where the data scientist relies on domain expertise to engineer the model inputs to make a higher performing model. One way this can be done is by assigning weights to specific input variables based on physical laws, experience, or other sources of privileged information [38], [39]. In machine learning, these weights are assigned manually. Deep learning utilizes hidden layers comprised of nodes which automatically assign weights to variables as the algorithm learns more about the data [39]–[41]. In this way, the deep learning algorithms are more of a black box than their machine learning kin. By using the training data to generate the weights automatically, deep learning algorithms can be more accurate than a human would otherwise be. As deep learning algorithms add additional complexity (i.e. increase the number of hidden layers or nodes per layer), it is critical that large datasets be used to train them. If not, the resulting model will not generalize well and, thereby, perform poorly on new data. The details of the specific deep learning method employed in this research are covered in ***Appendix E - Model Selection and Evaluation for Machine Learning: Deep Learning in Materials Processing*** section of this text.

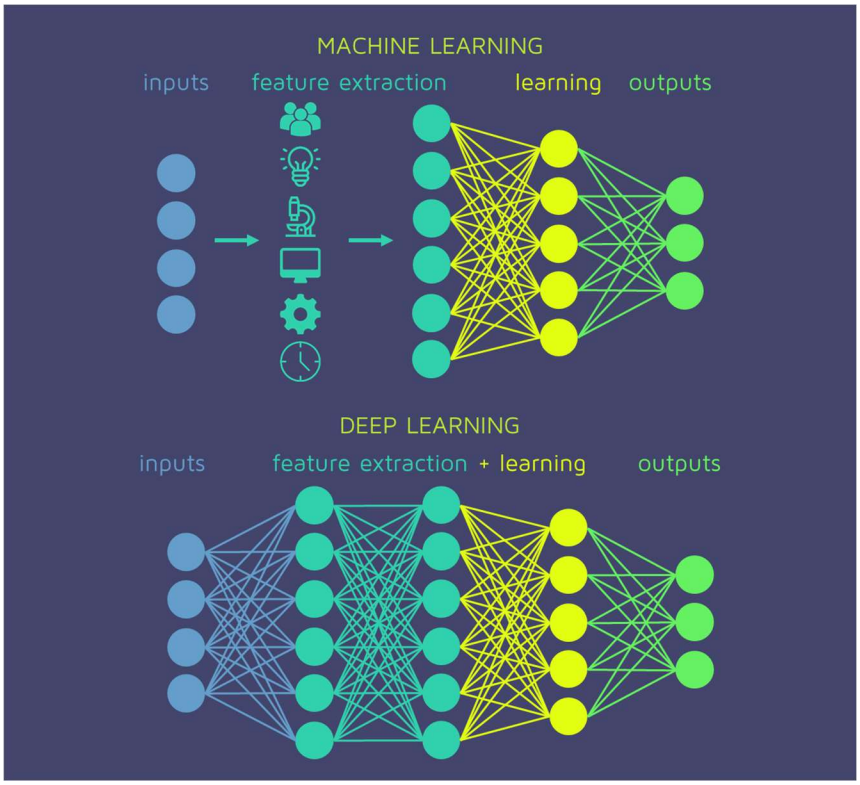


Figure 6. *The difference between machine learning and deep learning is how the feature engineering is performed* [37]. *Traditional machine learning utilizes subject matter expert input to the model while deep learning employs automatic processes.*

III. Materials Processing and High Pressure Die Casting

Materials processing is an interesting data processing and analysis challenge. Oftentimes, the product of the companies in materials processing industries is a raw material for the next operation which may be in the same facility or at a customer's operation where further value is added in the journey toward the final shape, assembly, etc. Its place in the product pipeline categorizes materials as commodities and pricing pressures are high. This is true for most engineering materials such as steel, aluminum, PVC, and plywood to name a few. Therefore, production processes are often large scale in terms of production tonnage and units per hour. In this climate, sampling each unit of the product for the purposes of quality assurance or process control slows productivity, adds cost, or may not be possible. Rather, once a process is running, the sampling is done by lot, by shift, or some such audit frequency at which the operation is confident in the product being produced. This creates a situation where the process is generating massive amounts of input data, but little output data. Adding to a general scarcity of output data, operational practices and situations can lead to missing data as heterogeneous data streams are fused together, where certain input data is captured for some parts, but not all parts. Lastly, for these businesses to be viable, they must be good at what they do. This leads to imbalanced datasets which are rich in good product but contain very few examples of bad product. The aforementioned are important challenges to working with manufacturing

data. The specifics of how to address these challenges through data science methods, and their appropriate cited works for the reader, are covered in ***Appendix C – Approach and Methodology*** and ***Appendix B – Machine Learning Pathway for Harnessing Knowledge and Data in Material Processing***.

Raw materials are not an isolated case, engineered components such as fasteners and metal castings reside in a similar economic climate. An important method for casting near net-shape components today is high pressure die casting (HPDC). HPDC has been in practice since the mid-nineteenth century. It started with the casting of low temperature lead and tin alloys used in the linotype industry [42]. Today, it is the most utilized casting method for aluminum components by tonnage in the United States and widely used throughout world [43], [44] (*Figures 7 and 8*). In terms of dollars, the North American Die Casting Association reported aluminum die castings to be over $8 billion in sales for 2019, while the American Foundry Society reports the entire aluminum foundry industry to be $9.67 billion [45].

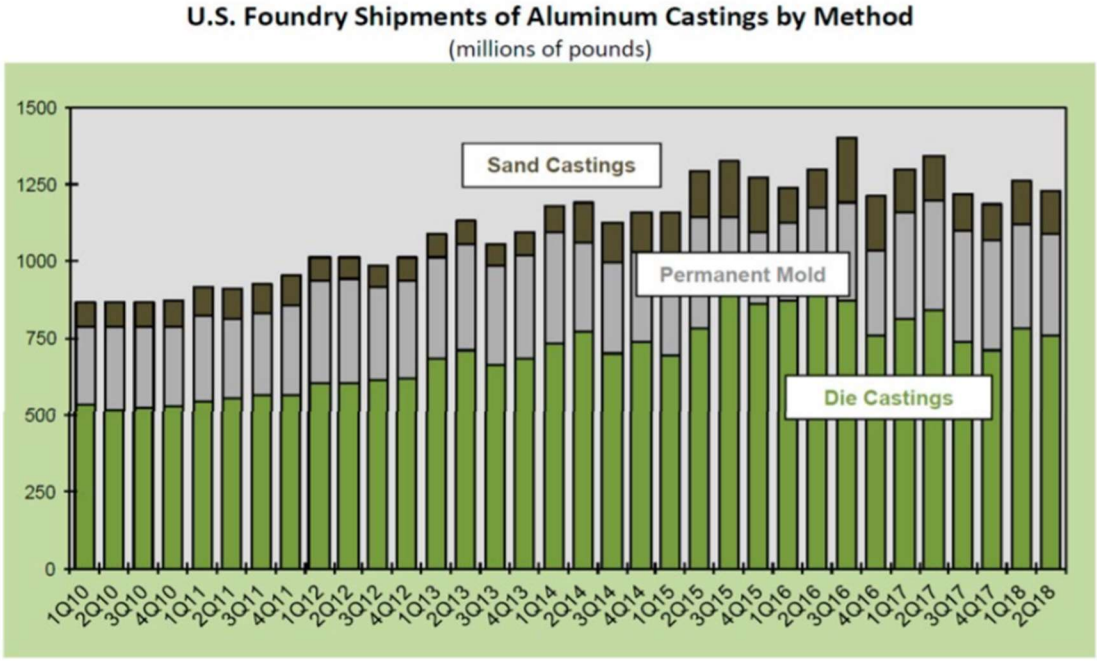


Figure 7. *Quarterly aluminum foundry shipment by process for the United States foundry industry 2010-2018. More die castings in tonnage are produced than sand and permanent mold combined* [43].

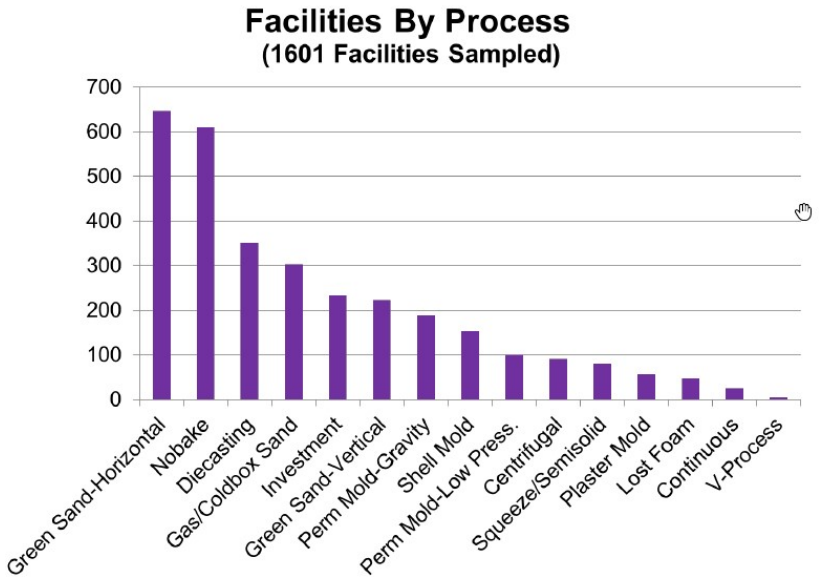


Figure 8. *Total North American foundry operations by process. Die casting trails two iron casting process as the third most employed process in North America* [44].

HPDC aluminum components are primarily employed where weight reduction and high annual production volumes are required. Die castings have evolved from largely housing-type components like covers and enclosures, sumps, and pump bodies to include more demanding applications like engine blocks and body-in-white nodes and pillars [46]–[50]. The demand for aluminum die castings is driven by automobile production, yet *Figure 9* shows many applications in recreational vehicles, marine propulsion, lighting, and household appliances [43].

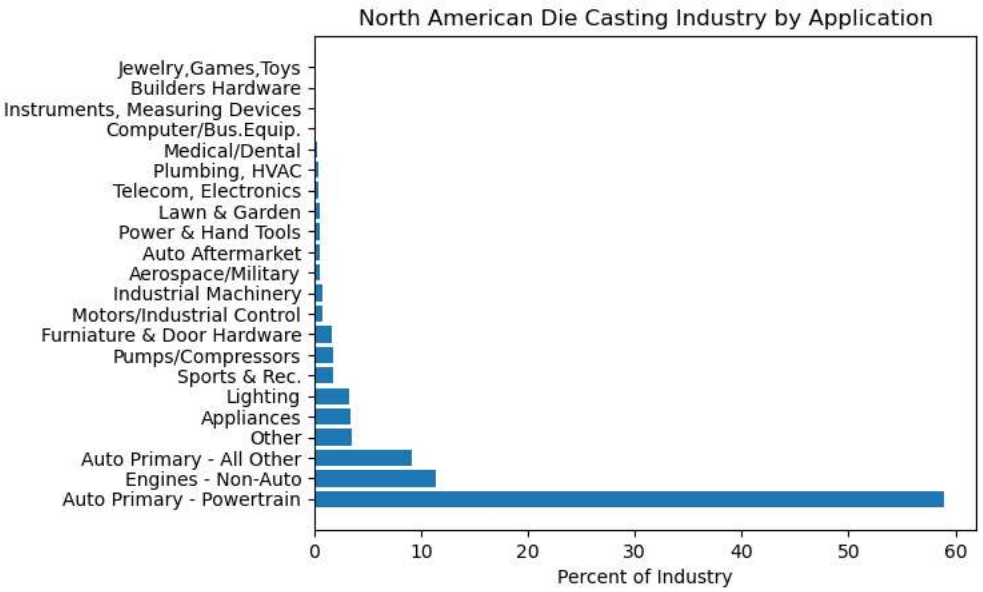


Figure 9. *Breakdown of the 2019 North American die casting industry by application* [43].

The 300-series of aluminum-silicon-magnesium alloys is the workhorse family of alloys in HPDC with varying chemistries aimed toward optimizing a combination of engineering considerations like cost, castability, machinability, corrosion resistance, strength, ductility, and wear resistance [51]. For this work in application of machine learning on materials processing, aluminum high pressure die casting will serve as the exemplar process. While the challenges in organizing and analyzing HPDC process data may be unique in some respects, they are believed to be similar, in general, across many materials processing disciplines. Machine learning and data science is of great interest to the materials processing industry. Imagine the benefit if these operations could model their quality by monitoring critical process inputs and running them through a machine learning algorithm. Such a future would result in increased uptime, rapid response to production issues in near real time, and data driven confidence that the product made between quality checks is acceptable.

A typical aluminum HPDC work cell consists of a furnace which holds the molten alloy, a transfer ladle, a cold chamber HPDC machine, a steel mold in which the casting is solidified, and a trim press for separating the parts from the runner system. Larger parts, which are awkward or too heavy for manual handling, benefit from additional automation such as an industrial robot for removing the solidified casting from the die and performing other finishing operations. *Figure 10* shows a typical HPDC casting work cell. *Figure 11* details key parts of the die casting machine (DCM).

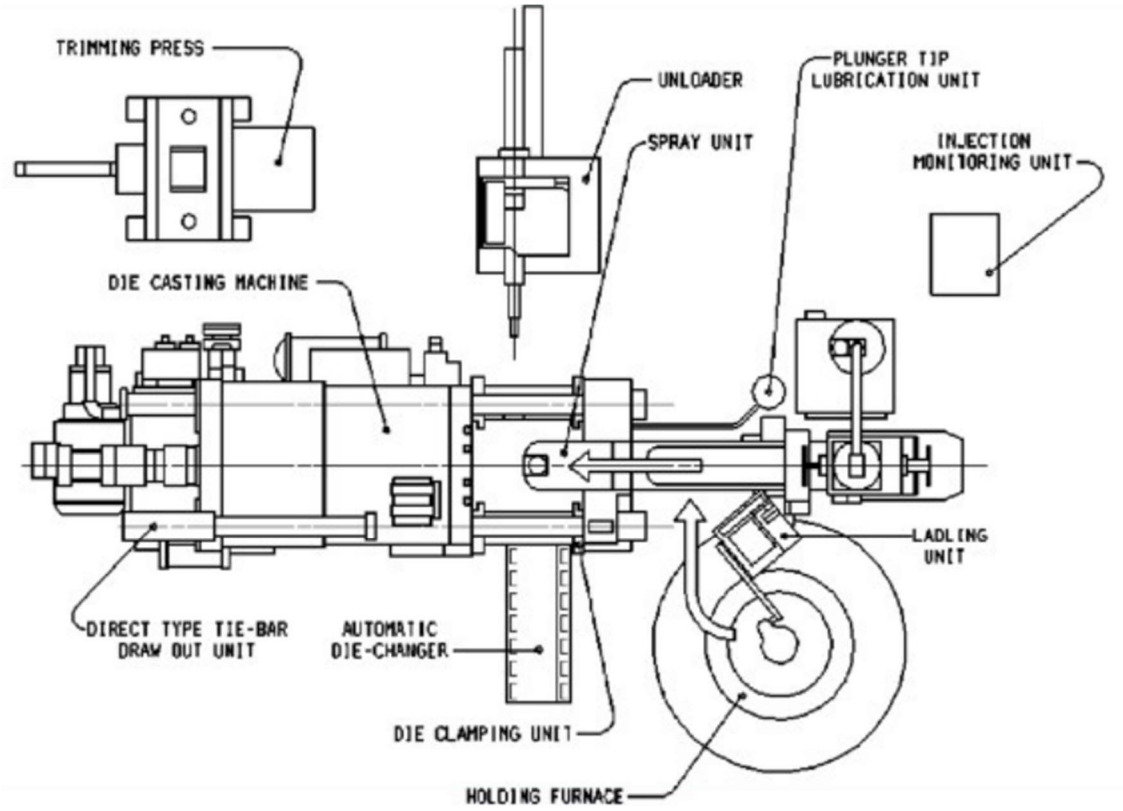


Figure 10. *A typical HPDC work cell layout* [42].

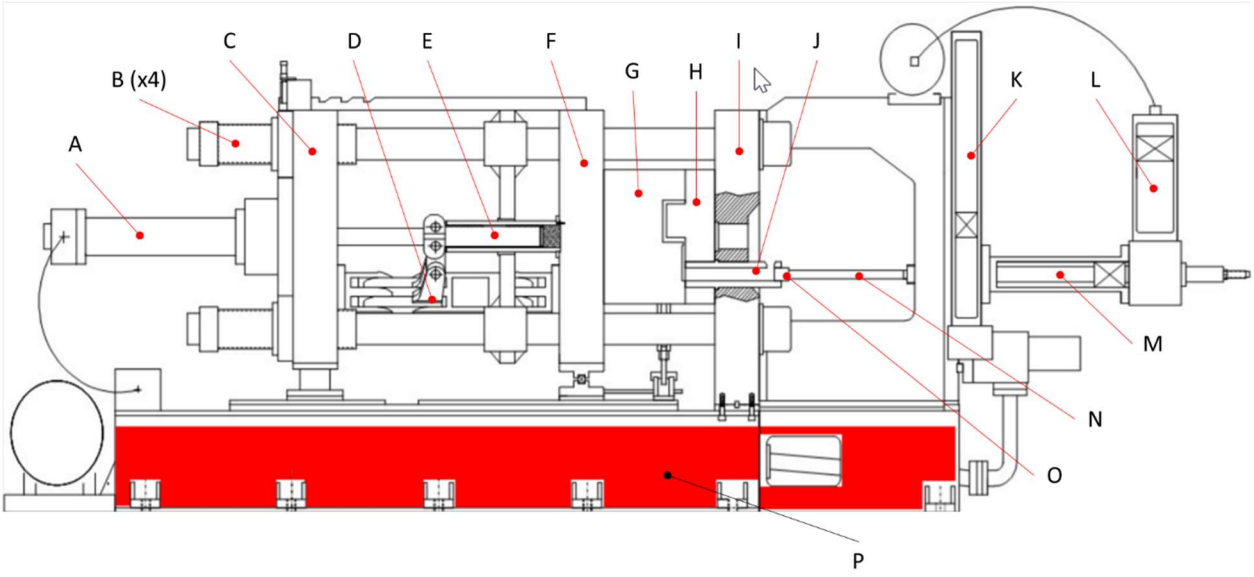


Figure 11. *Key components of a cold chamber die casting machine* [42].

A: Close Cylinder	*E: Ejection Cylinder*	*I: Stationary Platen*	*M: Shot Cyl.*
B: Tie Bars	*F: Moving Platen*	*J: Cold Chamber*	*N: Shot Rod*
C: Rear Platen	*G: Die (Ejector Half)*	*K: Intensifier Accumulator*	*O: Shot Tip*
D: Toggle	*H: Die (Stationary Half)*	*L: Shot Accumulator*	*P: Hyd. Tank*

For the sake of establishing an order to the cycle, one may choose die spraying as the start of the HPDC process and the steps of the cycle are as follows:

1. With the moving platen and, consequently, the die in the fully open position, the die cavity is sprayed with a release agent either manually or with an auto-sprayer. Auto-sprayers may be a purpose-built device or a six-axis industrial robot. Within the same step, the spraying is followed by an air blow-off to remove excess from the die cavity.
2. The plunger tip is drawn back to the pour ready position and the moving platen closes the die. The toggle clamp holds the die under locking tonnage. The tonnage is applied as the toggle puts the tie bars in tension imparting a force on the die to keep it closed against the intensification pressure of the machine cycle.
3. Molten alloy is ladled, or otherwise poured, into the cold chamber.
4. After pour, the shot cylinder valve is opened to begin the plunger movement forward. Slow shot is the term used for the portion of the plunger travel which moves the tip past the pour hole of the cold chamber.
5. Once the risk of metal splashing back out of the pour hole has passed, the shot control valve is opened further ramping up the velocity to the point where the metal has reached the gates. The gate is where the metal enters the die cavity.
6. When the metal has reached the gates, the shot control valve is fully opened the programmed amount and fast shot plunger velocities of 3.5 to 5 m/s (140 to 190 inches per second) are attained driving the metal to completely fill the cavity in a time period of 0.1-0.2 seconds.

7. Upon filling the cavity, the intensifier valve opens and applies additional pressure on the cavity as the casting solidifies. A range of 25 to 50 MPa (3500-7500 psi) [42] is suitable for most applications. This intensification pressure is used to feed porosity which results from volumetric shrinkage in the phase change from liquid to solid.
8. Once the gates are frozen, no further feeding takes place and solidification of the heavier sections continues in the part and the biscuit.
9. Upon solidification, the ejection platen retracts opening the die. The die is designed such that the casting (part, runners, and biscuit) rides with the ejector half of the tooling. The ejector plate is pushed forward driving ejector pins to free the part from the ejector half into a robotic gripper.
10. The extraction robot tends to post processing routines such as quenching, identification pin-stamping, trimming, and loading an exit conveyor. Once the extract robot is clear, the spray cycle is initiated, and the process begins again. Typical cycle times for large tonnage HPDC machines (>1600 ton) are in the range of two to three minutes.

HPDC is rich for data mining. Useful data can be pulled from the controllers of each piece of equipment in the cycle. Blondheim estimates that there are over 300,000 data which can be captured for each cycle [52]. If one includes thermal imaging data of the die cavity and the individual data points which make up the shot trace, this number explodes to over 2M input variable data per cycle. A reasonable estimate for an annual volume on one die casting machine is 100,000 cycles. That would equate to two-hundred billion data points per machine per year. Clearly, amassing features is not the challenge. Learning which features are most important and collecting enough observations to be sure of it is where the difficulty resides. Annual production volume of 200,000 pieces per year is a large number for HPDC but it is not Big Data. Small data presents challenges in the machine learning space and this thesis will serve to expand our understanding of these challenges and how to deal with them. This topic is covered in more detail in ***Appendix C*** of this volume.

IV. Application of Machine Learning to Die Casting

The high productivity nature of HPDC, where manufacturing outpaces the feedback the operation receives from quality checks and subsequent operations, is the right environment for applying machine learning tools to process data. Die casters would like to know which input variables are most important to control in their process and which outputs should be measured. They want to be confident that the process is in control and making quality castings. To keep costs low, it is preferable to add controls and measures to the casting process rather than add inspection and measurements post casting. It would be advantageous to know when equipment is running optimally and precisely when it is wearing out. To realize these benefits, the foundry industry must understand which machine learning tools fit their data environment.

Early applications of machine learning to HPDC center on the application of neural networks to predict virtual process outputs. Rai et al.. used supervised learning by creating datasets with process simulation software and then teaching a neural network to predict cavity fill time, solidification time, and porosity based on the process inputs of melt and die temperature and slow and fast shot velocities [53]. They found that the results of the neural network model compared well to those generated by commercially available finite element mesh-based simulation software but did so in near real time. Similarly, Yarlagadda et al.. predicted fill time from the melt temperature, die temperature, injection pressure, and casting weight with

a neural network trained via process simulation software and went a step further by including domain expertise from casting specialists [54]. Their predictions matched very closely to actual production die castings.

This is a worthwhile endeavor; however, simulation software packages are built utilizing assumptions which generate useful direction in building die casting tooling and choosing initial process settings. During process development, parameters are tuned more finely to optimize part quality. This tuning is done based on domain expertise and the results of actual castings. It is reasonable to expect a machine learning or deep learning algorithm to find the rules the simulation software is using and make very similar predictions. The next step is to apply the algorithms to serial production castings and determine which input variables are driving quality or mechanical performance metrics and direct the process engineer how to tune the process for best results.

The leap between the computationally trained algorithms and algorithms trained on observational data from casting operations may seem daunting. There are many variables which are not monitored or controlled on the factory floor (ambient environment, die temperature, cooling water flow rate) which either are not included in the simulation, can be held constant, or are tracked as an output. In a controlled experiment where 413-alloy aluminum is cast into simple cylindrical geometry under three levels of squeeze (intensification) pressure, die preheat temperature, and molten metal temperature, Soundararajan et al. were able to train and test a neural network predicting the UTS and YS of extracted tensile bars with a correlation coefficient of 0.95 and 0.96 respectively (*Figure 12*) [55]. In their experiment, the selected levels represent a wider range of process inputs than one might encounter on a fully developed production process. The objective is to see changes in the casting and have an algorithm learn and predict these changes. A production process, however, has one set of parameters. The objective is that there are no changes in the castings, part after part, 24/7. Predicting the UTS variation of each sample accounting natural process variation is a more difficult problem. This type of research lays the foundation from which the industry can build and develop algorithms which predict the UTS of serial production castings with low variation in input parameters.

A European research consortium called MUSIC (MUlti-layers control and cognitive System to drive metal and plastic production line for Injected Components) planned to investigate a broader range of process inputs via data analytics of experiments conducted on a highly instrumented HPDC cell. Results presented at the 2015 NADCA Die Casting Congress confirm our understanding about the effects of intensification pressure on porosity [56]. However, thus far, no predictive modeling based on experimental training data have been published. The application of machine learning tools to observed HPDC production data remains uncharted territory in the literature.

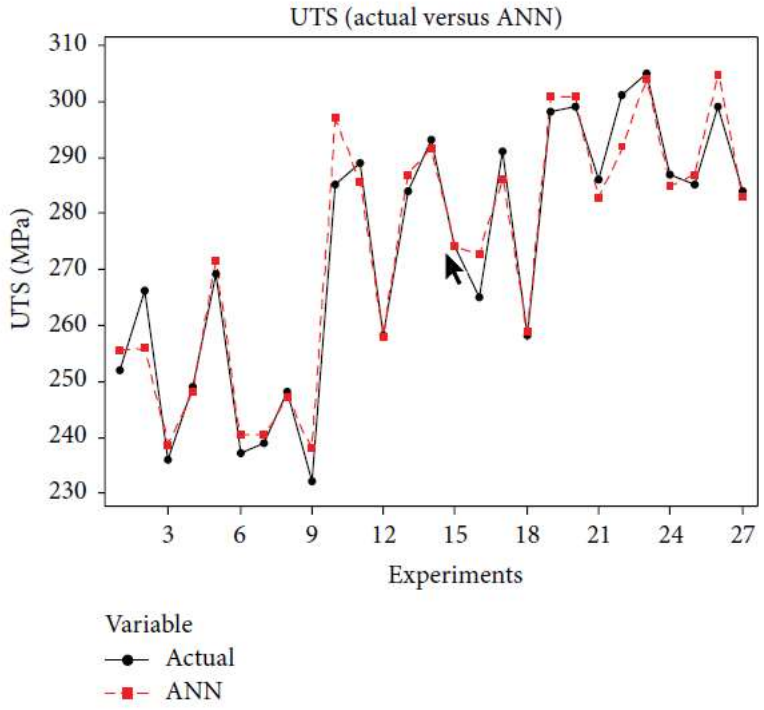


Figure 12. *Soundararajan et al. demonstrate the possibility for an ANN to predict mechanical properties of castings based on input parameters* [55].

V. Literature Review on Input and Output Variables in HPDC

For a HPDC company to be successful, they must produce good quality castings. One may ask, *what measurable outputs constitute a good quality casting?* A quality engineer will answer that a quality casting is one that meets the drawing and engineering specifications provided by the customer. Based on this definition, the acceptance criteria from one part to the next will vary according to the service requirements of the casting. Drawings and engineering specifications set the values and tolerances a foundry must target for acceptance by the customer. Examples include dimensions, surface finish, mechanical properties, hardness, internal soundness, and pressure tightness. Any of these can be considered as outputs for machine learning.

Dimensions are checked routinely at beginning and end of shifts and production runs in addition to die set-ups and in the case of die maintenance performed mid-run. Typical methods of dimension checks are via a coordinate measurement machine (CMM) or 3-D scanning technology. Surface finish is more often an issue as the die cavity begins to reach end of life. The steel die experiences thermal fatigue cracking on the surface which results in rougher and rougher casting surface [57]. Cast comparators and stylus profilometers are commonly employed to assign a roughness measure (Ra) to the part [58].

Alloy composition is a process input which is called out on engineering drawings. The specified alloy ties directly to mechanical properties and hardness as it is often the purpose of the elements added to the aluminum to drive these characteristics. Regular mechanical property testing is not typically required in traditional HPDC, but frequently is performed in other aluminum casting processes [59]–[61]. Even so, new structural applications in HPDC have increased industry awareness regarding specifying and measuring mechanical properties [46], [62], [63]. Tensile testing via methods outlined in ASTM B 557 is the accepted method for capturing yield strength, ultimate tensile strength, and elongation [64]. Microstructure is another measurable output which may be called out as a required range in grain size, eutectic silicon modification rating, and porosity limits [65]–[67].

Of the possible process outputs in HPDC which affect part performance, the amount of porosity and mechanical properties are the most widely studied. Many HPDC components have a fluid containing or transport function. Pump bodies, oil sumps and pans, compressors, valve bodies, various housings, and engine blocks are common examples. Porosity is the leading scrap issue for such parts and die castings in general (*Figure 13*) [68]. When these components are machined the porosity is exposed. Exposed porosity on sealing surfaces has the potential to undermine sealing gasket function. Interconnected porosity can link one machined surface with another creating an unacceptable leak path.

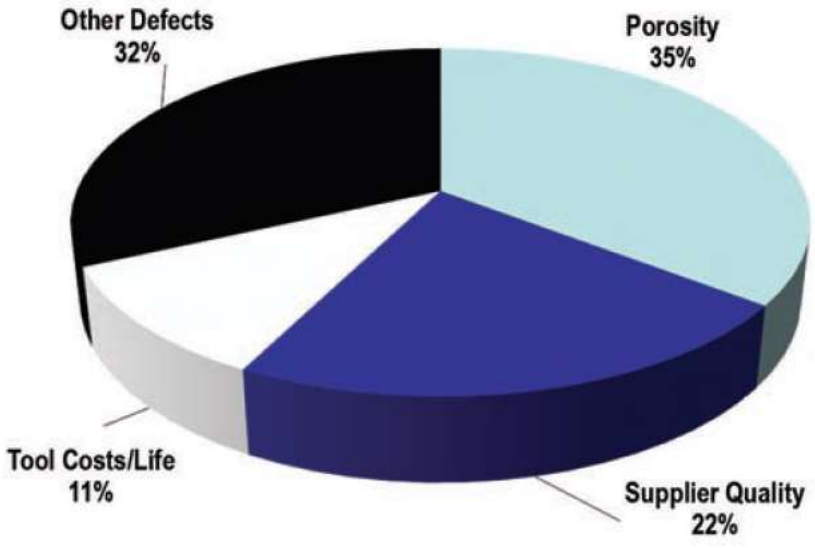


Figure 13. *Quality issues with die castings* [68].

It is of great importance that die casters gain as much knowledge about the root causes of porosity and how to avoid it in regions of the part where it is not allowed. It is widely known that the presence of porosity decreases the mechanical properties of tensile testing specimens, and this connection is often made in the literature (*Figure 14*). One may quantify porosity in terms of amount, or one may measure mechanical properties and link the lower performance test to increases in porosity [69]–[75].

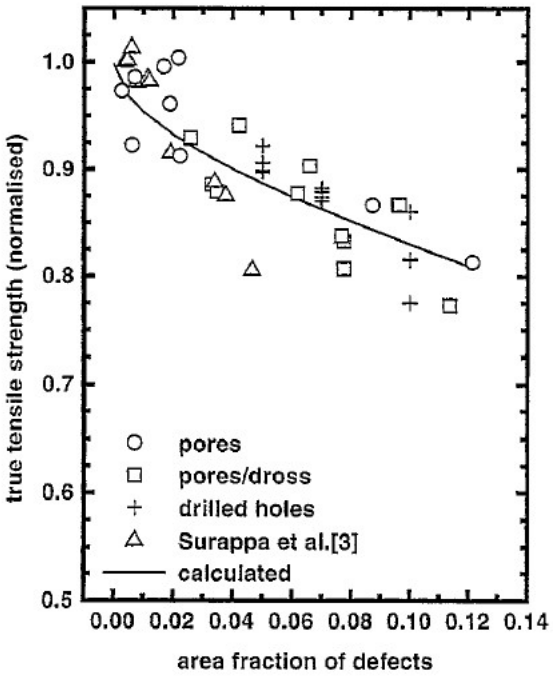


Figure 14. *Tensile strength suffers with increasing area fraction of defects at the fracture* [69].

In the North American Die Casting Association (NADCA) Operator Training course, students are taught that porosity can be generally assigned to two sources: gas and solidification shrinkage [42]. Gas porosity sources can be further broken down into entrapped air, gas from process lubricants, and hydrogen gas from the aluminum melt [76]–[78]. Shrinkage porosity results from the volume contraction during solidification and is more problematic in thicker regions of the casting which solidify last, after feed paths have been frozen off [77], [79], [80].

It is important to note that the condition of the starting melt plays a role in determining the resulting mechanical properties and porosity in the castings [65], [76], [81], [82]. Inclusions, such as oxide films and sludge, in the melt provide nucleation sites for hydrogen gas and block feeding paths exacerbating shrinkage [83]–[85]. The old phrase "garbage in, garbage out" certainly holds. This research is not focused on melt cleanliness as an input. The die-casting facility where these engine blocks are cast uses industry best practices in melt preparation and furnace maintenance. For the purposes of this research, the melt cleanliness is taken to be of high quality and practically constant. By making this assumption, the presence of porosity or mechanical property degradation due to hydrogen gas or inclusions in the melt are incorporated in the background noise of the data and not expected to be a cause of significant shifts in the results.

Gas from process lubricants can be severe [86], [87] , especially in manually operated die casting equipment where the amount of lubrication applied can vary from operator to operator and cycle to cycle. Fully automated HPDC work cells, such as those used in the generation of the data in this work, control the application and amount of lubricant very accurately [88]. Equipment issues will occur over normal production which lead to an increase in porosity. Leaking die spray manifolds, a leaking water line from a cracked die component, a damaged tip lube applicator, and heavy application of anti-solder compound on

the die are all issues for which die casters must be on the alert when a sudden increase in porosity shows up in their castings. What is common about these sources is that they arise and are remedied upon detection. They tend to affect most, if not all, castings while the issue is active.

While hydrogen porosity and external gas evolution can be argued to be in control and, for our purposes, constant, air entrapment is a different story. The two main sources of air in die castings are air being mixed with the metal in the cold chamber and die cavity air being mixed with the metal due to the turbulent filling of the mold [89]. In a 1979 study of process parameter effects on the density of A380-alloy HPDC castings, Garber and Draper established the baseline understanding of the importance of fast shot velocity, intensification pressure, die temperature gradient, melt temperature in the holding furnace, and die open time [90]. They report that decreasing the fast shot velocity and increasing the intensification pressure have the greatest effects on the improving the bulk density of their castings. They found only a small influence from die temperature gradient. With the exception of die temperature, the parameters they chose remain perhaps the most widely captured process input parameters in the industry today. The following paragraphs detail what researchers have found in relating die casting inputs to mechanical properties.

Effect of HPDC Process Parameter Inputs

Vacuum

Die casters have utilized vacuum assist to facilitate removal of air from the cavity during filling. The concept involves connecting a vacuum pump to the venting system of the die to draw out the cavity air and any evolved gases ahead of the advancing metal once the plunger passes the pour hole. Most producers in the business of structural automotive die castings, which require the best mechanical properties in the industry, are employing a vacuum assisted method [91], [92].

Early quantitative studies showed that applying vacuum in the die cavity increased castings density and the ultimate strength of HPDC components [93]. Another study from the same research center showed that process parameter variation (e.g. shot velocities and intensification pressure) had an overriding effect on the density results which masked the effect of the vacuum, however, parts cast under vacuum did show an improvement in leak testing [94]. In a more recent work, Cao et al. cast engine blocks while varying the vacuum over a range from 100 to 500 mbar holding all else constant. They observed an increase in elongation and tensile strength as well as reduced porosity as the amount of vacuum is increased [95].

Slow Shot Velocity

The amount of mixing in the cold chamber is controlled by two parameters, the “percent full” of the cold chamber and the velocity of the plunger at the initiation of the cycle. The term “percent full” refers to how full the cold chamber is with molten alloy for each cycle. The fuller the cold chamber is with alloy, the less air there is to be potentially entrapped. NADCA recommends 50-70% full to minimize air entrapment during slow shot [42]. Slow shot is the term given for the initial plunger velocity setting and travel distance at the beginning of the cycle. Alloy is ladled into the pour hole of the cold chamber just ahead of the retracted plunger tip. The molten alloy runs along the length of the cold chamber until it meets the parting line of the die. At this point a wave is ricocheted back toward the plunger tip. Timing is key and the plunger forward motion is initiated when the ricochet wave meets the tip. A proper slow shot velocity will maintain a built-up wave at the face of the shot tip and push all the chamber air out ahead of it through the

cavity and vents [42], [96]. Too fast and the wave will roll entrapping air. Too slow and the wave will disengage from the tip and bounce back entrapping air. Important process inputs to include in a machine learning model from this phase are slow shot velocity and biscuit length which is an indication of the amount of metal poured.

Slow shot velocity has typically been studied in efforts to control the wave motion in the cold chamber. Thome, Brevick, and Chu comprehensively modeled the wave formation and its relationship to air entrapment accounting not only for a critical velocity for a stable wave front but also a critical acceleration. While actual critical values are dependent on cold chamber geometry and the amount of metal poured, accelerating to reach the critical velocity too quickly will result in instability and roll over [89]. Verran et al.. conducted analysis of the slow shot velocity, fast shot velocity, and intensification pressure with respect to their influence on porosity and cold shuts. They compared density and visual porosity ratings to numeric simulations and determined that low porosity results from low velocity in both slow and fast shot in combination with high intensification pressure [97].

Fast Shot Velocity

The second source of entrapped air is from the die cavity during filling. As the plunger continues to move forward, it is accelerated though an intermediate transition from the slow shot velocity to the fast shot velocity. Fast shot is the term used for the portion of the cycle where the cavity is filled with molten alloy. The positions at which the velocity changes take place are determined by volume calculations or with the aid of process simulation software. The objective is to have the plunger at the desired fast shot velocity when the molten metal reaches the entrances into the cavity (the gates) [98], [99]. The cavity fill time is measured on the order of milliseconds. A cavity space in the shape of an engine block will accept roughly 40 pounds of aluminum in one tenth of a second. It is fascinating that, while filling a cavity in fractions of a second and applying enormous intensification pressure, die castings can exhibit incomplete fills, misruns, or “cold shuts”. At the low end of what one might traditionally use for fast shot velocity Verran et al. observed cold shuts at 1.23 m/s which were eliminated at 1.95 m/s [97]. Cleary, the high velocities are necessary, and 3-5 m/s is typical for the industry [42]. The complex geometry and high fill velocity combine to yield a highly turbulent filling pattern. While it is turbulent, the filling is not chaotic. Software packages simulate the filling pattern quite closely [100]–[102]. Based on the simulated results venting is placed at the edges of the casting to allow as much cavity air as possible to escape ahead of the advancing alloy. Intermediate and fast shot velocities and cavity fill time are expected to influence porosity and, thus, mechanical properties.

When it comes to linking fast shot velocity to porosity, the results in the literature are mixed and details on the venting strategy in the tooling are not always reported. Gate velocities are also reported in the literature in lieu of plunger velocity. Increased plunger velocity results in increased gate velocity as shown by the equations below:

$$volumetric\ fill\ rate = Area_{plunger} * velocity_{plunger} \qquad Eq.\ 1$$

$$velocity_{gate} = \frac{vol.fill\ rate}{Area_{gate}} \qquad Eq.\ 2$$

Dargusch et al. reported no clear trends linking fast shot velocity to porosity in the castings of their study of intensification pressure effects [103]. In another study, Okayasu and his team looked at gate velocities which at the low to very low end of what is viable in production (0.15-40 m/s). The connection to the plunger velocity is not given in the paper. High speed cold chamber die casting had lower mechanical properties than their "ultra slow" speed die castings. They attribute this difference to scattered chill microstructure structure (e.g. cold flakes) from the metal solidifying the chamber of the shot sleeve [104]. In Lumley's study on heat treatment of high pressure die-castings, he varied the gate velocity and found that increased gate velocity lead to improved mechanical properties upon heat treatment [105]. Gunasegaram et al. proposed that increased injection velocity paired with properly designed gating results in fragmentation of impurities such as oxides, air bubble, and cold flakes due to the resulting higher shear rates and turbulent energy dissipation during flow. The more broken up and dispersed these were, the better the properties would be. They found that higher UTS and elongation were achieved with higher injection velocity [106].

Intensification Pressure

Intensification pressure is applied at the end of the injection cycle to squeeze liquid metal from the thick biscuit and runners into the void space formed by contraction of the solidifying metal in the cavity. While the gates remain open and a feed path exists, shrinkage porosity can be supplied with additional metal. Once the feed path is frozen, typically this is at the gates but could be within the casting itself, thicker sections which contain liquid alloy cannot be fed and the result is shrinkage porosity. The amount of applied intensification pressure is typically in the range of 25 to 50 MPa (3500-7500 psi) [42]. Because the regions from which tensile bars can be extracted tend to be thick sections, microporosity from volumetric shrinkage is likely to be present. Thus, variation in the intensification pressure can be expected to influence porosity and, thereby, mechanical properties of the tensile bar.

The literature is consistent on the effect of intensification pressure. The higher the applied pressure during solidification, the better the mechanical properties, particularly ultimate tensile and yield strengths [107], and the less porosity is observed [97], [103], [108], [109]. Asquith reports that the relationship between intensification pressure and bulk porosity in the casting is linear [110] and follows the formula published by Kaye and Street in their book Die Casting Metallurgy [111]:

$$\% \, Porosity = a/P + b \qquad Eq.\ 3$$

Where P is the intensification pressure and a and b are empirically determined constants. The amount of intensification pressure available to apply in practice, however, is limited by the size of the die casting machine and the projected area of the casting itself [112].

Thermal Inputs

At its simplest, any type of casting is solidification processing. The fundamental laws of heat transfer dictate the solidification journey of a cast component. The initial thermal state of the system set the boundary conditions for the piece to be cast. Thermal inputs to the HPDC process include the ambient temperature of the factory, the holding temperature of the molten alloy, and the temperature of the die. Additional thermal inputs are the details of the die cooling system which determine the rate at which heat is removed from the system. Thermal effects of the die spray application are also considered. The durations

of time over which heat removal can take place are also inputs. Examples include the time the cooing water is on, die spray time, die open time, and delay time between retrieving alloy from the furnace and pouring the alloy. Of these, few are captured as data from shot to shot. Passive control tends to be the norm where a piece of equipment is programmed to do a task and send a signal that it has returned home. The next piece of equipment looking for that signal as an indication to "go" usually operates with a buffer where, if it takes too long to see that signal, it relays that something is wrong. This buffer is generally set to such a duration as to minimize a technician returning to the machine repeatedly to reset the same, perhaps benign, fault condition. For example, it may be that when a 40 second task takes 60 seconds, notification is given vie error message. However, smaller variations in time are forgiven, but may impact the thermal condition for the following cycle.

Of the thermal inputs introduced above, in practice and in the literature, melt temperature is the most captured and investigated. dos Santos et al. investigated combinations of melt temperature (579-709 C) and intensification pressure for AlSi9Cu3(Fe) (A380 alloy) and found that the porosity tends to increase with increasing melt temperature at high pressure, but no strong relationship was observed at low pressure [107]. Yang reports similar results in squeeze casting of near-eutectic aluminum silicon alloys with respect to mechanical properties; generally, the lower the melt temperature the better the mechanical properties [113].

The temperature of the die is difficult to fully characterize. A thermocouple can be used to capture the temperature in a specific location of the die, or several can be employed to get a better representation. The shortcoming of this is that the metal is affected not only by the temperature of the die where it came to rest, but every part of the die it encountered along the way. Thermal imaging shows the temperature profile of the entire cavity surface within its view but is cost prohibitive to widely employ. For this reason, researchers have long evaluated conditions which impact the die temperature, or otherwise alter the rate of heat extraction from the casting, by measuring the grain size or the secondary dendrite arm spacing (SDAS) of the resulting microstructure [66], [114]. These measures indicate the solidification rate of the casting. Solidification rate is highly influential on the mechanical properties of metals as fast solidification results in small grain size. Per the well-known Hall-Petch equation (*Equation 4*), smaller grains (and smaller SDAS) increase the yield strength of metals. This has been confirmed in HPDC 380 alloy [115], [116] and, specifically, on engine block castings [117].

$$\sigma_y = \sigma_0 + \frac{k_y}{\sqrt{d}} \qquad \textit{Eq. 4}$$

Where:

- σ_y is the yield stress of the material
- σ_0 and k_y are materials constants
- d is the grain size of the microstructure

Effect of Alloy Composition

Alloy composition follows the specifications of a registering body, such as the Aluminum Association, for popular die casting alloys such as A380, A383, B360, and 413 [118]. These specifications call out the elemental composition of the alloy in weight percent and the allowable range for each element. In some

circumstances, a tighter range may be specified by the customer, but this is rare due to the added costs associated with "off-spec" compositions. For HPDC, the most common family is the 300-series alloys which are primarily alloys of aluminum (Al), silicon (Si), and magnesium (Mg). Variants of Aluminum Association 380 alloy are often used in automotive powertrain castings like engine blocks and transmission housings. E380 is the alloy used in the engine blocks for this research. Additional alloying elements present in 380-type alloys include copper (Cu), zinc (Zn), iron (Fe), and manganese (Mn). Other trace elements are present as well. Trace elements are controlled to low levels as they are typically undesirable in the alloy.

The standard practice for measuring alloy composition is optical emission spectroscopy (OES) [119]. In OES, a sample of molten alloy is taken from the furnace and poured into a special mold which forms a test coupon casting. The test coupon is typically 2.5-3.0" in diameter and 0.5" thick. The surface is ground or machined and the OES uses a tungsten electrode to spark the prepared surface of the coupon. The light emitted from the spark is analyzed to determine the composition of the alloy. The purpose for laying out the steps of the test is to illustrate that the process is slow compared to production casting rates, thus it is not feasible to use OES as a method of capturing alloy composition on each part. Laser Induced Breakdown Spectroscopy (LIBS) is an interesting alternative method which would allow for real time collection of compositional data in the furnace for each part cast [120]. Its use is not widespread in die casting operations, though there is a real opportunity for data collection with LIBS in launder fed systems. The effects of various alloying elements in aluminum are described in the following paragraphs.

Silicon (Si)

In 380-type alloys, the largest elemental addition is Si with a nominal composition of 8.5% by weight. Because blending an alloy to a precise weight percent of each element is not practical, alloys are specified by an allowable range or a maximum allowable composition. For Si, that range in 380-aluminum is 7.5-9.5 wt%. Silicon plays an important role for die-cast aluminum alloys. Aluminum and silicon form a eutectic phase which increases the freezing range of the alloy. Long solidification range alloys have beneficial feeding characteristics. As a metal undergoes the phase transformation from liquid to solid, the volume contracts. This phenomenon is commonly called shrinkage. In long solidification range alloys, there is plently of liquid eutectic to feed the length of the mushy (solid + liquid) zone. Regions in the mushy zone which freeze off before being fed are small, even microscopic. This shrinkage, called microshrinkage, is preferred for die casting where there are no risers and limited means by which to feed the shrinking casting. Therefore, Si is an important element used to minimize the size of porosity and increase the castability [121]–[123]. The high melting point of Si imparts strength into the alloy at high temperatures allowing for parts to be ejected from the steel dies very shortly after solidification is complete [42].

The mechanical properties of Al are affected by additions of Si up to 7.0 wt%; strength increases and elongation decreases [124]. Additions beyond 7.0 wt% up to the eutectic composition of 12.7 wt% do not significantly increase the strength of the alloy [65], [124], [125].

Magnesium (Mg)

In the traditional die casting workhorse alloy A380, the Mg is held to a 0.1% by weight maximum. Similar alloys in other regions of the world such as ADC10 in Japan and ISO AlSi8Cu3Fe have long allowed a 0.3% by weight maximum Mg content [126]. E380 is an Aluminum Association designation which allows for the same 0.3% Mg as the international specifications. Magnesium combines with silicon to form an

effective precipitation hardening compound via T6 heat treatment in sand and permanent mold cast aluminum-silicon alloys which contain no copper. Magnesium is removed by common fluxing processes used for cleaning alloys in the smelting operation. Since traditional high-pressure die-castings were not heat treated, there was no cause to replace the Mg loss. Interest in investigating elevated Mg effects in traditional die-casting alloys peaked in the 2000's with a focus of gaining strength through precipitation hardening via heat treatment [105], [116], [127]. Yang et al. examined the effects of Mg on mechanical properties in Al-Si-Cu alloys as its weight percent was increased over the range of 0.01% to 0.88% [128]. They found that as the Mg level was increased, the yield strength and ultimate tensile strength increased. In the as-cast condition, additional Mg increased both strength values through the range, however, the impact on UTS was less. While strength increased, the elongation dropped. Once T6 heat treated, the addition of Mg beyond 0.3% offered no strength benefit and elongation continues to drop with increased Mg. Increased strength at the expense of elongation associated with increased Mg levels has been observed across similar alloys and other casting processes [116], [129], [130]. Fabrizi et al. examined the effect of various alloying elements on AlSi9Cu3(Fe) die-casting alloy [131]. They report an interesting interrelationship between the Cu, Strontium (Sr) and Mg content on the amount of microporosity in the resulting castings. While Cu and Sr were found to increase the porosity, additions of Mg were observed to counteract this phenomenon. Porosity analysis showed that adding Sr to a nominal AlSiCu3(Fe) alloy increases the porosity nearly three times. Doubling the Mg of this alloy from 0.24 to 0.46 wt% counteracted the Sr effect completely. Subsequently, increasing the Cu content of the alloy with added Mg brought the porosity nearly back where it was with the Sr addition alone.

Copper (Cu)

Aluminum-Copper alloys hold great promise with some of the the highest mechanical properties commercially available in a lightweight aluminum alloy, but they are challenging to cast because they behave in the opposite manner of AlSi alloys. AlCu alloys have a short freezing range which allows near complete feeding of the mushy zone. The challenge arises at the end of solidification when the shrinkage porosity is concentrated in the last areas to solidify. This porosity can be quite large macroporosity. There is so much concentrated volume contraction that the casting can literally pull itself apart in a phenomenon called hot tearing [77], [132]. Sand and semi-permanent mold casting processes use insulated risers to continue to provide feed metal into these areas. HPDC has no such flexibility and, therefore, AlCu alloys are not used in HPDC. This however does not preclude Cu from being added to AlSi alloys like 380, which allow 2-3 wt% Cu. Copper is added to 380-type aluminum alloys to impart strength and hardness in the absence of Mg [130]. In some alloys, Cu is restricted to improve the corrosion resistance of the alloy, particularly in marine applications [133].

The main influence of Cu content in aluminum alloys is its connection to porosity. Numerous studies have shown that microporosity increases with increasing copper content [108], [131], [134]–[136]. The cause of the increased porosity is generally attributed to the formation of a low melting point AlCu eutectic which solidifies after the feed paths for intensification pressure have been frozen off. The resulting volume contraction is observed as microporosity [137].

Iron (Fe)

It has been long understood that Fe is a bad actor negatively impacting the ductility of Al-Si-Mg alloys [138], [139]. The reason is attributed to the morphology of the intermetallic β-phase which it forms in

combination with Mg and Si [139]–[141]. The relatively large plate-like geometry is a microstructural stress concentrator embrittling the alloy. Seifeddine and Svennsson sought to predict mechanical properties of Al-Si alloys based on Fe content. A354 was studied with Fe between 0.35 and 0.65 wt% which is where, or toward the high end of where, structural die casting alloys are specified. They found that elongation is reduced with increasing iron content, however, the levels studied did not impact the yield and ultimate strength of the alloy [141].

Despite this knowledge, Fe concentrations of up to 1.3 and even 2 wt% are allowed in many die casting alloys. The necessity of Fe in 300-series die casting alloys is that it remedies parts being stuck to the die surface. This sticking has been shown to accompany the presence of iron-containing intermetallics. Consequently, the mechanism for sticking and soldering of aluminum parts to the die steel has been approached from the thermodynamics of phase formation [142]–[144]. Aluminum has a high affinity for Fe; a fact commonly observed in foundries where molten aluminum dissolves steel foundry tools such as skimmers and ladles. It has been long accepted that Fe pre-existing in the molten alloy would mitigate the driving force for further reaction between the molten aluminum and the die steel. Current research at Michigan Technological University by Monroe and Sanders challenges this understanding [145]. Nonetheless, Fe does provide a service in preventing parts sticking to the die steel even if the mechanism is under debate.

Manganese (Mn)

The amount of manganese in the alloy in relation to the Fe content can be chosen to promote the intermetallic α-phase. Donahue has shown that there is a critical ratio at which the β-phase formation is suppressed, and the morphologically favorable α-phase is formed [50]. The sweet spot for alloy smelting control is to keep the Mn around 0.35 wt% and the Fe below 0.40 wt%. Li et al. observed that morphology of the iron intermetallics are influenced by Fe and Mn content in gravity die casting as well. Increasing the ratio of Mn (0.01 to 0.51%) to Fe (0.14 to 0.80%) from 0.07 to 0.64 led to an increase in the ratio of alpha to beta intermetallics [146]. Modern structural die-casting alloys, such as A367, take advantage of this relationship to reduce the Fe content and increase the ductility of the alloy.

VI. Prediction Target Selection (Ultimate Tensile Strength)

The published research on the impact of alloy compositions and process parameters is a sizeable collection. Ultimately, the impact is determined by a measurable quantity. Mechanical properties are all-encompassing measures of the microstructural features, solidification discontinuities, and porosity from processing effects. Efforts to determine predictive equations have been performed to either predict mechanical properties from microstructure features and discontinuities or get a better sense of the microstructure and defect population from the mechanical properties. Okayasu et al. evaluated cast microstructures for SDAS, microporosity rating, diameter of eutectic structures, aspect ratio of eutectic structures, and dislocation density and found that, via multiple regression analysis, equations generated can predict UTS accurately [147]. Such work is interesting to predict mechanical properties from microstructural features, but too time consuming for characterizing each casting (i.e. from a coupon). It would be beneficial to the industry to have a similar model for process input parameters, and a method to develop such an equation for each casting.

Surrapa published his work on the effect of macroporosity at the fracture surface on tensile properties in gravity die cast samples. He reported that the projected pore area on the fracture surface was more significant than the bulk porosity calculation one might perform from a density measurement [148]. Caceras numerically modeled this behavior [149]. This work has expanded to apply similar methods to include the effects of microporosity at the fracture surface on the variability of tensile properties [69]–[72], [74], [75]. Unfortunately, no articles were discovered during this literature search applying the analysis to HPDC A380 alloy. An operational challenge is that this practice requires the destruction of a casting to measure the mechanical properties. However, for many foundries quantifying digital X-rays manually can be arduous and, ultimately, subjective [150]–[153]. On a sample basis, measuring the UTS in the area of interest as a quantitative value which can be connected to an indication of porosity via empirically determined formulas would benefit the die casting industry. Only an *indication* of porosity, though, since projected porosity is an area measure and actual porosity is a volumetric quantity. If a predictive model for UTS can be generated based on process input data, the UTS values can be converted to this porosity metric.

VII. Gap in the Literature

In the highly controlled experiments in the literature, it is generally assumed that the process should run as consistently as possible. Several cycles, perhaps 5 to 10, are run to achieve a thermal steady state before collecting samples for investigation. The number of samples collected for analysis tends to be small, less than 50. The industry has gained much from these studies, but there are some potentially significant parameters which cannot be accounted for in lab-scale or development-cell scale operations. In the late 1990's Balasubramaniam applied statistical analysis to 27 casting variables from regular production and found that higher intensification pressure rise time and lower cycle time were key inputs which improved the part density[154]. Interestingly, these parameters do not show up in any of the studies presented thus far. This highlights the need for more research in high-dimensional studies.

Die casting is a thermal process and time is an important factor that is often overlooked or simply held as constant as possible, but rarely measured and reported. The impact of variation in overall cycle time or timers for specific segments of the cycle are not published. Time impacts the die temperature. Running shorter cycle times will put more heat into the die raising the die temperature. But overall cycle time is not the whole picture. Increasing cycle time by increasing the dwell time (the time between casting and part ejection) will also put more heat into the die. Thus, it depends not only on if time is changing but when time is changing.

Variations in the process occur over serial production of shaped castings which have not been investigated. For example, in a small-scale tensile-bar casting study, the furnace level changes very little from the first shot to the last. Conversely, in production casting, the level in the furnace can drop (8 in) before being refilled (*Figure 15*). This can result in variation in the amount of metal delivered by the ladle [155].

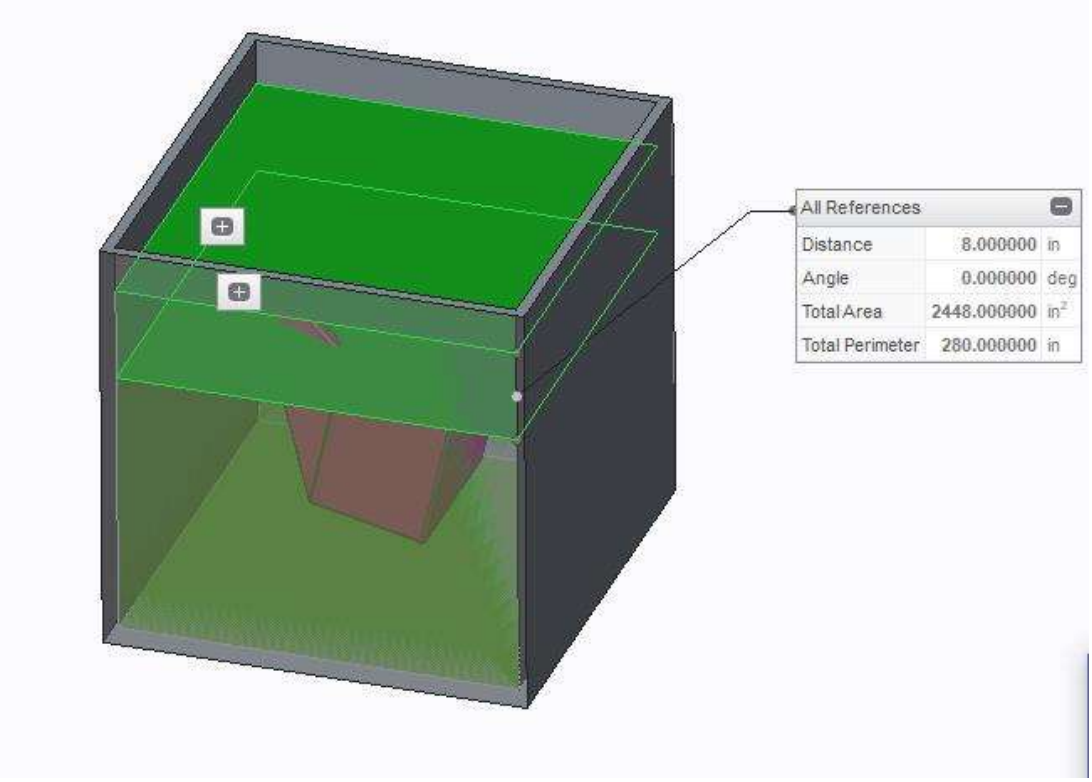


Figure 15. *Variation in furnace level in production operation is not experienced in small-scale experiments.*

Longer term variation effects are less obvious without careful data collection and monitoring. For example, as the equipment wears, its ability to deliver the programmed performance diminishes until the next scheduled maintenance is performed. Perhaps a die spray unit gradually slows down until it is greased. This may result in a longer cycle time which is difficult to diagnose without the proper data. These intricacies of the HPDC process and their effects on part quality require large, production-sized, datasets to understand.

For this research, the process data from one calendar year of engine block production along with the mechanical property quality checks were generously provided by the FCA Kokomo Casting Plant. The raw data is comprised of over 950,000 casting cycles and over 1600 tensile bar test results. Included in the process inputs are many of the process parameters shown to be significant in the literature with respect to tensile properties and porosity including average slow and fast shot velocities, intensification pressure, melt temperature, and cycle time. A complete list can be found in at the end of ***Appendix C – Approach and Methodology***. The goal of this project is to apply machine learning to the datasets and uncover the challenges associated with materials processing datasets. It is important to determine which machine learning tools work well with the type and size of data collected in HPDC operations. These findings will be based on our attempts to predict mechanical properties of extracted tensile bars from engine block castings and learn which process parameters are most important for making accurate predictions.

VIII. References

[1] A. M. Turing, "I.—COMPUTING MACHINERY AND INTELLIGENCE," *Mind*, vol. LIX, no. 236, pp. 433–460, Oct. 1950, doi: 10.1093/mind/LIX.236.433.

[2] W. D. Nordhaus, "Two Centuries of Productivity Growth in Computing," *J. Econ. Hist.*, vol. 67, no. 1, pp. 128–159, Mar. 2007, doi: 10.1017/S0022050707000058.

[3] W. D. Nordhaus, "The Progress of Computing," Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 285168, Sep. 2001. Accessed: Jun. 29, 2020. [Online]. Available: https://papers.ssrn.com/abstract=285168.

[4] J. McCarthy, M. Minsky, N. Rochester, and C. E. Shannon, "A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence." Aug. 31, 1955, Accessed: Feb. 17, 2020. [Online]. Available: https://wvvw.aaai.org/ojs/index.php/aimagazine/article/view/1904.

[5] G. E. Moore, "Cramming More Components Onto Integrated Circuits," *Proc. IEEE*, vol. 86, no. 1, pp. 82–85, Jan. 1998, doi: 10.1109/JPROC.1998.658762.

[6] M. Campbell, A. J. Hoane, and F. Hsu, "Deep Blue," *Artif. Intell.*, vol. 134, no. 1, pp. 57–83, Jan. 2002, doi: 10.1016/S0004-3702(01)00129-1.

[7] F.-H. Hsu, *Behind Deep Blue: Building the Computer that Defeated the World Chess Champion*. Princeton University Press, 2004.

[8] D. A. Ferrucci, "Introduction to 'This is Watson,'" *IBM J. Res. Dev.*, vol. 56, no. 3.4, p. 1:1-1:15, May 2012, doi: 10.1147/JRD.2012.2184356.

[9] "Build Watson | Proceedings of the 19th international conference on Parallel architectures and compilation techniques." https://dl.acm.org/doi/abs/10.1145/1854273.1854275 (accessed Jun. 29, 2020).

[10] G. D. Hutcheson, "Moore's Law: The History and Economics of an Observation that Changed the World," *Electrochem. Soc. Interface*, no. Spring 2005, pp. 17–21, 2005.

[11] G. D. Hutcheson, "The Economic Implications of Moore's Law," in *High Dielectric Constant Materials: VLSI MOSFET Applications*, H. R. Huff and D. C. Gilmer, Eds. Berlin, Heidelberg: Springer, 2005, pp. 1–30.

[12] J. Hruska, "Moore's Law is dead, long live Moore's Law - ExtremeTech." https://www.extremetech.com/extreme/203490-moores-law-is-dead-long-live-moores-law (accessed Feb. 17, 2020).

[13] P. Johnson, "Falling Cost of Memory: 1957 to Present – Let's Talk Data." https://letstalkdata.com/2014/04/falling-cost-of-memory-1957-to-present/ (accessed Feb. 17, 2020).

[14] M. Rieley, "Big data adds up to opportunities in math careers : Beyond the Numbers: U.S. Bureau of Labor Statistics." https://www.bls.gov/opub/btn/volume-7/big-data-adds-up.htm (accessed Feb. 17, 2020).

[15] "Data Science Major | Best 62 Data Science Bachelor's Programs for 2020," *DiscoverDataScience.org*. https://www.discoverdatascience.org/programs/bachelors-in-data-science/ (accessed Feb. 17, 2020).

[16] K.-D. Thoben, S. Wiesner, T. Wuest, BIBA – Bremer Institut für Produktion und Logistik GmbH, the University of Bremen, Faculty of Production Engineering, University of Bremen, Bremen, Germany, and Industrial and Management Systems Engineering, "'Industrie 4.0' and Smart Manufacturing – A Review of Research Issues and Application Examples," *Int. J. Autom. Technol.*, vol. 11, no. 1, pp. 4–16, Jan. 2017, doi: 10.20965/ijat.2017.p0004.

[17] Capgemini Consulting Group, "Industry_4.0_-The_Capgemini_Consulting_V.pdf." Capgemini, 2014, [Online]. Available: https://www.capgemini.com/consulting/wp-content/uploads/sites/30/2017/07/capgemini-consulting-industrie-4.0_0_0.pdf.

[18] "Press Release: Ford Invests in Argo AI, a New Artificial Intelligence Company, in Drive for Autonomous Vehicle Leadership," *businesswire*, Feb. 10, 2017. https://www.businesswire.com/news/home/20170210005537/en/Ford-Invests-Argo-AI-New-Artificial-Intelligence (accessed Jul. 01, 2020).

[19] L. Claussmann and M. Revilloud, “A Study on AI-based Approaches for High-Level Decision Making in Highway Autonomous Driving,” 2017, pp. 3671–3676, doi: 10.1109/SMC.2017.8123203.
[20] S. S. Shadrin, O. O. Varlamov, and A. M. Ivanov, “Experimental Autonomous Road Vehicle with Logical Artificial Intelligence,” *J. Adv. Transp.*, vol. 2017, pp. 1–10, 2017, doi: 10.1155/2017/2492765.
[21] R. A. Greenes, “Clinical Decision Support and Knowledge Management,” in *Key Advances in Clinical Informatics*, Elsevier, 2017, pp. 161–182.
[22] M. S. Bartlett, G. Littlewort, M. Frank, C. Lainscsek, I. Fasel, and J. Movellan, “Fully Automatic Facial Action Recognition in Spontaneous Behavior,” in *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, Southampton, UK, 2006, pp. 223–230, doi: 10.1109/FGR.2006.55.
[23] M. S. Bartlett, G. Littlewort, M. Frank, C. Lainscsek, I. Fasel, and J. Movellan, “Recognizing Facial Expression: Machine Learning and Application to Spontaneous Behavior,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, San Diego, CA, USA, 2005, vol. 2, pp. 568–573, doi: 10.1109/CVPR.2005.297.
[24] “Industry 4.0: the fourth industrial revolution- guide to Industrie 4.0.” https://www.i-scoop.eu/industry-4-0/ (accessed May 26, 2020).
[25] A. S. Vanli, A. Akdogan, K. Kerber, S. Ozbek, and M. N. Durakbasa, “SMART DIE CASTING FOUNDRY ACCORDING TO INDUSTRIAL REVOLUTION 4.0,” *Acta Tech. Napoc.*, vol. 61, no. IV, pp. 787–792, 2018.
[26] R. K. Jain, P. Banerjee, D. Baksi, and S. K. Samanta, “IoT Based Interface Device for Automatic Molding Machine towards SMART FOUNDRY-2020,” in *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, Kanpur, India, Jul. 2019, pp. 1–6, doi: 10.1109/ICCCNT45670.2019.8944549.
[27] J. Friedman, R. Tibshirani, and T. Hastie, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.
[28] D. Dietrich, B. Heller, and B. Yang, *Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data*, 1st ed. Wiley, 2015.
[29] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl, “Constrained K-means Clustering with Background Knowledge,” in *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001, pp. 577–584, Accessed: Jun. 29, 2020. [Online]. Available: https://www.cs.cmu.edu/~./dgovinda/pdf/icml-2001.pdf.
[30] A. Likas, N. Vlassis, and J. J. Verbeek, “The global k-means clustering algorithm,” *Pattern Recognit.*, vol. 36, no. 2, pp. 451–461, Feb. 2003, doi: 10.1016/S0031-3203(02)00060-2.
[31] M. Al-Maolegi and B. Arkok, “An Improved Apriori Algorithm for Association Rules,” *ArXiv14033948 Cs*, Mar. 2014, Accessed: Jun. 29, 2020. [Online]. Available: http://arxiv.org/abs/1403.3948.
[32] Yanbin Ye and Chia-Chu Chiang, “A Parallel Apriori Algorithm for Frequent Itemsets Mining,” in *Fourth International Conference on Software Engineering Research, Management and Applications (SERA’06)*, Aug. 2006, pp. 87–94, doi: 10.1109/SERA.2006.6.
[33] Y. Zhu and Y. Zhang, “The Study on Some Problems of Support Vector Classifier,” *Comput. Eng. Appl.*, no. 13, 2003, [Online]. Available: http://en.cnki.com.cn/Article_en/CJFDTotal-JSGG200313011.htm.
[34] M. Peixeiro, “The Complete Guide to Support Vector Machine (SVM),” *Towards Data Science*, Jul. 29, 2019. https://towardsdatascience.com/the-complete-guide-to-support-vector-machine-svm-f1a820d8af0b.
[35] D. Blondheim, “Unsupervised Machine Learning and Statistical Anomaly Detection Applied to Thermal Images,” *NADCA Trans. T18-071*, 2018, [Online]. Available: http://www.diecasting.org/transactions/T18-071.

[36] L. Deng and D. Yu, “Deep Learning: Methods and Applications,” *Found. Trends Signal Process.*, vol. 7, no. 3–4, pp. 197–387, Jun. 2014, doi: 10.1561/2000000039.
[37] Aporras, “What is the difference between Deep Learning and Machine Learning?,” *QuantDare*, Jan. 08, 2019. quantdare.com/what-is-the-difference-between-deep-learning-and-machine-learning/ (accessed Jun. 02, 2020).
[38] A. Zheng and A. Casari, *Feature Engineering for machine learning: Principles and techniques for data scientists*. Beijing: O-Reilly, 2018.
[39] Y. Jiang *et al.*, “Expert Feature-Engineering vs. Deep Neural Networks: Which Is Better for Sensor-Free Affect Detection?,” in *Artificial Intelligence in Education*, Cham, 2018, pp. 198–211, doi: 10.1007/978-3-319-93843-1_15.
[40] F. Nargesian, H. Samulowitz, U. Khurana, E. B. Khalil, and D. Turaga, “Learning Feature Engineering for Classification,” in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, Melbourne, Australia, Aug. 2017, pp. 2529–2535, doi: 10.24963/ijcai.2017/352.
[41] W. Long, Z. Lu, and L. Cui, “Deep learning-based feature engineering for stock price movement prediction,” *Knowl.-Based Syst.*, vol. 164, pp. 163–173, Jan. 2019, doi: 10.1016/j.knosys.2018.10.034.
[42] W. Butler, *Basic Operator Training Program*. NADCA, 2015.
[43] S. P. Udvardy, “2018 State of the Die Casting Industry,” *Cast. Eng.*, no. January 2019, pp. 10–15, 2019.
[44] A. Spada, “Revitalization of North American Metalcasting,” 2012, Accessed: May 24, 2020. [Online]. Available: https://www.diecasting.org/docs/statistics/North_America.pdf.
[45] J. Folk, “U.S. Aluminum Casting Industry - 2019,” *Cast. Eng.*, no. July 2019, pp. 16–19, Jun. 2019.
[46] D. Twarog, D. Apelian, and A. Luo, *High Integrity Casting of Lightweight Components*, Publication #307. NADCA, 2016.
[47] J. L. Jorstad and D. Apelian, *High Integrity Die Castings (Sound, Reliable and Heat Treatable)*, NADCA Publication #404. Wheeling, IL: NADCA.
[48] S. Wiesner, F. Niklas, and R. Miller, “HP-DC Alloys for Structural Castings: AlMg4Fe2 and AlMg4Zn3Fe2 (Castaduct-42 and -18),” *NADCA Trans. T18-061*, p. 10.
[49] C. Wu, X. Zeng, S. Shankar, G. Birsan, A. Lombardi, and G. Byczynski, “Microstructure and Uniaxial Tensile Properties of Heat Treatable Al-Zn Alloy for Structural HPDC Components,” *NADCA Trans. T18-103*, [Online]. Available: http://www.diecasting.org/transactions/T18-103.
[50] R. J. Donahue and G. K. Sigworth, “Die Casting Alloys that will Allow the Die Caster to Compete with Alloys A356, A357, 358 and 359 in PM Applications,” *NADCA Trans. T16-022*, 2016, [Online]. Available: http://www.diecasting.org/transactions/T16-022.
[51] L. Wang, M. Makhlouf, and D. Apelian, “Aluminium die casting alloys: alloy composition, microstructure, and properties-performance relationships,” *Int. Mater. Rev.*, vol. 40, no. 6, pp. 221–238, Jan. 1995, doi: 10.1179/imr.1995.40.6.221.
[52] D. Blondheim, “Artificial Intelligence, Machine Learning, and Data Analytics: Understanding the Concepts to Find Value in Die Casting Data,” presented at the 2020 NADCA Executive Conference, Clearwater Beach, FL, Feb. 25, 2020.
[53] J. K. Rai, A. M. Lajimi, and P. Xirouchakis, “An intelligent system for predicting HPDC process variables in interactive environment,” *J. Mater. Process. Technol.*, vol. 203, no. 1–3, pp. 72–79, Jul. 2008, doi: 10.1016/j.jmatprotec.2007.10.011.
[54] P. K. D. V. Yarlagadda and E. Cheng Wei Chiang, “A neural network system for the prediction of process parameters in pressure die casting,” *J. Mater. Process. Technol.*, vol. 89–90, pp. 583–590, May 1999, doi: 10.1016/S0924-0136(99)00071-0.
[55] R. Soundararajan, A. Ramesh, S. Sivasankaran, and A. Sathishkumar, “Modeling and Analysis of Mechanical Properties of Aluminium Alloy (A413) Processed through Squeeze Casting Route Using Artificial Neural Network Model and Statistical Technique,” *Adv. Mater. Sci. Eng.*, vol. 2015, pp. 1–16, 2015, doi: 10.1155/2015/714762.

[56] M. Winkler, L. Kallien, and T. Feyertag, “Correlation between Process Parameters and Quality Characteristics in Aluminum High Pressure Die Casting,” *NADCA Trans. T15-022*, 2015, [Online]. Available: http://www.diecasting.org/transactions/T15-022.
[57] D. Klobčar, L. Kosec, B. Kosec, and J. Tušek, “Thermo fatigue cracking of die casting dies,” *Eng. Fail. Anal.*, vol. 20, pp. 43–53, Mar. 2012, doi: 10.1016/j.engfailanal.2011.10.005.
[58] U. C. Nwaogu, N. S. Tiedje, and H. N. Hansen, “A non-contact 3D method to characterize the surface roughness of castings,” *J. Mater. Process. Technol.*, vol. 213, no. 1, pp. 59–68, Jan. 2013, doi: 10.1016/j.jmatprotec.2012.08.008.
[59] G. Dingus, “MECHANICAL PROPERTIES OF CAST ALUMINUM WHEELS,” *AFS Conf. Mech. Prop. Alum. Cast.*, pp. 295–304, 1987.
[60] J. G. Kaufman and E. L. Rooy, *Aluminum Alloy Castings Properties, Processes, and Applications*, 1st ed. ASM, 2004.
[61] G. K. Sigworth and F. DeHart, “Recent Developments in the High Strength Aluminum-Copper Casting Alloy 206,” *AFS Trans.*, pp. 341–354, 2003.
[62] A. I. Taub, P. E. Krajewski, A. A. Luo, and J. N. Owens, “The evolution of technology for materials processing over the last 50 years: The automotive example,” *JOM*, vol. 59, no. 2, pp. 48–57, Feb. 2007, doi: 10.1007/s11837-007-0022-7.
[63] S. Cecchel, G. Cornacchia, A. Panvini, and D. Ferrario, “Analysis and Development of a Safety Relevant Component for Commercial Vehicles,” *NADCA Trans. T17-053*, 2017, [Online]. Available: http://www.diecasting.org/transactions/T17-053.
[64] ASTM International, “ASTM B 557-15, Test Methods for Tension Testing Wrought and Cast Aluminum- and Magnesium-Alloy Products.” ASTM International, doi: 10.1520/B0557-15.
[65] G. K. Sigworth, S. Shivkumar, and D. Apelian, “The Influence of Molten Metal Processing on Mechanical Properties of Cast Al-Si-Mg Alloys,” *Trans. Am. Foundrymens Soc.*, pp. 811–824, 1989.
[66] I. R. McAdams and Q. Han, “Effect of Cooling Conditions on Silafont 36 Phase Formation and Secondary Dendrite Arm Spacing,” *NADCA Trans. T17-062*, 2017, [Online]. Available: http://www.diecasting.org/transactions/T17-062.
[67] D. Schwam, “Additive Manufacturing of Cores with Conformal Cooling Lines,” *NADCA Trans. T16-041*, 2016, [Online]. Available: http://www.diecasting.org/transactions/T16-041.
[68] D. L. Twarog, “State of the Die Casting Industry,” *Cast. Eng.*, no. January, pp. 16–25, 2011.
[69] C. H. Caceres and B. I. Selling, “Casting defects and the tensile properties of an Al-Si-Mg alloy,” *Mater. Sci. Eng. A*, vol. 220, pp. 109–116, 1996, doi: 10.1016/S0921-5093(96)10433-0.
[70] C. D. Lee, “Effects of microporosity on tensile properties of A356 aluminum alloy,” *Mater. Sci. Eng. A*, vol. 464, no. 1–2, pp. 249–254, Aug. 2007, doi: 10.1016/j.msea.2007.01.130.
[71] C. D. Lee and K. S. Shin, “Constitutive prediction of the defect susceptibility of tensile properties to microporosity variation in A356 aluminum alloy,” *Mater. Sci. Eng. A*, vol. 599, pp. 223–232, Apr. 2014, doi: 10.1016/j.msea.2014.01.091.
[72] C. D. Lee, T. I. So, and K. S. Shin, “Effect of geometric array of eutectic silicon particles and microscopic voids on the tensile behaviour of a cast aluminium alloy,” *Mater. Sci. Eng. A*, vol. 599, pp. 28–37, Apr. 2014, doi: 10.1016/j.msea.2014.01.063.
[73] C. Lee, “Effect of Ti-B addition on the variation of microporosity and tensile properties of A356 aluminium alloys,” *Mater. Sci. Eng. A*, vol. 668, pp. 152–159, Jun. 2016, doi: 10.1016/j.msea.2016.05.059.
[74] A. M. Gokhale and G. R. Patel, “Analysis of variability in tensile ductility of a semi-solid metal cast A356 Al-alloy,” *Mater. Sci. Eng. A*, vol. 392, no. 1–2, pp. 184–190, Feb. 2005, doi: 10.1016/j.msea.2004.09.051.
[75] A. M. Gokhale and G. R. Patel, “Quantitative fractographic analysis of variability in tensile ductility of a squeeze cast Al–Si–Mg base alloy,” *Mater. Charact.*, vol. 54, no. 1, pp. 13–20, Jan. 2005, doi: 10.1016/j.matchar.2004.10.003.

[76] J. E. Gruzleski and B. M. Closset, *Treatment of liquid aluminum-silicon alloys*. Des Plaines, IL: American Foundrymen's Society, 1999.
[77] J. Campbell, *Complete Casting Handbook: Metal Casting Processes, Metallurgy, Techniques and Design*. Butterworth-Heinemann, 2015.
[78] G. K. Sigworth and C. Wang, "Mechanisms of porosity formation during solidification: A theoretical analysis," *Metall. Trans. B*, vol. 24, no. 2, pp. 349–364, Apr. 1993, doi: 10.1007/BF02659138.
[79] M. C. Flemings, *Solidification Processing*, 1st ed. McGraw-Hill, 1974.
[80] X. P. Niu, K. K. Tong, B. H. Hu, and I. Pinwill, "Cavity pressure sensor study of the gate freezing behaviour in aluminium high pressure die casting," *Int. J. Cast Met. Res.*, vol. 11, no. 2, pp. 105–112, Sep. 1998, doi: 10.1080/13640461.1998.11819264.
[81] D. Apelian, "How Clean is the Metal You Cast? The Issue of Assessment: A Status Report," in *AFS International Conference on Molten Aluminum Processing*, Nov. 1992, vol. 3, pp. 1–15.
[82] S. DasGupta and D. Apelian, "INTERACTION OF INITIAL MELT CLEANLINESS, CASTING PROCESS AND PRODUCT QUALITY: CLEANLINESS REQUIREMENTS FIT FOR A SPECIFIC USE," in *AFS International Conference on Molten Aluminum Processing*, Nov. 1998, pp. 233; 235–258.
[83] M. Makhlouf, D. Apelian, and L. Wang, "Sludge Formation Tendency of Selected Aluminum Die Casting Alloys," *NADCA Trans. T01-083*, 2001, [Online]. Available: http://www.diecasting.org/archive/transactions/T01-083.pdf.
[84] J. Campbell, "An overview of the effects of bifilms on the structure and properties of cast alloys," *Metall. Mater. Trans. B*, vol. 37, no. 6, pp. 857–863, Dec. 2006, doi: 10.1007/BF02735006.
[85] J. L. Jorstad, "Understanding Sludge," *NADCA Trans. T87-011*, 1987, [Online]. Available: http://www.diecasting.org/archive/transactions/T87-011.pdf.
[86] A. Kopper, "Die Casting Plunger Lubricant Success Story: T6 Heat Treatable Die Castings," *NADCA Trans. T09-023*, 2009, [Online]. Available: http://www.diecasting.org/transactions/T09-023.
[87] Y. F. He, X. J. Xu, F. Zhang, D. Q. Li, S. P. Midson, and Q. Zhu, "Impact of Die and Plunger Lubricants on Blistering during T6 Heat Treatment of Semi-Solid Castings," *NADCA Trans. T13-012*, 2013, [Online]. Available: http://www.diecasting.org/transactions/T13-012.
[88] Q. Han, F. Yin, M. Rakita, J. Zhang, K. Blowers, and C. Vian, "Comparison of Lube Applications: Continuous and Pulse Spray," *NADCA Trans. T19-063*, 2019, [Online]. Available: http://www.diecasting.org/transactions/T19-063.
[89] M. C. Thome, J. R. Brevick, and Y.-L. Chu, "Modeling the Effect of Shot Plunger Acceleration on Wave Formation and Air Entrapment in Cold Chamber Die Casting (A Progress Report)," The Ohio State University, ERC-94-04, Dec. 1993. Accessed: May 25, 2020. [Online]. Available: https://www.diecasting.org/archive/erc/ERC-94-04.pdf.
[90] L. Garber and A. B. Draper, "The Effects of Process Variables on the Internal Quality of Aluminum Die Castings," *NADCA Trans. T79-022*, 1979, [Online]. Available: http://www.diecasting.org/archive/transactions/T79-022.
[91] Magna Cosma, "Cosma International Aluminum High Pressure Die Casting." Cosma International, 2014, Accessed: May 24, 2020. [Online]. Available: https://www.magna.com/docs/default-source/default-document-library/cosma_casting_brochure_english.pdf?sfvrsn=0.
[92] Shiloh Industries, "Castlight," *Castlight*, May 25, 2020. Shiloh.com/solutions/castlight.
[93] J. R. Brevick, I. Ziv, S.-H. (Patrick) Cheng, Y.-L. Chu, T. Altan, and B.-S. Chun, "Vacuum Assisted Die Casting: Casting Properties and Contained Gas Contents," The Ohio State University, ERC-94-52, Oct. 1994. Accessed: May 25, 2020. [Online]. Available: https://www.diecasting.org/archive/erc/ERC-94-52.pdf.
[94] K. A. Wyman, J. Brevik, Y.-L. Chu, and T. Altan, "A Preliminary Investigation into the Effects of Vacuum Assist in Cold Chamber Die Casting," The Ohio State University, ERC-92-04, Jan. 1992.

Accessed: May 25, 2020. [Online]. Available: https://www.diecasting.org/archive/erc/ERC-92-04.pdf.
[95] H. Cao, M. Hao, C. Shen, and P. Liang, "The influence of different vacuum degree on the porosity and mechanical properties of aluminum die casting," *Vacuum*, vol. 146, pp. 278–281, Dec. 2017, doi: 10.1016/j.vacuum.2017.09.048.
[96] C. Huang and W. Bishenden, "Venting Design and Process Optimization of Die Casting Process for Structural Components," *NADCA Trans. T14-042*, 2014, [Online]. Available: http://www.diecasting.org/transactions/T14-042.
[97] G. O. Verran, R. P. K. Mendes, and M. A. Rossi, "Influence of injection parameters on defects formation in die casting Al12Si1,3Cu alloy: Experimental results and numeric simulation," *J. Mater. Process. Technol.*, vol. 179, no. 1–3, pp. 190–195, Oct. 2006, doi: 10.1016/j.jmatprotec.2006.03.089.
[98] R. A. Miller, "Gate Speed, Fraction Solid, and the Effect on Mechanical Properties," *NADCA Trans. T18-042*, 2018, [Online]. Available: http://www.diecasting.org/transactions/T18-042.
[99] R. A. Miller, "The 'Gating Equation' Updated," *NADCA Trans. T15-092*, 2015, [Online]. Available: http://www.diecasting.org/transactions/T15-092.
[100] C. W. Kim and K. D. Siersma, "Shot profile optimization by moving Finite Element Method and Simulated Annealing," *NADCA Trans. T08-111*, p. 5.
[101] D. Gaddam and M. Gondek, "Next Generation Approach to Designing the Optimal High Pressure Die Casting Tooling and Process," *NADCA Trans. T15-093*, 2015, [Online]. Available: http://www.diecasting.org/transactions/T15-093.
[102] D. Gaddam, "Autonomous Optimization of Die Casting Processes," *AFS Trans.*, vol. 124, pp. 25–32, 2016.
[103] M. S. Dargusch, G. Dour, N. Schauer, C. M. Dinnis, and G. Savage, "The influence of pressure during solidification of high pressure die cast aluminium telecommunications components," *J. Mater. Process. Technol.*, vol. 180, no. 1–3, pp. 37–43, Dec. 2006, doi: 10.1016/j.jmatprotec.2006.05.001.
[104] M. Okayasu, S. Yoshifuji, M. Mizuno, M. Hitomi, and H. Yamazaki, "Comparison of mechanical properties of die cast aluminium alloys: cold *v* . hot chamber die casting and high *v* . low speed filling die casting," *Int. J. Cast Met. Res.*, vol. 22, no. 5, pp. 374–381, Oct. 2009, doi: 10.1179/174313309X380413.
[105] R. N. Lumley, R. G. O'Donnell, D. R. Gunasegaram, and M. Givord, "Heat Treatment of High-Pressure Die Castings," *Metall. Mater. Trans. A*, vol. 38, no. 10, pp. 2564–2574, Sep. 2007, doi: 10.1007/s11661-007-9285-4.
[106] D. R. Gunasegaram, M. Givord, R. G. O'Donnell, and B. R. Finnin, "Improvements engineered in UTS and elongation of aluminum alloy high pressure die castings through the alteration of runner geometry and plunger velocity," *Mater. Sci. Eng. A*, vol. 559, pp. 276–286, Jan. 2013, doi: 10.1016/j.msea.2012.08.098.
[107] S. L. dos Santos, R. A. Antunes, and S. F. Santos, "Influence of injection temperature and pressure on the microstructure, mechanical and corrosion properties of a AlSiCu alloy processed by HPDC," *Mater. Des.*, vol. 88, pp. 1071–1081, Dec. 2015, doi: 10.1016/j.matdes.2015.09.095.
[108] I. Outmani, L. Fouilland-Paille, J. Isselin, and M. El Mansori, "Effect of Si, Cu and processing parameters on Al-Si-Cu HPDC castings," *J. Mater. Process. Technol.*, vol. 249, pp. 559–569, Nov. 2017, doi: 10.1016/j.jmatprotec.2017.06.043.
[109] E. Fiorese and F. Bonollo, "Simultaneous Effect of Plunger Motion Profile, Pressure, and Temperature on the Quality of High-Pressure Die-Cast Aluminum Alloys," *Metall. Mater. Trans. A*, vol. 47, no. 12, pp. 6453–6465, Dec. 2016, doi: 10.1007/s11661-016-3732-z.
[110] B. M. Asquith, "The Use of Process Monitoring to Minimize Scrap in the Die Casting Process," *NADCA Trans. T97-063*, 1997, Accessed: May 25, 2020. [Online]. Available: http://www.diecasting.org/archive/transactions/T97-063.pdf.
[111] A. Kaye and A. C. Street, *Die Casting Metallurgy*. Butterworth Scientific, 1982.

[112] R. K. Kuppili and P. Praveen, "Design and Analysis of High Pressure Die Casting Die for Gear Box Cover," *Int. J. Sci. Eng. Res.*, vol. 7, no. 7, pp. 86–89, 2016.
[113] L. J. Yang, "The effect of casting temperature on the properties of squeeze cast aluminium and zinc alloys," *J. Mater. Process. Technol.*, vol. 140, no. 1–3, pp. 391–396, Sep. 2003, doi: 10.1016/S0924-0136(03)00763-5.
[114] J.-I. Cho and C.-W. Kim, "The Relationship between Dendrite Arm Spacing and Cooling Rate of Al-Si Casting Alloys in High Pressure Die Casting," *Int. J. Met.*, vol. 8, no. 1, pp. 49–55, Jan. 2014, doi: 10.1007/BF03355571.
[115] M. Okayasu and S. Yoshida, "Influence of solidification rate on material properties of cast aluminium alloys based on Al–Si–Cu and Al–Si–Mg," *Int. J. Cast Met. Res.*, vol. 28, no. 2, pp. 105–116, Apr. 2015, doi: 10.1179/1743133614Y.0000000128.
[116] E. Sjölander and S. Seifeddine, "Influence of alloy composition, solidification rate and artificial aging on plastic deformation behaviour of Al–Si–Cu–Mg casting alloys," *Int. J. Cast Met. Res.*, vol. 26, no. 1, pp. 28–36, Feb. 2013, doi: 10.1179/1743133612Y.0000000025.
[117] M. A. Irfan, D. Schwam, A. Karve, and R. Ryder, "Porosity reduction and mechanical properties improvement in die cast engine blocks," *Mater. Sci. Eng. A*, vol. 535, pp. 108–114, Feb. 2012, doi: 10.1016/j.msea.2011.12.049.
[118] The Aluminum Association, *Designations and Chemical Composition Limits for Aluminum Alloys in the Form of Castings and Ingot*, October 2018. Arlington, VA: The Aluminum Association, 2018.
[119] ASTM International, "E1251-17A, Standard Test Method for Analysis of Aluminum and Aluminum Alloys by Spark Atomic Emission Spectrometry." ASTM International, 2017, [Online]. Available: http://www.astm.org/cgi-bin/resolver.cgi?E1251-17a.
[120] S. W. Hudson, J. Craparo, R. De Saro, and D. Apelian, "Applications of Laser-Induced Breakdown Spectroscopy (LIBS) in Molten Metal Processing," *Metall. Mater. Trans. B*, vol. 48, no. 5, pp. 2731–2742, Oct. 2017, doi: 10.1007/s11663-017-1032-7.
[121] J. L. Jorstad and D. Apelian, "Hypereutectic Al-Si Alloys: Practical Processing Techniques," *Cast. Eng.*, no. May 2004, pp. 50–55, 2004.
[122] L. Wang, D. Apelian, and M. Makhlouf, "Development of High Performance Die Casting Alloys Part 1: Alloy Design," *NADCA Trans. T11-021*, 2011, [Online]. Available: https://www.diecasting.org/archive/transactions/T11-021.pdf.
[123] Y. C. Kim, S. W. Choi, C. W. Kim, J. I. Cho, and C. S. Kang, "Influence of Process Parameters on the Fluidity of High Pressure Die-Casting Al-Si Alloys," *Adv. Mater. Res.*, vol. 813, pp. 171–174, 2013, doi: https://doi.org/10.4028/www.scientific.net/amr.813.171.
[124] L. F. Mondolfo, *Aluminum Alloys: Structure & Properties*. Butterworth-Heinemann, 1976.
[125] M. Tsukuda, M. Harada, T. Suzuki, and S. Koike, "The effect of Si, Mg, Fe on the mechanical properties of Al-Si-Mg alloys for casting," *J. Jpn. Inst. Light Met.*, vol. 28, no. 3, pp. 109–115, Aug. 1977, doi: https://doi.org/10.2464/jilm.28.109.
[126] J. G. Gensure and D. L. Potts, *International Metallic Materials Cross-Reference*, 3rd ed. Schenectady, NY: Genium Publishing Corporation, 1988.
[127] R. N. Lumley, I. J. Polmear, and P. R. Curtis, "Rapid Heat Treatment of Aluminum High-Pressure Diecastings," *Metall. Mater. Trans. A*, vol. 40, no. 7, pp. 1716–1726, Jul. 2009, doi: 10.1007/s11661-009-9836-y.
[128] H. Yang, S. Ji, W. Yang, Y. Wang, and Z. Fan, "Effect of Mg level on the microstructure and mechanical properties of die-cast Al–Si–Cu alloys," *Mater. Sci. Eng. A*, vol. 642, pp. 340–350, Aug. 2015, doi: 10.1016/j.msea.2015.07.008.
[129] J. Y. Hwang, R. Banerjee, H. W. Doty, and M. J. Kaufman, "The effect of Mg on the structure and properties of Type 319 aluminum casting alloys," *Acta Mater.*, vol. 57, no. 4, pp. 1308–1317, Feb. 2009, doi: 10.1016/j.actamat.2008.11.021.
[130] Y. Zeden, A. M. Samuel, F. H. Samuel, and S. Alkahtani, "Effects of Cu, Mg, and Sr, on the Mechanical Properties and Machinability of Near-Eutectic Al-11%Si Casting Alloys," *Light Met.*

2012 Ed. Carlos E Suarez TMS Miner. Met. Mater. Soc., pp. 321–326, 2012, doi: https://doi.org/10.1007/978-3-319-48179-1_54.
[131] A. Fabrizi, S. Ferraro, and G. Timelli, "The influence of Sr, Mg and Cu addition on the microstructural properties of a secondary AlSi9Cu3(Fe) die casting alloy," *Mater. Charact.*, vol. 85, pp. 13–25, Nov. 2013, doi: 10.1016/j.matchar.2013.08.012.
[132] J. R. Brevick and S. N. Dubey, "OVERVIEW OF RECENT RESEARCH REGARDING HOT TEARING OF DIE CASTING ALLOYS," *NADCA Trans. T12-013*, 2012, [Online]. Available: http://www.diecasting.org/transactions/T12-013.
[133] J. L. Jorstad, D. L. Zalensas, and American Foundrymen's Society, Eds., *Aluminum casting technology*, 2. ed., reprint. Des Plaines, Ill: American Foundrymen's Society, 2001.
[134] C. H. Cáceres, M. B. Djurdjevic, T. J. Stockwell, and J. H. Sokolowski, "The effect of Cu content on the level of microporosity in Al-Si-Cu-Mg casting alloys," *Scr. Mater.*, vol. 40, no. 5, pp. 631–637, Feb. 1999, doi: 10.1016/S1359-6462(98)00492-8.
[135] F. Sanna, A. Fabrizi, S. Ferraro, G. Timelli, P. Ferro, and F. Bonollo, "Multiscale characterisation of AlSi9Cu3(Fe) die casting alloys after Cu, Mg, Zn and Sr addition," *Metall. Ital.*, vol. 105, no. 4, pp. 13–24, Apr. 2013.
[136] S. G. Shabestari and H. Moemeni, "Effect of copper and solidification conditions on the microstructure and mechanical properties of Al–Si–Mg alloys," *J. Mater. Process. Technol.*, vol. 153–154, pp. 193–198, Nov. 2004, doi: 10.1016/j.jmatprotec.2004.04.302.
[137] Z. Li, A. M. Samuel, F. H. Samuel, C. Ravindran, and S. Valtierra, "Effect of alloying elements on the segregation and dissolution of CuAl2 phase in Al-Si-Cu 319 alloys," *J. Mater. Sci.*, vol. 38, no. March 2003, pp. 1203–1218, 2003, doi: https://doi.org/10.1023/A:1022857703995.
[138] W. Bonsack, "Iron problematic factor in quality of alumimnum alloy die castings," *Trans. Am. Foundrymens Soc.*, pp. 712–720, 1961.
[139] P. N. Crepeau, "Effect of Iron in Al-Si Casting Alloys: A Critical Review," *Trans. Am. Foundrymens Soc.*, pp. 361–366, 1995.
[140] M. A. Moustafa, "Effect of iron content on the formation of β-Al5FeSi and porosity in Al–Si eutectic alloys," *J. Mater. Process. Technol.*, vol. 209, no. 1, pp. 605–610, Jan. 2009, doi: 10.1016/j.jmatprotec.2008.02.073.
[141] S. Seifeddine and I. L. Svensson, "Prediction of mechanical properties of cast aluminium components at various iron contents," *Mater. Des.*, vol. 31, pp. S6–S12, Jun. 2010, doi: 10.1016/j.matdes.2009.11.023.
[142] S. Shankar and D. Apelian, "Die soldering: Effect of process parameters and alloy characteristics on soldering in the pressure die casting process," *Int. J. Cast Met. Res.*, vol. 15, no. 2, pp. 103–116, Sep. 2002, doi: 10.1080/13640461.2002.11819469.
[143] S. Shankar and D. Apelian, "Die soldering: Mechanism of the interface reaction between molten aluminum alloy and tool steel," *Metall. Mater. Trans. B*, vol. 33, no. 3, pp. 465–476, Jun. 2002, doi: 10.1007/s11663-002-0057-7.
[144] J. Song, X. Wang, T. DenOuden, and Q. Han, "Evolution of Intermetallic Phases in Soldering of the Die Casting of Aluminum Alloys," *Metall. Mater. Trans. A*, vol. 47, no. 6, pp. 2609–2615, Jun. 2016, doi: 10.1007/s11661-016-3454-2.
[145] A. K. Monroe and P. G. Sanders, "Reducing Die Lubricant by Understanding the Importance of Draft Angle and Casting Alloy," *NADCA Trans. T19-083*, 2019, [Online]. Available: https://www.diecasting.org/archive/transactions/T19-083.pdf.
[146] Z. Li, N. Limodin, A. Tandjaoui, P. Quaegebeur, P. Osmond, and D. Balloy, "Influence of Sr, Fe and Mn content and casting process on the microstructures and mechanical properties of AlSi7Cu3 alloy," *Mater. Sci. Eng. A*, vol. 689, pp. 286–297, Mar. 2017, doi: 10.1016/j.msea.2017.02.041.
[147] M. Okayasu, K. Ota, S. Takeuchi, H. Ohfuji, and T. Shiraishi, "Influence of microstructural characteristics on mechanical properties of ADC12 aluminum alloy," *Mater. Sci. Eng. A*, vol. 592, pp. 189–200, Jan. 2014, doi: 10.1016/j.msea.2013.10.098.

[148] M. K. Surappa, E. Blank, and J. C. Jaquet, “EFFECT OF MACRO-POROSITY ON THE STRENGTH AND DUCTILITY OF CAST,” *Scr. Metall.*, vol. 20, no. 9, pp. 1281–1286, 1986, doi: 10.1016/0036-9748(86)90049-9.
[149] C. H. Cáceres, “On the effect of macroporosity on the tensile properties of the Al-7%Si-0.4%Mg casting alloy,” *Scr. Metall. Mater.*, vol. 32, no. 11, pp. 1851–1856, Jun. 1995, doi: 10.1016/0956-716X(95)00031-P.
[150] ASTM International, “E 505 - 01, Standard Reference Radiographs for Inspection of Aluminum and Magnesium Die Castings.” ASTM International, 2001.
[151] H. Boerner and H. Strecker, “Automated x-ray inspection of aluminum castings,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 1, pp. 79–91, 1988, doi: 10.1109/34.3869.
[152] A. P. Rale, D. C. Gharpure, and V. R. Ravindran, “Comparison of different ANN techniques for automatic defect detection in X-Ray images,” in *2009 International Conference on Emerging Trends in Electronic and Photonic Devices & Systems*, Dec. 2009, pp. 193–197, doi: 10.1109/ELECTRO.2009.5441138.
[153] L. A. Dobrzański, M. Krupiński, and J. H. Sokolowski, “Methodology of automatic quality control of aluminium castings,” *J. Achiev. Mater. Manuf. Eng.*, vol. 20, no. 1–2, pp. 69–78, 2007.
[154] S. Balasubramaniam and R. Shivpuri, “Improving the Quality in Die Casting Production Using Statistical Analysis Procedures,” *NADCA Trans. T99-071*, 1999, [Online]. Available: http://www.diecasting.org/transactions/T199-071.
[155] A. K. Monroe, “Personal Communication.”

Appendix B – Machine Learning Pathway for Harnessing Knowledge and Data in Material Processing

Ning Sun[1], Adam Kopper[2], Rasika Karkare[3], Randy C. Paffenroth[4], and Diran Apelian[5]

[1] Metso, Shrewsbury, MA 01545 USA
[2]Mercury Marine, Fond du Lac, WI 54935 USA
[3] WPI, Data Science, Worcester, MA 01609 USA
[4] WPI, Mathematical Sciences, Computer Science, Data Science, Worcester, MA 01609 USA
[5] UCI, Materials Science and Engineering, Irvine, CA 92967 USA

Keywords. Industry 4.0, machine learning, smart factory, IoT, artificial intelligence, classification models, random forest, XGBoost, unbalanced, semi-supervised, dimension reduction, principal component analysis, feature importance, data standardization

Abstract

Artificial Intelligence (AI) is integral to Industry 4.0 and the evolution of Smart Factories. To realize this future, material processing industries are embarking on adopting AI technologies into their enterprise and plants; however, like all new technologies, there is always the potential for misuse or the false belief that the outcomes are reliable. The goal of this paper is to provide context for the application of machine learning to materials processing. The general landscapes of data science and materials processing are presented, using the foundry and the metal casting industry as an exemplar. The challenges that exist with typical foundry data are that the data are unbalanced, semi-supervised, heterogeneous, and limited in sample size. Data science methods to address these issues are presented and discussed. The elements of a data science project are outlined and illustrated by a case study using sand cast foundry data. Finally, a prospective view of the application of data science to materials processing and the impact this will have in the field are given.

I. Introduction

The fourth industrial revolution that ushered the Internet of Things (IoT) and the Internet of Services (IoS) has come to be known as Industry 4.0. At the Hannover Messe in 2011, Germany launched a project called *"Industrie 4.0"* designed to fully digitize manufacturing. The larger vision of Industry 4.0 is the digital transformation of manufacturing, leveraging advanced technologies and innovation accelerators in the convergence of IT (Information Technology) and OT (Operational Technology). The purpose is to integrate connected factories within industry, decentralized and self-optimizing systems and the digital supply chain in the information-driven cyber-physical environment of the fourth industrial revolution [1], [2]. The evolution toward Industry 4.0 is given in Figure 1.

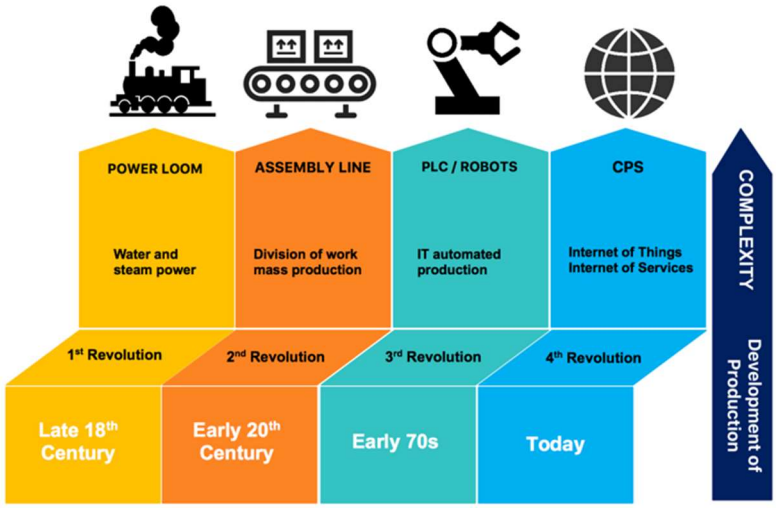


Figure 1. *Industrial Revolutions* [1].

The initial goals of Industry 4.0 typically have been automation, manufacturing process improvement and productivity optimization. The more advanced goals are innovation and the transition to new business models and revenue sources using information technologies and services as cornerstones. These developments will transform manufacturing plants into smart factories or foundries. Three keystone digital technologies will enable the transformation to smart factories: (i) connectivity, which implies executing industrial IoT to collect data from various segments of the plant; (ii) intelligent automation which includes advanced robotics, machine vision, digital twins, distributed control; and (iii) cloud-scale data management and analytics (AI and Machine Learning) [3].

In the metal processing field, particularly in the metal casting industry, whether it be ferrous or non-ferrous foundries, many data are collected at various locations within the plant. However, these data are usually siloed within operational departments without an intentional strategy for data fusion and transformation into knowledge. It is a fact that many of our plants and plant infrastructures were built prior to the rise of data science capabilities and tools. The time is now to make the transformation of our plants into smart factories in the context of Industry 4.0.

In this paper, our goal is to establish some context of AI and machine learning and how it can be appropriately utilized in materials processing where physical laws govern the process. We use metal casting as an example in this work as it is a well-established industry from which we have access to process data via the industrial membership of the Advanced Casting Research Center at UCI. In metal casting, the quality of the final product is influenced by many factors: metal composition, processing conditions, the solidification journey where transport phenomena influence the resultant microstructure, post processing treatments, etc. Even with our understanding of the materials processing world, working with its manufacturing data is not without challenges [4]. Because the industry is well-established, the foundries do not produce components with many defects. In other words, scrap rates are low, making it difficult to utilize algorithms, based on supervised learning**,** which learn from successes and failures [5]. This is where the need for using machine learning algorithms that can treat unbalanced data arises.

Moreover, real-world manufacturing processes are complex and appropriate data may not always be available for all parts. Accordingly, the need for advanced unsupervised or semi-supervised machine learning algorithms also exists [5]. Section II describes these types of algorithms in detail. In this work, we want to show how these techniques can be used to answer the questions: *How can we develop algorithms and apply AI/Machine Learning to processes where one does not have many defective, or otherwise labeled, parts to teach and learn from?* It should be noted that the fundamentals and the principles presented here are applicable to a host of manufacturing processes.

II. The Landscape of Machine Learning

What is Machine Learning?

Machine learning is a branch of Artificial Intelligence (AI) where one constructs computer algorithms intended to mimic tasks commonly performed by humans. Algorithms for image recognition, health analytics, natural language processing, and self-driving vehicles are all examples of AI that have transformed industries that affect our daily lives [6]. More specifically, AI clearly has a role to play in advanced manufacturing where there are myriad of tasks that could be automated by algorithms such as defect detection, process optimization, and new materials development, to name but a few [2].

For many years, classical philosophers have attempted to describe human thinking as a symbolic system. Babbage in the 1830's realized that punched cards used in the Jacquard loom could control operations [6]. Alan Turing in England (1935-1940 era) developed a machine that could compute using a set of rules transitions/states to solve mathematical functions [7], [8]. Subsequently, Turing went on to expand his view by posing the question: "*Can a machine think*"? The Term AI was formally established in 1956 at a conference at Dartmouth College, Hanover, NH USA by pioneers John McCarthy and Marvin Minsky. McCarthy challenged the community to make machines that *"behave in ways that would be called intelligent if a human were so behaving"*; whereas Minsky focused on making machines that would do things that *"would require intelligence if done by men"* [9].

AI and machine learning are closely related to fields such as pattern recognition (an umbrella term that covers many different approaches), statistics and statistical learning (where the focus tends to be on formal mathematical relationships), and neural networks (a field which has seen great advancements in the past few years) [10]. For example, one class of approaches that was common in AI's early years was that of rules-based systems**.** In a manufacturing plant, the convention has been that engineers develop a knowhow enabling them to detect defects in the final product; in turn, they pass on this knowledge to those who follow their footsteps. It is tempting to distill how a human performs such tasks by enumerating a set of rules for defect recognition. Once such a collection of rules is developed, they can then be encoded in a computer language to allow a machine to mimic what a human does. Unfortunately, such rules-based systems tend to be quite fragile as the interactions in the system can be subtle. As a result, rules-based systems do not achieve human level performance. The field of machine learning takes a different perspective by developing algorithms that learn by example. Rather than constructing hand-crafted rules, in machine learning, one designs systems that can construct their own rules given a collection of examples where the desired task is performed correctly.

Elements of Machine Learning

Algorithms

Algorithms are the "machines" that can learn and generate the rules. It is reasonable to ask whether it is easier to write down explicit rules for a task or to create an algorithm that can generate its own rules. Perhaps counterintuitive, the latter is often much easier, more effective, and less error prone. Rulemaking algorithms abound, from simple linear regression, to the more complicated support vector machines, to cutting edge neural networks [5], [11], [12]. As expected, algorithms are imperfect if the training data is inadequate. In machine learning, and specially in semi-supervised learning mode, one requires a large set of training data; without this the algorithms developed may be unreliable.

Training data

For algorithms to be effective, they require many examples to learn from. The question is: *Given the amount of training data, which algorithm is the most suitable?* The choice of algorithms is dependent upon the size of training data available. Using techniques such as cross-validation, we can test the performance of different algorithms in terms of generalizing on unseen test data [13]. This can be done by training the algorithms on different subsets of the data and then testing on the rest. Choosing between algorithms should be based on comparing their performance on the test set.

Feature Engineering

Feature engineering is the process of using domain knowledge of the data to create features that optimize machine learning algorithms [14], [15]. This is where technical prowess provided by the practitioner or the engineer plays an important role. Feature engineering increases the predictive power of the algorithms by selecting specific features or creating new features from the data that assist in the learning process.

Feature engineering determines what information is given as input to the machine learning algorithm. The danger, however, is that one may carry this out and over-engineer, in the engineering parlance, and over-fit, in the machine learning parlance; there is a sweet spot for feature engineering. An example may be useful to explain the concept. Many parameters are collected during metal casting: alloy composition, environmental conditions in the foundry, superheat, temperature changes during the solidification process, etc. It is not unusual to have 40 columns of data for a given cast part. *Feature engineering helps us address how these parameters are communicated to the algorithm.* A close collaboration is needed in generating appropriate training datasets and the appropriate feature engineering by experts in both manufacturing and machine learning. To a large extent the authors of this paper have formed such a team.

Data Pre-Processing

Many machine learning algorithms require that their input data be numeric. In the example above, how should the chemical composition be represented numerically for a fair comparison with melt temperature and foundry environmental conditions? In the original training data, the amount of Si is 0.07 weight fraction, the melt temperature is 704 ºC, and the temperature of the foundry is 24°C. *Many machine learning algorithms depend on an appropriate definition of distance, and the rules they generate hinge on the distances between the training examples*. By setting Si=0.07, T_{melt}=704, and T_{floor}=24, one is implicitly informing the algorithm that melt temperature is a more important

parameter as compared to the composition of silicon or the temperature of the foundry. However, such inferences may be neither intended nor correct. In order to avoid such inferences, we pre-process the data such as normalizing the dataset so that all the columns are on the same scale [16]. Details of how we normalize using a Z-transform are given in section IV of this paper [17].

Cross-Validation

One important part of machine learning that we have not yet touched upon is the evaluation of the performance of the algorithms we construct. Cross-validation is a technique that is used for algorithm evaluation on unseen data. *Cross-validation can be thought of as testing the algorithm in an environment that is faithful to how it will be used during the manufacturing process.* For example, given a set of training data (e.g., labeled X-ray images of parts), one can train the algorithm in the task of detecting defective parts. When the algorithm is utilized on the factory floor, one would be interested in knowing how well it performs on images of parts as they roll off the assembly line, when the true label is not yet known.

There are important differences between how a machine learning algorithm performs on its training data, and how it might perform in practice on the factory floor. For example, consider a machine learning algorithm that merely memorizes all the X-ray images in its training data and whether the image corresponds to a good part. Such an algorithm would be able to perfectly label every image in its training set but would have no ability to correctly label new images. In data science terms, it would not be able to generalize from its training data to new examples, such as the current production parts shipping from the foundry.

The machine learning terminology for such an algorithm that performs well on training data but fails to generalize is known as overfitting [18]. Avoiding overfitting is an essential part of machine learning and a place where the expertise of machine learning practitioners can play a pivotal role. Constructing a machine learning algorithm that appears to be quite effective during training but fails in the field can be surprisingly easy to do. However, such situations are clearly to be avoided and require a measure of machine learning expertise.

III. The Landscape of Materials Processing – *Metal Casting*

Machine learning promises to have a transformative impact on the advanced manufacturing landscape, where applications of machine learning alongside the IoT is projected to generate $1.2 to $3.7 trillion of value globally by 2025 [1]. In the metal casting industry, which is one of the core building blocks of advanced manufacturing industries, machine learning has only seen negligible adoption to date. Thus, there exists a huge opportunity to utilize AI and machine learning in the metal casting industry [19]–[24]. The metal casting industry is at the cusp of its data revolution.

Modern foundries have the capability to capture a vast amount of process data on a daily basis [25]. These include molten metal preparation details, casting process data, simulation data, part geometry data (CAD files), Non-Destructive Evaluation/Testing data, etc. However, these many types of data from various sources throughout the operation are often kept in departmental silos where their value might have limited utility (Figure 2). Integrated data is the prerequisite for performing machine learning, and it is a lost opportunity for the foundry industry if no effort is

made to compile, fuse, and analyze these data to better understand the process factors influencing the quality of the castings.

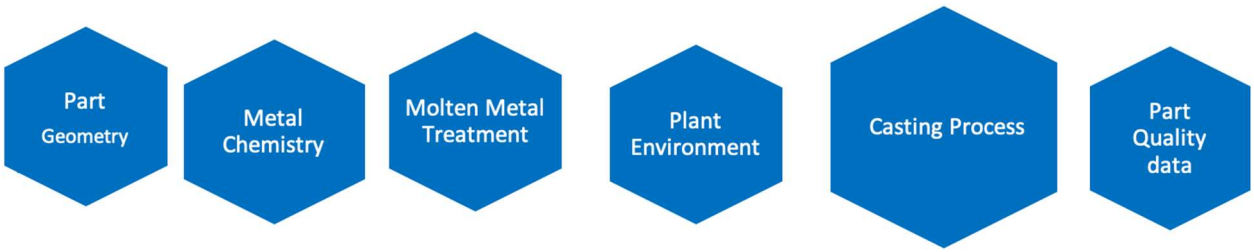


Figure 2. *Data from various sources throughout the casting operation are kept in silos*.

Implementation of machine learning in the metal casting industry requires *knowledge workers* who are trained in both data science and materials science and engineering domains [26]. Unfortunately, most engineers are not trained in data science. Efforts are underway in academia (e.g., WPI, UCI, University at Buffalo, Northwestern U., U. of Wisconsin, etc.) to develop curricula for engineering students who can navigate in both domains.

At the Advanced Casting Research Center (ACRC), a consortium consisting of 35 corporations has made a commitment to study how machine learning and deep learning can yield transformative improvements to metal casting processes. The long-term goal is to develop a framework that can be adopted by foundries to transform their data into process cognition and knowledge. In this project, the research team is multidisciplinary comprising of faculty and graduate students from Data Science as well as Materials Science and Engineering. The data scientists apply their expertise in seeking or developing the effective and appropriate data analysis techniques. Material scientists and engineers determine how to treat anomalous data points in the raw dataset and can assess whether the predicted results and the feature importance are in-line with observations on the shop floor.

IV. Process Cognition and Harnessing of Knowledge in Metal Casting

In the following sections, we review some technical challenges and pitfalls in applying machine learning to industrial foundry data and cover some potential solutions to resolve these challenges. Subsequently, we navigate the critical steps of machine learning as applied in a case study to showcase the implementation of machine learning to metal casting step-by-step.

Challenges of Metal Casting Datasets

Our team is in a fortunate position to have access to cast data from the industrial partners of ACRC. All the data are treated confidentially and are collected from three different casting processes – die casting, permanent mold, and sand casting. Though knowledge extracted directly from these front-line datasets can provide meaningful guidance on process and quality control to foundries, the process of converting these data into knowledge is quite challenging to our data scientists due to three notable attributes of foundry data as discussed below.

The data are unbalanced

In machine learning, the algorithm is designed to construct its own rules given a collection of examples. The aim is to develop a machine learning model that can predict the quality of cast

components. The algorithm is trained by providing it with a large set of processing data, with each or some of the parts being labeled. The algorithm can construct its own set of rules for distinguishing between the labels. Based upon these examples, the algorithm applies the rules to make predictions on parts whose label is unknown. Ideally, the algorithm would learn from an approximately equal number of examples representing each label, however, in reality, the labels are unbalanced. The lifeblood of successful foundries is large-scale production of defect-free products. Accordingly, only a small percentage of defective products are available to train the machine learning algorithm. For example, in metal casting, the defect rate of a mature product can be as low as 2-5%, which introduces significant challenges in developing and testing a robust predictive solution. Moreover, the generation of the quality data could be further complicated by the fact that it is too expensive to perform quality inspection for all of the products. As a result, while it is possible and straightforward to measure the processing data (the inputs to the machine learning model) of each casting, to generate the quality data (the response variable of the model) can be quite difficult. In sum, metal casting is an unbalanced, semi-supervised learning problem which is challenging for even state-of-the-art machine learning algorithms.

One of the main tasks in developing the machine learning model is to work meaningfully with unbalanced raw datasets supplied by foundries. As shown in Figure 3a, in a dataset containing 500 castings, only 6% of the total production is categorized as Class 3 and are considered defective. The population of good quality parts (Class 1 and Class 2) is much larger than that of the defective parts. Several algorithms were explored for data balancing. These algorithms can learn from the structure of the minority class in the original dataset and construct their own rules for generating new datapoints, or oversample. For illustration, an example of data balancing is shown in Figure 3. Employing such an algorithm can make the population of all three classes nearly equal.

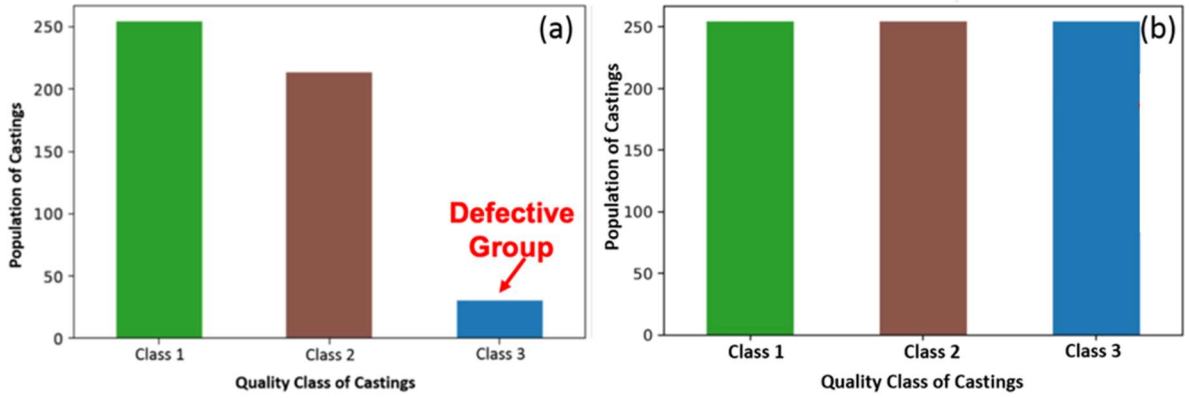


Figure 3. *(a) Original and (b) oversampled casting data of each class*.

Figures 3a and 3b show the original and the oversampled data respectively. The oversampling is done using a popular data balancing approach known as Synthetic Minority Oversampling TEchnique (SMOTE) [27]. This approach is used when the number of samples in one class is significantly higher than the samples in the other classes, as is typical in manufacturing datasets. As the name suggests, this technique generates synthetic samples of the minority class by interpolating between two instances of the minority class. The oversampling is done until a point that the proportion of the minority class matches that of the majority class, and we have a balanced dataset for training. There are also variations of the SMOTE approach that can be used, for

example, Borderline-SMOTE is widely used which focuses on the minority class samples that are at the border of the majority and the minority class, since these samples are more prone to misclassification errors as compared to those that are away from the border [28].

The data are sporadically labeled

For example, a classic problem in machine learning would be detecting defects in X-ray images of manufactured parts. The algorithm is trained by providing it with a large set of images of parts, with each image being *labeled* by whether the quality of this particular part is acceptable. From a given set of labeled images, the algorithm learns and constructs its own set of rules for distinguishing between acceptable and non-acceptable parts, and subsequently applies these rules to make predictions on *unlabeled* images. Labeled data means that processing and quality data of the parts manufactured are known. In the machine learning literature, such methods are called supervised machine learning, where supervision arises from the availability of labeled training data. As shown in Table I, for each individual sample, in the training dataset both the input variable (X) and its corresponding output variable (Y) are known. The algorithm can construct rules (e.g., $Y=f(x;\theta)$) to perform tasks such as predicting the quality of new parts. When labeled training data are not available, the machine learning problem becomes more difficult, and such algorithms are referred to as unsupervised machine learning. Whereas in semi-supervised machine learning only a fraction of the training data is labeled; both labeled, and the unlabeled training data are used to develop the appropriate algorithm.

Table I. Classes of machine learning tasks and techniques.

	Supervised			Unsupervised			Semi-Supervised		
		X	Y		X	Y		X	Y
Training Data	Sample 1	✓	✓	Sample 1	✓	×	Sample 1	✓	✓
	Sample 2	✓	✓	Sample 2	✓	×	Sample 2	✓	×
	Sample 3	✓	✓	Sample 3	✓	×	Sample 3	✓	×
	Sample 4	✓	✓	Sample 4	✓	×	Sample 4	✓	✓
	Sample 5	✓	✓	Sample 5	✓	×	Sample 5	✓	×
	Sample 6	✓	✓	Sample 6	✓	×	Sample 6	✓	✓
	Y = f (x;θ)			Pattern of the data			Y = f (x;θ)		
Predictions	Sample 7	✓	✓	Sample 7	✓		Sample 7	✓	✓
	Sample 8	✓	✓	Sample 8	✓		Sample 8	✓	✓
	Sample 9	✓	✓	Sample 9	✓		Sample 9	✓	✓

Most metal casting data are not "Big Data"

In our dataset, each individual row represents one cast component. The various columns in each row contain the recorded parameters when the cast component was produced. Unlike datasets

generated from social media activities, the scale of metal casting dataset is quite small. Although more sensors can be installed to capture additional processing data during casting (to add more columns), the total number of rows in the dataset is still limited by the volume or production capability of the foundry. For instance, we have collected data from three casting manufactures over the past two years, depending upon the casting method and the size of the casting component, the total number of one part produced in one year varies from 300 parts to 7000 parts per year. Even if the foundry can save and extract 10 years of historical data, there would only be about 70,000 rows in this dataset, which is well short of being considered appropriate in the realm of "Big Data".

Along with the oversampling techniques such as SMOTE, we can also use Generative Adversarial Networks (GANs), a class of artificial intelligence algorithms, to generate rows of new data by learning the structure of the original data and generating new samples that follow the same distribution [29], [30]. The original application of this technique was to generate photographs with many realistic characteristics that were superficially authentic to human observers. Applying GANs to generate more datapoints in metal casting datasets has shown promise. Mixing synthetic and real data is one way to overcome the drawback of having a small sized dataset. Synthetic data can be used to increase the volume of the data in case of small size datasets such as these, and the real data is used so that it is faithful to the original dataset.

Case Study of Machine Learning in Metal Casting

The following paragraphs provide detail for training and evaluating machine learning algorithms on production foundry data. Analysis begins with an investigation of the training data. The output for prediction is a binary pass or fail rating of the porosity classification. SMOTE is applied to overcome the class imbalance between pass and fail samples, and the newly balanced training data is standardized. Several machine learning algorithms are trained on datasets with and without dimension reduction. Algorithm performance is evaluated with a metric of minimizing false negative classifications on the testing dataset.

The results shown below are generated using the scikit-learn v0.24 [31], [32], matplotlib v2.0.2 [33], and pandas v1.0.4 [34], [35] libraries within the Python [36], [37] programming language.

Training Data

Shown below (Table II) is a snapshot of a portion of the dataset collected from a sand cast foundry. This dataset has 510 rows and 28 columns; each individual row represents a cast component. The various columns in each row give the processing parameters captured: component ID, metal chemistry, casting processing details, and quality data (X-ray inspection results). All the cast components were inspected, and the quality results were labeled with varying levels depending upon the appearance of porosity. Class 1 indicates that the casting was porosity-free, Class 2 indicates fine porosity, and Class 3 indicates large porosity voids. A dummy variable was used to divide quality data into a binary quality condition as given in Equation 1.

$$Y = \begin{cases} Pass: \ if\ the\ quality\ level\ of\ the\ part\ is\ Class\ 1\ or\ Class\ 2 \\ Fail: \ if\ the\ quality\ level\ of\ the\ part\ is\ Class\ 3 \end{cases} \qquad Eq.\ 1$$

Table II. A snapshot showing portion of one dataset containing 510 rows and 28 columns.

PourID	Temp_Floor	RH_Floor	Gr_Floor	LadleTemp	LadleDensityPP
6750	79	60	88.77	1320	2.639
6756	73	50	60.50	1333	2.644
6758	78	51	72.85	1330	2.637
6766	75	56	72.43	1332	2.638
6768	70	57	62.30	1333	2.635
6770	71	57	64.44	1330	2.636
6773	76	54	72.20	1327	2.635
6835	62	40	33.42	1332	2.641
6837	73	30	36.56	1338	2.644
6839	68	22	22.59	1340	2.642
6841	68	22	22.59	1338	2.639
6844	65	19	17.39	1335	2.644
6849	71	40	45.40	1330	2.640

Data Standardization

Since the physical meaning and the scale of all processing parameters incorporated into a given dataset varies significantly, the raw data in each column that represents a particular class needs to be standardized to ensure the data are unitless and are of comparable scale. Once all processing parameters are incorporated into a given dataset, the data in each column are standardized using a statistical method, called the Z-transform [17], [38], which converts the values in each column using the following equation:

$$Z_{i,j} = \frac{X_{i,j} - \mu_j}{\sigma_j} \qquad Eq.\ 2$$

Where

- $Z_{i,j}$ is the Z-transformed value of the parameter in one data cell
- $X_{i,j}$ is the original value of the parameter in the data cell
- μ_j is the mean of the original values of the parameter in the data column
- σ_j is the standard deviation of the original values of the parameter in the data column

Table III is a snapshot of the dataset after normalizing using a Z-transform. Compared with the original dataset shown in Table II, the values in each cell of the transformed dataset are on the same scale regardless of the physical meaning and the scale of these processing parameters. The data are now unitless.

Table III. A snapshot showing portion of the dataset after applying Z-transform.

PourID	Temp_Floor	RH_Floor	Gr_Floor	LadleTemp	LadleDensityPP
6750	1.34	1.48	2.03	-1.69	-0.48
6756	0.43	0.85	0.78	0.20	0.58
6758	1.19	0.91	1.33	-0.23	-0.90
6766	0.73	1.23	1.31	0.06	-0.69
6768	-0.03	1.29	0.86	0.20	-1.32
6770	0.12	1.29	0.95	-0.23	-1.11
6773	0.88	1.10	1.30	-0.67	-1.32
6835	-1.26	0.21	-0.42	0.06	-0.05
6837	0.43	-0.43	-0.28	0.93	0.58
6839	-0.34	-0.94	-0.90	1.22	0.16
6841	-0.34	-0.94	-0.90	0.93	-0.48
6844	-0.80	-1.13	-1.13	0.49	0.58
6849	0.12	0.21	0.11	-0.23	-0.26

Dimension Reduction:

All datasets collected from foundries are comprised of many columns regardless of the type of casting process. If the whole dataset were to be plotted on a scatter plot, this plot would need to have as many axes as the data has columns. However, the human perceptual system is designed to process three dimensions. As a result, foundry engineers will often have difficulty producing meaningful visual representations of their data. Fortunately, representing high-dimensional data in a low-dimensional space is a well-studied problem. In particular, Principal Component Analysis (PCA) is a classic dimension reduction technique allowing us to blend a large set of correlated variables in the original dataset into a smaller number of newly created representative variables [39]–[41]. This is a powerful tool explored in our study to compress the dimensionality of the original dataset and allow visualization of the complicated dataset on a two- or three-dimensional plot whose axes correspond to the newly created principal components. We can then use these principal components as the predictors in the machine learning model in place of the original larger set of variables. We evaluated this technique against the original high-dimension data and found that the dimension reduction via PCA was not necessary in this case study. However, PCA is an important method employed in many machine learning projects so we offer the following detailed description.

Compared with the original dataset which contained 28 columns, the dataset is now represented with three newly created columns, PC1, PC2, and PC3; therefore, the complicated dataset can be visualized with the three-dimensional plot shown in Figure 4b. The PCA plot is a scatter plot, in other words, PC1 is not a function of PC2 or PC3. These three components were used to display the dataset into several groupings of points. Each point in Figure 4 represents a cast component, and the position of the point is determined by all the input variables describing how the casting was manufactured. The output variable of the casting, in this case, the quality of the part (Class 1, 2, or 3), is marked by color.

The formation of clusters is most likely accounted for by the variations in production conditions. We investigated castings in the small cluster and found that they were manufactured in the last quarter of 2016. The cluster separation, most likely, is related to the seasonal changes when these parts were manufactured. This type of variation can more easily be detected once the data are visualized on a plot.

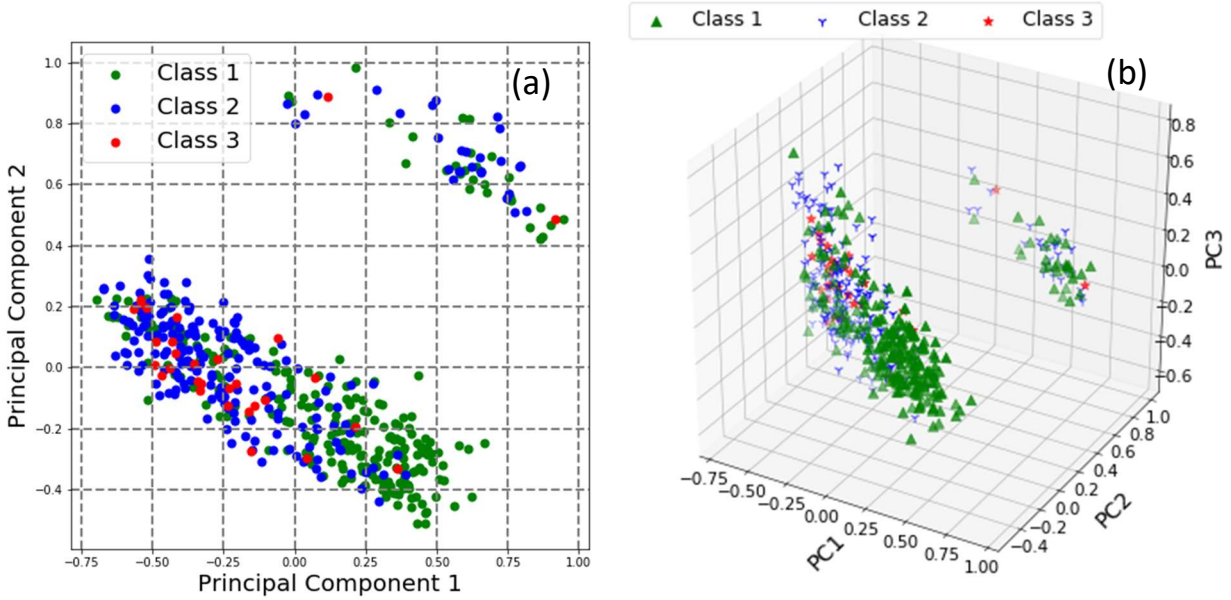


Figure 4. *(a) Two-dimensional PCA plot with color-coded quality feature of the original dataset. (b) Three-dimensional PCA plot with color-coded quality feature of the original dataset*.

The Singular Value Decomposition, or SVD, is a computational method often employed to calculate principal components for a dataset. Using SVD to perform PCA is efficient and numerically robust [41]. The singular value plot of the dataset is shown in Figure 5. The *x*-axis of this plot represents the first six principal components, and the *y*-axis shows the singular values of these components. The singular values of these principal components are plotted in the order from largest to smallest. The statistical interpretation of singular values is in the form of variance in the data explained by the various components. It can be interpreted that if a component has a high singular value, it represents a high percentage of variance in the dataset.

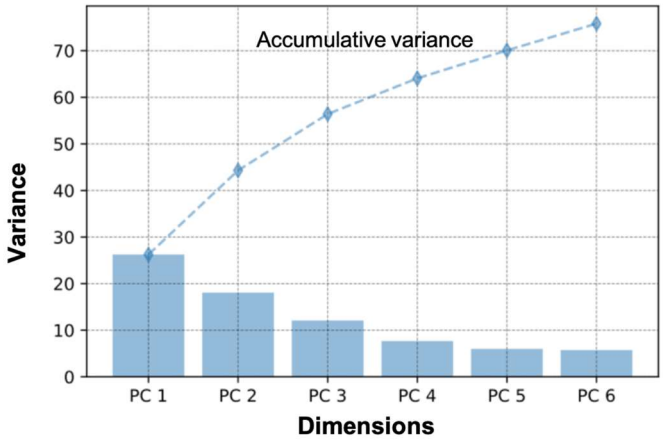


Figure 5. *Singular value decomposition plot of the dataset*.

As shown in Figure 5, the first principal component (PC1) and the second principal component (PC2) respectively represent about 27% and 18% of the variance of the dataset. Since the first two principal components represent less than 50% of the original data, it is necessary to introduce more principal components to better capture the essence of the original dataset.

Machine Learning Classifiers for Quality Prediction

The main objective of the data analysis work is to develop a machine learning based model to be used for part quality prediction. The performance of the models developed are evaluated by cross-validation. The complete dataset was divided into two sets of data, one for training the algorithms, and the other set for testing the performance of the algorithms. The testing dataset is about 10% the size of the original dataset. Since the quality of each casting in the test set is known, and the quality result is simplified into two possible classifications, "pass" or "fail", the performance of the model is measured via four numbers obtained from applying the algorithm to the testing dataset. These numbers are called True Positives TP, False Positives FP, True Negatives TN, and False Negatives FN. They can be presented using a two by two matrix called a *confusion matrix*. In our study, since the "fail" class is more critical to the foundry operation, we call the "fail" class Positives (P) and the "pass" class Negatives (N). The confusion matrix we used to present the output of cross validation in our study is shown in Table IV. If the model mistakenly predicted a bad part as a good part, it created a False Negative case. A model with good performance should give very few False Negatives, because the cost of this error would be high for the foundry.

Table IV. Confusion Matrix to Visualize Model Performance.

	Predicted # of Good Part	Predicted # of Bad Part
Actual # of Good Part	True Negative	False Positive
Actual # of Bad Part	False Negative	True Positive

Several algorithms were evaluated to predict part quality, and the confusion matrices of these algorithms are shown in Table V [5]. We use the SMOTE technique for oversampling and increasing the number of minority class samples in the dataset. We then use the oversampled data for training the classification algorithms such as Random Forest [42], Logistic Regression [43] and Support Vector Classifier (SVC) [11]. Specifically, Logistic Regression is a machine learning algorithm that is used for performing classification tasks based on the logistic function using probabilities. For example, anything above a probability threshold of 0.5 is predicted as one class and anything below 0.5 is predicted as another. Random Forest is a decision tree [44] based machine learning algorithm that is widely used in a number of classification as well as regression applications. Random Forest makes the prediction using the average of the predictions of the trees that build the forest in case of regression tasks and using the majority vote of the trees for label prediction in case of a classification task. SVC is a machine learning algorithm that is defined by a hyperplane that separates the classes in a dataset. It uses a labeled set of data as the training set and then categorizes new data on the correct side of the optimal separating hyperplane. Ensemble Learning combines the predictions obtained using all the classifiers mentioned above and then makes a prediction based on the majority vote for a certain class for every sample in the test dataset. Combining SMOTE with SVC performed best as seen in Table V below. No False Negatives were assigned by this model with only two False Positives.

Employing this method in a production environment allows for targeted selection of production parts for detailed quality inspection. Instead of a random sampling of parts, the system can select suspect parts identified by a validated model.

Table V. Several algorithms and their confusion matrix for performance evaluation.

SMOTE + Algorithm	**Confusion Matrix**
Random Forest [42]	$\begin{bmatrix} 22 & 5 \\ 6 & 14 \end{bmatrix}$
Logistic Regression [43]	$\begin{bmatrix} 22 & 5 \\ 9 & 11 \end{bmatrix}$
Ensemble Learning	$\begin{bmatrix} 23 & 4 \\ 8 & 12 \end{bmatrix}$
SVC [11] (Best Performance)	$\begin{bmatrix} 23 & 2 \\ 0 & 19 \end{bmatrix}$

Important Features Influencing Part Quality

In addition to making a quality prediction, another application of machine learning is the identification of critical features which are predicted to have the greatest influence on the quality of the product [45]. Some algorithms, for instance, Random Forest and other ensemble methods, can rank the various features (variables) in the dataset in terms of their importance to the predicted quality. Foundry engineers can benefit from this function to identify parameters to monitor and control product quality.

For this case study, Figure 6 shows the rankings of feature importance relevant to the quality calculated by two machine learning algorithms, namely the XGBoost [46] and Random Forest. Like Random Forest, XGBoost is a decision tree-based algorithm that is used for classification and regression problems. These algorithms are used for finding the most important features in a dataset in terms of the label predictions for many applications. The two algorithms differ in the way that most important features are selected. XGBoost uses a criterion known as F-score to decide which features are the most important in terms of the label prediction. F-score for a feature is defined as the number of times that a feature in the dataset is used for prediction. Higher F-scores represent the most important features. Similarly, for Random Forest, permutation importance is used as the criterion for selecting the top features in the dataset. Permutation importance permutes, or randomly shuffles, the values of every feature in the dataset by taking one column at a time and checking by how much the predictions change. Moreover, if after permuting the values of a column in the dataset, the predictions change significantly, then that column is deemed as important in terms of the predictions. We check the feature importance using two different algorithms to compare and see if they agree with each other. Figure 6 shows the top four features found by using these two techniques and it can be seen that at least three of the top features found by these algorithms are common. The Random Forest determined environmental conditions such as the grains of moisture content in the air on the foundry floor (Gr_Floor), relative humidity (RH_Floor), and ambient temperature (Temp_Floor) in addition to the metal temperature in the ladle (LadleTemp) to be important factors in predicting casting quality. XGBoost replaces ambient temperature with the density of the metal inside the riser, or feeder. We validated these predictions using domain expertise of foundry engineers and these were indeed the top features related to part quality according to domain experts. This is where machine learning can be exploited to target important inputs for better control over the casting process. Further, techniques such as feature importance can be used to drive designs of experiment, identify issues more quickly through targeted monitoring, and improve the overall cognition of the process.

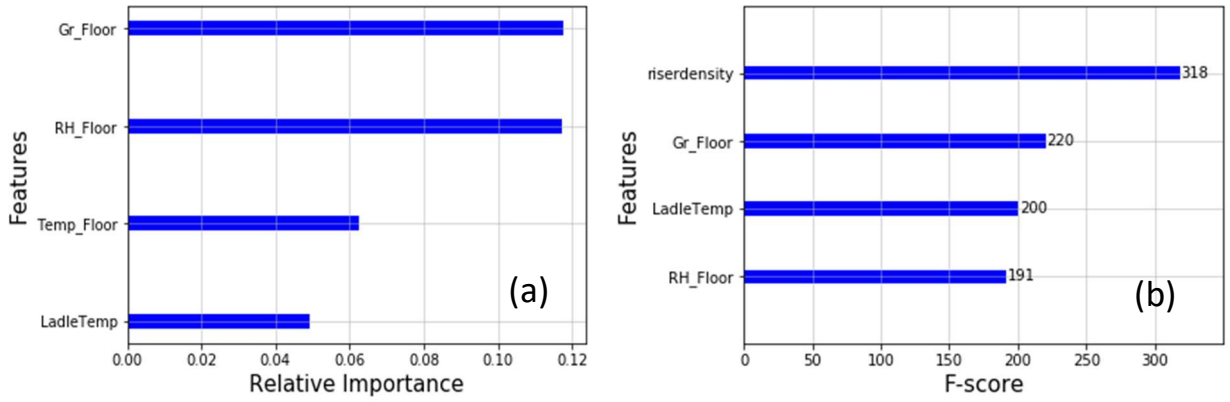


Figure 6. *Feature importance relevant to quality by rankings: (a) generated by Random Forest and (b) generated by XGBoost.*

V. Prospective View of Machine Learning for Manufacturing

The abilities to gather data, mine it for knowledge, and apply the insights that we gain are transforming almost everything that we do as a society and manufacturing is no exception. On the other hand, modern automated factors are well-springs of data where more information can be measured about manufacturing processes than ever dreamed before. These two ideas make the manufacturing industry ripe for a data revolution in which quality can be improved, production can be accelerated, and waste can be minimized, if only the right collaborations between data scientists, material engineers and the industrial sector can be achieved.

Modern machine learning, and deep learning in particular, provide tantalizing opportunities for progress, but the very power and generality of such data science methods bring along an important measure of responsibility. If used wisely, then such techniques allow for unprecedented improvements in the full flowering of Industry 4.0. If used unwisely, then the unbalanced, semi-supervised, and partially observed data that naturally arise in manufacturing problems, if not treated correctly from a data science and statistical perspective, can lead us astray. Perhaps as put best by the National Research Council of the National Academies [47]:

> *"Overlooking this foundation may yield results that are not useful at best, or harmful at worst. In any discussion of massive data and inference, it is essential to be aware that it is quite possible to turn data into something resembling knowledge when actually it is not. Moreover, it can be quite difficult to know that this has happened."*

However, through the opportunity that we have had working with so many industrial partners of the ACRC, who have generously worked with us and shared their data, we have at least helped begin a conversation on how best to use data science, machine learning, and deep learning in manufacturing problems. We see tremendous opportunities in improving the quality of cast components via the enabling tools we are developing, and there is also an implicit opportunity for major advances in planning for manufacturing and supply chain management. We are at the beginning of a revolution.

VI. Concluding Comments

Almost over six decades ago, C. P. Snow wrote a critical essay titled "The Two Cultures" [48] that not only pointed out the gap between the sciences and the arts, but also the opportunities if we could cross the bridge between the two cultures. If he were alive today, C.P. Snow may have written about the "Three Cultures"- the Arts, Sciences, and Engineering/Manufacturing. The authors of this paper are a good example of individuals from "three cultures" who have worked together and learned much from each other. However, to do so required emotional as well as time commitments and investments. The dividends those investments have paid for the authors have been invaluable and impactful. In a similar way, the execution of the Fourth Industrial Revolution will require a cultural diffusion and much discourse between data scientists and manufacturing engineers. There is no question that future of work will be transformed in the 21st century, as well as the future of the worker. But as has been stated before, the future is for us to make.

VII. References

[1] "Industry 4.0: the fourth industrial revolution- guide to Industrie 4.0." https://www.i-scoop.eu/industry-4-0/ (accessed May 26, 2020).

[2] K.-D. Thoben, S. Wiesner, T. Wuest, BIBA – Bremer Institut für Produktion und Logistik GmbH, the University of Bremen, Faculty of Production Engineering, University of Bremen, Bremen, Germany, and Industrial and Management Systems Engineering, "'Industrie 4.0' and Smart Manufacturing – A Review of Research Issues and Application Examples," *Int. J. Autom. Technol.*, vol. 11, no. 1, pp. 4–16, Jan. 2017, doi: 10.20965/ijat.2017.p0004.

[3] Capgemini Consulting Group, "Industry_4.0_-The_Capgemini_Consulting_V.pdf." Capgemini, 2014, [Online]. Available: https://www.capgemini.com/consulting/wp-content/uploads/sites/30/2017/07/capgemini-consulting-industrie-4.0_0_0.pdf.

[4] T. Prucha, "From the Editor - Big Data," *Int. J. Met.*, vol. 9, no. 3, p. 5, 2015.

[5] J. Friedman, R. Tibshirani, and T. Hastie, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

[6] L. Hauser, "Internet Encyclopedia of Philosophy," *Artificial Intelligence*. https://www.iep.utm.edu/art-inte/ (accessed May 26, 2020).

[7] A. M. Turing, "I.—COMPUTING MACHINERY AND INTELLIGENCE," *Mind*, vol. LIX, no. 236, pp. 433–460, Oct. 1950, doi: 10.1093/mind/LIX.236.433.

[8] C. Bernhardt, *Turing's Vision - The Birth of Computer Science*. MIT Press, 2016.

[9] J. McCarthy, M. Minsky, N. Rochester, and C. E. Shannon, "A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence." Aug. 31, 1955, Accessed: Feb. 17, 2020. [Online]. Available: https://wvvw.aaai.org/ojs/index.php/aimagazine/article/view/1904.

[10] K. P. Murphy, *Machine learning: a probabilistic perspective*. Cambridge, MA: MIT Press, 2012.

[11] Y. Zhu and Y. Zhang, "The Study on Some Problems of Support Vector Classifier," *Comput. Eng. Appl.*, no. 13, 2003, [Online]. Available: http://en.cnki.com.cn/Article_en/CJFDTotal-JSGG200313011.htm.

[12] M. W. Craven and J. W. Shavlik, "Using neural networks for data mining," *Data Min.*, vol. 13, no. 2, pp. 211–229, Nov. 1997, doi: 10.1016/S0167-739X(97)00022-8.

[13] J. D. Rodriguez, A. Perez, and J. A. Lozano, "Sensitivity Analysis of k-Fold Cross Validation in Prediction Error Estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 3, pp. 569–575, Mar. 2010, doi: 10.1109/TPAMI.2009.187.

[14] C. Reid Turner, A. Fuggetta, L. Lavazza, and A. L. Wolf, "A conceptual basis for feature engineering," *J. Syst. Softw.*, vol. 49, no. 1, pp. 3–15, Dec. 1999, doi: 10.1016/S0164-1212(99)00062-X.

[15] A. Zheng and A. Casari, *Feature Engineering for machine learning: Principles and techniques for data scientists*. Beijing: O-Reilly, 2018.

[16] I. Gibson and C. Amies, "Data normalization techniques," 6259456, Jul. 10, 2001.

[17] "Z-Transform," *Wolfram MathWorld*. https://mathworld.wolfram.com/Z-Transform.html (accessed May 26, 2020).

[18] W. M. P. van der Aalst, V. Rubin, H. M. W. Verbeek, B. F. van Dongen, E. Kindler, and C. W. Günther, "Process mining: a two-step approach to balance between underfitting and

overfitting," *Softw. Syst. Model.*, vol. 9, no. 1, p. 87, Nov. 2008, doi: 10.1007/s10270-008-0106-z.
[19] J. K. Kittur, G. C. Manjunath Patel, and M. B. Parappagoudar, "Modeling of Pressure Die Casting Process: An Artificial Intelligence Approach," *Int. J. Met.*, vol. 10, no. 1, pp. 70–87, Jan. 2016, doi: 10.1007/s40962-015-0001-7.
[20] E. Kocaman, S. Şirin, and D. Dispinar, "Artificial Neural Network Modeling of Grain Refinement Performance in AlSi10Mg Alloy," *Int. J. Met.*, 2020, [Online]. Available: https://doi.org/10.1007/s40962-020-00472-9.
[21] P. K. D. V. Yarlagadda and E. Cheng Wei Chiang, "A neural network system for the prediction of process parameters in pressure die casting," *J. Mater. Process. Technol.*, vol. 89–90, pp. 583–590, May 1999, doi: 10.1016/S0924-0136(99)00071-0.
[22] J. K. Rai, A. M. Lajimi, and P. Xirouchakis, "An intelligent system for predicting HPDC process variables in interactive environment," *J. Mater. Process. Technol.*, vol. 203, no. 1–3, pp. 72–79, Jul. 2008, doi: 10.1016/j.jmatprotec.2007.10.011.
[23] A. Krimpenis, P. G. Benardos, G.-C. Vosniakos, and A. Koukouvitaki, "Simulation-based selection of optimum pressure die-casting process parameters using neural nets and genetic algorithms," *Int. J. Adv. Manuf. Technol.*, vol. 27, no. 5–6, pp. 509–517, Jan. 2006, doi: 10.1007/s00170-004-2218-0.
[24] J. Zheng, Q. Wang, P. Zhao, and C. Wu, "Optimization of high-pressure die-casting process parameters using artificial neural network," *Int. J. Adv. Manuf. Technol.*, vol. 44, no. 7–8, pp. 667–674, Oct. 2009, doi: 10.1007/s00170-008-1886-6.
[25] D. Blondheim, "Artificial Intelligence, Machine Learning, and Data Analytics: Understanding the Concepts to Find Value in Die Casting Data," presented at the 2020 NADCA Executive Conference, Clearwater Beach, FL, Feb. 25, 2020.
[26] T. Prucha, "From the Editor: AI Needs CSI: Common Sense Input," *Int. J. Met.*, vol. 12, no. 3, pp. 425–426, Jul. 2018, doi: 10.1007/s40962-018-0235-2.
[27] R. Blagus and L. Lusa, "SMOTE for high-dimensional class-imbalanced data," *BMC Bioinformatics*, vol. 14, no. 1, p. 106, Dec. 2013, doi: 10.1186/1471-2105-14-106.
[28] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning," in *Advances in Intelligent Computing*, vol. 3644, D.-S. Huang, X.-P. Zhang, and G.-B. Huang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 878–887.
[29] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative Adversarial Networks: An Overview," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 53–65, Jan. 2018, doi: 10.1109/MSP.2017.2765202.
[30] I. Goodfellow, "NIPS 2016 Tutorial: Generative Adversarial Networks," *ArXiv170100160 Cs*, Apr. 2017, Accessed: May 27, 2020. [Online]. Available: http://arxiv.org/abs/1701.00160.
[31] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011, doi: 10.1016/j.patcog.2011.04.006.
[32] A. Geron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*, 1st ed. O'Reilly, 2017.
[33] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 90–95, 2007, doi: 10.1109/MCSE.2007.55.
[34] The pandas development team, *pandas-dev/pandas: Pandas*. Zenodo, 2020.

[35] W. McKinney, “Data Structures for Statistical Computing in Python,” in *Proceedings of the 9th Python in Science Conference*, 2010, pp. 51–56, Accessed: Jan. 09, 2020. [Online]. Available: http://conference.scipy.org/proceedings/scipy2010/pdfs/mckinney.pdf.
[36] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.
[37] T. E. Oliphant, “Python for Scientific Computing,” *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 10–20, Jun. 2007, doi: 10.1109/MCSE.2007.58.
[38] T. Wuest, D. Weimer, C. Irgens, and K.-D. Thoben, “Machine learning in manufacturing: advantages, challenges, and applications,” *Prod. Manuf. Res.*, vol. 4, no. 1, pp. 23–45, Jan. 2016, doi: 10.1080/21693277.2016.1192517.
[39] C. Eckart and G. Young, “The approximation of one matrix by another of lower rank,” *Psychometrika*, vol. 1, no. 3, pp. 211–218, Sep. 1936, doi: 10.1007/BF02288367.
[40] H. Abdi and L. J. Williams, “Principal component analysis: Principal component analysis,” *Wiley Interdiscip. Rev. Comput. Stat.*, vol. 2, no. 4, pp. 433–459, Jul. 2010, doi: 10.1002/wics.101.
[41] S. Wold, K. Esbensen, and P. Geladi, “Principal Component Analysis,” *Chemom. Intell. Lab. Syst.*, vol. 2, pp. 37–52, 1987, doi: 10.1016/0169-7439(87)80084-9.
[42] M. Pal, “Random forest classifier for remote sensing classification,” *Int. J. Remote Sens.*, vol. 26, no. 1, pp. 217–222, Jan. 2005, doi: 10.1080/01431160412331269698.
[43] R. E. Wright, “Logistic regression.,” in *Reading and understanding multivariate statistics.*, Washington, DC, US: American Psychological Association, 1995, pp. 217–244.
[44] D. Dietrich, B. Heller, and B. Yang, *Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data*, 1st ed. Wiley, 2015.
[45] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, “Permutation importance: a corrected feature importance measure,” *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, Apr. 2010, doi: 10.1093/bioinformatics/btq134.
[46] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 785–794, Aug. 2016, doi: 10.1145/2939672.2939785.
[47] National Research Council, *Frontiers in Massive Data Analysis*. Washington, D.C.: National Academies Press, 2013.
[48] C. P. Snow, *The Two Cultures*. London: Cambridge University Press, 1959.

VIII. Acknowledgements

The authors would like to thank the ACRC consortium members for their support and data for this project.

Appendix C – Approach and Methodology

The primary objective of the research is to delve into the nexus of materials processing and data science to begin our understanding of the challenges unique to the materials processing field. Materials processing is an interesting machine learning challenge. While it is the author's belief that the application of machine learning is beneficial to any materials manufacturing technology, this work is centered on the high pressure die casting (HPDC) of aluminum alloys. The literature review in this volume (***Appendix A***) gives a comprehensive overview of the HPDC process and examples of the research which has driven our understanding about which HPDC process parameters influence microstructural discontinuities, such as porosity, and their impact on the resulting mechanical properties.

The published literature contains many papers which report observations of various process inputs on mechanical properties and porosity. Forward focused HPDC facilities do a good job of capturing many of these data for each casting they produce. As an industry, we believe that we are collecting the correct data. The literature confirms the importance and die casters document and demonstrate process control to their customers by this data. The hypothesis that this work aims to test is that die casters collect the correct input information and, given a large enough dataset, quality and performance properties can be predicted from that data.

I. High Pressure Die Casting Data

Die casting is a thermal process where molten metal is delivered to a machine, injected into a die, and allowed to solidify. Key input data to predict solidification phenomena are those variables which affect the thermal system: temperatures, times, pressures, filling velocities, flow rate of cooling lines, and amount of die spray applied to aid part removal are some of the data which could be collected and analyzed. Other data is useful for machine health and predictive maintenance such as motor amperage draw and cycle times for each piece of equipment. At the holding furnace, the temperature of the metal and metal level can be captured as time series data or at the start of each cycle. Alloy composition is periodically sampled at the machine or upstream in the melting operation. From the literature, the reported HPDC input variables which drive mechanical properties and porosity are intensification pressure, slow shot velocity, fast shot velocity, vacuum pressure, and melt temperature. Alloy composition is also an important factor, though the ranges investigated to measure an effect are wider than the variation in the alloy of this study. More detail and a complete list of references is included in ***Appendix A - Literature Review***.

Modern foundry equipment is PLC driven and integrated such that input and output signals are passed between the equipment in the work cell throughout the cycle. Being a thermal process, time is an important factor in HPDC. Timing of signal activity, when sensors are made on periphery equipment and on the die casting machine, can be captured. The level of detail is up to the operation. For example, one could capture the time it takes an extraction robot to complete its entire cycle, or record each segment of that cycle: extract, trim, pin stamp, etc. as separate variables. The DCM is the hub of the cell and all the peripheral equipment relay their signals through the DCM. This is convenient for organizing the data and assigning each value to a serialized part number.

The DCM is programmed by the process engineer to perform movements which have been developed to produce acceptable castings which meet the specifications of the design. The shot velocity profile dictates the velocities the machine will move the plunger forward and when to change from one speed setting to another as it travels. An example of a shot velocity profile is shown in *Figure 1*. Modern machines are

equipped with fast acting valves which give the engineer great flexibility and precision in performance [1]. The intensification pressure is another important parameter programmed into the DCM for the purpose of increasing the soundness of the casting. The onboard injection monitoring computer captures the velocity, pressure, and position inputs as well as how the machine performed with respect to those inputs in a combination diagram called a shot trace.

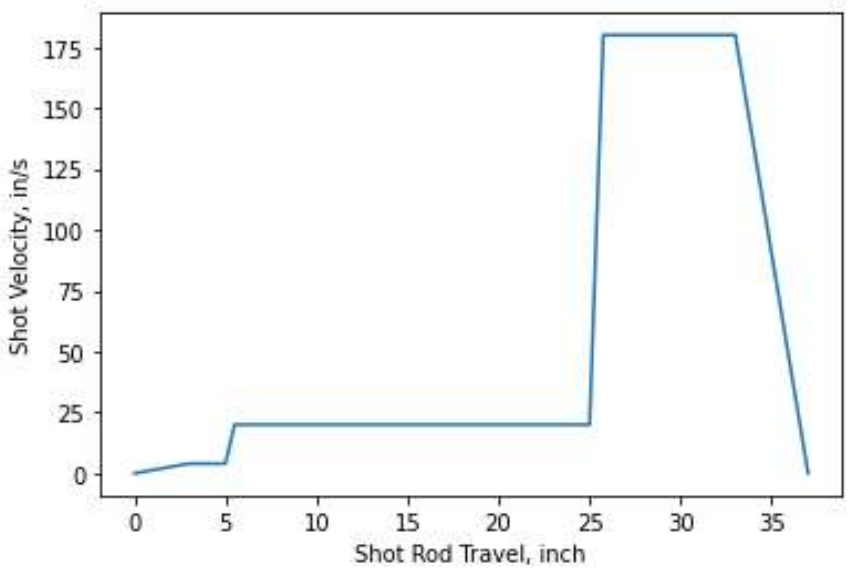


Figure 1. *Schematic of a shot trace in aluminum high pressure die casting.*

From the shot trace, key process outputs can be summarized and displayed to the operator. Even now, it is common practice for this summary data to be stored on the injection monitoring computer. Storage constraints dictate that the oldest cycle data be deleted as new machine cycles were performed and stored. This allows technicians and engineers to review recent history and make comparisons to aid in troubleshooting, but data is being lost. With the improved connectivity and memory storage options of 2010's era technology, this data can be uploaded to cloud-based data storage after each cycle and maintained indefinitely. The cycle summary data is historically the best, and often the only, information available to troubleshoot the process and make intuitive, experience-based predictions regarding the quality of the resulting castings. For this reason, it is also the most easily and widely stored data. Additional data from the cycle are collected and appended to this information prior to uploading into long-term storage.

To make robust predictive models one needs relevant data to the problem at hand and prefers a large quantity of it; Big Data. Unfortunately, HPDC is not a Big Data environment when compared to YouTube or Facebook. Conceding that there are always exceptions, production demand for a newly designed HPDC component varies from approximately 10,000 to over 1 million pieces per year. Consider the following hypothetical scenario at the high end of the production demand spectrum. An imaginary part is cast in a single-cavity die. Each cycle of the process yields one part and 1 million are needed per year. For simplicity, assume the useful life of each tool is 100,000 cycles and each machine can run 100,000 cycles per year. Ideally, ten identical dies in ten identical die casting machines would cover the demand.

However, as uncovered in ***Appendix D***, combinations of die and machine can behave as unique processes unto themselves dividing the 1 million sample dataset into 10 subsets of 100,000 each. Further complicating the picture, die casting operations do not set a die in a machine and run it until the useful life

is spent. During this year of production, an operation would need to pull dies periodically and perform maintenance on them, while another back-up die runs in its place. Perhaps 15 identical dies are required to allow for the maintenance rotation. If divided evenly over the first year, each die would have a rounded 67,000 cycles on it. For flexibility in dealing with which dies require maintenance, a die may be approved to run on three of the ten machines. Assuming an even split, each combination of die cavity and machine would have 22,000 samples cast with their process input data recorded. 22,000 pieces would require nearly 60 days to cast. Sampling one piece per day for mechanical properties would yield 60 measured outputs for each die cavity/machine pair (0.27% sampled). This scenario shows how a seemingly expansive dataset can become quite small. Small data is a challenge for machine learning. Research in this space, such as this project, strives to improve a difficult data science domain.

As an industry, die casters are generally not to the level where every potential important variable is captured, and has been for years, such that large datasets are commonplace. There is also the challenge of accessibility to the data for analysis. Leaders in the industry recognize the importance of taking the first steps in bringing machine learning into die casting. The Aluminum Casting Research Center (ACRC) at WPI is an industry-university consortium where a cross-section of the aluminum casting industry including alloy producers, casters, industry suppliers, and end users meet and sponsor pre-competitive fundamental research [2]. FCA, a major automobile manufacturer with a large die casting operation and longtime member of the ACRC, partnered with the research team to provide a calendar year worth of HPDC process data, alloy composition checks, and mechanical property testing data. The size of the datasets is given in Table I. The details of the datasets with respect to which inputs and outputs are available and descriptions of each are given in Tables C-I and C-II at the end of the appendix.

Table I. FCA datasets size details.

Dataset Name	**Raw Dataset (Rows x Columns)**
HPDC Process	956,986 x 109
Alloy Composition	980 x 17
Tensile Testing	1,634 x 14

The HPDC process data can be thought of as a spreadsheet with each row representing an individual casting and each column containing a piece of information about that casting. The columns are the input variables which produced the castings and output variables which are data determined about the castings after they are made. Similarly, in the mechanical property dataset each row represents a tensile bar and the columns contain the input and output variables associated with each bar. The composition dataset has rows which represent each test and columns containing the amount of each elemental constituent in the melt at that time. This description is rather straightforward; however, visualization is difficult. The raw HPDC dataset has 109 columns. Humans are finite beings and, as such, have no ability to visualize what is happening in 109 dimensions. Fortunately, machines can perform these tasks on our behalf via machine learning algorithms that analyze high-dimensional data.

The literature provides an understanding of which variables are important for mechanical properties. Today's HPDC equipment is more interconnected than ever to facilitate data organization and collection in the die casting cell. Platforms now exists for storing and accessing large amounts of data with which to train machine learning models. The need largely remains within the die casting industry to begin taking

advantage of this reality and start investigating how to process data and train algorithms to create knowledge for data-driven decision making.

II. Data Science Approach

Data science projects are more intricate than collecting data and plugging it into an algorithm. There are steps one must take in an iterative process to generate reliable predictions and actionable results. An overview is given below with more detail on the methods used and references in the subsections that follow. The main building blocks of a data science project are:

Data exploration: Here we gain an understanding of the data in front of us. We determine how much data we have, what data types make up the features, generate statistical summaries of the features, initial visualization to uncover relationships between features. For example, it would save us a lot of time to see that that one of our many input features is linearly correlated to our target output. One has to look at their data before moving any further.

Pre-processing: This makes up the bulk of the effort. Many necessary actions fall under the pre-processing step and iterations of development tend to bring us back to this step. Pre-processing involves the cleaning of the dataset, feature engineering, standardization or normalization of the data, and dimension reduction.

Machine Learning Algorithms: Once the data is ready for analysis, the algorithms are selected in regard to the problem at hand, data collected, and the information one is looking to gain. Within the off-the-shelf algorithms, there are hyperparameters which can be adjusted for optimum performance.

Model Evaluation: To evaluate the model, cross-validation is conducted to get a better idea of how the model will operate in general rather than on one training instance. Performance is evaluated by comparing how the algorithm did on the training and testing data with respect to the chosen performance metric.

Iteration: Most of the time, model performance is not the best it can be on the first attempt. It is not as accurate as we hoped, the model is overfitting the training data, or perhaps we want to try looking at a subset of features or a different set of hyperparameters. Iteration is critical to optimizing model performance.

Reporting: Performing data analytics is most effective when accompanied with an effective method to communicate what we have found. Creating graphs and figures to show our results help tell the story of the data.

This section describes the software, pre-processing methods, machine learning algorithms, and evaluation techniques used in this project. For the purpose of an illustrative example for data pre-processing and machine learning algorithms, a faithful subset of the FCA HPDC process data was created named *toydata* (Table II). This small dataset was built by selecting features which capture the range in scale of the HPDC process data from three arbitrarily chosen machines. The process inputs included in *toydata* are biscuit length, cavity fill time, final intensification pressure, average velocity of the plunger during slow shot, spray robot cycle time, molten metal temperature in the furnace, and the ladle pour time. Incorporating the machine, which was set to 1, 2, and 3 respectively, provides *toydata* with a categorical feature in the dataset.

Table II. The *toydata* dataset for illustration purposes.

BiscuitLength	CavityFillTime	FinalIntensifierPressure	SlowShotVelAve	SprayRobotTime	MetalTemp	LadlePourTime	machine
2.6	103	6379	4.3	61.30	1217	11.97	1
2.6	103	6453	3.9	54.97	1217	11.42	1
2.5	102	6473	3.6	58.33	1216	10.85	1
2.7	103	6440	3.8	62.43	1222	11.00	1
2.7	105	6343	3.9	63.62	1219	11.14	1
2.6	103	6421	3.9	56.22	1235	11.50	1
2.5	104	6314	4.1	58.40	1219	10.82	1
2.6	105	6460	3.6	55.07	1219	11.30	1
2.7	104	6356	4.0	60.60	1224	10.82	1
2.6	102	6541	4.2	56.65	1224	10.52	1
2.9	106	6446	2.4	59.03	1232	9.02	2
2.5	105	6432	2.3	58.72	1229	11.95	2
2.5	103	6525	2.4	61.90	1234	9.12	2
2.5	105	6640	2.9	59.55	1221	9.27	2
2.6	104	6336	2.4	58.15	1225	12.59	2
2.7	105	6484	2.6	60.50	1232	12.62	2
2.9	107	6397	2.4	61.80	1229	9.15	2
2.8	102	6522	2.4	57.88	1235	9.70	2
2.7	107	6358	2.5	61.33	1222	9.62	2
2.5	104	6398	2.4	59.35	1215	11.12	2
2.7	103	6269	5.6	57.70	1230	9.35	3
2.6	101	6365	6.1	59.35	1225	10.92	3
2.7	101	6468	4.5	55.67	1231	11.50	3
2.6	100	6423	4.3	55.40	1228	11.69	3
2.5	101	6339	5.8	59.22	1223	11.42	3
2.8	101	6298	4.8	54.97	1227	11.62	3
2.8	100	6265	5.7	58.95	1224	11.35	3
2.8	102	6350	5.2	56.20	1230	11.32	3
2.5	100	6284	5.3	57.95	1230	11.75	3
2.5	99	6445	4.3	54.53	1212	11.65	3

Software/Libraries/Helpful Resources

The benefits of machine learning are realized on very large datasets such as the HPDC process dataset. There is still a place for spreadsheet software for manageable datasets. However, when datasets grow too large, purpose-built software is required. Excessively large datasets, Big Data, utilize other tools for performing advanced statistical analysis and model generation. Two such tools are computing codes R and Python [3]–[5]. Both are open source and free to download. R and Python have numerous ready-made packages perform data analysis, organization and visualization. A key advantage is an engaged user community with an endless amount of helpful information on the internet and continual adaption to the needs of the data science community. Both are computing codes so there is a more involved learning curve as compared to point and click commands of Microsoft Excel, for example [6]. The flexibility in analysis tools available and reproducibility of custom graphics make R and Python programming indispensable skills for performing machine learning in materials processing.

The following few paragraphs highlight open source libraries, modules, and packages recommended for someone who is interested in getting started in machine learning in Python. The examples are provided in context of how they were utilized for this research. To ascertain the full functional extent of each, the

reader is encouraged to visit the cited websites and read for themselves the capability of these packages, work through the tutorials, and determine their suitability for your data.

pandas [7]–[9]
Pandas is a data structuring package for conducting data analysis in Python. Data can be structured as either a one-dimensional series or two-dimensional data frame. While the diversity of data which pandas can handle is wide ranging, the data in the present study is of the tabular kind as an Excel spreadsheet, to name a common example. Pandas is capable to import files such as .xlsx and .csv files directly into Python as a pandas DataFrame. The first important feature is the ability to summarize and review the data after importing. This includes statistical summaries, data type identification by column, the shape (dimensions) of the data frame, and the presence of missing data (or NaN). Once the basic information about the data is understood, within pandas one can change the data type of specific columns, create subsets, and compare and merge different data frames. Columns and rows can be created, removed, renamed, and reordered. The creation of a new column can be based on, but not limited to, a mathematical operation, splitting character strings, Boolean conditions, and discretization. The flexibility is high so that in one line of code a new column may be created utilizing more than one of these conditions. In addition to this built in functionality, pandas allows the user to create their own functions and apply the data to them. In short, pandas is a very useful tool in accessing and working with data frames.

NumPy [10]–[12]
NumPy (pronounced Num-Pie) is a basic building block of working with numerical data in Python. The other libraries listed in this section are built upon NumPy. NumPy utilizes efficient n-dimensional homogenous arrays (ndarrays) typically for mathematical calculations. Though pandas is built upon NumPy, there are times where the algorithm with which one is working requires an array or a data frame specifically, so it is a good practice to import both into your Python code. Similar to a pandas data frame, NumPy ndarrays can be sliced, merged, subset, etc. to create new ndarrays. A very useful function in NumPy is the creation of random numbers or arrays of random numbers. This is a great way to generate data with which to practice new data science techniques.

Matplotlib [13], [14]
Matplotlib is the go-to library for data visualization. A critical component to any data science project is the creation of plots and charts which tell the story of that data. Matplotlib offers a wide selection of plot styles that are highly customizable, and easily reproduced once the code is written. Matplotlib.org offers many examples and tutorials to aid the user in getting started.

Scikit-learn [15], [16, p. 1]
Machine learning algorithms can be composed in Python from scratch. For efficiency, the scikit-learn library is a comprehensive suite of popular machine learning algorithms which are plug and play into Python code. The user simply loads the algorithm from the proper scikit-learn module and adjusts the parameters of the model as needed. In addition to these algorithms, scikit-learn offers modules for pre-processing (standardization, one-hot encoding), dimension reduction (feature selection, Principal Component Analysis), and model selection (cross-validation, metrics).

colab.research.google.com [17]– Google's Colaboratory, Colab for short, offers the ability to write Python code in a web browser. The benefit is free access to Google's graphic processing units (GPUs). GPUs are gaming hardware which has been appropriated for machine learning. Large data science projects can

overwhelm computers set-up for general home and office use. In such cases, running machine learning algorithms on Colab GPUs will accelerate one's progress.

stackoverflow.com [18]– User community driven, one-stop answer site for your coding questions. If you have an error in your code the answer is almost certainly on this website. Learning to ask the right question takes time. Building out one's data science coding lexicon will help in asking the right questions.

towardsdatascience.com [19]– A community of bloggers posting useful quick tutorials on a specific topic of data science from basic beginner tips to advanced and specialized techniques. These insightful blogs help explain the highly technical in plain English.

kaggle.com [20]– An excellent source for data with which to hone one's skills.

The FCA dataset was analyzed using both R and Python. The initial data fusion and cleaning was conducted in R to create a comma separated values (.csv) file. Python has this capability via pandas and NumPy as described above. This file is uploaded into Python for the predictive model analysis for part quality and mechanical properties. The steps for preparing the data and the analyses performed are detailed below using *toydata* where graphics benefit the reader. At the end of each section, there is a supplemental table with the Python libraries used in the examples given along with citations of where to find more information on each.

Data Pre-processing

Cleaning the Dataset

Real world production data is messy. Missing values, erroneous sensor readings, duplicated entries, typos, format changes in the source file, etc. must be sorted out before one can engage in meaningful analysis. Considering the FCA HPDC process data set with over 950,000 observations and 109 variables, one cannot simply scroll through and hope to catch these issues by eye. Running summaries of the data, examining the data class, and locating *NaN* values are a few of the tasks to accomplish in this step. Fortunately, there are simple commands to execute which reveal issues in the data, but how to address them is up to the data scientist and the domain experts involved in the project. One example of how bad data comes to be is when a monitoring sensor fails, it is common for production to continue until such a time when a technician is available to replace it. Each cycle while the sensor is malfunctioning, its column in the dataset will be left blank or populated with bad data. Bad data can be recognized by its scale or upon comparison to typical values when the sensor was working properly. There is often no way to recover the actual missing values and erroneous sensor readings. Even so, this does not mean that these columns should simply be deleted. For example, one can impute the mean or median value depending on whether the distribution is Gaussian or not.

Another source of missing data is not a result of something going awry, rather it is a result of sampling frequency. This is called heterogeneous data, and alloy composition is a good example. Every casting has a composition; however, the furnace is only evaluated once per shift as a quality control audit. Parts cast near the time when the composition check occurred can be estimated to be near that composition, but after hours have passed, that assumption is less valid. For this study, elemental compositions were imputed into the merged dataset based on proximity in time to when the alloy was measured. The composition of the

E380 aluminum alloy cast was quite consistent and no measures were taken showing the alloy to be out of specification.

A popular domain for Data Science research is in the area of synthetic data generation for AI [21]–[23]. These methods are promising and more faithful attempts to create new synthetic data which better represent the original data than mean imputation or sample duplication. The downside of these naïve methods is that the statistical distribution of the data will change upon imputation and duplication provides no new information to the algorithm. The objective is to fill gaps of the dataset, inputs and outputs, with created values which maintain the statistical measures of the true data from which it was derived. Synthetic data generation can be used for data masking as a security measure and creating data when gathering additional real data is very expensive or risky such as autonomous vehicle training. [24], [25].

An important area where synthetic data creation is being explored is the balancing of imbalanced datasets. The Synthetic Minority Oversampling TEchnique (SMOTE) provides new information to the model via the creation of new minority class samples with which to *train* the algorithm [26], [27]. It is very important that a testing dataset be set aside which represents the properties of the original dataset prior to using SMOTE. SMOTE creates each new minority class sample by selecting an example of the minority class, finding its nearest neighbors (k-neighbors = 5 is the default), and drawing a line between the example and one of its neighbors at random. The new sample is created along the connection line. This is done repeatedly until the minority class balances out the majority class or meets a prescribed ratio. A disadvantage of SMOTE is that it is prone to making synthetic data which is not representative of the process under investigation. Another is that SMOTE will generate data far from the borders of classes which the algorithm can easily classify already. *Figure 2a* shows a randomly generated dataset created from scikit-learn make_classification module [28]–[30]. The dataset is comprised of 2000 samples representing two clustered classes. The weight of the majority class is 0.96 (1920 samples). *Figure 2b* shows the same dataset after the application of SMOTE. The new dataset is balanced via SMOTE at 1920 samples in each class.

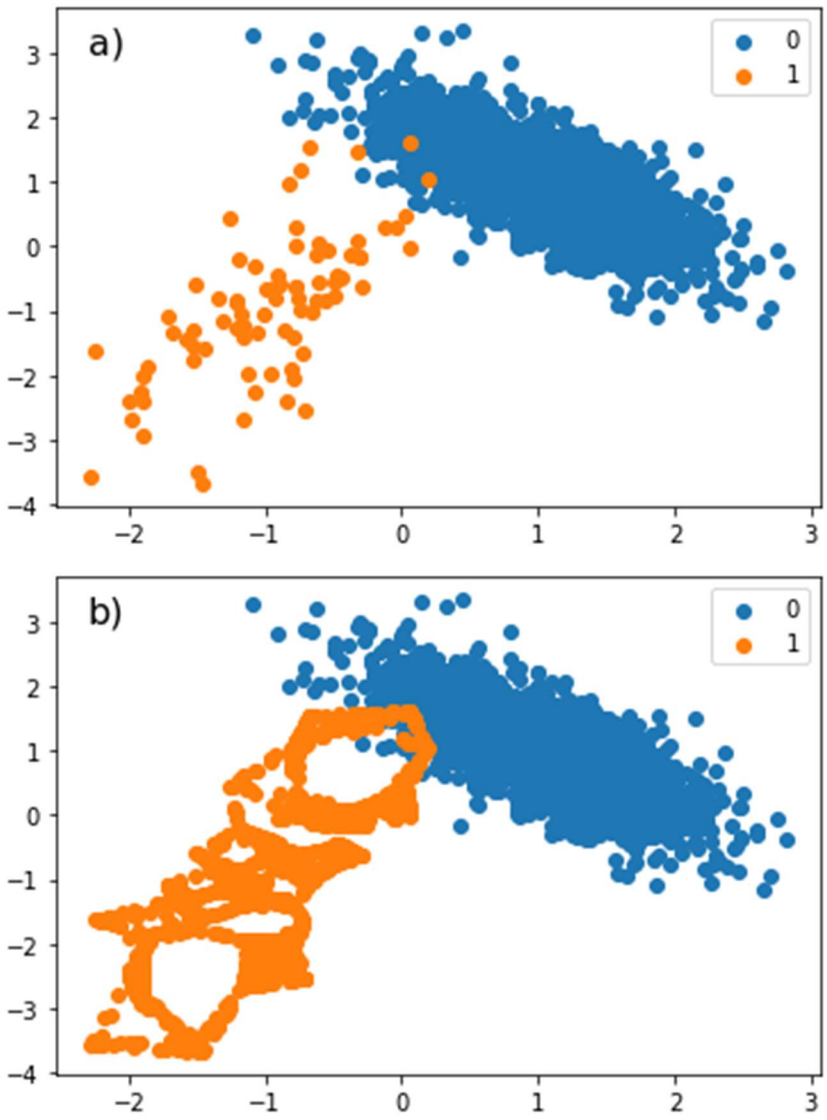


Figure 2. *An illustration of SMOTE application showing a) the unbalanced data prior to synthetic data generation and b) the balanced dataset after SMOTE. In this classification example, many synthetic data are generated far from the border of the two class clusters which is less useful for making predictions between classes.*

Borderline SMOTE is a selective oversampling method which increases the number of minority class samples in the region of the border between two class clusters [31]. When two samples of opposite class are not similar, i.e. far apart in *Figure 2a*, training the algorithm to classify them correctly is easy. The error increases along the border where minority class samples are more likely to be grouped into the majority class. Generating more minority class in this region will accentuate what is unique about one class versus the other and improve predictive performance. *Figure 3* displays Borderline SMOTE applied to the data from *Figure 2a*.

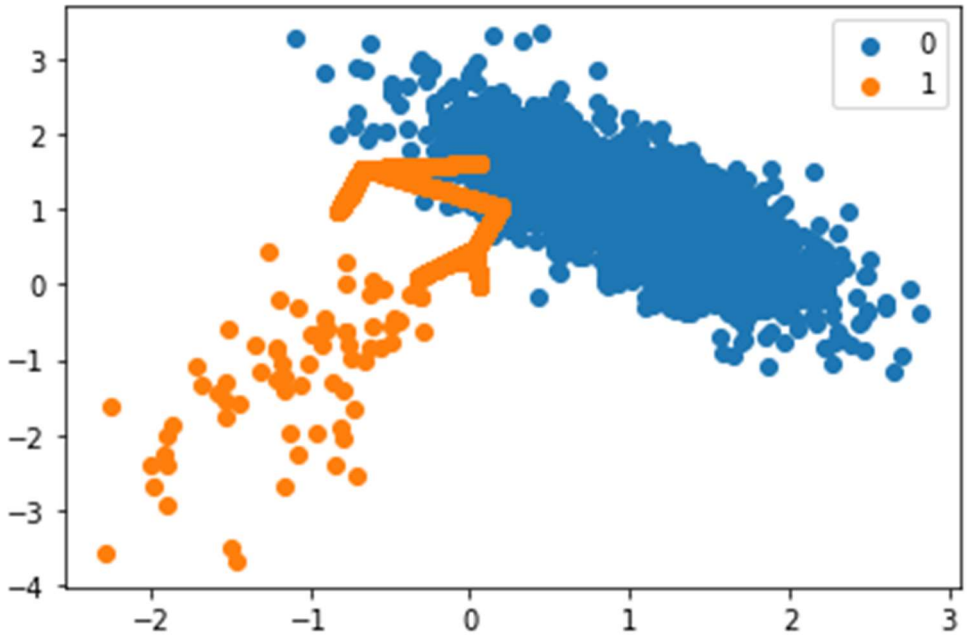


Figure 3. *An illustration of Borderline SMOTE application to the example dataset in Figure 2a. Here, the synthetic data are generated along the border of the two class clusters for the purposes of training the model to better differentiate between similar samples of different classes.*

Typos are problematic with manually entered datasets. In the FCA data, typos in the serial numbers on the mechanical property dataset were encountered. These typos had to be corrected since the HPDC process data and the mechanical property data were to be merged on the serial number and it was important to retain as much of the mechanical property dataset as possible. The erroneous serial numbers were identified by reviewing only the serial numbers from the tensile bars which were not found in the HPDC process database. A needle in the haystack search for humans that takes a computer seconds to perform. The next step is manual, but the number of instances was not daunting. The serial number is a code which contains the Julian day, year, model letter designation, die casting machine number, die cavity number, work shift, and shot number. Errors could be identified such as transposed digits based on limited possibilities and date time information about that part. If the instances were sufficiently large, a script could be written to perform the same tasks and typos could be autocorrected. If the serial number could not be corrected with a high confidence, the row was omitted from the combined process and property dataset.

Cleaning the data is often an iterative process as downstream operations will fail if the data is not organized correctly. Upon cleaning, the data sets the dimensions are changed as shown in Table III.

Table III. Effect of cleaning on the data set shape (*n*-rows x *n*-columns).

Dataset Name	Raw Dataset	Cleaned Dataset	Combined Dataset
HPDC Process	956,986 x 109	954,313 x 95	1,485 x 140
Alloy Composition	980 x 17	933 x 17	
Tensile Testing	1,634 x 14	1,623 x 15	

Observation count drops as rows with missing data or duplicated data are removed. The number of features can increase as combinations of variables are added such as the calculation of the Quality Index [32] in the tensile test data set. On the other hand, features are removed when they do not contain variation or are otherwise deemed unimportant to the analysis. Therefore, the resulting number of features after working with the data may increase or decrease. The combined data set is the result of matching the data sets first on serial number between the HPDC data and the tensile testing data then on machine number and date of the alloy composition data set.

Supplement *i*. Libraries, modules, and algorithms used in this example along with citations to assist the reader in accessing them. The purpose served by each is given in the Notes column.

Library.module	Algorithm	Notes	Reference
pandas		Use: Dataframe management	[7]–[9]
sklearn.datasets	make_classification	Use: Create sample data clusters for SMOTE demonstration n_samples = 2000 n_features = 2 n_clusters_per_class = 1 weights = 0.96	[15], [16], [28]
imblearn.over_sampling	SMOTE BorderlineSMOTE	Use: Generate synthetic data in the minority class. Default parameters used.	[27], [29]–[31]
matplotlib	pyplot	Use: Scatter plots for SMOTE visualization	[13], [14], [33]

Continuous vs Discrete Data

In the dataset, there are continuous variables such as melt temperature, fast shot velocity, and intensification pressure. Likewise, there are discrete variables such as machine ID, cavity number, and work shift. Continuous variables fit nicely into machine learning algorithms which base predictions on the distance between two values. However, discrete data, especially discrete data which is represented by numeric identifiers such as those listed above, cannot be properly characterized by finding the difference. Even so, it is useful data and can be incorporated into the analysis. Using *toydata*, the categorical feature, *machine*, is a good example for illustrating how to deal with discrete data. In Table II, the column for machine identifier contains ones, twos and threes. Many machine learning algorithms struggle with this type of data entry as it is computing a distance between numerical machine identifiers which it interprets as continuous data. While the distance between 1 and 2 and the distance between 2 and 3 both have a value of one, the pair 1 and 3 have a distance of two (*Figure 4*). This is interpreted as parts cast on machines 1 and 3 are less alike than parts cast on machines 1 and 2. Knowing that all three machines are identical, machine 1 and machine 2 are no further apart than machine 1 and machine 3. We need data representation which the algorithm will correctly interpret.

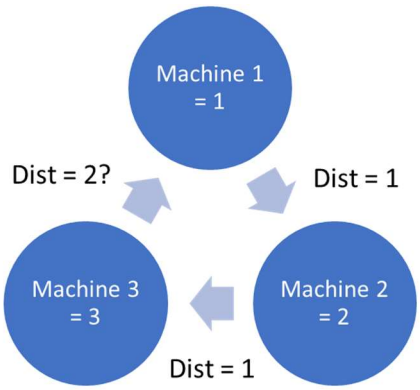


Figure 4. *Discrete numerical data challenge of machine identifier.*

To deal with the challenge of discrete data, data scientists utilize a method known as one-hot encoding [34]. One-hot encoding takes the tall vector which has discrete machine number data consisting of 1's, 2's, and 3's and converts it into a wide set of dummy variables. Call them machine_1, machine_2, and machine_3. Where machine has a value of one (1), machine_1 also has a value of one (1) while machine_2 and machine_3 both have a value of zero (0) (Table IV).

Table IV. After one-hot encoding the *machine* variable, *toydata* has three new columns of machine ID data each specific to one machine. The original machine column is dropped from the dataset.

BiscuitLength	CavityFillTime	FinalIntensifierPressure	SlowShotVelAve	SprayRobotTime	MetalTemp	LadlePourTime	machine_1	machine_2	machine_3
2.6	103	6379	4.3	61.3	1217	11.97	1	0	0
2.6	103	6453	3.9	54.97	1217	11.42	1	0	0
2.5	102	6473	3.6	58.33	1216	10.85	1	0	0
2.7	103	6440	3.8	62.43	1222	11	1	0	0
2.7	105	6343	3.9	63.62	1219	11.14	1	0	0
2.6	103	6421	3.9	56.22	1235	11.5	1	0	0
2.5	104	6314	4.1	58.4	1219	10.82	1	0	0
2.6	105	6460	3.6	55.07	1219	11.3	1	0	0
2.7	104	6356	4	60.6	1224	10.82	1	0	0
2.6	102	6541	4.2	56.65	1224	10.52	1	0	0
2.9	106	6446	2.4	59.03	1232	9.02	0	1	0
2.5	105	6432	2.3	58.72	1229	11.95	0	1	0
2.5	103	6525	2.4	61.9	1234	9.12	0	1	0
2.5	105	6640	2.9	59.55	1221	9.27	0	1	0
2.6	104	6336	2.4	58.15	1225	12.59	0	1	0
2.7	105	6484	2.6	60.5	1232	12.62	0	1	0
2.9	107	6397	2.4	61.8	1229	9.15	0	1	0
2.8	102	6522	2.4	57.88	1235	9.7	0	1	0
2.7	107	6358	2.5	61.33	1222	9.62	0	1	0
2.5	104	6398	2.4	59.35	1215	11.12	0	1	0
2.7	103	6269	5.6	57.7	1230	9.35	0	0	1
2.6	101	6365	6.1	59.35	1225	10.92	0	0	1
2.7	101	6468	4.5	55.67	1231	11.5	0	0	1
2.6	100	6423	4.3	55.4	1228	11.69	0	0	1
2.5	101	6339	5.8	59.22	1223	11.42	0	0	1
2.8	101	6298	4.8	54.97	1227	11.62	0	0	1
2.8	100	6265	5.7	58.95	1224	11.35	0	0	1
2.8	102	6350	5.2	56.2	1230	11.32	0	0	1
2.5	100	6284	5.3	57.95	1230	11.75	0	0	1
2.5	99	6445	4.3	54.53	1212	11.65	0	0	1

By employing one-hot encoding we now have three columns which capture the machine identifier as numerical data and the distance between each machine is one. The original machine data column is

removed from the input dataset prior to running the algorithm. One-hot encoding can be applied to character string data as well.

Supplement *ii*. Libraries, modules, and algorithms used in this example along with citations to assist the reader in accessing them. The purpose served by each is given in the Notes column.

Library.module	Algorithm	Notes	Reference
pandas		pandas.get_dummies() command used for one-hot encoding. Within the parentheses, one can specify the dataframe and which columns to convert.	[7]–[9]

Standardizing the Data

Once the data set is fully numeric and discrete variables have been managed, the issue of scale is addressed. The data collected in high pressure die casting contains a wide range in scale. Also, different equipment manufacturers may capture data in only English or metric units. In round numbers, intensification pressure of 10,000 psi, melt temperature of 1300 F/ 704 C, cycle time of 150 seconds, biscuit size of 2 inches, and an iron content of 0.60% are a few examples which show the range of scale is in orders of magnitude. If left in this format, the intensification pressure would register as highly significant and outweigh any influence the iron content would show simply because the numbers are larger. Recent developments in high pressure die casting alloys show that iron content is very influential on mechanical properties [35] and this significance would perhaps not come to light because of the issue of scale. The standardization method which was employed in this study is the Z-transform (*Equation 1*), which brings all the variables into the same scale, resolves the issue of units, and leads to meaningful distances when considering multiple columns of data [36], [37]. Table V shows the Z-transformed *toydata*.

$$Z_{i,j} = \frac{X_{i,j} - \mu_j}{\sigma_j} \qquad \textit{Eq. 1}$$

Where:

- $Z_{i,j}$ is the Z-transform value in the *i*th row of the *j*th column
- $X_{i,j}$ is the original value in the *i*th row of the *j*th column
- μ_j is the mean of the original values in the *j*th column
- σ_j is the standard deviation of the original values in the *j*th column

An advantage of the Z-transform is that anomalies are easy to detect upon performing a summary of the standardized data set. A summary of the standardized data set will reveal anomalous values. For this data set, variable with maximum standardized values greater than 10 were examined. Large standardized values for a given variable signify that examples of this condition are rare. Machine learning relies on many examples of each condition in order to generate the most accurate model. It is important to investigate these in the original data; applying domain expertise to decide if it should be removed from the data set for the purpose of the analysis. Removing anomalies from the data set does not necessarily mean deleting those rows and never looking back. Anomalies provide information about the fringes of the process window that may prove valuable if more data were to be collected in that space. Once the additional data is collected, then that can be added back into the machine learning model.

Table V. The *toydata* data frame after Z-transformation. Each column is now on the same scale and has an average value of 0 and a standard deviation of 1.

BiscuitLength	CavityFillTime	FinalIntensifierPressure	SlowShotVelAve	SprayRobotTime	MetalTemp	LadlePourTime	machine_1	machine_2	machine_3
-0.3259	0.0000	-0.3301	0.3828	1.1413	-1.2643	1.0270	1.4142	-0.7071	-0.7071
-0.3259	0.0000	0.5281	0.0400	-1.4617	-1.2643	0.4809	1.4142	-0.7071	-0.7071
-1.1406	-0.4841	0.7600	-0.2171	-0.0800	-1.4250	-0.0851	1.4142	-0.7071	-0.7071
0.4888	0.0000	0.3773	-0.0457	1.6059	-0.4607	0.0639	1.4142	-0.7071	-0.7071
0.4888	0.9682	-0.7476	0.0400	2.0953	-0.9429	0.2029	1.4142	-0.7071	-0.7071
-0.3259	0.0000	0.1569	0.0400	-0.9477	1.6286	0.5603	1.4142	-0.7071	-0.7071
-1.1406	0.4841	-1.0839	0.2114	-0.0513	-0.9429	-0.1148	1.4142	-0.7071	-0.7071
-0.3259	0.9682	0.6092	-0.2171	-1.4206	-0.9429	0.3617	1.4142	-0.7071	-0.7071
0.4888	0.4841	-0.5969	0.1257	0.8534	-0.1393	-0.1148	1.4142	-0.7071	-0.7071
-0.3259	-0.4841	1.5486	0.2971	-0.7709	-0.1393	-0.4127	1.4142	-0.7071	-0.7071
2.1182	1.4524	0.4469	-1.2454	0.2078	1.1464	-1.9021	-0.7071	1.4142	-0.7071
-1.1406	0.9682	0.2845	-1.3311	0.0803	0.6643	1.0071	-0.7071	1.4142	-0.7071
-1.1406	0.0000	1.3631	-1.2454	1.3880	1.4679	-1.8028	-0.7071	1.4142	-0.7071
-1.1406	0.9682	2.6967	-0.8169	0.4216	-0.6214	-1.6539	-0.7071	1.4142	-0.7071
-0.3259	0.4841	-0.8288	-1.2454	-0.1541	0.0214	1.6426	-0.7071	1.4142	-0.7071
0.4888	0.9682	0.8876	-1.0740	0.8123	1.1464	1.6724	-0.7071	1.4142	-0.7071
2.1182	1.9365	-0.1214	-1.2454	1.3469	0.6643	-1.7730	-0.7071	1.4142	-0.7071
1.3035	-0.4841	1.3283	-1.2454	-0.2651	1.6286	-1.2269	-0.7071	1.4142	-0.7071
0.4888	1.9365	-0.5737	-1.1597	1.1536	-0.4607	-1.3063	-0.7071	1.4142	-0.7071
-1.1406	0.4841	-0.1098	-1.2454	0.3394	-1.5857	0.1830	-0.7071	1.4142	-0.7071
0.4888	0.0000	-1.6058	1.4968	-0.3391	0.8250	-1.5744	-0.7071	-0.7071	1.4142
-0.3259	-0.9682	-0.4925	1.9252	0.3394	0.0214	-0.0156	-0.7071	-0.7071	1.4142
0.4888	-0.9682	0.7020	0.5541	-1.1739	0.9857	0.5603	-0.7071	-0.7071	1.4142
-0.3259	-1.4524	0.1801	0.3828	-1.2849	0.5036	0.7490	-0.7071	-0.7071	1.4142
-1.1406	-0.9682	-0.7940	1.6681	0.2859	-0.3000	0.4809	-0.7071	-0.7071	1.4142
1.3035	-0.9682	-1.2695	0.8112	-1.4617	0.3429	0.6795	-0.7071	-0.7071	1.4142
1.3035	-1.4524	-1.6522	1.5824	0.1749	-0.1393	0.4114	-0.7071	-0.7071	1.4142
1.3035	-0.4841	-0.6665	1.1540	-0.9559	0.8250	0.3816	-0.7071	-0.7071	1.4142
-1.1406	-1.4524	-1.4319	1.2397	-0.2363	0.8250	0.8086	-0.7071	-0.7071	1.4142
-1.1406	-1.9365	0.4353	0.3828	-1.6427	-2.0679	0.7093	-0.7071	-0.7071	1.4142

To present benefits of the Z-transform graphically, *toydata* with and without Z-transformation are run through a K-Nearest Neighbors (KNN) algorithm. KNN is a supervised learning method highly sensitive to distances [38]. Thus, benefits from standardizing data are readily shown. Importing the KNeighborsClassifier from scikit-learn.neighbors, the machine column from *toydata* is predicted as the target output. The remaining *toydata* columns are the input parameters. Since machine is being predicted, the one-hot encoded machine columns are not included with the Z-transform input data. Due to the small size of *toydata* the model was trained on all the data and predictions were made on the same training data. One should expect a high performing model in this case. *Figure 5* clearly shows the impact of standardizing the data. Without the Z-transform, the model accuracy is poor, misclassifying one-third of the samples. After Z-transform, 90% of the samples are classified correctly.

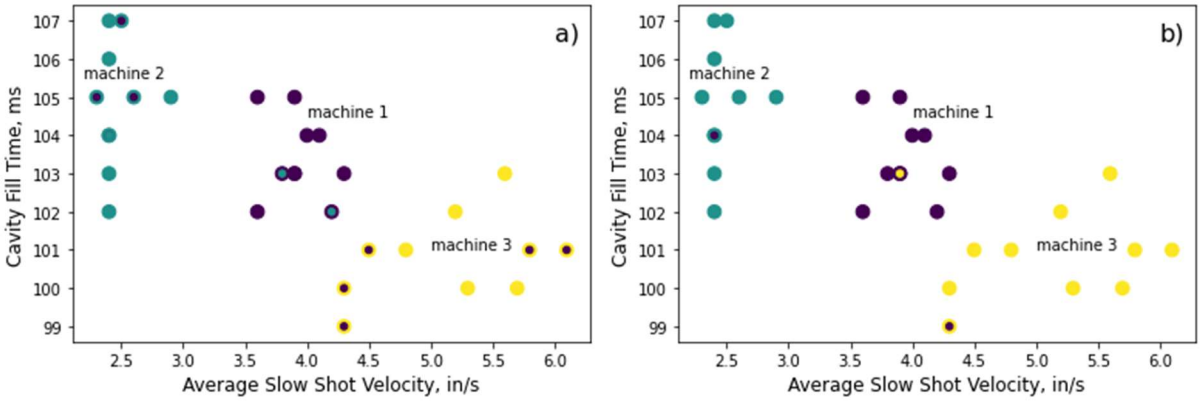


Figure 5. *K-Nearest Neighbor classification of machine identifier a) without Z-transform and b) with Z-transform. Each point represents the true and predicted machine class. Misclassified samples display as bi-colored points. The axes were chosen to show separation between the three machines. With the Z-transform, the model performed much better, classifying at a 90% accuracy.*

Supplement *iii*. Libraries, modules, and algorithms used in this example along with citations to assist the reader in accessing them. The purpose served by each is given in the Notes column.

Library.module	Algorithm	Notes	Reference
pandas		Use: Dataframe management	[7]–[9]
sklearn.pre-processing	StandardScaler	Use: Applies Z-transform to a dataframe	[15], [16], [39]
sklearn.neighbors	KNeighborsClassifier	Use: Machine learning algorithm n_neighbors = 3 weights = 'uniform'	[15], [16], [40]
matplotlib	pyplot	Use: Scatter plots for KNN visualization	[13], [14], [33]

Correlation Between Variables

Some machine learning algorithms are sensitive to correlation between variables in the data set. The effect of keeping two correlated variables in the data set is that the underlying contribution to variance is doubled. Correlation coefficients are not only an indication of how two variables are related, but also a measure of how strongly related they are (*Equation 2*) [41]. Correlation coefficients range between -1 and 1. Any variable correlated to itself has a coefficient of 1. Therefore, if two variables x_i and x_j have a correlation of, or near, 1 they are highly positively correlated and one of them should be removed from the data. Similarly, coefficients near -1 are highly negatively correlated and one of the variables should be removed. Comparing *Equations 1 and 2*, it can be seen that the data need not be standardized prior to checking for correlation; the values are standardized within the correlation coefficient equation.

$$r = \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{x_{ij}-\mu_j}{\sigma_j}\right)\left(\frac{x_{ik}-\mu_k}{\sigma_k}\right) \qquad \textit{Eq. 2}$$

Where:

- r is the correlation coefficient between variables (columns) x_j and x_k

- n is the number of samples (rows) in the dataset
- x_{ij} and x_{ik} are the ith value in variables x_j and x_k respectively
- μ_j and μ_k are the mean of the values in x_j and x_k respectively
- σ_j and σ_k are the standard deviation of the values in x_j and x_k respectively

For the purposes of this research, variables with a correlation coefficient greater than 0.85 or less than -0.85 were considered correlated enough to remove one of them. A correlation matrix is a comparison tool which displays the correlation coefficients between variables in a dataset. An example of a correlation matrix is shown in *Figure 6*.

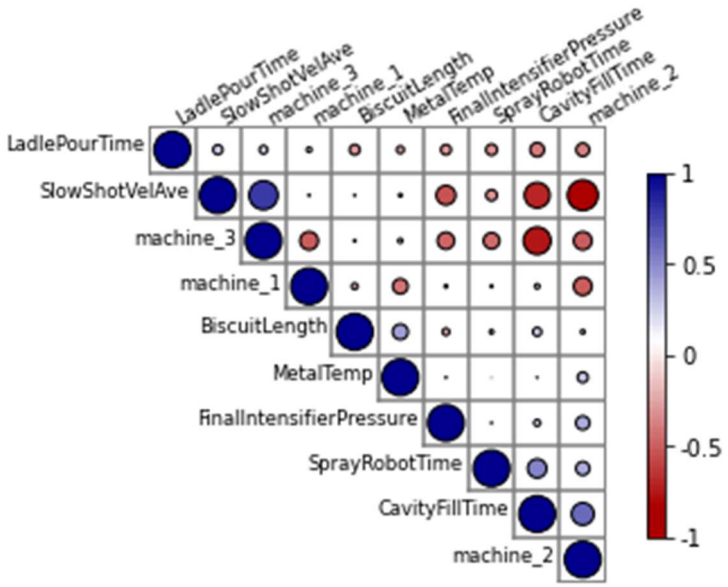


Figure 6. *Correlation matrix of toydata. The amount of correlation is depicted by the size and color of the circle. Blue circles show a positive correlation between two features and red circles show a negative correlation. The size of the circle and depth of color are redundant depictions of the magnitude of the correlation coefficient. Note that correlating any feature to itself yields a coefficient of 1.0.*

Figure 6 shows a strong negative correlation (-0.75) between the categorical feature machine_3 and CavityFillTime. This means that when a casting is run on machine_3, CavityFillTime decreases. This works the other way around too, as CavityFillTime increases, the casting is not run on machine_3.

Supplement *iv*. Libraries, modules, and algorithms used in this example along with citations to assist the reader in accessing them. The purpose served by each is given in the Notes column.

Library.module	Algorithm	Notes	Reference
biokit.viz	corrplot	Use: Plotting correlation matrix	[31]

Dimension Reduction Techniques

Feature Selection:
Feature selection is a dimension reduction technique which is performed by the data scientist much like the name implies. From the high-dimensional dataset, specific features are selected to run through the algorithm. These selections are not made haphazardly. A key resource in deciding which inputs to keep is the domain expert. A domain expert is a specialist in the field from which the data was generated who has reliable knowledge and experience useful to making choices regarding the data to be used. In the absence of domain expertise some machine learning algorithms, like Random Forest, track feature importance in the model generated from the high-dimension data [43]. A new Random Forest model can be created using the top *n* most important features. The actual number of features chosen may be based on an arbitrary number, a minimum feature importance value, or top percentage of features.

Principal Component Analysis (PCA):
PCA is an unsupervised dimension reduction technique [44], [45]. The goal of PCA is to determine linear combinations of the input variables, called principal components (PCs), which capture the most variation in the dataset while minimizing the error when the dataset is reconstructed from the PCs. In doing so, a high-dimensional dataset can be condensed into a smaller number of PCs. Python packages like scikit-learn [15] do all the mathematics behind the scenes, but it is important to understand what the algorithm is doing. The steps for PCA are as follows:

Step 1: Start with the *standardized* dataset *A,* with *n* rows and *d* columns. In data science terms, the *shape* of matrix *A* is *n* x *d*.

Step 2: Calculate the covariance of matrix *A, Cov(A)* [46]. Covariance describes how one variable changes in relation to another. The covariance matrix is given by *Equation 3*. *Cov(A)* is a square matrix, *d* x *d.*

$$Cov(A) = \frac{A^T A}{n-1} \qquad \textit{Eq. 3}$$

Where A^T is the transpose of A; a *d* x *n* matrix.

Step 3: Perform Singular Value Decomposition (SVD) [47] on the $Cov(A)$ matrix (*Equation 4*), and obtain the singular values $(\sigma_1, \sigma_2, \ldots \sigma_d)$ from Σ which are arranged in decreasing order along the diagonal of Σ $(Equation\ 5)$.

$$Cov(A) = U\Sigma V^T \qquad \textit{Eq. 4}$$

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & \sigma_d \end{bmatrix} \qquad \textit{Eq. 5}$$

Where:

- U and V^T are orthogonal matrices [48] where the columns of V^T are the right singular vectors if $A^T A$ is used. In the case where AA^T is used, the left singular vectors are the columns of U [48], [49].
- Σ is a diagonal matrix which contains the singular values, σ_i, of $Cov(A)$

Calculate the explained variance (*Equation 6*).

$$explained\ variance = [\sigma_i / \sum_{i=1}^{n} \sigma] * 100 \qquad Eq.\ 6$$

Step 4: Choose the number of singular vectors, *k*, which yields the desired cumulative explained variance from Step 3; 80-85% is a good minimum value. The value of *k* must be between 1 and *d*. These singular vectors make up a new matrix, *W*, which has the shape *d* x *k*.

Step 5: Project *A* onto the new feature space by taking the dot product of matrix *A* and matrix *W* (*Equation 7*). This new matrix, *Y*, has the shape *n* x *k* where the columns are the PCs.

$$Y = AW \qquad Eq.\ 7$$

Now, the standardized dataset *A* has been reduced from *d*-dimensional space to *k*-dimensional space. In the case where *k* is equal to two or three, the data can be plotted and visually inspected. Also, the new matrix, *Y,* can be input into regression and classification algorithms to create models less susceptible to overfitting with less computational time required to run them.

In Table VI, the PCA transformation of standardized *toydata* (Table V) is shown. In PCA, the number of PCs determined is equal to the number of columns, *d*, in the dataset. The goal is dimension reduction, so a number less than *d* is desired. In the scikit-learn PCA algorithm, the number of PCs can be specified as an integer or the amount of explained variance can be entered as a decimal between 0 and 1 [50]. In this example, the minimum explained variance is specified to be 0.85 and the algorithm yields five PCs (*Figure 7*). Reducing the dimensionality of *toydata* from ten to five. Unlike feature selection, where entire columns are removed, in PCA all the original columns are represented in each of the PCs. Thus, no feature has been removed though the dimensions are reduced.

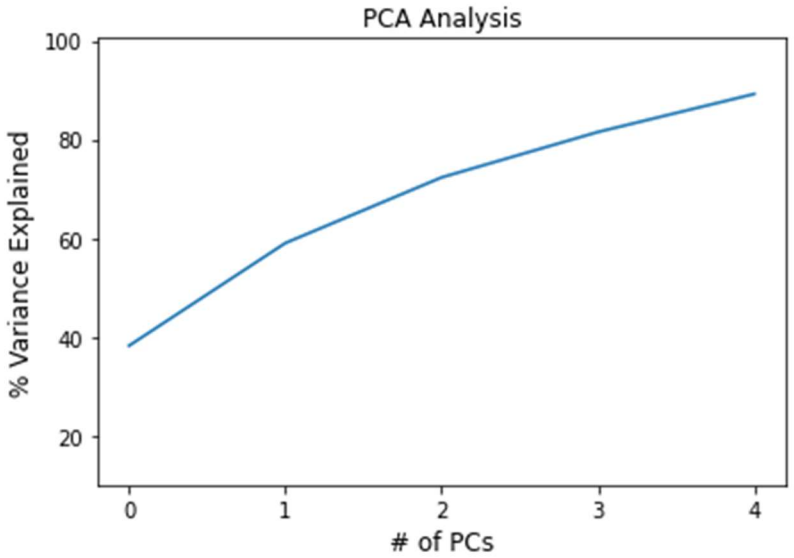


Figure 7. *Plot of the cumulative explained variance as PCs are added in rank order. The first five PCs, in total, explain 89.2% of the variance in the original toydata dataset.*

Table VI. The first five PCs from the *toydata* PCA. Note, there are 10 PCs in the PCA, only the first five are shown as that is what is required to achieve a minimum explained variance of 0.85.

PC1	PC2	PC3	PC4	PC5
-0.3425	2.1937	1.1681	0.5850	-0.1636
-0.5536	2.1344	-0.4662	-0.9768	-0.5579
-0.1246	2.4797	-0.4458	-0.4541	0.6473
0.5593	1.3762	1.3861	-0.4321	0.1943
0.7116	1.5664	2.4890	0.5223	0.1071
-0.2977	0.6253	-0.1563	-1.4142	-1.4152
-0.3287	2.0107	0.8973	0.2770	0.1997
0.0768	2.0234	-0.2823	-1.0395	-0.7444
0.2956	1.0377	1.7499	-0.4186	-0.2037
-0.0962	1.4867	-0.6929	-1.9095	0.2613
3.0641	-2.1458	0.6680	-1.1008	-0.0885
1.7317	-0.0180	-1.3512	1.3757	-1.0718
2.7294	-0.9024	-1.2176	-0.0402	1.4797
2.8224	0.3993	-1.9553	-0.3053	1.8973
0.9743	-0.0720	-0.6589	1.8799	-1.7520
1.9772	-0.6876	-0.6445	0.8904	-1.8119
3.3776	-1.8868	1.6178	-0.1853	0.1793
2.1074	-1.9283	-1.0306	-1.4026	-0.2202
2.8153	-0.6236	1.0145	1.0417	0.5802
1.4926	0.8871	-0.9332	1.9255	0.3432
-1.5888	-1.9684	1.3231	-0.3764	1.1518
-2.2704	-0.7969	0.3001	0.4156	1.1611
-1.7603	-1.3212	-1.0658	-0.9307	-0.4497
-2.2160	-0.8009	-1.3898	-0.2515	-0.2793
-2.4706	-0.2301	-0.0537	1.1388	1.0222
-2.4888	-1.5835	0.3034	-0.2414	-0.9351
-2.6597	-1.4580	1.3428	0.4828	0.1397
-2.0192	-1.7835	0.4377	-0.5664	-0.5905
-2.7955	-0.8532	-0.2610	1.0304	0.1201
-2.7226	0.8394	-2.0926	0.4806	0.7997

The primary reason to use PCA is for dimension reduction which offers benefits of reduced computation time in predictive modeling and the ability to visualize high-dimensional datasets in two or three dimensions. *Figure 8* shows the two-dimensional scatter plot of PCs 1 and 2. As one reads this figure, it is important to recognize that this is a simple scatter plot. PC1 and PC2 are not functions of one another. The main disadvantage of PCA is that it is limited to linear principal components.

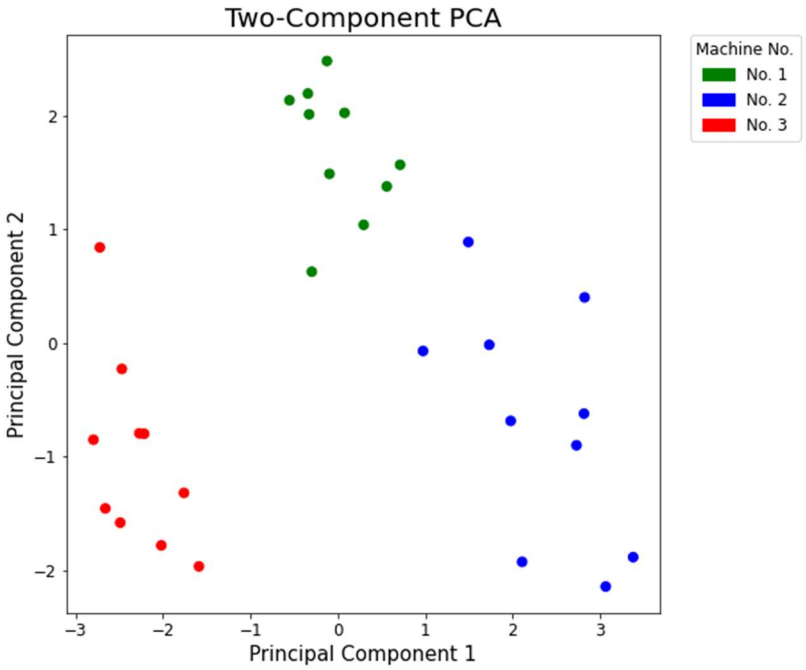


Figure 8. *Two-component PCA. On the toydata dataset, we see a separation of the castings into clusters by machine number.*

Supplement *v*. Libraries, modules, and algorithms used in this example along with citations to assist the reader in accessing them. The purpose served by each is given in the Notes column.

Library.module	Algorithm	Notes	Reference
pandas		Use: Data frame management	[7]–[9]
sklearn.decomposition	PCA	Use: PCA transform the standardized *toydata* dataset n_components = 0.85 svd_solver = 'full' Used pca.explained_variance_ratio_ to calculate cumulative sum of explained variance	[15], [16], [50]
matplotlib	plyplot	Use: Create and customize plots	[13], [14], [33]
matplotlib	patches	Use: Generate custom legend for scatter plot	[13], [14], [51]

Machine Learning Algorithms

Classification vs. Regression Models

When making machine learning predictions the data scientist has two common options at their disposal. The first is to predict that new sample data belongs in a certain category, or class. This is classification modeling. In some cases, the classes may be outputs included in the raw dataset. For example, a classification model is used when one is interested in predicting whether new data is representative of good parts or process scrap. Other datasets require the creation of a new output columns based on the

values of another output column. Here, continuous data can be binned into ranges of values and these bins are now the classes being predicted by the model. Classification models are useful when predicting general outputs like good or bad, greater than or less than, low/medium/high, etc.

Regression models, on the other hand, are used to make specific prediction values for continuous output variables. In regression models, one is looking to predict the actual temperature, material strength, selling price of housing, etc. based on the input data provided.

Over-fitting and the Bias-Variance Trade-Off

There are many machine learning algorithms and neural networks available for analyzing data. When choosing which to implement, the two most important considerations are size of the available data and bias-variance trade-off. The FCA dataset of 1494 tensile tests are exceedingly large when compared to typical mechanical property studies in the literature. However, in the world of data science, this is not Big Data. The amount of data available is a limiting factor in the complexity of the model chosen (*Figure 9*). It is difficult to provide a numerical direction to the reader on what should be considered a small dataset versus a large dataset. A large dataset is one which captures enough examples representative of the most variation within the space one is looking to model.

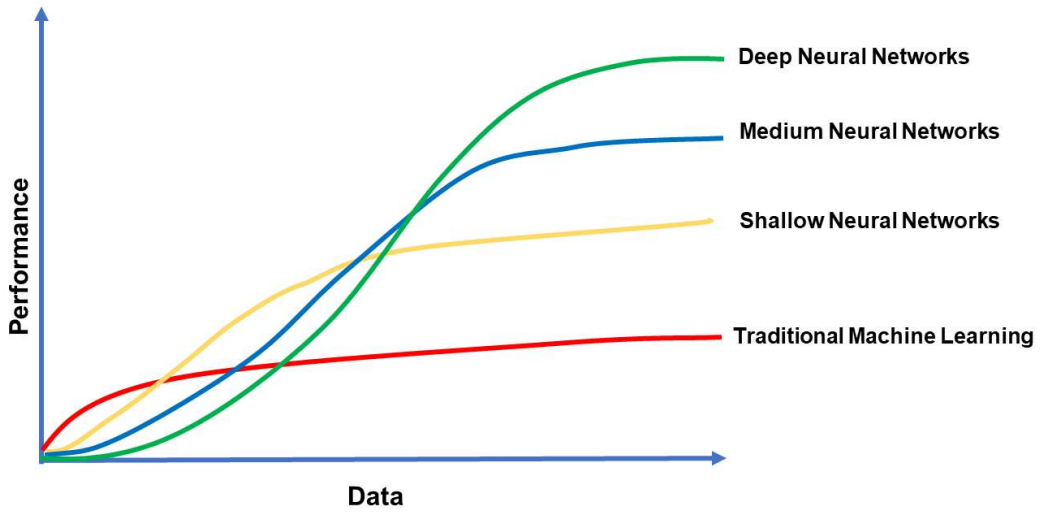


Figure 9. *Performance comparison of Neural Network models with traditional machine learning models as training data size increases. On smaller datasets, traditional algorithms outperform deep learning models however, as the amount of data increases, deep learning models perform better.*

Figure 9 shows a performance comparison of the models as data size increases. For smaller datasets, one would pick traditional machine learning algorithms. However, as the quantity of data increases, deep learning models perform better because traditional algorithms reach a saturation point and do not improve any further whereas deep learning models performance keeps increasing with training data size [52].

Understanding the bias-variance trade-off is essential in deciding which algorithms to select for a given dataset and application. Bias is error in the model driven by the underlying assumptions in the algorithm.

For example, in linear regressions, the bias is high because the algorithm only has linear equations at its disposal to fit the data. Variance refers to the error in the model due to its sensitivity to noise in the training dataset. When variance is high, the algorithm will model the specifics of training set and not be able to generalize to new data. In *Figure 10*, the X-axis shows model complexity and the Y-axis is generalization error. As model complexity increases, variance increases and bias decreases. An increase in the variance causes the model to overfit to the training data and it fails to generalize on new data. The left side of the plot shows a high bias but low variance region. This implies that the model is too simple and, hence, it is highly biased. It fails to learn the complexity of the data. The ideal point is where bias and variance intersect, as shown by the optimum model complexity in the plot below [53].

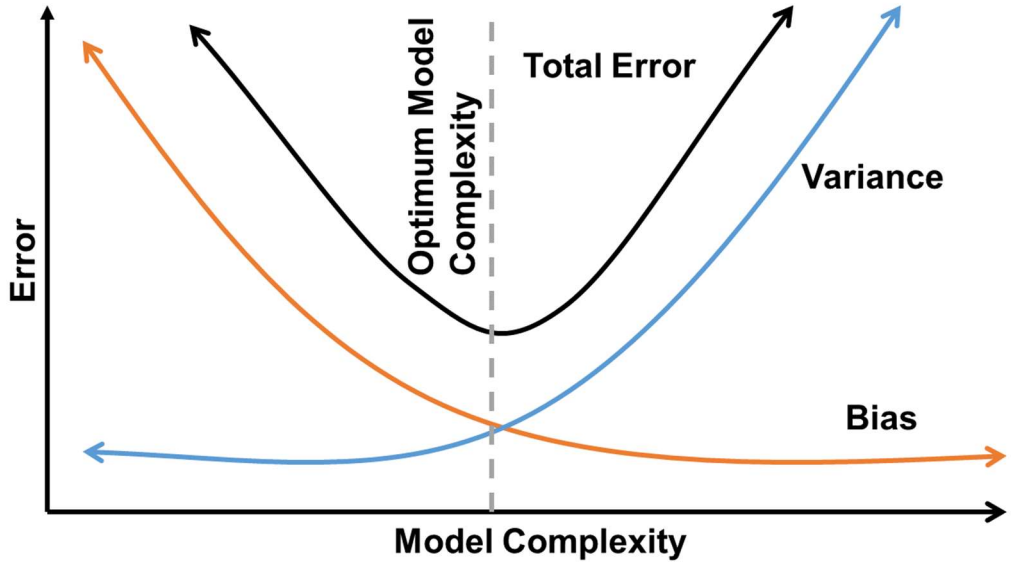


Figure 10. *Bias-variance trade-off* [53], [54] *shows how error changes as the complexity of the model increases. The region on the right is that of high variance and low bias whereas the region on the left is that of high bias and low variance. These regions are where the model overfits or underfits the training data and should be avoided. The optimal model complexity is where variance and bias intersect, and one should utilize algorithms in this region.*

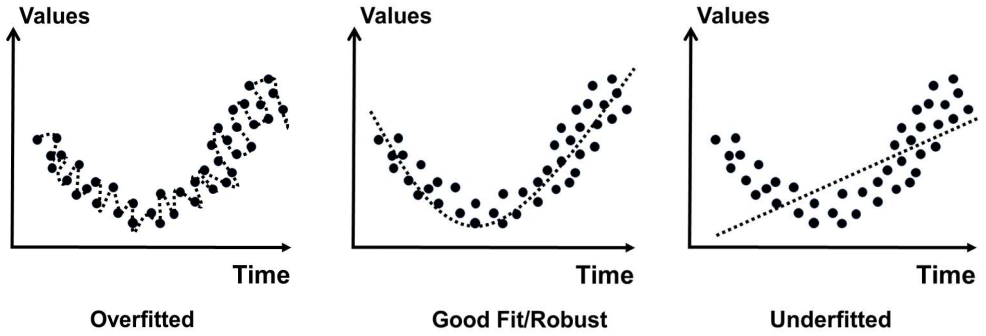


Figure 11. *The phenomenon of underfitting and overfitting is seen in this figure* [55]. *We want a model that is optimal for the kind of data and application that we are exploring. For example, a good fit is illustrated in the center plot. The plots on the right and left show underfitting and overfitting respectively and should be avoided.*

Figure 11 shows the phenomena of overfitting and underfitting. It can be seen in the leftmost plot that the model follows the data very closely and, thus, overfits. This is the region of high variance in the bias-variance trade-off where the model will fail to generalize on testing data because it almost memorizes the training data. The middle figure shows the optimum model which corresponds to the lowest point of bias and variance in the bias variance trade-off and gives a robust fit to the data. The rightmost figure shows an example of high bias in the bias-variance trade-off. Here, the model fails to learn enough complexity in the dataset and underfits [54], [55].

Decision Tree / Random Forest

The Decision Tree is a supervised machine learning method with known outputs applicable to classification and regression problems [56]. Decision Tree classifiers build a predictive model by evaluating the input variables and sorting the observations at various nodes into classes. The nodes split, forming branches of the tree which terminate at a node which does not split, a leaf. Each split creates another layer of depth in the model. Without placing restrictions on the model, the sorting will continue until each branch of the tree ends at a pure leaf consisting of one class. While this results in a high scoring model, overfitting to the training data is highly likely and the model will not score well on new data. This propensity to overfit is a by-product of what makes the Decision Tree so advantageous to materials manufacturing problems. The algorithm is not limited to a functional form. Rather, it evaluates each feature and makes decisions on where to split based on the information gained by doing so.

To combat misclassification, an improved model which uses the results from hundreds, or thousands, of Decision Trees to make predictions is called Random Forest [43], [57]. Random Forest is an ensemble learning, or prediction by committee, approach where the observations are randomly broken into subsets and built into trees splitting on a random subset of the features. The predictions of many trees built from the training data are compiled to make a final prediction for each observation. In a classification model, group voting among the trees is conducted to determine the predicted class. The result of the group vote can be better than what any one of the trees would determine on its own.

To explain Random Forest, we apply the scikit-learn RandomForestClassifier model to the *toydata* database [15], [58]. To have enough data to split into training and testing sets, *toydata* was expanded to 99 samples. Supervised learning methods require an output by which to train the data, so we will attempt to predict the machine (1, 2, or 3) which made the casting based on the process inputs. Due to the small data size, a 90/10 training to testing data split was performed. For the Random Forest classifier itself, default parameters were used with the exceptions of setting the number of estimators (trees) to 1000, maximum features to consider at each node to 3, and maximum samples per estimator to 12.

Once the Random Forest algorithm has been run, the trees created on the training data can be viewed to understand upon which features the cuts were made. A sample tree from a Random Forest classifier is shown in *Figure 12* followed by an explanation of the details behind the Random Forest algorithm. In *Figure 12*, each node has data associated with it. First, on nodes where a split is made, the criterion for the split is shown. This information is not applicable for the leaf nodes. Next, a Gini index is given, which is a measure of the purity of the node [59]. Gini is followed by the number of samples in the node. The value shows three numbers in brackets which represent the class make-up of the samples in the node. Lastly, the class to which the node is assigned is displayed as determined by the majority class in the node.

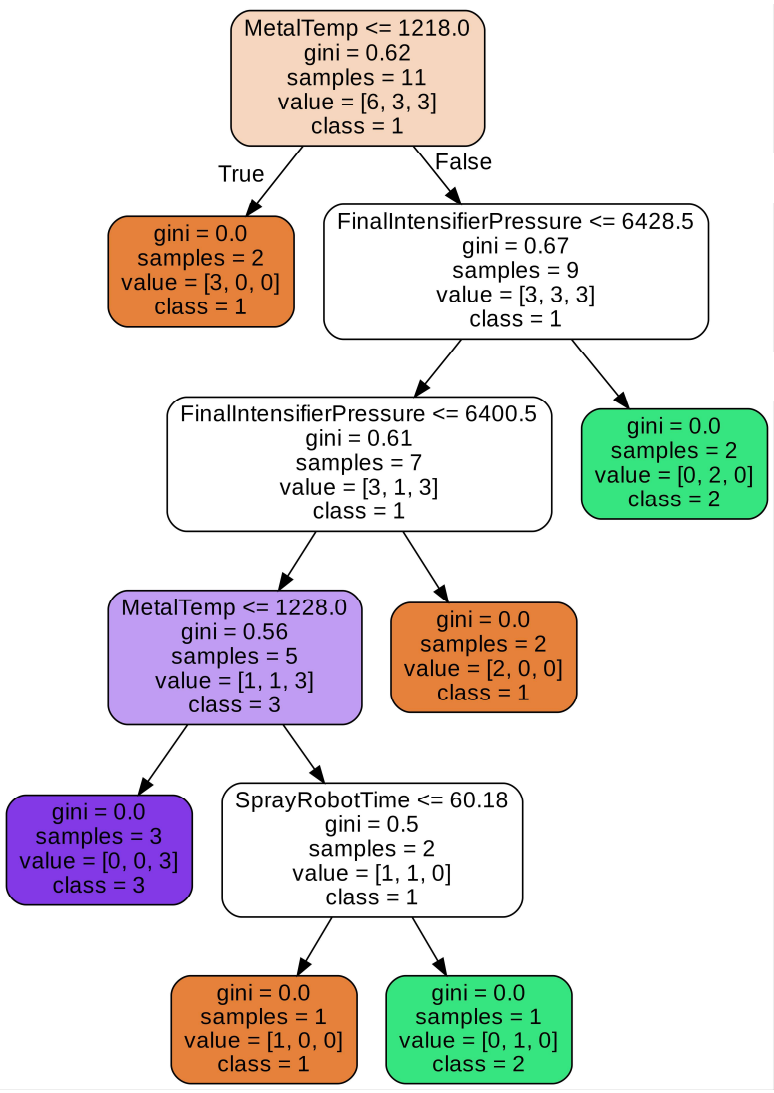


Figure 12. *Sample tree from a Random Forest model of the toydata dataset. In this example, 1000 trees were used to train the algorithm with maximum features set to a value of 3 and maximum samples set to 12. Note how each leaf is pure as no restrictions were placed with respect to minimum samples per leaf or maximum depth of the tree.*

In the scikit-learn RandomForestClassifer, Gini index is the default calculation upon which the model splits a node [60]. Gini has a value between zero and one. The value indicates the probability of misclassifying an observation. A Gini of zero represents a pure node where all the observations are of the same class, no split is made. A high Gini value means that the various classes are mixed and there is a high probability that a new observation would be misclassified. The equation for Gini is given in *Equation 8*.

$$Gini = 1 - \sum_{i=1}^{n} P_i^2 \qquad Eq.\ 8$$

Where:

- $Gini$ is the Gini index
- n is the number of classes
- P_i is the probability of finding each class in the node

For a split, the Gini index is calculated for each feature with respect to the observations in the node. The split is made on the feature which results in the lowest Gini values in the resulting child nodes. This is repeated down the tree until a stopping criterion is met. Examples of stopping criteria include:

- All the leaf nodes are pure
- All the leaf nodes are sufficiently pure by setting a predetermined minimum Gini index
- All the leaf nodes contain too few samples to split by setting a *minimum samples to split* value.
- The maximum depth of the tree is met by setting a *maximum tree depth* value.

Classification Random Forests can be summarized in a confusion matrix [61]. The confusion matrices in *Figure 13* show how the model performed on the training data and the testing data. Across the matrix rows are the actual classes and the columns are the predicted classes. Correct classifications reside along the diagonal. With no restrictions placed on training the model, the training trees terminate at pure nodes and *Figure 13b* shows a very nearly perfect model. On the testing data (*Figure 13c*), however, the model demonstrates it has overfit to the training data and does not perform quite as well.

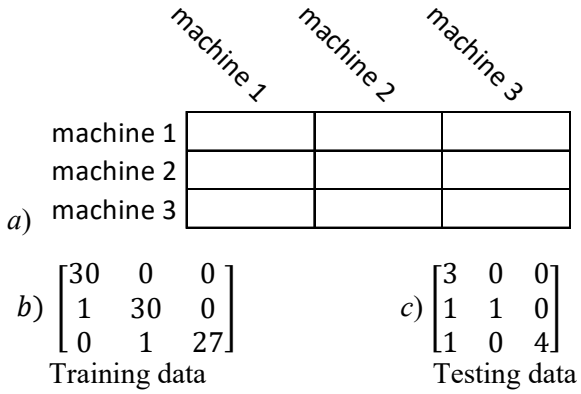


Figure 13. *a) depicts the row and column layout of the confusion matrices for the toydata Random Forest: b) the training data and c) the testing data. The testing data shows two misclassified observations where one machine 2 and one machine 3 sample are both classed as machine 1.*

Thus far, we have focused on using Random Forest as a classification model. Random Forest can build regression models to predict continuous variables as well. Scikit-learn RandomForestRegressor operates in a similar way as the classifier [62]. Rather than Gini index, the regressor uses error metric reduction (mean squared error or mean absolute error) as the splitting criterion to create the purest nodes as the tree is constructed. At the leaf nodes, an average value for the output of interest is calculated from the known output values of the samples within that leaf. For the final reported value, the predictions of the observations from all the trees are averaged.

A helpful output that Random Forest offers is *feature importance* [63]. This report resonates with materials processing engineers because it boils the model down to buttons and knobs that can be pressed and turned to conduct experiments. Data-driven decisions of which parameters to consider for a design of experiment. The alternative is hard earned experience. When the two line up, the team can be confident that they have now captured knowledge from data. When they diverge, the opportunity to move away from long held beliefs and feelings which have yet to produce solutions is opened to the group. The feature importance

from the *toydata* example Random Forest is given in Table VII. The results show that the most distinguishing feature between machines 1, 2, and 3 is the average plunger velocity during slow shot.

Table VII. Rank ordered feature importance from the *toydata* Random Forest example

Feature	**Importance**
Average Slow Shot Plunger Velocity	0.343
Cavity Fill Time	0.210
Spray Robot Cycle Time	0.134
Ladle Pour Time	0.119
Final Intensification Pressure	0.083
Molten Metal Temperature	0.073
Biscuit Length	0.038

In addition to feature importance, Random Forest models are advantageous for their ability to handle high-dimensional data, capability of learning complex non-linear relationships, and the lack of dependence on a particular distribution of the data. The main drawback is the tendency to overfit the training data. Without placing limits on the model, Random Forest will learn according to the specifics of the training data and not be general enough for new data fed into the model. Another weakness of Random Forests is a sensitivity to irrelevant and noisy data. An example of a very noisy input in the FCA data are the timers associated with the opening and closing of slides in the die. Slides are tooling cavity components that form features in the part which would be undercut in a purely open/close die configuration. Slides pull in directions perpendicular to the parting direction of the tool halves. Timers measure how many seconds it takes from when a signal to move the slides is given until the slide meets a limit switch indicating that it is open or closed. This data becomes noisy due to how hydraulics work in a scenario when two slides are ordered to be moved at once. In short, when two slides are activated, whichever slide takes the least amount of force to start moving will open fully before the second begins to move. If it takes t seconds to open each slide, the timer data for Slide 1 is t or $2t$ depending on whether Slide 2 is $2t$ or t respectively. Any bit of debris or metal flash can be the reason one slide moves ahead of the other and results in a noisy variable. Once a split is made, there is no going back, and the trees continue to build themselves. The split on a piece of irrelevant data effects the rest of the model. Careful consideration of the data input in the model must be taken to avoid this scenario. These slide timers were ultimately removed from the data prior to running algorithms.

The Random Forest itself is an improvement on the Decision Tree, since the weaknesses listed above are more severe when utilizing a single tree with one final outcome. Nonetheless, employing Random Forest alone does not guarantee the elimination of overfitting. Tuning the hyperparameters, the settings within the algorithm, aids in reducing overfitting. The model can be tuned to be a more general predictor by limiting the maximum depth of the tree, the maximum number of features evaluates at each node, and setting a minimum sample required to split a node. Random Forest classification and regression models are implemented in this research to predict quality data and ultimate tensile strength respectively.

Supplement *vi*. Libraries, modules, and algorithms used in this example along with citations to assist the reader in accessing them. The purpose served by each is given in the Notes column.

Library.module	Algorithm	Notes	Reference
pandas		Use: Dataframe management	[7]–[9]
sklearn.model_selection	train_test_split	Use: Make train and test datasets test_size = 0.1 random_state = 0	[15], [16], [64]
sklearn.ensemble	RandomForestClassifier	Use: Machine learning algorithm n_estimators = 1000 bootstrap = True max_samples = 12 max_features = 3	[15], [16], [58]
sklearn.metrics	confusion_matrix	Use: Create confusion matrix to assess algorithm performance	[15], [16], [65]
sklearn.metrics	classification_report	Use: Reports accuracy, precision, recall, and f1-score for each class	[15], [16], [66]
sklearn.tree	export_graghviz	Use: Tree visualization. Export tree as .dot file	[15], [16], [67]
subprocess	call	Use: Convert .dot to .png	[68]
IPython.display	Image	Use: Display tree in Jupyter notebook	[69]

Support Vector Machine (SVM)

Like Random Forest models, Support Vector Machines can also be used as classification or regression models. SVMs look to separate classes of data by determining a decision boundary between the populations with the widest separation between them (*Figure 14*). This decision boundary and its associated margins can be linear or non-linear. The boundary is an (n-1)-dimensional subspace for an n-dimensional space. For example, the hyperplane for one-dimensional data is a point, for two-dimensional data a line, for three-dimensional data a plane, and for more than three dimensions and the term *hyperplane* is used [70]. The margin hyperplanes are determined by the observations closest to boundary. The boundary is optimized when the margin is maximized. Simply put there are many hyperplanes which can be drawn to separate the data; the algorithm selects the one in the middle.

When data populations are not linearly separable by a hyperplane, soft margins or a kernel trick can be used (*Figure 15*). Soft margins allow for some misclassification of observations while maintaining a wider margin. The user determines the allowance of misclassifications in the model. Kernel tricks seek a non-linear decision boundary by projecting the data into a higher dimensional space where the examples are linearly separable. Kernels are selected and tuned in the code to optimize the SVM. In scikit-learn SVM [71], the soft margin is controlled by the regularization parameter, C, and the kernel is tuned by the parameter gamma. A large C value will result in a narrower margin for fewer misclassifications, but the risk of overfitting increases. The gamma value determines how wiggly or meandering the boundary can be [72]. Higher gamma values create more winding boundaries; however, one must guard against overfitting when adjusting the gamma parameter.

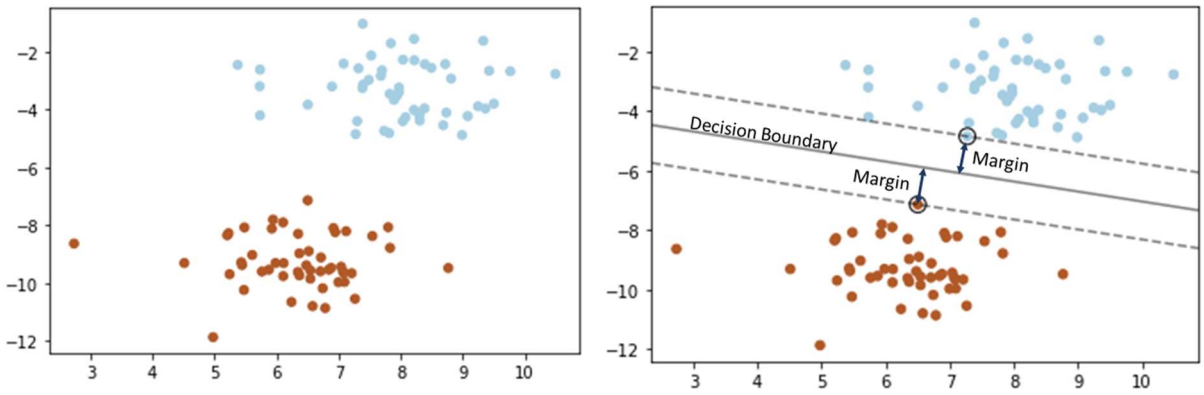


Figure 14. *Support vector classifier example with linearly separable populations. Class predictions are made based on which side of the hyperplane the sample lies.*

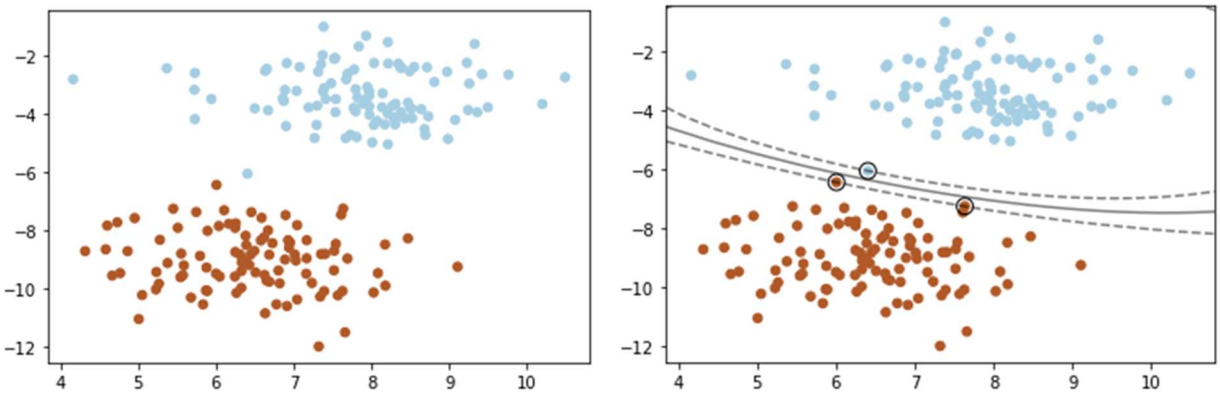


Figure 15. *Support vector classifier example with non-linearly separable populations.*

The Radial Basis Function (RBF) kernel trick was used in an SVM regressor to predict ultimate tensile strength in this work [73]. For the regressor SVM (SVR), the objective is to predict continuous values. This is accomplished by setting an allowable error using the epsilon parameter. The model seeks to construct a line with margins set by epsilon such that the most points fall between the margins [74]. Thereby, the error, or length of the support vectors, between the actual values and predicted values is reduced. The C parameter provides slack to the model to better fit the data by allowing more data to fall outside the margin. A plot of an example SVR is shown in *Figure 16*. Here, 25 random samples are selected from the combined HPDC process and tensile testing dataset. An SVR was run to predict the UTS based on cycle time. Over these 25 samples, the results show that as cycle time increases one can expect a decrease in UTS.

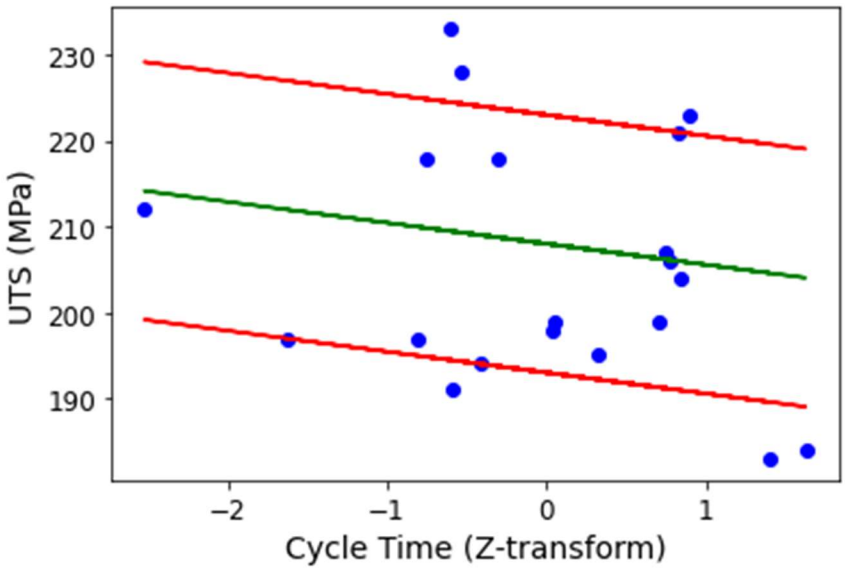


Figure 16. *Support vector regression for UTS prediction based on cycle time. The blue points are the actual values. Predictions are made along the green line. Eplison was set to 15 and is shown as the margins in red. The linear kernel was employed and C=1.0. As cycle time increases ultimate strength decreases.*

Supplement *vii*. Libraries, modules, and algorithms used in this example along with citations to assist the reader in accessing them. The purpose served by each is given in the Notes column.

Library.module	Algorithm	Notes	Reference
pandas		Use: Dataframe management	[7]–[9]
sklearn.pre-processing	StandardScaler	Use: Applies Z-transform to a dataframe	[15], [16], [39]
sklearn.svm	SVC	Use: Support Vector Classifier algorithm	[15], [16], [71]
sklearn.svm	SVR	Use: Support Vector Regressor	[15], [16], [73]
matplotlib	pyplot	Use: Create and customize plots	[13], [14], [33]

XGBoost

XGBoost, short for extreme gradient boosting, is an advanced tree based algorithm developed by Chen [75]–[77]. This method has risen in popularity for supervised machine learning due to its computational speed and model performance and can be used to work both classification and regression problems. Bridging the evolution from Random Forests to XGBoost is beyond the scope of this document. The method was chosen to determine how a state-of-the-science tree-based algorithm would compare to traditional Random Forest and SVR. The following brief definitions and references are given to assist the reader. Boosting is a method of combining relatively weak models improving the accuracy overall at the risk of overfitting [78]–[80]. In boosting, the models are built sequentially with prior knowledge of the errors from the preceding models. A modification to boosting is Gradient Boosting, a method which improves the predictive power along the direction of the gradient to optimize the objective [81].

Supplement *viii*. Libraries, modules, and algorithms along with citations to assist the reader in accessing them. The purpose served by each is given in the Notes column.

Library.module	Algorithm	Notes	Reference
pandas		Use: Dataframe management	[7]–[9]
xgboost	XGBRegressor	Use: Machine learning algorithm	[75], [77]

Machine Learning Model Evaluation

Training, Testing, and Cross-Validation

Prior to running an analysis, it is a best practice to divide the dataset into two parts: a training set and a test set. In this study, 80/20 and 90/10 training to test splits were most often employed (*Figure 17*). The goal of the split is to have enough training data to create a strong predictive model while holding back a testing dataset enough to capture the essence of the complete dataset. This allows the data scientist to train the model with a larger dataset then test the performance on a smaller population that the model has not seen.

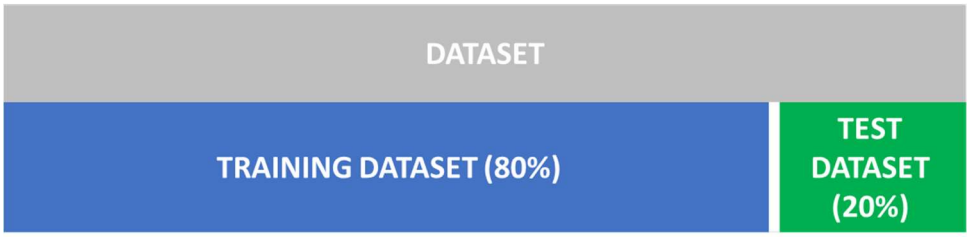


Figure 17. *It is best practice to split the pre-processed data prior to training the machine learning algorithm into training and testing datasets. 80/20 is a good starting point. For the smaller tensile bar dataset, 90/10 was used.*

The train/test split can influence the model. For that reason, cross-validation is conducted to determine how different splits of the data affect the model performance metrics. K-folds is a common method of cross-validation [82]. In K-folds, the user sets the number of folds and the model is run that many times on the training data. Each time, a different segment of the training population is set aside as the test data and run through a model created on the balance of the training data for that fold as seen in *Figure 18*.

Figure 18. *K-folds cross-validation where the number of folds is equal to five.*

The performance of the algorithm can be measured in many ways. Mean absolute error and standard deviation values can be used to score regression algorithms. Accuracy, precision, recall, and f1-scores are often chosen to evaluate classifiers. Regardless of the algorithm, it is common for the error on the training data to be less than the test data. When the difference between the two is large, the model is said to be overfit to the training data. Data scientists are keenly aware of over-fitting because such a model does not generalize. The model shows amazing accuracy on the training data, however, when fed new data, the predictions of the algorithm are unreliable. The goal of a robust model is to minimize the difference in error between the training data and testing data results.

The above methods in pre-processing, algorithm selection, and model evaluation were performed on the FCA HPDC dataset. Algorithms modeled classification of part quality and ultimate tensile strength as well regression predictions of ultimate tensile strength. These studies are published in three technical articles presented in Appendices B, D, and E of this thesis.

References:

[1] J. I. Moore and P. J. Van Huis, "US4493362.pdf," 4493362, Jan. 15, 1985.
[2] "ACRC - Advanced Casting Research Center." https://wp.wpi.edu/acrc/ (accessed Jun. 02, 2020).
[3] R Core Team, *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing, 2013.
[4] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.
[5] "Welcome to Python.org," *Python.org*. https://www.python.org/ (accessed Jul. 12, 2020).
[6] "Microsoft Excel, Spreadsheet Software, Excel Free Trial." https://www.microsoft.com/en-us/microsoft-365/excel (accessed Jul. 12, 2020).
[7] The pandas development team, *pandas-dev/pandas: Pandas*. Zenodo, 2020.
[8] W. McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, 2010, pp. 51–56, Accessed: Jan. 09, 2020. [Online]. Available: http://conference.scipy.org/proceedings/scipy2010/pdfs/mckinney.pdf.
[9] "pandas - Python Data Analysis Library." https://pandas.pydata.org/ (accessed Jul. 12, 2020).
[10] T. E. Oliphant, "Python for Scientific Computing," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 10–20, Jun. 2007, doi: 10.1109/MCSE.2007.58.
[11] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation," *Comput. Sci. Eng.*, vol. 13, no. 2, pp. 22–30, Mar. 2011, doi: 10.1109/MCSE.2011.37.
[12] "NumPy." https://numpy.org/ (accessed Jul. 12, 2020).
[13] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 90–95, Jun. 2007, doi: 10.1109/MCSE.2007.55.
[14] "Matplotlib: Python plotting — Matplotlib 3.2.2 documentation." https://matplotlib.org/ (accessed Jul. 12, 2020).
[15] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011, doi: 10.1016/j.patcog.2011.04.006.

[16] “scikit-learn: machine learning in Python — scikit-learn 0.23.1 documentation.” https://scikit-learn.org/stable/ (accessed Jul. 12, 2020).
[17] “Google Colaboratory.” https://colab.research.google.com/notebooks/intro.ipynb (accessed Jul. 12, 2020).
[18] “Stack Overflow - Where Developers Learn, Share, & Build Careers,” *Stack Overflow*. https://stackoverflow.com/ (accessed Jul. 12, 2020).
[19] “Towards Data Science,” *Towards Data Science*. https://towardsdatascience.com/ (accessed Jul. 12, 2020).
[20] “Kaggle: Your Machine Learning and Data Science Community.” https://www.kaggle.com/ (accessed Jul. 12, 2020).
[21] T. D. Pigott, “A Review of Methods for Missing Data,” *Educ. Res. Eval.*, vol. 7, no. 4, pp. 353–383, Dec. 2001, doi: 10.1076/edre.7.4.353.8937.
[22] M. Pampaka, G. Hutcheson, and J. Williams, “Handling missing data: analysis of a challenging data set using multiple imputation,” *Int. J. Res. Method Educ.*, vol. 39, no. 1, pp. 19–37, Jan. 2016, doi: 10.1080/1743727X.2014.979146.
[23] G. E. A. P. A. Batista and M. C. Monard, “An analysis of four missing data treatment methods for supervised learning,” *Appl. Artif. Intell.*, vol. 17, no. 5–6, pp. 519–533, May 2003, doi: 10.1080/713827181.
[24] A. Kantarci, “Synthetic Data Generation in 2020: in-Depth guide,” *appliedAI*, Jul. 15, 2020. https://research.aimultiple.com/synthetic-data-generation/ (accessed Jul. 17, 2020).
[25] C. Dilmegani, “The Ultimate Guide to Synthetic Data in 2020,” *appliedAI*, Jul. 19, 2018. https://research.aimultiple.com/synthetic-data/ (accessed Jul. 17, 2020).
[26] R. Blagus and L. Lusa, “SMOTE for high-dimensional class-imbalanced data,” *BMC Bioinformatics*, vol. 14, no. 1, p. 106, Dec. 2013, doi: 10.1186/1471-2105-14-106.
[27] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002, doi: 10.1613/jair.953.
[28] “sklearn.datasets.make_classification — scikit-learn 0.23.1 documentation.” https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html (accessed Jul. 17, 2020).
[29] J. Brownlee, “SMOTE for Imbalanced Classification with Python,” *Machine Learning Mastery*, Jan. 16, 2020. https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/ (accessed Jul. 17, 2020).
[30] G. Lemaitre, F. Nogueira, D. Oliveira, and C. Aridas, “imblearn.over_sampling.SMOTE — imbalanced-learn 0.5.0 documentation.” https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTE.html (accessed Jul. 17, 2020).
[31] H. Han, W.-Y. Wang, and B.-H. Mao, “Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning,” in *Advances in Intelligent Computing*, vol. 3644, D.-S. Huang, X.-P. Zhang, and G.-B. Huang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 878–887.
[32] M. Drouzy, S. Jacob, and M. Richard, “Interpretation of Tensile Results by Means of Quality Index and Probable Yield Strength,” *AFS Int. Cast Met. J.*, no. June 1980, pp. 43–50, 1980.

[33] “matplotlib.pyplot.scatter — Matplotlib 3.2.2 documentation.” https://matplotlib.org/3.2.2/api/_as_gen/matplotlib.pyplot.scatter.html (accessed Jul. 12, 2020).
[34] K. Potdar, T. S., and C. D., “A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers,” *Int. J. Comput. Appl.*, vol. 175, no. 4, pp. 7–9, Oct. 2017, doi: 10.5120/ijca2017915495.
[35] R. J. Donahue and G. K. Sigworth, “Die Casting Alloys that will Allow the Die Caster to Compete with Alloys A356, A357, 358 and 359 in PM Applications,” *NADCA Trans. T16-022*, 2016, [Online]. Available: http://www.diecasting.org/transactions/T16-022.
[36] “Z-Transform,” *Wolfram MathWorld*. https://mathworld.wolfram.com/Z-Transform.html (accessed May 26, 2020).
[37] T. Wuest, D. Weimer, C. Irgens, and K.-D. Thoben, “Machine learning in manufacturing: advantages, challenges, and applications,” *Prod. Manuf. Res.*, vol. 4, no. 1, pp. 23–45, Jan. 2016, doi: 10.1080/21693277.2016.1192517.
[38] J. Friedman, R. Tibshirani, and T. Hastie, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.
[39] “sklearn.preprocessing.StandardScaler — scikit-learn 0.23.1 documentation.” https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html (accessed Jul. 12, 2020).
[40] “sklearn.neighbors.KNeighborsClassifier — scikit-learn 0.23.1 documentation.” https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html (accessed Jul. 12, 2020).
[41] R. Taylor, “Interpretation of the Correlation Coeffcient: A Basic Review,” *J. Diagn. Med. Sonogr.*, vol. 6, no. 1, pp. 35–39, 1990, doi: 10.1177/875647939000600106.
[42] T. Cokelaer, *biokit: Access to Biological Web Services from Python*. .
[43] L. Breiman, “Random Forests,” *Mach. Learn.*, vol. 45, no. October 2001, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.
[44] H. Abdi and L. J. Williams, “Principal component analysis: Principal component analysis,” *Wiley Interdiscip. Rev. Comput. Stat.*, vol. 2, no. 4, pp. 433–459, Jul. 2010, doi: 10.1002/wics.101.
[45] S. Wold, K. Esbensen, and P. Geladi, “Principal Component Analysis,” *Chemom. Intell. Lab. Syst.*, vol. 2, pp. 37–52, 1987, doi: 10.1016/0169-7439(87)80084-9.
[46] “6.5.4.1. Mean Vector and Covariance Matrix,” *Engineering Statistics Handbook*. https://www.itl.nist.gov/div898/handbook/pmc/section5/pmc541.htm (accessed Jul. 14, 2020).
[47] G. H. Golub and C. Reinsch, “Singular Value Decomposition and Least Squares Solutions,” in *Linear Algebra*, J. H. Wilkinson, C. Reinsch, and F. L. Bauer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1971, pp. 134–151.
[48] G. Strang, *Introduction to linear algebra*, 4. ed. Wellesley, Mass: Wellesley-Cambridge Press, 2009.
[49] J. Tan, “Parallel Singular Value Decomposition.” http://www.cs.csi.cuny.edu/~gu/teaching/courses/csc76010/slides/SVD%20by%20Jiaxing.pdf (accessed Jul. 13, 2020).

[50] "sklearn.decomposition.PCA — scikit-learn 0.23.1 documentation." https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html (accessed Jul. 12, 2020).
[51] "matplotlib.patches — Matplotlib 3.2.2 documentation." https://matplotlib.org/3.2.2/api/patches_api.html (accessed Jul. 12, 2020).
[52] A. Oppermann, "Artificial Intelligence vs. Machine Learning vs. Deep Learning," *Towards Data Science*, Oct. 29, 2019. https://towardsdatascience.com/artificial-intelligence-vs-machine-learning-vs-deep-learning2210ba8cc4ac (accessed Jun. 08, 2020).
[53] E. Briscoe and J. Feldman, "Conceptual complexity and the bias/variance tradeoff," *Cognition*, vol. 118, no. 1, pp. 2–16, Jan. 2011, doi: 10.1016/j.cognition.2010.10.004.
[54] "Bias-Variance Tradeoff in Machine Learning," *AI Pool*, Oct. 20, 2019. https://ai-pool.com/a/s/bias-variance-tradeoff-in-machine-learning (accessed Jun. 02, 2020).
[55] A. Bhande, "What is underfitting and overfitting in machine learning and how to deal with it," *medium.com*, Mar. 11, 2018. https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-howto-deal-with-it-6803a989c76 (accessed Jun. 08, 2020).
[56] D. Dietrich, B. Heller, and B. Yang, *Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data*, 1st ed. Wiley, 2015.
[57] G. Drakos, "Random Forest Regressor explained in depth," *GDCoder*, Jun. 04, 2019. https://gdcoder.com/random-forest-regressor-explained-in-depth/ (accessed Jul. 12, 2020).
[58] scikit-learn developers, *sklearn.ensemble.RandomForestClassifier*. .
[59] R. I. Lerman and S. Yitzhaki, "A note on the calculation and interpretation of the Gini index," *Econ. Lett.*, vol. 15, pp. 363–368, 1984, doi: 10.1016/0165-1765(84)90126-5.
[60] "3.2.4.3.1. sklearn.ensemble.RandomForestClassifier — scikit-learn 0.23.1 documentation." https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (accessed Jul. 12, 2020).
[61] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Inf. Process. Manag.*, vol. 45, no. 4, pp. 427–437, Jul. 2009, doi: 10.1016/j.ipm.2009.03.002.
[62] "3.2.4.3.2. sklearn.ensemble.RandomForestRegressor — scikit-learn 0.23.1 documentation." https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html (accessed Jul. 12, 2020).
[63] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, "Permutation importance: a corrected feature importance measure," *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, Apr. 2010, doi: 10.1093/bioinformatics/btq134.
[64] scikit-learn developers, *sklearn.model_selection.train_test_split*. .
[65] scikit-learn developers, *sklearn.metrics.confusion_matrix*. .
[66] scikit-learn developers, *sklearn.metrics.classification_report*. .
[67] scikit-learn developers, *sklearn.tree.export_graphviz*. .
[68] Python Software Foundation, *subprocess — Subprocess management*. .
[69] The IPython development team, *IPython*. .
[70] W. S. Noble, "What is a support vector machine?," *Nat. Biotechnol.*, vol. 24, no. 12, pp. 1565–1567, Dec. 2006, doi: 10.1038/nbt1206-1565.

[71] "sklearn.svm.SVC — scikit-learn 0.23.1 documentation." https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html (accessed Jul. 14, 2020).
[72] M. Peixeiro, "The Complete Guide to Support Vector Machine (SVM)," *Towards Data Science*, Jul. 29, 2019. https://towardsdatascience.com/the-complete-guide-to-support-vector-machine-svm-f1a820d8af0b.
[73] "sklearn.svm.SVR — scikit-learn 0.23.1 documentation." https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html (accessed Jul. 14, 2020).
[74] A. Geron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*, 1st ed. O'Reilly, 2017.
[75] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 785–794, Aug. 2016, doi: 10.1145/2939672.2939785.
[76] J. Brownlee, "A Gentle Introduction to XGBoost for Applied Machine Learning," *Machine Learning Mastery*, Aug. 17, 2016. https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/ (accessed Jun. 02, 2020).
[77] xgboost developers, "XGBoost Documentation," *XGBoost Documentation*, 2020. https://xgboost.readthedocs.io/en/latest/index.html (accessed Jul. 14, 2020).
[78] T. G. Dietterich, "An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization," *Mach. Learn.*, vol. 40, no. August 2000, pp. 139–157, 2000, doi: https://doi.org/10.1023/A:1007607513941.
[79] Y. Freund and R. E. Schapire, "Experiments with a New Boosting Algorithm," in *13th International Conference on Machine Learning,* San Francisco, 1996, pp. 148–156.
[80] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997.
[81] J. H. Friedman, "Stochastic gradient boosting," *Comput. Stat. Data Anal.*, vol. 38, no. 4, pp. 367–378, Feb. 2002, doi: 10.1016/S0167-9473(01)00065-2.
[82] M. Sanjay, "Why and how to Cross Validate a Model?," *Towards Data Science*, Nov. 12, 2018. towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f (accessed Jun. 04, 2020).

Table C-I. FCA HPDC process dataset descriptions.

Variable Name	Description	Category
MachineID	Die casting machine on which the part was cast	Input
SerialNumb	Unique identifier given to the casting when it is made. Can be used to merge datasets.	
CreatedDateStamp	Date and Time the casting was produced.	Input
AvgFastHeadPressure	Average pressure reading on the head side of the shot cylinder during fast shot.	Input
AvgFastRodPressure	Average pressure reading on the rod side of the shot cylinder during fast shot.	Input
AvgIntermediateHeadPressure	Average pressure reading on the head side of the shot cylinder during intermediate shot.	Input
AvgIntermediateRodPressure	Average pressure reading on the rod side of the shot cylinder during intermediate shot.	Input
AvgSlowHeadPressure	Average pressure reading on the head side of the shot cylinder during slow shot.	Input
AvgSlowRodPressure	Average pressure reading on the rod side of the shot cylinder during slow shot.	Input
BiscuitLength	The thickness of the biscuit calculated based on the end of stroke position of the shot rod.	Input
CavityFillTime	The time taken to fill the part geometry cavity in the die. Calculated from CavityFillTimeWinStartPos until the end of the shot velocity is detected.	Input
CavityFillTimeWinStartPos	Position programmed by the engineer to begin timing cavity fill time. This is typically the position of the shot cylinder when the metal is at the gates.	Setting
DieCloseTankLevel	Level of the hydraulic fluid reservoir	Input
DieCloseTankTemp	Temperature of the hydraulic fluid reservoir.	Input
DwellTimePre	Programmed time to allow the part to solidify between the end of the shot and the opening of the die.	Setting
DwellTime2Pre	Same as DwellTimePre. For any given row if one was null the other had a value.	Setting
EndofShotPosition	Position where fast shot velocity decelerates to the end of shot velocity.	Input
FastShotVelAve	Calculated average shot velocity at which the plunger moved forward during fast shot.	Input
FastShotVelWinEnd	Position of the shot cylinder when the DCM stops measuring velocity for the average fast shot calculation.	Setting
FastShotVelWinStart	Position of the shot cylinder when the DCM begins measuring velocity for the average fast shot calculation.	Setting
FastSpeedSetting	Programmed plunger velocity during fast shot.	Setting
FastStartPosSetting	Programmed position at which the DCM increases to fast shot velocity.	Setting

FinalIntensifierPressure	Maximum pressure applied to the biscuit during intensification phase.	Input
FinalIntensPressWinEnd	End position for measuring intensification pressure.	Setting
FinalIntensPressWinStart	Start position for measuring intensification pressure.	Setting
IntensificationStroke	Amount of plunger forward movement after intensification is initiated.	Input
IntensPressRiseTime	Time measured to reach the programmed intensification pressure.	Input
IntensPressRiseTimeWinEndPress	Measured pressure at the end of the intensification rise time window.	Input
IntensPressRiseTimeWinStartVelocity	Measured plunger velocity at the start of the intensification rise time window.	Input
IntensStartPosSetting	Programmed position at which the DCM engages the intensifier accumulator.	Setting
IntensVelRiseTime	Calculated average velocity at which the plunger moved forward during intensification rise window.	Input
IntensVelRiseTimeWinEndPOS	End position for measuring intensification rise time velocity (position controlled).	Setting
IntensVelRiseTimeWinEndVEL	End velocity for measuring intensification rise time velocity (velocity controlled).	Setting
IntensVelRiseTimeWinStartPOS	Start position for measuring intensification rise time velocity (position controlled).	Setting
IntensVelRiseTimeWinStartVEL	Start velocity for measuring intensification rise time velocity (velocity controlled).	Setting
IntermediateSpeedSetting	Programmed plunger velocity during intermediate shot.	Setting
IntermediateStartPosSetting	Programmed position at which the DCM increases to the intermediate shot velocity.	Setting
IntermediateVelAve	Calculated average shot velocity at which the plunger moved forward during intermediate shot.	Input
IntermediateVelWinEnd	Position of the shot cylinder when the DCM stops measuring velocity for the average intermediate shot calculation.	Setting
IntermediateVelWinStart	Position of the shot cylinder when the DCM begins measuring velocity for the average intermediate shot calculation.	Setting
MetalTemp	The temperature of the molten alloy in the holding furnace at the die cast cell.	Input
ShotDecelSetting	Not used.	
ShotDecelSpeedSetting	Not used.	
ShotDecelStartPosSetting	Not used.	
ShotDelayTimePre	Programmed delay time between pouring into the cold chamber and initiating slow shot.	Setting
ShotForwardPosSetting	Not used.	
ShotTankLevel	Not used.	
ShotTankTemp	Not used.	

SlowShotVelAve	Calculated average shot velocity at which the plunger moved forward during slow shot.	Input
SlowShotVelWinEnd	Position of the shot cylinder when the DCM stops measuring velocity for the average slow shot calculation.	Setting
SlowShotVelWinStart	Position of the shot cylinder when the DCM begins measuring velocity for the average slow shot calculation.	Setting
SlowSpeedSetting	Programmed plunger velocity during slow shot.	Setting
TieBarTon1	Tons of force measured by the load cell on tie bar #1 when the die is closed and locked.	Input
TieBarTon2	Tons of force measured by the load cell on tie bar #2.	Input
TieBarTon3	Tons of force measured by the load cell on tie bar #3.	Input
TieBarTon4	Tons of force measured by the load cell on tie bar #4.	Input
TieBarTonTotal	Sum of the tonnage of all four tie bars.	Input
TipLubeTimePre	Programmed time for which tip lube is applied to the plunger tip.	Setting
Overflows_OK	Post casting check to determine that all the overflows ejected with the part.	
Trimmed	Post casting log that the part was run on the trim press.	
PartDegated	Post casting log that the part was de-gated.	
Quenched	Post casting log that the part was quenched.	
Reject	Operator override label of DCM label	Output
Scrap	DCM quality label based on specific process parameter values.	Output
WarmUp	DCM quality label based on specific process parameter values.	Output
PinMarked	Post casting log that the part was pin marked (serial ID).	
CycleTime	Elapsed time for the entire process to produce one piece.	Input
PhaseTime00 (Cycle Time)	See above.	Input
PhaseTime01 (Dwell Time)	Elapsed time between end of shot and die open.	Input
PhaseTime02 (Dwell Time 2)	See above. One or the other is active on each row, so the two columns were merged for analysis.	Input
PhaseTime03 (Die Open Time)	Elapsed time to open the die.	Input
PhaseTime04 (Extract Robot Time)	Elapsed time for the extract robot to complete its full cycle.	Input
PhaseTime05 (Spray Robot Time)	Elapsed time for the spray robot to complete its full cycle.	Input
PhaseTime06 (Liner Load Time)	Elapsed time to load cast in liners into the die.	Input
PhaseTime07 (Core Insert Time)	Elapsed time to insert core feature into the die.	Input
PhaseTime08 (Die Close Time)	Elapsed time to close the die.	Input
PhaseTime09 (Ladle Pour Time)	Elapsed time for the ladle to pour molten alloy into the cold chamber.	Input

PhaseTime10 (Shot Delay Time)	Elapsed time between pour complete and shot forward.	Input
PhaseTime11 (Core 1 Insert Time)	Elapsed time to move Core 1 slide forward.	Input
PhaseTime12 (Core 2 Insert Time)	Elapsed time to move Core 2 slide forward.	Input
PhaseTime13 (Core 3 Insert Time)	Elapsed time to move Core 3 slide forward.	Input
PhaseTime14 (Core 4 Insert Time)	Elapsed time to move Core 4 slide forward.	Input
PhaseTime15 (Core 5 Insert Time)	Elapsed time to move Core 5 slide forward.	Input
PhaseTime16 (Core 6 Insert Time)	Elapsed time to move Core 6 slide forward.	Input
PhaseTime17 (Core 1 Pull Time)	Elapsed time to pull Core 1 slide open.	Input
PhaseTime18 (Core 2 Pull Time)	Elapsed time to pull Core 2 slide open.	Input
PhaseTime19 (Core 3 Pull Time)	Elapsed time to pull Core 3 slide open.	Input
PhaseTime20 (Core 4 Pull Time)	Elapsed time to pull Core 4 slide open.	Input
PhaseTime21 (Core 5 Pull Time)	Elapsed time to pull Core 5 slide open.	Input
PhaseTime22 (Core 6 Pull Time)	Elapsed time to pull Core 6 slide open.	Input
PhaseTime23 (Ejection Forward Time)	Elapsed time to move the ejection cylinder forward and free the casting from the die.	Input
PhaseTime24-29	Not Used.	
TubesLoaded	Pre-casting log that cast-in tubes were inserted into the die.	
LinersLoaded	Pre-casting log that cast-in liners were loaded into the die.	
VacuumCheckPosition	Programmed position at which the vacuum pressure on the cavity is measured.	Setting
VacuumPressDuringShot	Measured vacuum pressure.	Input
VacuumPurgeResult	Measured pressure when clearing the vacuum chill block of debris.	Input
LubeVolTotal	Not used.	

Table C-II. FCA mechanical property testing dataset descriptions.

Variable Name	**Description**	**Category**
start_date	Date of the tensile test.	
SerialNumb	Unique identifier given to the casting when it is made. Can be used to merge datasets.	
heat_treat_date	Heat treat date of the block casting.	Input
cavity	Cavity number in which the block was cast.	Input
Bulkhead	Location in the block where the tensile bars was extracted.	Input
diameter_mm	Measured diameter of the tensile bar. Used to calculate material strength.	Input
final_length_mm	Measured final length of the tensile bar after failure.	Output
UTS_MPa	Ultimate Tensile Strength. The peak strength measured prior to failure.	Output
YS_MPa	Yield Strength. Calculated by the testing software as the strength at the intersection of the 0.2% offset of the elastic portion of the stress-strain curve and the measured stress-strain curve.	Output
Elong	Percent elongation. Calculated by dividing the posttest distance between gage markers by the original distance between the gage marks of 25.4 mm.	Output

tensile_strain	Tensile strain measured by the extensometer. This was used in place of percent elongation due to the increased accuracy.	Output
fracture_loc	Operator text notes on the location of fracture in relation to the gage marks. Also visible defects on the fracture surface are noted in this column.	Input
furnace_no	Heat treat furnace number used for the casting.	Input
hardness_BHN	Brinell hardness taken on the tensile bar.	Output

Appendix D – Predicting Quality of Cylinder Block Castings via Supervised Learning Method

Adam E. Kopper[1]
Diran Apelian[2]

1. Mercury Marine, Fond du Lac, WI
2. University of California - Irvine, Irvine, CA

ABSTRACT

The process input data which materials processing operations can collect for each unit of production is extensive. Large datasets have long been difficult to work with as computing power to execute analysis in a timely fashion was unavailable. Further, the great velocity at which the data is generated makes near real-time decision making unwieldy without a new set of tools with which to do the work. When troubleshooting by a small dataset, such as the last few hours of production, observations made on the measured parameters can be misleading. Machine learning is opening doors to high-dimensional data analysis in material processing. In this work, high-pressure die-casting (HPDC) is explored as an exemplar of high-volume materials processing. HPDC process summary data from a full year of production data covering over 950,000 machine cycles is analyzed via supervised machine learning methods to successfully model the prediction of good parts and process scrap as determined by the die casting machine. Additionally, the prediction of ultimate tensile strength via a classification method of extracted tensile bars is performed and the important features identified are discussed. Supervised learning is found to be a useful tool for materials processing applications.

I. INTRODUCTION

Machine learning has been sparked by a simultaneous decrease in cost of computer memory and increase in computing power [1], [2]. While applications such as advertisements, coupon targeting, credit card fraud detection, and streaming media service recommendations are the average person's daily interaction with machine learning, the development of next wave applications is well underway [3]. Artificial intelligence is driving object detection and sign recognition for autonomous vehicles on land and sea [4], [5]. The medical field is using predictive modeling in disease diagnosis which is a game changing technology for rural areas and developing nations where doctors are few and collaboration is limited [6]. Facial recognition is going beyond finding individuals for national security applications to emotion detection from facial pattern recognition [7]–[9]. Machine learning is often associated with advanced computing technology sectors; however, it is of great interest to manufacturing and materials processing industries as well.

Oftentimes, the products of companies in materials processing industries are raw materials for the next operation. This may be in the same facility or at a customers' operation where further value is added in the journey toward the final shape, assembly, etc. Its place in the product pipeline categorizes many materials as commodities and pricing pressures are high. For efficiency, production processes tend to be large scale in terms of production tonnage and units per hour. These processes are often thermally controlled. Once they are running, any interruption has significant quality and downtime implications in getting the process back to operating temperature. In this climate, sampling each unit of the product for the purposes of quality assurance or process control slows productivity, adds cost, or simply is not possible (see *Figure 1*). Even with 100% inspection of the product, analysis of the input parameters is required to gain knowledge and improve the outcome. Machine learning is a toolset which can analyze the input parameters and, in near real time, provide actionable direction on the output product. Thereby, increasing confidence in the product being made without increasing sampling.

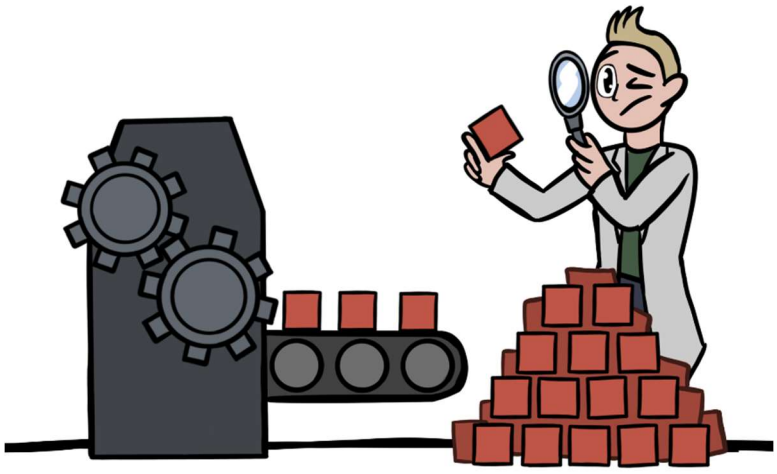


Figure 1 [10]. *In many materials processing operations, the product is generated too quickly for 100% inspection. New tools are required to garner insight into the product between quality checks.*

Early opportunities for implementing machine learning in foundries are entering into the literature. Traditional casting process simulation used for optimizing tool designs and process settings involved a significant amount of trial and error and running modified simulations in series, which is highly time consuming. Autonomous optimization routines have been built into the simulation software which now runs many iterations aimed at improving certain outputs such as fill time, porosity, or air entrapment [11]. Rather than one value for each parameter, the user sets up a range of values to test and the software finds the optimum solution. This capability greatly increases the efficiency of process development. In other applications, companies are monitoring the data they collect to detect anomalous behavior in the process [12]. While no predictions are being made, the idea is that the process is under control and making good quality parts while as the parameters are within a three-sigma variation of their respective means. If a parameter falls out of those

specifications, the part serial number is flagged for additional inspection and the equipment is serviced to determine the root cause and solution. Similarly, machine maintenance is moving away from preventative, where consumables are replaced or service performed on a schedule based on history or recommendations, toward predictive [13], [14]. In predictive maintenance, sensors and meters are applied throughout the equipment in a manufacturing cell to ensure that various machines are maintaining a consistent amperage draw, cycle time, hydraulic pressure, etc. The goal is for the machine to indicate, through data, to the maintenance team when it requires service.

In all these examples, machine learning is implemented to improve the operation and save cost. Autonomous optimization in process simulation increases speed to market and reduces the cost of trial and learn process development. Anomaly detection in process parameters aims to reduce scrap costs through early detection. Predictive maintenance reduces downtime costs through identifying declines in equipment performance. Such cost saving measures are great places to apply machine learning and build the culture in the materials processing industry. Ultimately, materials processing operations are headed toward truly smart factories where machines can correct for their own performance variation given a window in which they can self-adjust. To realize this future, our industry must capture the knowledge of domain experts, build a data science skillset, cultivate a culture of data driven decision making, and begin creating knowledge from the data already being generated in our operations.

So, why has this not happened yet? It turns out that analyzing materials processing data is not straightforward [15]. As materials processing companies bring their data to the data science community to find answers, new insight into how the data is traditionally collected and the challenges which are created thereby are brought to light. Let us look at three of them: a culture of departmental data keeping, collection of many input data and few outputs, and an imbalance in output data class where high quality samples far outweigh unacceptable samples.

The first obstacle is cultural and centers on organizing data for analysis. Data fusion for machine learning is more difficult when the data is stored within operational silos (see *Figure 2*). The type of data and methods used by siloed departments within the same facility have evolved, in isolation, over years. Methods range from high-tech automatic uploading to a cloud database to handwritten records in a logbook. This creates challenges for combining the various sources of data into a cohesive dataset as the collection frequency and identifiers often differ. Communication among stakeholders through the entire process is critical to identify which, how, and how often data should be collected to give the best description of the system to be modeled. A culture of uniformity, traceability, and trust is required to tie the data together in meaningful ways.

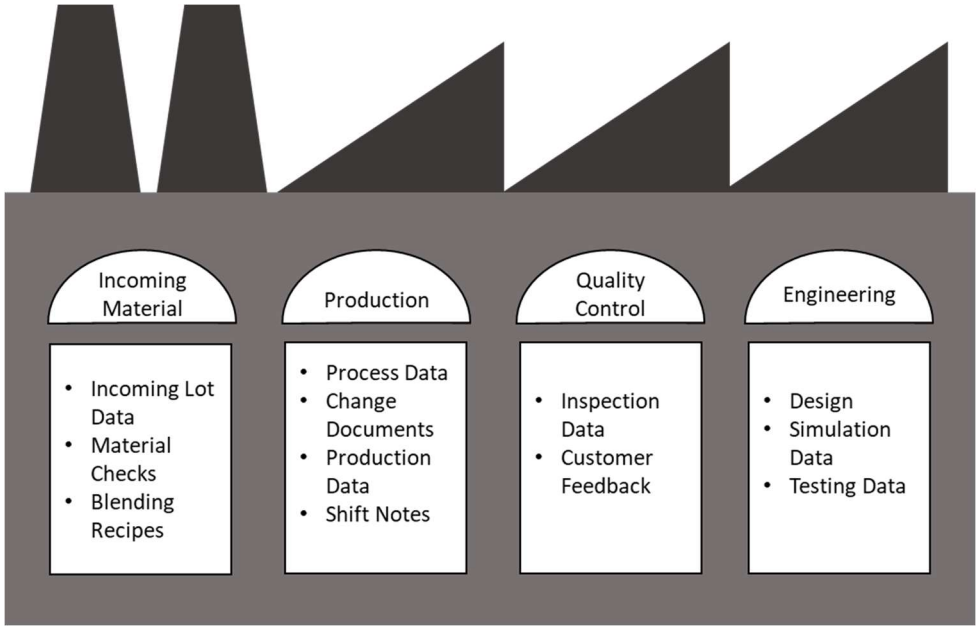


Figure 2. *Departmental data silos are a challenge to implementing machine learning in many materials manufacturing operations.*

The next challenge is specific to machine learning. Materials processes are established and controlled on the input end while output measures are commonly performed on an audit basis. A set of quality checks represents a production lot, or a shift, which occurs over several hours. Thus, there is a wealth of data on the process inputs and timeseries records, but scant output data. Similarly, there is missing process input data. Owing to the method of data collection, some inputs are audited as well. In other example, a non-essential monitoring sensor fails, and maintenance is not available to replace it for some time, creating a gap in data collection for a specific measurement. Collecting data on different frequencies results in heterogeneous data where certain data exists for a subset of the population, but not all the population. How do work with instances of missing data require the assistance of subject matter expert. Working with heterogeneous data of interest to many researchers in Data Science [16]–[18].

Lastly, manufacturers are very capable at what they do. Well-developed manufacturing operations are looking for improvements in production yields, for example, from 96% good product to 97%. There are two difficulties here. The first is the dataset is highly imbalanced where there are many more good products than nonconforming (*Figure 3*). Robust machine learning algorithms need to be trained on both. In the example given above, if the model predicts all the production to be good, it would be 96% accurate but, in effect, unhelpful. The second difficulty is that very large amounts of data are required to know if a small improvement in predictability is real or a result of error and noise in the data [19].

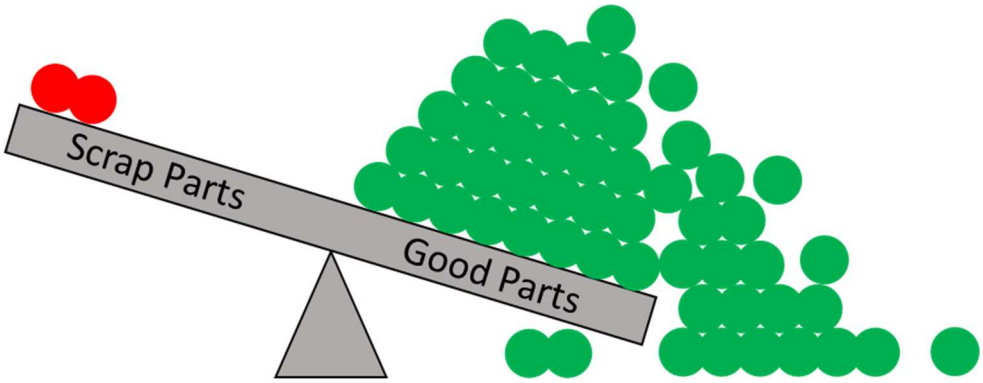


Figure 3. *Unbalanced data is a challenge for making predictions on manufacturing data. Robust models need to be trained on many good and scrap parts.*

The computing power and the tools exist to increase data-driven decision making in materials processing. Materials processing is rich with data from which knowledge can be created and incorporated into a future of smart factories. Active application of machine learning in the industry exists and the challenges are coming to the surface. The opportunity is now for industry and academia to work together to determine the right set of tools and methods which are most applicable to accelerate progress.

High Pressure Die Casting

As an exemplar materials processing method, consider high-pressure die-casting (HPDC) of aluminum alloys. HPDC is the most utilized process for casting aluminum alloy components [20], [21]. The process offers the advantages of high productivity and complex part geometries [22]. At the center of the HPDC work cell is the die casting machine (DCM). Ancillary equipment fills out the cell to execute tasks of metal delivery, die preparation, casting removal and trimming the part of excess material like runners and overflows. The DCM can be programmed to identify a casting as being good, scrap, or a warm-up shot based on the parameters which created the casting. This is accomplished by setting upper and lower control limits (UCL and LCL) for key variables as determined by the manufacturing engineer. The DCM is using a series of Boolean checks (Is parameter n between LCL_n and UCL_n?), all of which must be TRUE for a part to be good [23]. If one of the process parameters falls out of the programmed window, the check returns FALSE and the part is labeled scrap. If a part is cast with too low of intensification pressure, for example, the machine will identify the casting as scrap and send a signal to the operator or a robot to place the part into the scrap hopper or set it aside for inspection. Parts cast within the prescribed limits are labeled good parts and further processed as normal.

When envisioning HPDC process data, imagine a spreadsheet where rows represent individual castings, and columns are various input parameters and process outputs. The number of inputs can be very high. Blondheim estimates that a fully monitored die casting cell with thermal imaging for die temperature data collection can exceed 2,000,000 inputs per cycle [24]. If the imaging data is removed, the total opportunity still tops 300,000 inputs per cycle. Humans can visualize two-

and three-dimensional data quite well, but we have no way to visualize 300,000-dimensional data. Data science and machine learning provide the tools to analyze high-dimensional data.

There are two objectives of this research: The first is to use machine learning via a classification model to predict the quality label assigned by the DCM: good part, process scrap, and warm-up. The second is to determine which other classifications can be determined by this method; specifically, the presence of discontinuities (e.g. porosity) in a tensile bar machined from the casting was examined.

II. METHOD

The Decision Tree classifier is a supervised machine learning method used to build a predictive model for a given process output by sorting the castings into classes at various nodes using an input variable as the sorting criteria [2]. This input variable is chosen by the algorithm because sorting by it provides the greatest information gain to the model. Random Forest classifiers use many Decision Trees together to make predictions of what class each casting belongs [25]. Both methods are examined in this study.

Decision Trees are effective, easy to analyze machine learning algorithms which can be applied in both classification and regression problems. The Decision Tree classifier is a *supervised* machine learning method which means that the model is trained and tested on data with known output classifications. Once the model is developed, it can be used to make predictions on castings where the class is not known. Decision Trees build a predictive model by evaluating the variables and sorting the observations at various nodes into classes. The nodes split, forming branches of the tree which terminate at a leaf. Without placing restrictions on the model, the sorting will continue until each branch of the tree ends at a pure leaf consisting of one class. While this may result in a high scoring model, overfitting to the training data can be a problem, and it will not score highly on new data which the model has not seen before.

One method by which the splits are determined is Gini factor (*Equation 1*), which is a measure of the purity of the resulting nodes by making a split [26]. The Gini factor varies between zero and one. A Gini of zero represents a pure node where all the observations are of the same class. A high Gini value means that the various classes are mixed and there is a high probability that an observation may be misclassified.

$$Gini = 1 - \sum_{i=1}^{n} P_i^2 \qquad Eq.\ 1$$

Where:

- $Gini$ is the Gini index
- n is the number of classes
- P_i is the probability of finding each class in the node

Splits in the tree are made by evaluating the Gini over each input for the node in question. The feature with the lowest Gini for the samples in that node will become the split criterion and the tree continues to build. If the node is pure, or the Gini is not reduced, the branch terminates at a leaf. The user can specify other stopping criteria such as the maximum depth of the tree, minimum Gini required for a split, and minimum samples required to split a node. Doing so will result in a more general predictive model which is desirable.

An improvement upon the Decision Tree, Random Forest uses many tree estimators for making predictions. Random Forest works well with high-dimensional data, is robust to non-linear data, has low bias, and variance is reduced through bagging [25]. The randomization of the model resides in the building of the trees in the forest. The samples and features available to build each tree are randomly selected according to user defined limits. Random Forest is a prediction by committee approach. The results of many trees trained on the subsets are compiled to classify each observation. The number of tree estimators is set in the algorithm prior to creating the model.

The data used for this study is a large production dataset from a HPDC production of engine block castings at FCA Kokomo Casting Plant. The full dataset consists of over 950,000 observations, each row representing a production casting, and 83 columns of input/output variable data. The data is collected from 12 HPDC work cells and 20 die casting tools. Periodically, the production castings are destructively evaluated for mechanical property testing via a tensile test. In this dataset, there are 1495 observations for which both the HPDC process variables and the mechanical property data are collected into 159 columns. The blocks are cast in E380 aluminum alloy [27] and subjected to T5 heat treatment post castings. For a specific application, a subset of the blocks receives an additional 24-hour natural age prior to T5.

Prior to training the models, the data was cleaned to remove missing data, drop columns with no variation, and remedy bad data. Bad data, for example, may be intermittent sensor glitches, mis-scaled data, format errors, etc. Effective remedies rely on domain expertise of those close to the process to decide whether to impute a suitable value such as a median value, adjust to the proper scale, or remove the row or column as unreliable.

When developing the models, the datasets are split into training and test subsets. Unless otherwise noted, the split uses 80% of the rows for training and the remaining 20% for testing the model. Model performance metrics include accuracy, precision, recall, *f1-score*. Accuracy is percentage of correctly classified observations. This calculation is shown in *Equation 2* where *TP* is the number of true positives, *TN* is the number of true negatives, *FP* is the number of false positives, and *FN* is the number of false negatives. The equations for calculating precision, recall, and *f1-score* are given in *Equations 3, 4, and 5* respectively. *f1-score* is the harmonic mean of the precision and recall and is useful for unbalanced datasets, such as the one in this study, where there are many more good parts than process scrap.

$$Accuracy = (TP + TN) / (TP + TN + FP + FN) \quad Eq\ 2.$$

$$(Precision)_i = TP_i / (TP_i + FP_i) \quad Eq\ 3.$$

$$(Recall)_i = TP_i / (TP_i + FP_j) \quad Eq\ 4.$$

$$(f1\_score)_i = 2 * ((Precision_i * Recall_i)/(Precision_i + Recall_i)) \quad Eq\ 5.$$

III. RESULTS AND DISCUSSION

DCM Label Classification

Our first objective is to create a classification model which evaluates die-casting process data and the known class assigned to the part and determines what the rules are such that the model will accurately assign new part data to the correct class. The three labels are: good part, process scrap, and warm-up. The full HPDC data set was used in this exercise split into training and test populations.

Summaries of the model performance will be shown via *confusion matrix* (Table I). The matrix rows track the actual known classifications of the test population and the columns correspond to the classifications of the test population as predicted by the model. A perfect model would have zero *FN* and *FP* predictions.

Table I. Interpretation of the confusion matrix. A perfect model would have zero *FN* and *FP* predictions.

		Predicted Value	
		Positive	**Negative**
Actual Value	**Positive**	*TP*	*FN*
	Negative	*FP*	*TN*

Decision Tree Classifier

The results of the initial Decision Tree classifier are plotted in *Figure 4*. The graphic is displayed not for legibility, rather to show how complex Decision Trees can be.

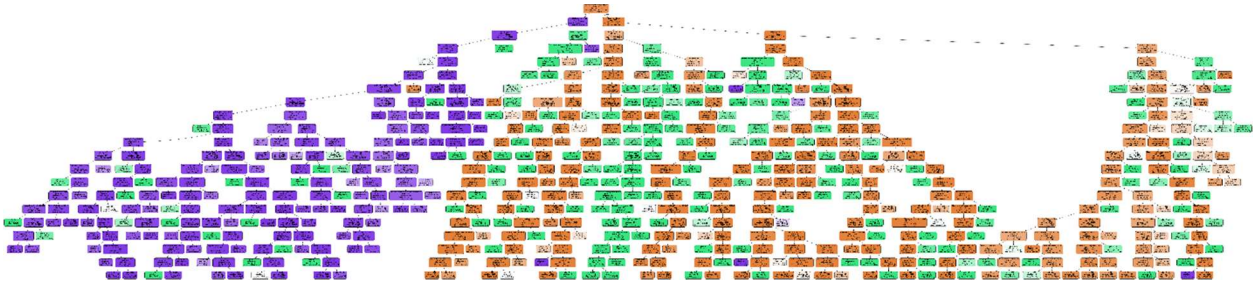


Figure 4. *Complete Decision Tree model for prediction of quality classification.*

This tree would be even larger if limits had not been placed on it. The depth of the tree was limited to 20 levels and the minimum requirement to make a split was set to 0.1% of the population. These settings are optimized by adjusting them in turn and evaluating performance metrics. What can be understood from *Figure 4* is that warm-up shots which are colored purple are easily separated from the good parts (orange) and process scrap (green). This is because warm-up shots typically have different process settings from production shots utilizing low shot velocity and minimal intensification pressure to reduce wear and tear on the DCM and die while bringing the die up in temperature. The challenge is separating the good parts from the process scrap. For the remainder of this exercise the warm-up shots have been utilized in a calculation as a process input which designates the number of shots performed since the last warm-up shot was made. This value is included to serve as a directional proxy for die temperature. Since this column equals zero for all warm-up shots, the prediction of warm-up shots by the model becomes automatic. Thus, the warm-up shots are removed from the dataset.

Table II contains the results of the part quality Decision Tree model.

Table II. Confusion matrix showing the performance of the Decision Tree classifier model for part quality.

Decision Tree Model – Part Quality		**Predicted Value**	
		Good Part	**Process Scrap**
Actual Value	**Good Part**	167,470	528
	Process Scrap	1,577	5,294

The Decision Tree performed well on the test data which is made up of 174,869 rows of data which the model had not seen before. The model does very well in predicting good parts. Although, manufacturing operations make many good parts, it is predicting the process scrap which is of the greatest value. 5,294 of the 6,871 process scrap rows are correctly classified. The 1,577 process scrap castings which are classified as good parts are FPs. False Positives must be minimized, as these would have a negative impact on downstream operations.

Random Forest Classifier

In the Random Forest, 10 estimators were used with a maximum depth of 35 levels and a minimum split size of 100 observations. The confusion matrix for the Random Forest is shown in Table III. Interestingly, the Random Forest model performs slightly worse on the process scrap class than the Decision Tree. Unbalanced data is a challenge for modeling production manufacturing data. Since each split in each tree is done without the consideration of any other splits, the best Gini split may sweep many process scrap samples into a node which is overwhelmingly good parts. This results in misclassification if the node is a leaf. Fortunately, there are methods to working with unbalanced data.

Table III. Confusion matrix showing the performance of the Random Forest model for part quality.

Random Forest Model – Part Quality		Predicted Value	
		Good Part	**Process Scrap**
Actual Value	**Good Part**	167,793	205
	Process Scrap	1,613	5,258

Oversampling for Imbalanced Data

To reduce the amount of FP, the issue of data imbalance is addressed by generating more process scrap data by which to train the model. The simplest way to do this is to reproduce samples from the process scrap class, but this provides no new information to the algorithm. A better method, which does provide new information to the model via the creation of new minority class samples, is called Synthetic Minority Oversampling TEchnique (SMOTE) [28], [29]. SMOTE creates each new minority class sample by selecting an example of the minority class, finding its nearest neighbors, and drawing a line between the example and one of its neighbors at random. The new sample is created along the connection line. This is done repeatedly until the minority class balances out the majority class. A disadvantage of SMOTE is that it is challenged by datasets where classes overlap.

To investigate, the training data was oversampled using SMOTE and new Decision Tree and Random Forest models were trained. Predictions were made on the same testing data using the new models. It is important that SMOTE be applied to the training data only, and not the testing data. This way the testing data is of a distribution faithful to the process. The results are shown in Table IV and Table V for the Decision Tree and Random Forest respectively. By balancing out the process scrap with the good parts, the new models are more adept at recognizing process scrap and FPs are reduced. The increase in FN is potentially due to overlap in the classes. A potential cause of overlap is from castings where the classification of the machine was overruled by the operator. In such an instance, a good part would be labeled process scrap. Unfortunately, no record of operator intervention is kept by which to verify.

Table IV. Confusion matrix showing the performance of the Decision Tree with SMOTE classifier model for part quality.

Decision Tree w/ SMOTE – Part Quality		Predicted Value	
		Good Part	**Process Scrap**
Actual Value	**Good Part**	164,716	3,282
	Process Scrap	924	5,947

Table V. Confusion matrix showing the performance of the Random Forest w/ SMOTE model for part quality.

Random Forest w/ SMOTE – Part Quality		**Predicted Value**	
		Good Part	**Process Scrap**
Actual Value	**Good Part**	165,443	2,555
	Process Scrap	1038	5,833

SMOTE improves the model performance on the minority class in both Decision Tree and Random Forest models. Comparing Tables IV and V, it is difficult to see which model is best suited for our data. Both exhibit false positives and false negatives. To determine the better performing model, it is useful to use scoring metrics. These measures are tabulated for both models below (Table VI). The metrics associated with the minority class (process scrap) are more telling for model performance. The models perform quite similarly, however, the Random Forest is the better model due to the higher *f1-score* for the process scrap class. The results between the testing and training datasets are nearly the same, therefore, it can be said that neither model is overfitting to the training data.

Table VI. Key scoring metrics for the part quality Decision Tree and Random Forest classifiers with SMOTE training data. Mean values are reported from 5-fold cross validation.

	Decision Tree w/ SMOTE		**Random Forest w/ SMOTE**	
	Training Data	Test Data	Training Data	Test Data
Model Accuracy	98.81 %	98.70 %	98.40 %	98.66 %
Precision	98.76 %	98.64 %	98.41 %	98.63 %
Recall	98.81 %	98.70 %	98.40 %	98.66 %
f1-Score	98.76 %	98.64 %	98.40 %	98.55 %
f1-Score (Process Scrap)	97 %	74 %	99 %	76 %

Once the model is run, a useful summary for the process engineer can be pulled from the model, *feature importance* [30]. Understanding the influence of each variable on the model helps the engineer determine which variables to monitor more frequently or accurately, and where to invest in process control measures for best results. Feature importance of the Random Forest and Decision Tree with SMOTE models are given in Table VII below. The list of 83 variables was truncated at values > 0.02. The feature importance table shows how Random Forest ensemble learning softens and enhances the importance of individual features in comparison to the Decision Tree algorithm.

Table VII. Feature importance for the part quality Decision Tree and Random Forest classifiers with SMOTE generated training data.

Part Quality Decision Tree w/ SMOTE		**Part Quality Random Forest w/ SMOTE**	
Feature Name	**Importance**	**Feature Name**	**Importance**
Time Between Cycles	0.4910	Time Between Cycles	0.2830
Shots Since Last Warm-up Shot	0.1779	Biscuit Length	0.0868
Biscuit Length	0.0793	Final Intensifier Pressure	0.0641
Final Intensifier Pressure	0.0595	Plunger position at the end of shot	0.0505
Average Intermediate Shot Velocity	0.0471	Cycle Time	0.0480
Cycle Time	0.0207	Average Intermediate Shot Velocity	0.0426
		Cavity Fill Time	0.0392
		Average Fast Shot Velocity	0.0381
		Shots Since Last Warm-up Shot	0.0314
		Intensification Velocity Rise Time	0.0239
		Dwell Time	0.0239
		Intensification Stroke	0.0215

Feature importance can also be used to assist in feature selection for creating more efficient models which take less time to run and perform better when noisy features are removed. Ultimately, the final set of features is based on trial and error and the preferred performance metric. The same Random Forest model set-up was run using only the top 12 features (Table VII). Dropping the low importance input variables minimally reduces predictive power, and overfitting to the training data is still avoided (Table VIII).

Table VIII. Scoring metrics for the part quality Random Forest with SMOTE classifier models using the top 12 features. Mean values are reported from 5-fold cross validation.

	Random Forest Classifier	
	Training Data	**Test Data**
Model Accuracy	98.81 %	98.55 %
Precision	98.82 %	98.48 %
Recall	98.81 %	98.55 %
f1-Score	98.81 %	98.45 %
f1-Score (Process Scrap)	99 %	79 %

Breaking the data down into unique combinations of DCM number and die cavity number yielded interesting results. It was observed that, when subsets representing each combination of DCM number and die cavity number were run across the general part quality Random Forest classifier, the metrics of the predictions varied. This suggests that each DCM and cavity combination is to some degree a unique process. The five best and five worst results are presented in Table IX.

Table IX. Performance metrics of the part quality Random Forest with SMOTE classifier when each unique combination of DCM and die cavity subset is run as the test sample.

Combination ID	**Accuracy**	**Precision**	**Recall**	**F1-score**
Top 5 Results				
Combination 17	0.99	0.96	0.89	0.93
Combination 16	0.99	0.89	0.95	0.92
Combination 14	0.98	0.87	0.93	0.90
Combination 13	0.99	0.85	0.92	0.88
Combination 06	0.99	0.87	0.87	0.87
Bottom 5 Results				
Combination 23	0.98	0.72	0.81	0.76
Combination 26	0.98	0.63	0.96	0.76
Combination 24	0.99	0.72	0.78	0.75
Combination 25	0.98	0.66	0.77	0.71
Combination 02	0.98	0.65	0.76	0.70

Combination 16 which offered the largest subset of the overall dataset was examined on its own to see if the quality label prediction could be improved. The dataset is reduced from 874,344 to 76,226 observations. However, the noise from multiple processes grouped together is minimized.

For the Combination 16 Random Forest classifier model, 100 estimators were used with a maximum depth of 11 splits. The top 12 features as determined by the all-inclusive Random Forest were used as the input features. The confusion matrix is shown in Table X. The model classifies some process scrap in the good part class, but the *f1-score* for process scrap specifically increases to 0.89 as compared to 0.76 when all combinations of DCM and cavity numbers are grouped together.

Table X. Confusion matrix for the Combination 16 test dataset run across the part quality Random Forest classifier specific to the Combination 16 dataset.
Top 12 important features used as inputs.

Random Forest Model – Part Quality Combination 16 Data		**Predicted Value**	
		Good Part	**Process Scrap**
Actual Value	**Good Part**	14,698	83
	Process Scrap	22	443

The takeaway from this analysis is that machine learning algorithms, specifically Random Forest classifiers, are adept at analyzing high-dimensional datasets and identifying the quality thresholds established for materials processes. In learning the acceptable range for each variable to make quality parts, accurate predictions can be made on new parts of unknown quality. Oversampling techniques such as SMOTE, helps to address the imbalance in the data and makes the model a better predictor of the minority class.

The example of predicting part quality assigned by the DCM is a straightforward example where the dataset is very large and contains all the information available to the DCM for labeling parts good or process scrap. Many materials processing problems are more difficult due to the challenges of small datasets. Next, we turn our attention to how well Random Forest classification modeling can be applied to predicting porosity in castings using process data by which they were made.

Porosity Classification

HPDC process input data is used by manufacturing operations as a real-time quality check. Thus, it is of interest to test if these data can be analyzed further to predict levels of porosity in good parts. For the cast component of this study, production castings are selected for destructive mechanical property testing via testing tensile bars extracted from the casting itself. Ultimate tensile strength (UTS), yield strength, tensile strain (elongation), and hardness data are collected for the purpose of quality assessment [31]. In most HPDC products, the location of the tensile bars is limited to the few heavy areas of the casting which can accommodate their geometry. This constraint applies to the engine block geometry in this study. Thick walled sections are difficult

geometries in die cast parts because there is no ability to use risers, as other casting processes do, to feed volume contraction during solidification [32]. Intensification pressure is applied during solidification to compress gas porosity and feed shrinkage; however, once the gates freeze, pressure is no longer transmitted to the last areas to solidify (e.g. heavy walled sections). In the long freezing range aluminum alloys commonly utilized in HPDC, like 380-alloy, the resulting shrinkage is often microshrinkage which is difficult to detect via NDA methods such as digital X-ray. Thus, the presence of porosity is a characteristic of HPDC which must be controlled and not necessarily an indication of a poor casting. Discontinuities do impact the measured mechanical properties resulting in additional work and cost to reproduce the test. It has been shown that mechanical properties are dependent on the amount of porosity in the area of fracture [33]–[36]. Making a connection between mechanical properties and porosity is of interest to die casters because, in many applications, the presence of porosity can result in scrap due to uncovered porosity after machining or loss of pressure tightness or leaking.

Finding which process inputs contribute to porosity in a mature process is challenging for humans to solve. It is also a difficult problem to model because all the castings in the new dataset are classified as good parts, so the difference between any given input variable from one observation to the next is likely small.

The HPDC process dataset was merged with the tensile bar dataset using the part serial number to match the observations. The result is a much smaller dataset with 1495 rows. The comments column from the tensile bar dataset was text mined to determine which bars exhibited visual discontinuities in the tensile bar fracture surface [37]. Next, the data was run through a Random Forest classifier to see if porosity could be predicted. This newly generated Random Forest classifier was unable to discern a difference between the parts which had confirmed porosity in the tensile bar and those for which no porosity was observed. Most of the porosity samples are predicted to not have porosity as shown in Table XI.

Table XI. Confusion matrix showing the performance of the porosity Random Forest classifier model on the test dataset for all tensile bars.

Random Forest Model – Porosity All Tensile Bar Data		**Predicted Value**	
		No Porosity	**Porosity**
Actual Value	**No Porosity**	220	0
	Porosity	6	2

The output in the above model is a binary categorization of porosity observed or not. Cáceres' work shows that a binary classification for porosity is not adequate since the amount of porosity affects the mechanical properties [33]. Whether or not the porosity was observed by the tester in the tensile bar has no bearing on how the bar performed. The observation is based on unaided

visual inspection. Porosity in 380-aluminum from volumetric contraction is expected to be microshrinkage which may go undetected in visual inspection [38]. *Figure 5* shows an empirical cumulative distribution function for the bars with and without observed discontinuities. The curve for the data with observed discontinuity is shifted to lower UTS values. There is considerable overlap which supports the supposition that microshrinkage porosity is often undetectable via visual inspection.

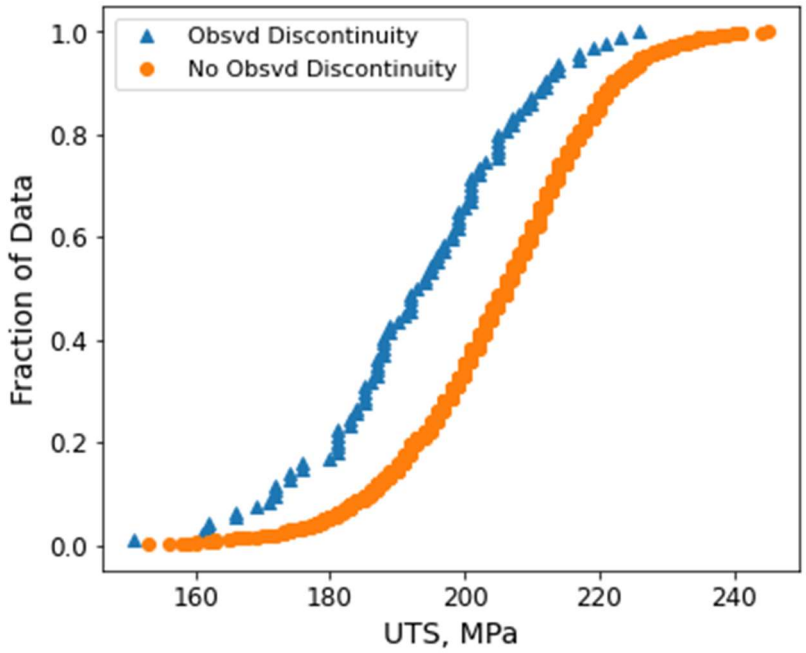


Figure 5. *Empirical cumulative distribution functions for the UTS of tensile bars with and without observed discontinuities on the facture surface by visual inspection.*

A classification model based on a UTS value has two benefits: undetected porosity gets captured in the lower performing bars and the two classes can be set up to be more balanced. Two classes were selected: UTS < 205 MPa and UTS ≥ 205 MPa. The groupings have no reflection on engine block performance. There is no assignment of "good" versus "bad" implied in selecting the ranges. The value of 205 MPa is chosen as it is the median value of the tensile bar dataset. Importantly, 80% of the bars with observed discontinuities exhibited less than 205 MPa of UTS as well. A Random Forest classifier was used to predict which UTS class each bar in the test dataset would fall into using the HPDC process input data. Table XII shows the Random Forest classifier results cross validated over 10 iterations. This model is overfitting to the training data as there is more of a difference between the training and testing metrics than we saw in the DCM quality label example. One of the better performing models is shown in Table XIII.

Table XII. Key scoring metrics for the Random Forest classifier model predicting UTS over or under 205 MPa. The support of the test dataset is: 138 UTS < 205 MPa samples and 161 UTS ≥ 205 MPa samples.

	Random Forest Classifier: UTS over/under 205 MPa	
	Training Data	**Test Data**
Model Accuracy	60.62 %	56.87 %
Precision (weighted)	60.76 %	57.13 %
Recall (weighted)	60.62 %	56.87 %
f1-Score (weighted)	60.57 %	56.37 %

Table XIII. Confusion matrix of the Random Forest classifier for UTS tensile bars above and below 205 MPa using HPDC process inputs only.

Random Forest Model – UTS		**Predicted Value**	
		> 205 MPa	**< 205 MPa**
Actual Value	**> 205 MPa**	98	63
	< 205 MPa	47	91

If die casting operations examine their data in this way, there is benefit gained even from imperfect models. Referring to Table XIII, the test dataset consists of 299 samples of which 161 were of the higher UTS class. This amounts to 53.8% high UTS samples. This model suggests that there are operating conditions where high UTS bars can be expected. If those conditions are employed, one would find that 98 of 145 are high UTS bars, or 67.6%. The parameters which rise to the top of the feature importance list in Table XIV are worthy of study since splitting on their value has the largest impact on UTS prediction.

Individual tree estimators can be pulled from the Random Forest and viewed to understand which feature and values were chosen for splitting nodes. Geometry and performance requirements are design specific, expect important features and the values set as thresholds for splitting the data to vary part number to part number. In the data for this engine block study, higher UTS parts are associated with lower cycle times: in every timer listed in Table XIV, a lower timer value is associated with a higher percentage of parts with greater than 205 MPa UTS. Intensification stroke refers to how much the shot rod moves forward under intensification pressure; the forward movement is tied to feeding of shrinkage porosity and higher values for this variable improve UTS. Also, consistent production associated with less time between cycles and longer continuous runs of parts result in stronger parts.

Table XIV. Average feature importance calculated over ten iterations of the Random Forest classifier.

Feature Name	Importance	Feature Name	Importance
Ejection Forward Time	0.0747	Total Tie Bar Tonnage	0.0379
Spray Robot Time	0.0505	Final Intensifier Pressure	0.0357
Die Close Tank Level	0.0498	Avg Head Pressure during Intermediate Shot	0.0349
Avg Head Pressure during Fast Shot	0.0466	Extract Robot Cycle Time	0.0348
Shot Count Since Last Warm Up Shot	0.0414	Cycle Time	0.0347
Die Close Time	0.0403	Die Opening Time	0.0329
Intensification Pressure Rise Time	0.0396	Vacuum Pressure during Shot	0.0321
Average Fast Shot Velocity	0.0386	Ladle Pour Time	0.0332
Avg Head Pressure during Slow Shot	0.0383	Intensification Stroke	0.0304

IV. CONCLUSIONS

- Supervised learning performed better on the larger HPDC process dataset. The complete population has 874,344 observations and we know that the DCM is making quality determinations based on this data, so the right data is collected. The result is a good model.
- Oversampling using SMOTE is effective for teaching the model to better predict the minority class.
- The Random Forest classifier outperforms a single Decision Tree by reducing variance. The ability to differentiate good parts from process scrap improve when focusing on unique combinations of machine and cavity number as stand-alone processes.
- For predicting porosity, UTS has been shown to be a better output for predictive modeling than relying on porosity observation alone. Microshrinkage porosity can easily be missed by the unaided eye, but its effect is apparent in the UTS measured.
- A key difference between the DCM part label problem and the porosity prediction problem is the size of the dataset available to the model. The smaller tensile bar dataset is impacted by overfitting issues that the larger dataset avoids.

V. REFERENCES

[1] W. D. Nordhaus, "Two Centuries of Productivity Growth in Computing," *J. Econ. Hist.*, vol. 67, no. 1, pp. 128–159, Mar. 2007, doi: 10.1017/S0022050707000058.

[2] D. Dietrich, B. Heller, and B. Yang, *Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data*, 1st ed. Wiley, 2015.

[3] Capgemini Consulting Group, “Industry_4.0_-The_Capgemini_Consulting_V.pdf.” Capgemini, 2014, [Online]. Available: https://www.capgemini.com/consulting/wp-content/uploads/sites/30/2017/07/capgemini-consulting-industrie-4.0_0_0.pdf.
[4] H. Cho, Y. Seo, B. V. K. V. Kumar, and R. R. Rajkumar, “A multi-sensor fusion system for moving object detection and tracking in urban driving environments,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, Jun. 2014, pp. 1836–1843, doi: 10.1109/ICRA.2014.6907100.
[5] T. Huntsberger, H. Aghazarian, A. Howard, and D. C. Trotz, “Stereo vision–based navigation for autonomous surface vessels,” *J. Field Robot.*, vol. 28, no. 1, pp. 3–18, Jan. 2011, doi: 10.1002/rob.20380.
[6] R. A. Greenes, “Clinical Decision Support and Knowledge Management,” in *Key Advances in Clinical Informatics*, Elsevier, 2017, pp. 161–182.
[7] M. Xiaoxi, L. Weisi, H. Dongyan, D. Minghui, and H. Li, “Facial emotion recognition,” in *2017 IEEE 2nd International Conference on Signal and Image Processing (ICSIP)*, Singapore, Aug. 2017, pp. 77–81, doi: 10.1109/SIPROCESS.2017.8124509.
[8] M. S. Bartlett, G. Littlewort, M. Frank, C. Lainscsek, I. Fasel, and J. Movellan, “Recognizing Facial Expression: Machine Learning and Application to Spontaneous Behavior,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, San Diego, CA, USA, 2005, vol. 2, pp. 568–573, doi: 10.1109/CVPR.2005.297.
[9] M. S. Bartlett, G. Littlewort, M. Frank, C. Lainscsek, I. Fasel, and J. Movellan, “Fully Automatic Facial Action Recognition in Spontaneous Behavior,” in *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, Southampton, UK, 2006, pp. 223–230, doi: 10.1109/FGR.2006.55.
[10] B. Kopper, *Untitled*. 2020.
[11] D. Gaddam, “Autonomous Optimization of Die Casting Processes,” *AFS Trans.*, vol. 124, pp. 25–32, 2016.
[12] D. Blondheim, “Unsupervised Machine Learning and Statistical Anomaly Detection Applied to Thermal Images,” *NADCA Trans. T18-071*, 2018, [Online]. Available: http://www.diecasting.org/transactions/T18-071.
[13] M. Paolanti, L. Romeo, A. Felicetti, A. Mancini, E. Frontoni, and J. Loncarski, “Machine Learning approach for Predictive Maintenance in Industry 4.0,” in *2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, Oulu, Jul. 2018, pp. 1–6, doi: 10.1109/MESA.2018.8449150.
[14] B. Cline, R. S. Niculescu, D. Huffman, and B. Deckel, “Predictive maintenance applications for machine learning,” in *2017 Annual Reliability and Maintainability Symposium (RAMS)*, Orlando, FL, USA, 2017, pp. 1–7, doi: 10.1109/RAM.2017.7889679.
[15] T. Prucha, “From the Editor: AI Needs CSI: Common Sense Input,” *Int. J. Met.*, vol. 12, no. 3, pp. 425–426, Jul. 2018, doi: 10.1007/s40962-018-0235-2.
[16] A. Chatterjee and A. Segev, “Data Manipulation in Heterogeneous Databases,” *SIGMOD Rec*, vol. 20, no. 4, pp. 64–68, Dec. 1991, doi: 10.1145/141356.141385.
[17] W.-S. Li and C. Clifton, “Semantic Integration in Heterogeneous Databases Using Neural Networks,” p. 12.
[18] W.-S. Li and C. Clifton, “SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks,” *Data Knowl. Eng.*, vol. 33, no. 1, pp. 49–84, Apr. 2000, doi: 10.1016/S0169-023X(99)00044-0.

[19] J. Cohen, “Statistical Power Analysis,” *Curr. Dir. Psychol. Sci.*, vol. 1, no. 3, pp. 98–101, Jun. 1992, doi: 10.1111/1467-8721.ep10768783.
[20] J. Folk, “U.S. Aluminum Casting Industry - 2019,” *Cast. Eng.*, no. July 2019, pp. 16–19, Jun. 2019.
[21] A. Spada, “Revitalization of North American Metalcasting,” 2012, Accessed: May 24, 2020. [Online]. Available: https://www.diecasting.org/docs/statistics/North_America.pdf.
[22] R. Lumley, Ed., *Fundamentals of aluminium metallurgy: production, processing and applications*. Oxford: Woodhead Publ, 2011.
[23] J. I. Moore and P. J. Van Huis, “US4493362.pdf,” 4493362, Jan. 15, 1985.
[24] D. Blondheim, “Artificial Intelligence, Machine Learning, and Data Analytics: Understanding the Concepts to Find Value in Die Casting Data,” presented at the 2020 NADCA Executive Conference, Clearwater Beach, FL, Feb. 25, 2020.
[25] L. Breiman, “Random Forests,” *Mach. Learn.*, vol. 45, no. October 2001, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.
[26] R. I. Lerman and S. Yitzhaki, “A note on the calculation and interpretation of the Gini index,” *Econ. Lett.*, vol. 15, pp. 363–368, 1984, doi: 10.1016/0165-1765(84)90126-5.
[27] The Aluminum Association, *Designations and Chemical Composition Limits for Aluminum Alloys in the Form of Castings and Ingot*, October 2018. Arlington, VA: The Aluminum Association, 2018.
[28] R. Blagus and L. Lusa, “SMOTE for high-dimensional class-imbalanced data,” *BMC Bioinformatics*, vol. 14, no. 1, p. 106, Dec. 2013, doi: 10.1186/1471-2105-14-106.
[29] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002, doi: 10.1613/jair.953.
[30] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, “Permutation importance: a corrected feature importance measure,” *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, Apr. 2010, doi: 10.1093/bioinformatics/btq134.
[31] ASTM International, “ASTM B 557-15, Test Methods for Tension Testing Wrought and Cast Aluminum- and Magnesium-Alloy Products.” ASTM International, doi: 10.1520/B0557-15.
[32] J. Campbell, *Complete Casting Handbook: Metal Casting Processes, Metallurgy, Techniques and Design*. Butterworth-Heinemann, 2015.
[33] C. H. Cáceres, “On the effect of macroporosity on the tensile properties of the Al-7%Si-0.4%Mg casting alloy,” *Scr. Metall. Mater.*, vol. 32, no. 11, pp. 1851–1856, Jun. 1995, doi: 10.1016/0956-716X(95)00031-P.
[34] M. K. Surappa, E. Blank, and J. C. Jaquet, “EFFECT OF MACRO-POROSITY ON THE STRENGI~ AND DUCTILITY OF CAST,” vol. 20, no. 9, p. 6.
[35] C. D. Lee and K. S. Shin, “Constitutive prediction of the defect susceptibility of tensile properties to microporosity variation in A356 aluminum alloy,” *Mater. Sci. Eng. A*, vol. 599, pp. 223–232, Apr. 2014, doi: 10.1016/j.msea.2014.01.091.
[36] C. D. Lee, “Effects of microporosity on tensile properties of A356 aluminum alloy,” *Mater. Sci. Eng. A*, vol. 464, no. 1–2, pp. 249–254, Aug. 2007, doi: 10.1016/j.msea.2007.01.130.
[37] V. Gupta and G. S. Lehal, “A Survey of Text Mining Techniques and Applications,” *J. Emerg. Technol. Web Intell.*, vol. 1, no. 1, pp. 60–76, Aug. 2009, doi: 10.4304/jetwi.1.1.60-76.

[38] J. A. Spittle, “Grain refinement in shape casting of aluminium alloys,” *Int. J. Cast Met. Res.*, vol. 19, no. 4, pp. 210–222, Sep. 2006, doi: 10.1179/136404606225023444.

Appendix E – Model Selection and Evaluation for Machine Learning: Deep Learning in Materials Processing

Adam Kopper[1*], Rasika Karkare[2], Randy C. Paffenroth[3], Diran Apelian[4]

[1]Mercury Marine, Fond du Lac, WI 54935 USA
[2]WPI, Data Science, Worcester, MA, 01609 USA
[3]WPI, Mathematical Sciences, Computer Science, Data Science, Worcester, MA 01609 USA
[4]UCI, Materials Science and Engineering, Irvine, CA 92967 USA

[*]Corresponding author, email: adam.kopper@mercmarine.com

Keywords. Machine learning; deep learning; random forest; support vector machine; neural network; high pressure die casting; principal component analysis; bias-variance trade-off.

Abstract

Materials processing is a critical subset of manufacturing which is benefitting by implementing machine learning to create knowledge from the data mined/collected and gain a deeper understanding of manufacturing processes. In this study, we focus on aluminum high-pressure die-casting (HPDC) process, which constitutes over 60% of all cast Al components. Routinely collected process data over a year's time of serial production is used to make predictions on mechanical properties of castings; specifically, the ultimate tensile strength (UTS). Random Forest, Support Vector Machine (SVM), and XGBoost regression algorithms were selected from the machine learning spectrum along with a Neural Network, a deep learning method. These methods were evaluated and assessed and were compared to predictions based on historical data. Machine learning, including Neural Network, regression models do improve the predictability of UTS above that of predicting the mean from prior tests. Choosing the correct models to use for the data requires an understanding of the bias-variance trade-off such that a balance is struck between the complexity of the algorithms chosen and the size of the dataset in question. These concepts are reviewed and discussed in context of HPDC.

I. Introduction

A recent boom in machine learning has been sparked by continuous decrease in the cost of computer memory and increases in computing power [1], [2]. This, coupled with increased access to machine learning algorithms and open source software, has broadened the scope of interested parties beyond the early adopters like social media, banking, and marketing and retail sectors into manufacturing operations. Materials processing is a critical subset of manufacturing which is benefitting by implementing machine learning to create knowledge from the data mined/collected and to gain a deeper understanding of manufacturing processes.

Many materials manufacturing processes tend to be large-scale in terms of production tonnage and units per hour. Efficiency is a core metric for materials processing plants. In thermally controlled processes, interruptions have significant downtime implications in returning the process to the operating temperature. In such an environment, sampling each unit of the product for the purpose

of quality assurance or process control reduces productivity, adds cost, or just simply is not practical. Machine learning is an enabling technology with the potential to minimize sampling and testing while boosting the confidence that both producers and customers have in the end product.

We consider aluminum high-pressure die-casting (HPDC) for this study, which is the most utilized process in the world for aluminum alloy near-net shaped components [3], [4]. In brief, the process consists of a machine which holds a steel die where the casting is formed, and an injection system for delivering the metal at high speed and holding the solidifying metal under pressure. The application of machine learning, including Neural Networks, to HPDC has been studied in the numerical simulation realm. Rai et al. used a supervised learning method by creating datasets with process simulation software and teaching a Neural Network to predict cavity fill time, solidification time, and porosity based on the process inputs: melt and die temperature and slow and fast shot velocities [5]. They found that the results of the Neural Network model compared well to those generated by commercially available finite element mesh-based simulation software but did so in much less time. Similarly, Yarlagadda et al. predicted fill time from the melt temperature, die temperature, injection pressure, and casting weight with a Neural Network trained via process simulation software and domain expertise from casting specialists [6]. Moving into the experimental realm, Soundararajan et al. were able to train and test a Neural Network predicting the ultimate tensile strength (UTS) and yield strength (YS) of extracted tensile bars from gravity cast aluminum with a correlation coefficient of 0.95 and 0.96, respectively [7]. Their experimental settings represent a wider range in process input values than one might encounter on a fully developed production casting, exaggerating the differences for the algorithm to recognize and learn. In volume production, process parameters are established to ensure uniformity in the final product. To predict the UTS of each sample to a high accuracy based on typical input variation is a difficult problem at which this research is directed. It is common practice to collect HPDC cycle summary data with respect to plunger velocity, pressures, and various timers for each shot as captured by the shot monitoring software on the die casting machine. While these data are routinely reviewed for troubleshooting purposes, utilizing such information to make predictions about the castings themselves is not the norm, and thus the opportunity or a need that this work addresses.

It is paramount to understand the type of data HPDC operations generate, and the machine learning and deep learning methods that are best suited for analysis. One is often introduced to the terms machine learning, deep learning, and Neural Network as buzz words used interchangeably in marketing or general audience publications. All of these are subsets of artificial intelligence and defining where machine learning ends and deep learning begins is somewhat blurry. Perhaps it is best to look at these as a continuum of complexity. Machine learning algorithms reside on the lower end of the complexity spectrum making use of linear and other low-order functions [8]. While deep learning is at the other end employing layers of mathematical transformations and activation functions for creating models [9]. The most suitable method depends on the data available.

This study was conducted to compare the performance of various machine learning and deep learning methods in predicting the UTS of tensile bars excised from engine block castings. The mean absolute error of the algorithm is used to score the methods. Furthermore, an explanation of the importance of bias-variance trade-off is given to provide context for the results [10], [11].

II. Methodology

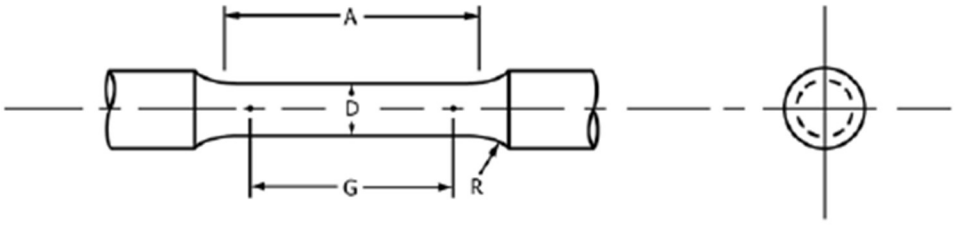
Casting Details

The data used for this machine learning study represent a large production dataset from a HPDC operation. The full dataset consists of over 950,000 observations, each row representing a production casting, and 83 columns of input/output variable data collected from 12 HPDC work cells and 20 die casting tools in the production of engine block castings at FCA Kokomo Casting Plant. Briefly, the core of the set is HPDC process summary data collected via shot monitoring software (plunger velocities, intensification parameters, timers, melt temperature, to name a few); specifics can be found in Appendix A. Periodically, the production castings are destructively evaluated for mechanical property testing via a tensile test. In this dataset, there are 1495 observations for which both the HPDC process variables and the mechanical property data are collected into 159 columns. The blocks are cast in E380 aluminum alloy [12] and subjected to T5 heat treatment post castings. For a specific application, a subset of the blocks receives an additional 24-hour natural age prior to T5.

Mechanical Property Testing of Castings

When designing castings, minimum mechanical properties may be specified by the designer which are required for the final product. Process and alloy selection are largely driven by these requirements [13]. Testing mechanical properties such as UTS, YS, and elongation requires destructive methods which can only be conducted on an audit basis. Tensile testing of test bars extracted from the cast part itself, or cast alongside the part, is the most employed method to measure these properties [14]. The tensile bars come from four different locations in the engine block and are machined to a 0.350 inch (9 mm) diameter sub-sized geometry based on ASTM B 557 (*Figure 1*) [15]. The bars are pulled using an Instron tensile testing machine. A load cell is used to measure the force on the tensile bar. An extensometer is affixed to the bar to determine the point of yielding. The dataset captures the UTS, YS, and the tensile strain. The 0.2% offset method is used to calculate the YS [16]. The tensile strain is measured with the extensometer over the course of the test and is reported as a percentage. Included in the dataset is a notes column which is text mined for mentions of fracture location and the presence of observed discontinuities such as porosity or an inclusion [1]. Each bar is classified accordingly. Tests with no indication of a discontinuity are classified as *unknown*, rather than to assume none were present.

For traceability, each engine block is assigned a serial number when it is cast. This unique character string is stored for each row of the HPDC process dataset. The serial number of the block casting is recorded in the tensile bar data as well. This identifier is used to merge the two datasets together, such that the data for each tensile test is expanded to include the HPDC process data as well. After an initial feature selection exercise [8], to remove columns with no variation and highly correlated columns, the resulting dataset consists of 1494 observations, or rows, and 80 variables, or columns. Of the 80 columns, 77 are inputs and 3 are outputs (UTS, tensile strain, and the Quality Index (QI) [17]). QI is an empirical relationship which aids in the interpretation of tensile test data. Mapping UTS-Elongation data over a grid of iso-QI and iso-YS lines provides the materials engineer directional insight into how to adjust alloy chemistry, solidification rate, and heat treatment to achieve the desired result.



G – Gage Length	1.400 +/- .005 (35.5 +/- 0.1)	**R – Radius (min)**	0.25 (6.35)
D – Diameter	0.350 +/- .007 (9.0 +/- 0.2)	**A – Reduced Section Length (min)**	1.650 (41.9)

Figure 1. *Tensile bar geometry per ASTM B557* [15, p. 55]. *Dimensions in inches (mm).*

Tensile bar data were examined to determine which output to target for prediction. Like most production manufacturing data, there is noise in the data that can be difficult to filter out with certainty as the actual tensile bars are not typically retained and were not available for this study. Statistical analysis via Welch's t-test is performed to detect significant shifts in the mean value of UTS and tensile strain from one population to another [18]. Location of bar extraction, presence of observed discontinuities, and the heat treatment were analyzed, and key results are shown in Tables I and II. The fracture location along the bar (middle vs. gauge) was not found to significantly move the mean UTS to a 95% confidence level.

Table I. Mean values for UTS and tensile strain comparing tensile bars from two heat treatments for all tensile bars. The null hypothesis is that there is no difference with respect to heat treatment.

	Standard T5 Heat Treatment	**T5 w/ Additional Natural Age**	**p-value**	**Reject the null hypothesis?**
Ultimate Strength, MPa (All Bars)	201	212	2.8E-29	Yes
Tensile Strain, % (All Bars)	1.4	1.6	9.3E-14	Yes

Table II. Mean values for UTS and tensile strain comparing tensile bars based on noted discontinuity on the fracture surface. The null hypothesis is that there is no difference between bars with an observed discontinuity and those with no noted observation.

	No Observed Discontinuity	Observed Discontinuity (unaided eye)	p-value	Reject the null hypothesis?
Ultimate Strength, MPa (All Bars)	204	191	1.2E-11	Yes
Tensile Strain, % (All Bars)	1.4	1.3	0.01	Yes

The results of Welch's t-test confirmed that the mean UTS value is statistically different based on the presence of defects and heat treatment used. UTS was selected over QI and tensile strain for its sensitivity to the presence of observed anomalies in the tensile bars. The literature has shown that UTS is sensitive to the presence of such casting features in tensile bars. Surappa reported that the mechanical properties of A356 permanent mold castings are less dependent on the bulk porosity than they are on the porosity in the test bars themselves [19]. Cáceres and Selling observed a power law relationship between the UTS and the area fraction of defects on the fracture surface of tensile bars [20]. This connection of UTS to porosity is very useful to die casting producers, since quality issues in die casting are largely porosity related [21]. Preliminary modeling efforts confirmed that UTS was showing less error in the model performance as compared to prediction of tensile strain and QI.

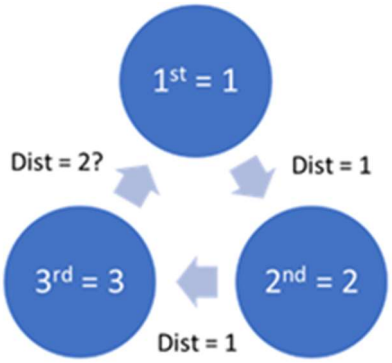
Data Pre-Processing

Before the data can be processed through machine learning algorithms, there is a significant amount of pre-processing which must be done to obtain meaningful results. In particular, the idea of *distance between observations* is essential to machine learning algorithms and getting such distances wrong can severely hamper the functioning of machine learning algorithms. Consequently, common pre-processing operations were performed for the purpose of cleaning the dataset, dealing with discrete data, and standardizing the data.

Real world production data is messy. Missing values, erroneous sensor readings, duplicated entries, typos, format changes in the source file, etc. must be sorted out before one can engage in meaningful analysis. Considering a data set of over 950,000 rows and 109 columns, one cannot simply scroll through and identify the anomalies. Running summaries of each column, examining the data class, and locating missing values are a few of the tasks to be carried out. This is where the expertise of the data scientist and the domain experts are invaluable.

In the dataset, there are continuous variables such as melt temperature, fast shot velocity, and intensification pressure. Likewise, there are discrete, categorical, variables such as machine number, cavity number, and work shift. Continuous variables often have easily defined distances since the distance between two melt temperatures, for example, is easily calculated and meaningful. However, categorical data, especially those which are represented by numeric identifiers cannot be properly calculated by simply finding the difference between two numeric

labels. Even so, it is useful data and can be incorporated into machine learning algorithms. Work shift is a good example for illustrating how to deal with discrete data. Work shift is often represented by numerical representation of first (1), second (2), and third (3) shift. While the distance between 1 and 2 and the distance between 2 and 3 both have a value of one, the pair 1 and 3 have a distance equal to two. Logically, first and third shift are no further apart than first and second (*Figure 2a*). To deal with the challenge of discrete data, data scientists utilize a method known as one-hot encoding [22]. One-hot encoding takes the tall vector which has discrete work shift data consisting of 1's, 2's, and 3's and converts it into a wide set of three vectors, we will call them Work Shift 1, Work Shift 2, and Work Shift 3 as shown in *Figure 2b*.



Convert from this... ... to this

Sample ID	Work Shift	Work Shift 1	Work Shift 2	Work Shift 3
1	1	1	0	0
2	1	1	0	0
300	2	0	1	0
301	2	0	1	0
600	3	0	0	1
601	3	0	0	1

Figure 2. *a) Discrete numerical data challenge of work shift; b) the data after one-hot encoding.*

We now have three vectors which capture the work shift as numerical data and the distance between each shift is one. The original Work Shift column is removed prior to executing the algorithm. One-hot encoding can be applied to character string data as well.

Once the data set is fully numeric and discrete variables have been managed, the issue of scale is addressed. The data collected in HPDC contains a wide range in scale. Also, different equipment manufacturers may capture data in only English or metric units. In round figures, intensification pressure of 10,000 psi, melt temperature of 1300 ˚F (704 ˚C), cycle time of 150 seconds, biscuit size of 2 inches, and an iron content of 0.60% are a few examples which show that the range of scale is in orders of magnitude. If left in this format, the intensification pressure would register as highly significant and outweigh any influence the iron content would show just the because the numbers are larger. The standardization method employed in this study is the Z-transform [23] (*Equation 1*), which brings all the variables into the same scale, resolves the issue of units, and leads to meaningful distances when considering multiple columns of data.

$$Z_{i,j} = \frac{X_{i,j} - \mu_j}{\sigma_j} \qquad Eq.\ 1$$

Where

- $Z_{i,j}$ is the Z-transformed value of the parameter in one data cell
- $X_{i,j}$ is the original value of the parameter in the data cell

- μ_j is the mean of the original values of the parameter in the data column
- σ_j is the standard deviation of the original values of the parameter in the data column.

Dimension Reduction Methods

The data is now in the proper format to analyze via machine learning or deep learning methods. It is possible that performing the analysis on the full dimensional dataset will not produce the best results. One can start with as much potentially relevant data as possible, but too much meaningless data adds noise to the model, such as redundant columns and those which are pure noise. To counteract this, dimension reduction methods, such as feature selection and principal component analysis (PCA) [24], [25], are conducted on the data. Both methods were implemented in this work.

Feature selection, as the name implies, is the specific selection of which inputs to run through the algorithm. The decisions are not made carelessly, rather, with the input and direction of a subject matter expert. In the absence of this resource, the decision can be made based on the feature importance [26] from the full dimensional model. The number of features ultimately selected is based on trial and error and the preferred performance metric.

PCA is another dimension reduction technique. The goal of PCA is to determine linear combinations of the input variables which capture the most variation in the dataset while minimizing the error when the dataset is reconstructed from the principal components. In doing so, a high-dimensional dataset can be condensed into a smaller number of principal components. PCA is an excellent tool for visualizing a high-dimensional dataset in two or three dimensions.

Bias-Variance Trade-Off

The above pre-processing methods apply to both the machine learning and Neural Network algorithms. When choosing which path to take, the two most important parameters that need to be considered are size of the available data and bias-variance trade-off. This dataset of 1494 tensile tests are exceedingly large when compared to typical mechanical property studies. However, in the world of data science, this is not “big data”. The amount of data available is a limiting factor in the complexity of the model.

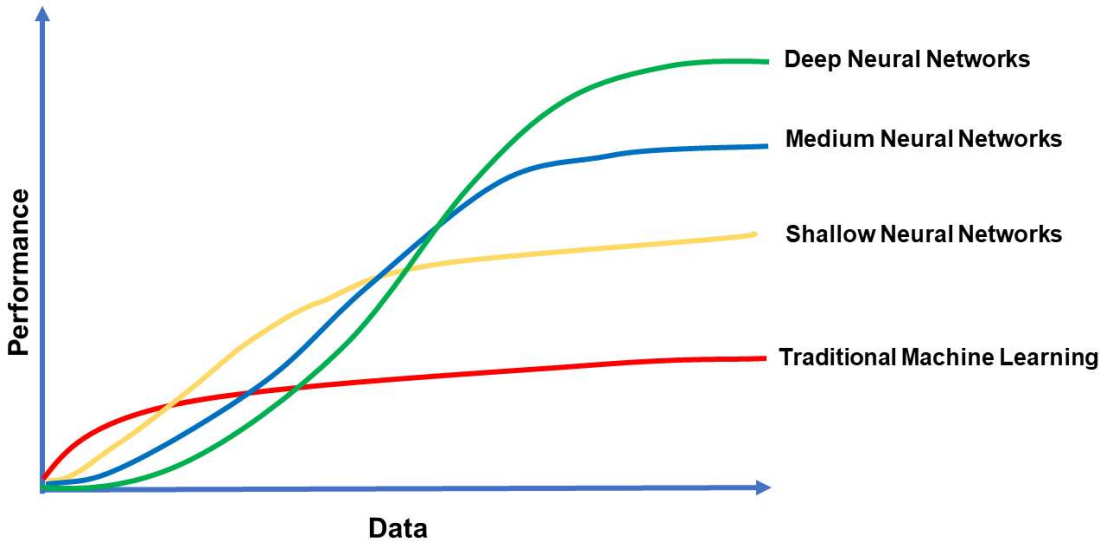


Figure 3. *Performance comparison of Neural Network models with traditional machine learning models as training data size increases. On smaller datasets, traditional algorithms outperform deep learning models however, as the amount of data increases, deep learning models perform better.*

Figure 3 shows a performance comparison of the models as data size increases. For smaller datasets, one would pick traditional algorithms as compared to deep learning models. However, as the quantity of data increases, deep learning models perform better because traditional algorithms reach a saturation point and do not improve any further whereas deep learning models performance keeps increasing with training data size [27].

Understanding the bias-variance trade-off is essential in deciding which algorithms to select for a particular dataset and application. In *Figure 4*, the X-axis shows model complexity and the Y-axis is predictive error. As model complexity increases, variance increases and bias decreases. An increase in the variance causes the model to overfit to the training data and it fails to generalize on new data. The left side of the plot shows a high bias but low variance region. This implies that the model is too simple and, hence, it is highly biased. It fails to learn the complexity of the data. The ideal point is where bias and variance intersect, as shown by the optimum model complexity in the plot below [10].

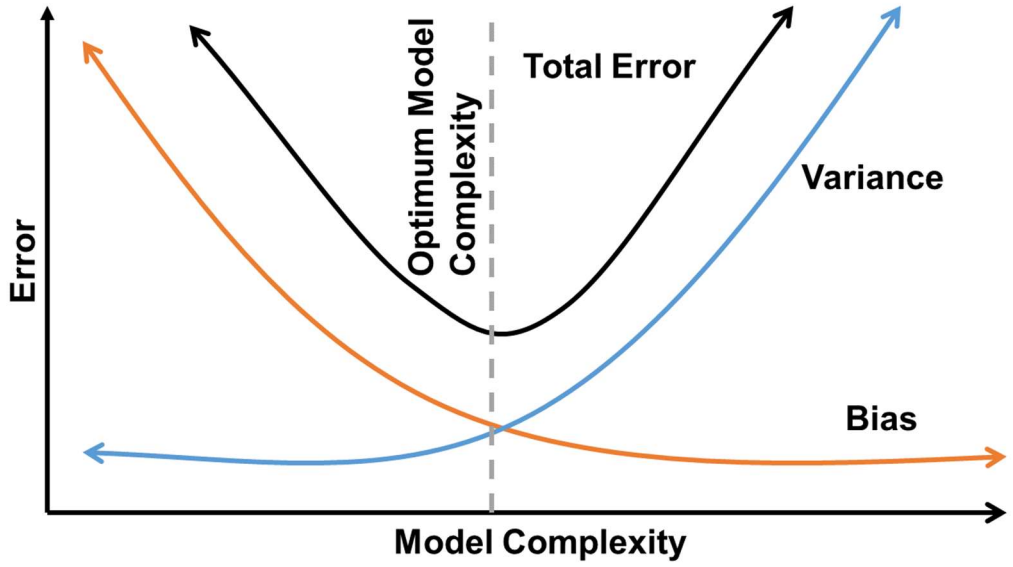


Figure 4. *Bias-variance trade-off [10], [11] shows how error changes as the complexity of the model increases. The region on the right is that of high variance and low bias whereas the region on the left is that of high bias and low variance. These regions are where the model overfits or underfits the training data and should be avoided. The optimal model complexity is where variance and bias are minimized, and one should utilize algorithms in this region.*

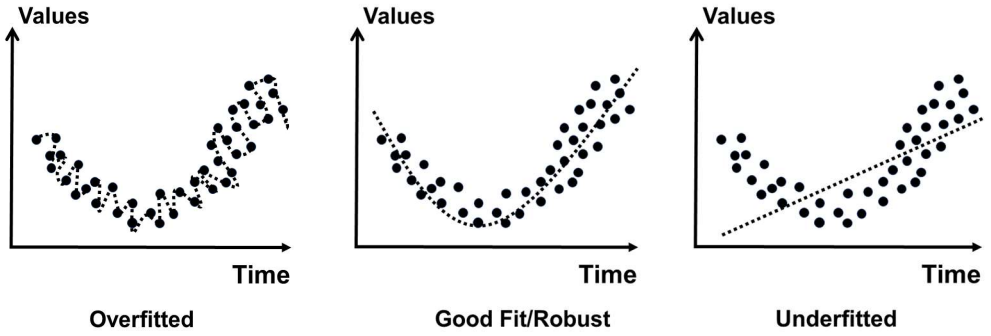


Figure 5. *The phenomenon of underfitting and overfitting is seen in this figure* [28]. *We want a model that is optimal for the kind of data and application that we are working on. For example, a good fit is illustrated in the center plot. The plots on the right and left show underfitting and overfitting respectively and should be avoided.*

Figure 5 shows the phenomena of overfitting and underfitting. It can be seen in the leftmost plot that the model follows the data very closely and, thus, overfits. This is the region of high variance in the bias-variance trade-off where the model will fail to generalize on testing data because it almost memorizes the training data. The middle figure shows the optimum model which corresponds to the lowest point of bias and variance in the bias variance trade-off and gives a robust fit to the data. The rightmost figure shows an example of high bias in the bias-variance trade-off. Here, the model fails to learn enough complexity in the dataset and underfits [11], [28].

Training, Testing, and Cross-Validation

Recognizing overfitting and underfitting in 2D plots like those shown in *Figure 5* is fine for illustrative purposes, but machine learning and deep learning are often applied to high dimensional datasets. In these problems, the fit of the model is evaluated by performance metrics comparison between the training data and faithful testing dataset which captures the essence of the complete dataset. To accomplish this, the dataset is split into two parts prior to analysis: a training set and a testing set. In this study a 90/10 training to testing split was most often employed. This allows the data scientist to train the model with a larger dataset and then test the model performance on a representative subset not previously seen by the algorithm.

The train/test split can influence the model. To avoid being misled, cross-validation is conducted to minimize the effect the split has on scoring the model performance metrics [29]–[31]. K-folds is a common method of cross-validation. In K-folds, the user sets the number of folds and the model is run as many times taking a different segment of the population as the testing data (*Figure 6*).

	MODEL 1	MODEL 2	MODEL 3	MODEL 4	MODEL 5
FOLD 1	TEST 1	TRAIN 1	TRAIN 1	TRAIN 1	TRAIN 1
FOLD 2	TRAIN 2	TEST 2	TRAIN 2	TRAIN 2	TRAIN 2
FOLD 3	TRAIN 3	TRAIN 3	TEST 3	TRAIN 3	TRAIN 3
FOLD 4	TRAIN 4	TRAIN 4	TRAIN 4	TEST 4	TRAIN 4
FOLD 5	TRAIN 5	TRAIN 5	TRAIN 5	TRAIN 5	TEST 5

Figure 6. *K-folds cross-validation where the number of folds is equal to five.*

Mean absolute error (MAE) values are reported to score the algorithm (*Equation 2*). It is common for some overfitting to the training data to exist in the model, so the error on the training data tends to be less than the test data. The goal of a robust model is to minimize the difference in error between the training data and testing data results.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right| \qquad Eq.\ 2$$

Where

- MAE is the mean absolute error
- n is the number of samples in the dataset
- Y_i is the actual value of the output
- $\hat{Y}_i$ is the predicted value of the output

III. Results and Discussion

First, a few comments are in order about machine learning and deep learning via neural network. It is generally agreed that both are forms of artificial intelligence (AI) rather than something entirely unique unto each other. Machine learning represents a family of methods which use statistical and probabilistic models trained on historical data to make predictions about new observations. In machine learning, feature

engineering, such as weighing one input more heavily or taking a logarithm of an input, is performed manually. Deep learning is similar, however, weight assignment to features is performed automatically by the algorithm.

For machine learning algorithms, Random Forest is selected because it works well with high-dimensional data, is robust to non-linear data, has low bias, and variance is reduced through bagging [32]. XGBoost was chosen to evaluate a more recent adaptation of Random Forest which, in addition to bagging, uses boosting to reduce bias by training the subsequent model on the errors of its predecessor. Bagging reduces overfitting while boosting improves accuracy at the cost of possible overfitting [33], [34]. SVM was chosen for its ability to determine non-linear decision functions via the *kernel trick.* The kernel trick maps the input data into a higher dimensional feature space where the data is linearly separable resulting in non-linear boundaries between the input data [35], [36]. These methods are compared to a Neural Network which is effective for handling nonlinearity, tolerant of noise, utilizes advanced learning methods, and generalizes well. The following sections cite numerous sources for the reader to delve further into the specifics of the methods chosen.

Results of Machine Learning Regression Methods

With the pre-processing complete, the machine learning algorithms can be run on the data. Since the objective is to predict the value of the UTS in extracted tensile bars based on the HPDC process parameters by which it was made, this is a regression problem [1]. Three algorithms were chosen: Random Forest, SVM, and XGBoost. The detail on how these algorithms operate can be found in these references [31], [37], [38]. For each of these methods, there are default parameters used when none are defined by the user. *Figure 7* shows the results in terms of the MAE in UTS prediction for the default models. For comparison, an additional model was evaluated where the UTS of the testing data is predicted by the mean UTS of the training data. The default setting for the Random Forest and XGBoost show significant overfitting where the error on the testing data exceeds that of the training data. The case of the default Random Forest illustrates well the danger of misuse. If this model were to be implemented, the expectation would be low error in predictions. However, the actual experience would show much higher error because the model is too specific to the training data.

The process of adjusting the controlling parameters within the algorithm is called tuning [39]. The chosen method of tuning selected for these models is Grid Search Cross-Validation (GSCV) [40]. In GSCV, multiple parameters can be tuned at once optimizing the model with respect to the target metric rather than each parameter at a time. The goal of tuning is to minimize the difference between the training and testing data results. *Figure 8* shows the improvement realized from tuning. In the Random Forest and the XGBoost the difference between the training and testing error decreases. The model becomes more general. The tuned SVM is not far from where the default parameters started.

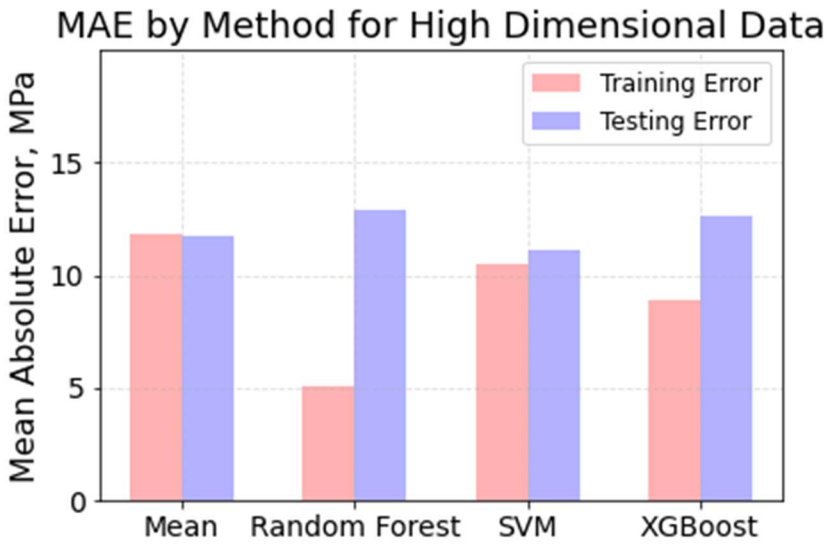


Figure 7. *MAE in UTS prediction results for the high dimension dataset using default settings. Both the Random Forest and the XGBoost are showing significant overfitting to the training data.*

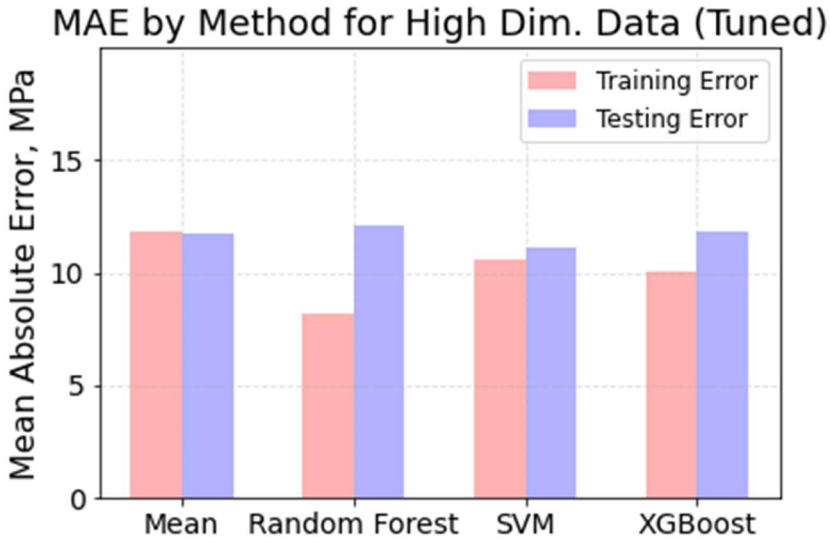


Figure 8. *MAE in UTS prediction results for the high dimensional dataset using tuned parameters. Compared to the default algorithm results in Figure 7, the amount of overfit in the Random Forest and XGBoost is lessened. The SVM improvement is imperceptible in the graph.*

The results presented thus far represent the algorithm performances on the full dataset containing all process input columns. Dimension reduction techniques were applied to the data to reduce noise of marginal features and further reduce the gap between the testing and training error. The Random Forest and XGBoost algorithms have an output called *feature importance* that shows which parameters have the most influence in training the model. The top 15 features from the tuned high dimensional Random Forest are shown in *Figure 9*. These features were selected as

the process inputs for the Random Forest and SVM. Beyond the top 15, the importance of additional features continues the gradual tailing off seen in *Figure 9*. The prediction results are shown in *Figure 10*. The Random Forest further reduced it overfitting.

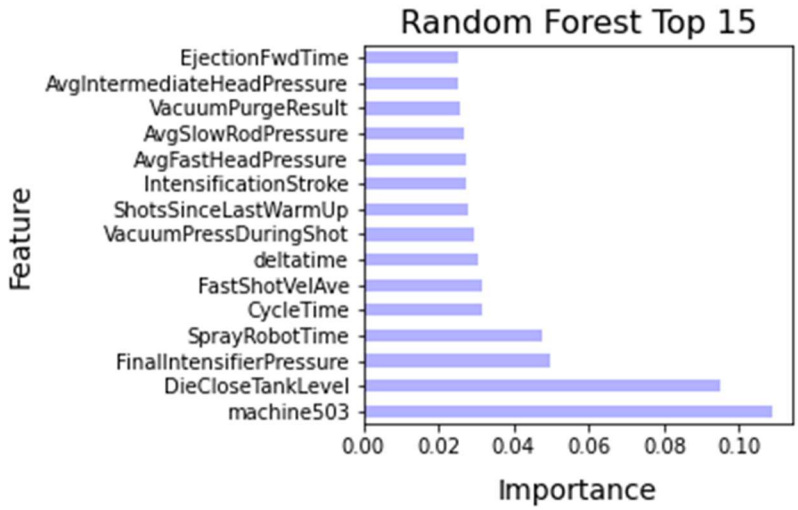


Figure 9. *Feature importance generated from the tuned Random Forest regressor. The top 15 features are shown.*

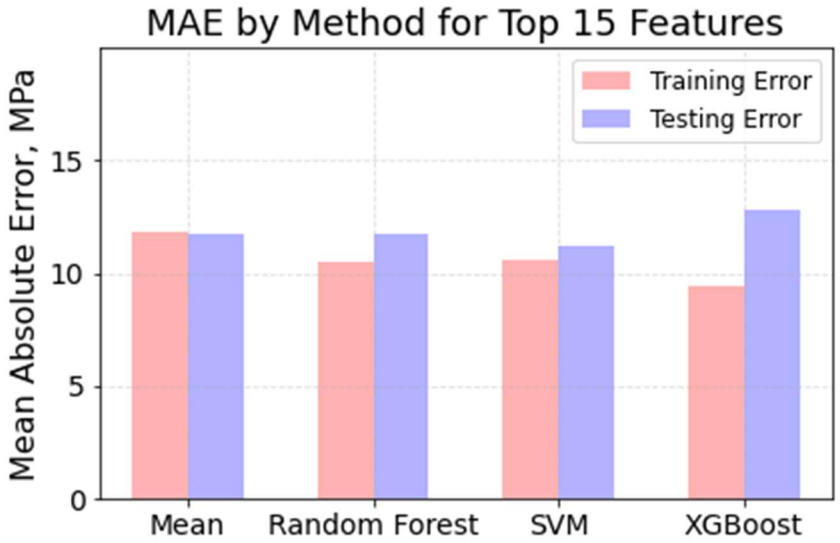


Figure 10. *Machine learning results on the feature selected dataset using the top 15 important features from the high-dimensional tuned model. Overfitting in the Random Forest is further reduced from the tuned model.*

Machine 503 is connected to the heat treat schedule including the natural age step which resulted in a statistically significant higher UTS than the standard heat treatment and the Random Forest was able to identify that as being important. The die close tank level variable refers to the fluid level in the hydraulic tank. It is showing up as important because of a highly positive correlation

to Machine 503 of 0.82. Beyond these two, the important features uncovered by the high dimensional tuned Random Forest look much like the parameters one finds in the literature when investigating the impact of process settings on mechanical properties or defects [41]–[45]. Based on this observation, a new feature selected dataset, "LitRev Features", was evaluated. The selected features are: Machine 503, average slow shot velocity, average fast shot velocity, average intermediate shot velocity, cycle time, intensification pressure, intensification pressure rise time, melt temperature, robot spray time (a proxy for amount of time the die was open between shots), and vacuum pressure during the shot. The predictive performance is displayed in *Figure 11*. The MAE for the training and testing data for all machine learning models dropped slightly with the biggest gain being in the XGBoost test error.

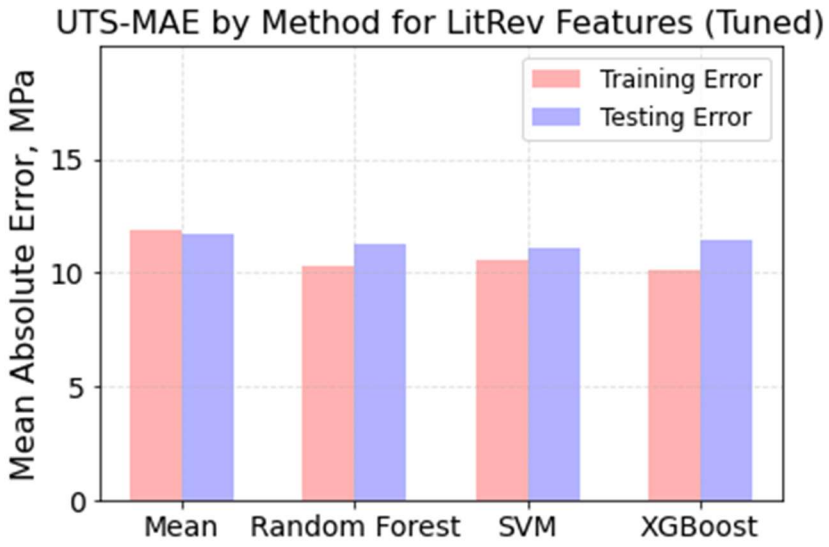


Figure 11. *Machine learning results on the feature selected dataset using important features from the literature. Small reductions in the training and testing error were found in all three algorithms with the most improvement in the XGBoost test data.*

Additionally, a different dimension reduction method, PCA, was applied to the high dimensional dataset. The number of principal components to explain 85% of the variation in the original dataset is 27. Each principal component is a linear combination of the original 77 dimensions, thus none of the inputs are completely dropped from the analysis as they are in feature selection. The PCA transformed data can be run through the same machine learning algorithms as the original data and the same tuning methods are employed. In *Figure 12*, the PCA Random Forest demonstrates the best performance overall in terms of UTS MAE and the degree of overfit.

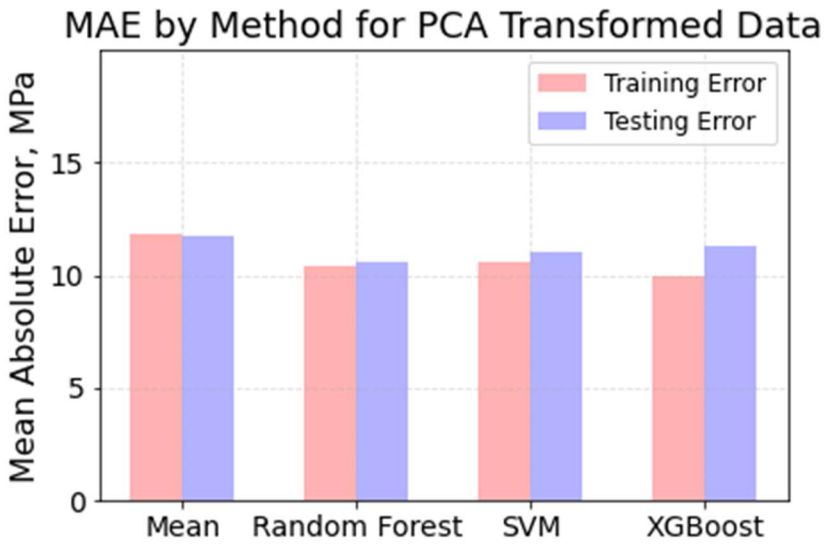


Figure 12. *Machine learning results on the PCA transformed dataset using top 27 principal components which explain 85% of the variation in the high-dimensional dataset. The Random Forest applied to the PCA dataset is the best machine learning performance in this study.*

Results of Neural Network Regression Method

Deep learning based Neural Networks have proved useful for advanced analytics of big manufacturing datasets [9]. In this section, we will show results of a Neural Network model for predicting the UTS and also show a comparison of the Neural Network with traditional state-of-the-art machine learning models namely, the Random Forest and XGBoost for the same dataset as shown above.

The plots below (*Figures 13-16*) demonstrate the significance of hyperparameter tuning in case of the Neural Network models [46]. The metric used for comparison is the same as that used in the machine learning section, i.e., the MAE (*Equation 2*). Selecting the right combination of parameters is critical for optimizing the target metric and reducing the overfitting phenomenon. We choose the parameters of the network in a way such that the model generalizes and does well on data that it has not seen during training.

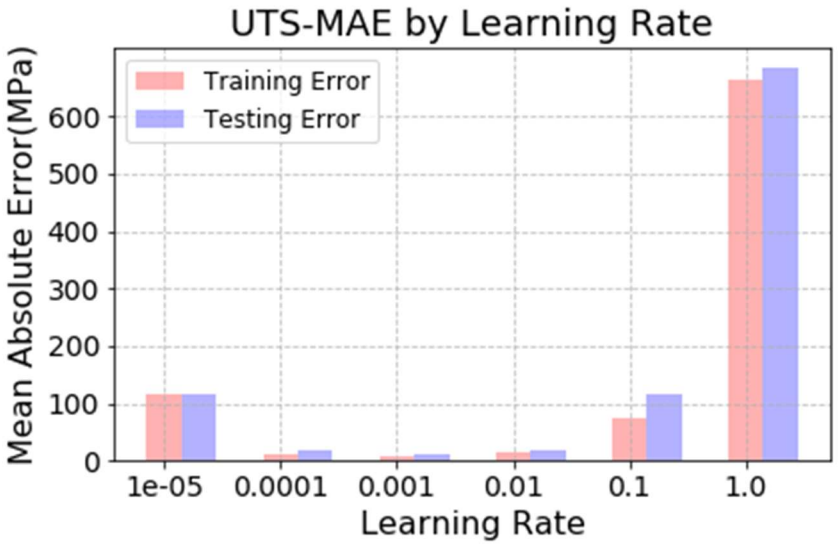


Figure 13. *Comparison of MAE across multiple learning rates of the Neural Network model. It can be seen that intermediate learning rates give the lowest errors as compared to lower or higher values.*

Figure 13 shows a comparison of the MAE values using different learning rates for the ADAM optimization technique [47]. The learning rate parameter should be chosen in a way such that it is low enough that the model is able to reach the minimum error solution, while at the same time, it should be high enough such that the model does not take excessive time to converge [48].

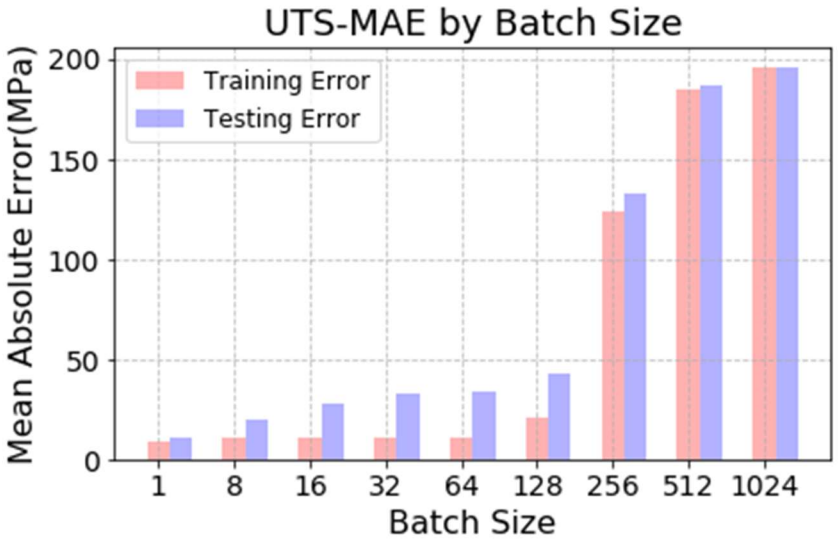


Figure 14. *Comparison of the MAE values with different batch sizes of the Neural Network. Smaller batch sizes show better performance as compared to higher sizes with this dataset.*

Figure 14 shows a comparison of the MAE values with respect to batch size. The results show that smaller batch sizes give lower error as compared to higher batch sizes. The difference in the MAE is more significant as the batch size is increased above 128. We train the model using

batches instead of training the entire data at once in order to make it computationally efficient and have other desirable properties such as avoiding local minima [46], [48]. We use a batch size of one for this analysis since it yields the lowest MAE as well as minimal difference between the training and testing errors.

Figure 15 shows a comparison of the MAE using different number of hidden layers [49]. It can be seen that using one hidden layer not only gives the best performance in terms of the MAE value but also gives the lowest difference between the training and the testing errors as compared to using higher number of hidden layers.

Hidden layers of a Neural Network are comprised of nodes, which are the basic units of a Neural Network. The hidden layer is where the learning of the data takes place which includes learning important features of the dataset; also, obtaining a compressed representation of the data. Contrast this with machine learning where this step is accomplished by human input during pre-processing. The complexity of the model increases as the number of hidden layers is increased.

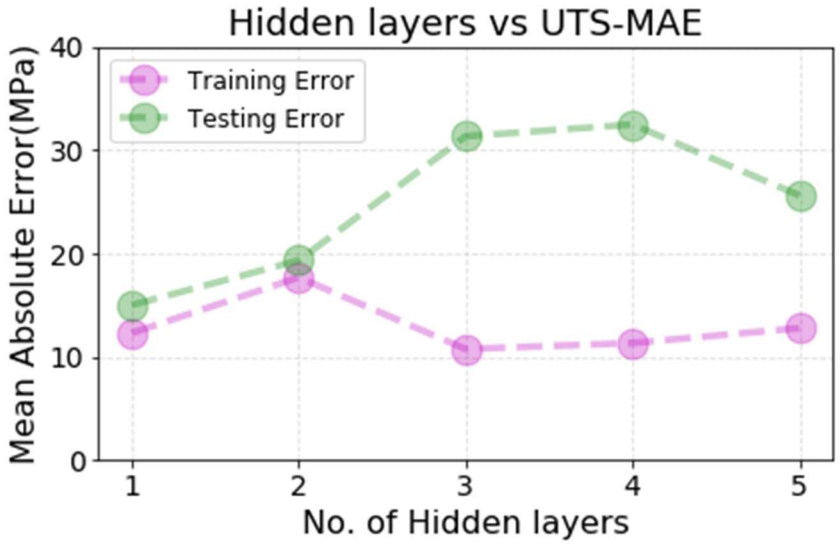


Figure 15. *A comparison of MAE as the number of hidden layers changes. The MAE value is lowest for one hidden layer as compared to higher number of hidden layers. Using one hidden layer optimizes the performance in terms of the metric itself and reduces overfitting.*

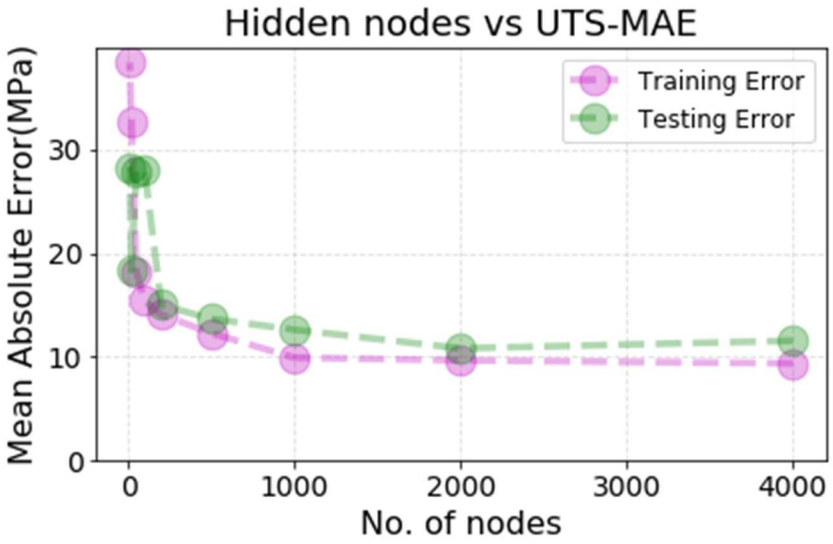


Figure 16. *Evaluation of MAE values in terms of number of nodes in the hidden layer. A higher number of nodes in the hidden layer performs better than fewer nodes.*

Figure 16 shows a comparison in terms of MAE with number of nodes in the hidden layer. A larger number of nodes gives lower errors as compared to lesser nodes in the hidden layer for this dataset. The difference between the training and testing errors is also low which shows that the model would generalize better on unseen data.

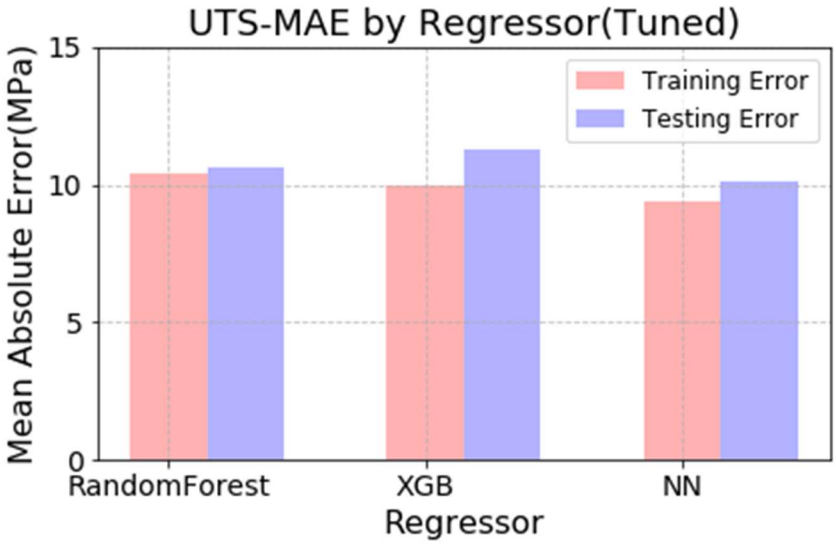


Figure 17. *Comparison of the optimized Neural Network model with optimized state-of-the-art traditional algorithms namely, Random Forest and XGBoost* [27]. *The Neural Network model gives the best performance in terms of the training as well as testing errors as compared to the traditional algorithms.*

Figure 17 shows a comparison of the optimized Neural Network model using the best combination of parameters with traditional machine learning algorithms namely, Random Forest and XGBoost.

The Neural Network gives the best performance in terms of MAE as compared to the other two models on this dataset.

Figures 12 and 17 illustrate that by using either traditional machine learning methods or a Neural Network we can reduce the error in predicting UTS below that of predicting the mean value. Our results demonstrate the importance of understanding the relationship between algorithm complexity and the predictive error on a particular dataset. In the context of *Figure 3*, the tensile dataset fits in the area where the traditional machine learning and shallow Neural Networks cross. It is crucial to appreciate the bias-variance trade-off for this relationship, so that we select the appropriate algorithm, with optimal parameters, to improve the predictive performance.

The dataset available for this research is typical and highly accessible for modern HPDC operations. Modern HPDC machines collect much of the data and make it available. In production data, the performance of the machine follows closely along with the process settings with some amount of natural variation. The objective is a repeatable cast result day in day out, 24/7. In contrast, experimental casting parameter studies explore a much wider range for the parameters under investigation to more clearly see a response in the cast component. The objective is to see a difference. In the present work, we saw improved prediction of UTS working with production data, but not a drastic improvement. Therefore, we must rethink about what other parameters can be included to increase our predictive power. Especially, consider parameters moving within a wide band because accurate control is difficult or costly. Perhaps there are parameters our industry has never considered measuring or controlling. It has been stated that one cannot control anything unless one has measures; the question is which measures? The question begs itself: are we measuring the correct parameters? One of the indirect key results of AI is the realization that perhaps what we have been measuring in the past is not appropriate, and that there are other key parameters that we should be capturing.

The temperature of the die cavity where the metal is solidified into its final shape is perhaps the most influential input parameter that is to a large extent passively controlled. Most die casters rely on a condition of *steady state*, which is a somewhat nebulous combination of cycle time, dwell time, die spray application parameters, cooling water temperature and flow rate, the melt temperature and amount of metal delivered during each shot, alloy chemistry, and the ambient environment in the factory. If all of these are constant, then the die temperature will take care of itself at steady state. Readers can decide for themselves if this is realistic. Miller demonstrated in a 1-D model to challenge common notions of how many cycles it takes to attain a quasi-steady state [50]. Other studies, especially modeling based investigations, have shown that die temperature is a high impact parameter on castings and the dies themselves [51]–[54]. For these reasons, a robust and reliable method of collecting the die temperature is the next source of data to drive predictive modeling forward. Knowing how and what to measure will lead the industry toward active control of die cavity temperature [55], [56].

IV. Conclusions

- Machine learning and Neural Network regression models utilizing HPDC process data as inputs can improve the predictability of UTS above that of predicting the mean from prior tests. It is reasoned that the predictive power can be improved by increasing the number of rows and adding new input data columns.

- Principal component analysis is an effective dimension reduction technique to reduce complexity and overfitting of a dataset. A Random Forest of a PCA transformed dataset was the top performing machine learning method in this study.
- To optimize the models, parameter tuning must be performed with the objective of minimizing the error in the model predictions as well as the difference between training data and testing data errors.
- It can be seen that given the right combination of parameters for a Neural Network such as learning rate, batch size and number of hidden layers, the predictive performance of a Neural Network can be optimized not only in terms of the error metric but also in terms of obtaining a robust model fit for a given dataset without overfitting or underfitting.
- Selecting the correct models to use for the data being considered requires an understanding of the bias-variance trade-off such that a balance is struck between algorithm complexity and size of the dataset in question.

V. References

[1] D. Dietrich, B. Heller, and B. Yang, *Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data*, 1st ed. Wiley, 2015.

[2] Capgemini Consulting Group, "Industry_4.0_-The_Capgemini_Consulting_V.pdf." Capgemini, 2014, [Online]. Available: https://www.capgemini.com/consulting/wp-content/uploads/sites/30/2017/07/capgemini-consulting-industrie-4.0_0_0.pdf.

[3] J. Folk, "U.S. Aluminum Casting Industry - 2019," *Cast. Eng.*, no. July 2019, pp. 16–19, Jun. 2019.

[4] A. Spada, "Revitalization of North American Metalcasting," 2012, Accessed: May 24, 2020. [Online]. Available: https://www.diecasting.org/docs/statistics/North_America.pdf.

[5] J. K. Rai, A. M. Lajimi, and P. Xirouchakis, "An intelligent system for predicting HPDC process variables in interactive environment," *J. Mater. Process. Technol.*, vol. 203, no. 1–3, pp. 72–79, Jul. 2008, doi: 10.1016/j.jmatprotec.2007.10.011.

[6] P. K. D. V. Yarlagadda and E. Cheng Wei Chiang, "A neural network system for the prediction of process parameters in pressure die casting," *J. Mater. Process. Technol.*, vol. 89–90, pp. 583–590, May 1999, doi: 10.1016/S0924-0136(99)00071-0.

[7] R. Soundararajan, A. Ramesh, S. Sivasankaran, and A. Sathishkumar, "Modeling and Analysis of Mechanical Properties of Aluminium Alloy (A413) Processed through Squeeze Casting Route Using Artificial Neural Network Model and Statistical Technique," *Adv. Mater. Sci. Eng.*, vol. 2015, pp. 1–16, 2015, doi: 10.1155/2015/714762.

[8] J. Friedman, R. Tibshirani, and T. Hastie, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

[9] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, "Deep learning for smart manufacturing: Methods and applications," *J. Manuf. Syst.*, vol. 48, pp. 144–156, Jul. 2018, doi: 10.1016/j.jmsy.2018.01.003.

[10] E. Briscoe and J. Feldman, "Conceptual complexity and the bias/variance tradeoff," *Cognition*, vol. 118, no. 1, pp. 2–16, Jan. 2011, doi: 10.1016/j.cognition.2010.10.004.

[11] "Bias-Variance Tradeoff in Machine Learning," *AI Pool*, Oct. 20, 2019. https://ai-pool.com/a/s/bias-variance-tradeoff-in-machine-learning (accessed Jun. 02, 2020).

[12] The Aluminum Association, *Designations and Chemical Composition Limits for Aluminum Alloys in the Form of Castings and Ingot*, October 2018. Arlington, VA: The Aluminum Association, 2018.

[13] D. Twarog, D. Apelian, and A. Luo, *High Integrity Casting of Lightweight Components*, Publication #307. NADCA, 2016.

[14] J. G. Kaufman and E. L. Rooy, *Aluminum Alloy Castings Properties, Processes, and Applications*, 1st ed. ASM, 2004.
[15] ASTM International, "ASTM B 557-15, Test Methods for Tension Testing Wrought and Cast Aluminum- and Magnesium-Alloy Products." ASTM International, doi: 10.1520/B0557-15.
[16] W. D. Callister, *Materials Science and Engineering An Introduction*, 3rd ed. Wiley, 1994.
[17] M. Drouzy, S. Jacob, and M. Richard, "Interpretation of Tensile Results by Means of Quality Index and Probable Yield Strength," *AFS Int. Cast Met. J.*, no. June 1980, pp. 43–50, 1980.
[18] L. M. Surhone, M. T. Timpleton, and S. F. Marseken, *Welch's T Test*. VDM Publishing, 2010.
[19] M. K. Surappa, E. Blank, and J. C. Jaquet, "EFFECT OF MACRO-POROSITY ON THE STRENGTH AND DUCTILITY OF CAST," *Scr. Metall.*, vol. 20, no. 9, pp. 1281–1286, 1986, doi: 10.1016/0036-9748(86)90049-9.
[20] C. H. Caceres and B. I. Selling, "Casting defects and the tensile properties of an Al-Si-Mg alloy," *Mater. Sci. Eng. A*, vol. 220, pp. 109–116, 1996, doi: 10.1016/S0921-5093(96)10433-0.
[21] D. L. Twarog, "State of the Die Casting Industry," *Cast. Eng.*, no. January, pp. 16–25, 2011.
[22] K. Potdar, T. S., and C. D., "A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers," *Int. J. Comput. Appl.*, vol. 175, no. 4, pp. 7–9, Oct. 2017, doi: 10.5120/ijca2017915495.
[23] "Z-Transform," *Wolfram MathWorld*. https://mathworld.wolfram.com/Z-Transform.html (accessed May 26, 2020).
[24] S. Wold, K. Esbensen, and P. Geladi, "Principal Component Analysis," *Chemom. Intell. Lab. Syst.*, vol. 2, pp. 37–52, 1987, doi: 10.1016/0169-7439(87)80084-9.
[25] H. Abdi and L. J. Williams, "Principal component analysis: Principal component analysis," *Wiley Interdiscip. Rev. Comput. Stat.*, vol. 2, no. 4, pp. 433–459, Jul. 2010, doi: 10.1002/wics.101.
[26] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, "Permutation importance: a corrected feature importance measure," *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, Apr. 2010, doi: 10.1093/bioinformatics/btq134.
[27] A. Oppermann, "Artificial Intelligence vs. Machine Learning vs. Deep Learning," *Towards Data Science*, Oct. 29, 2019. https://towardsdatascience.com/artificial-intelligence-vs-machine-learning-vs-deep-learning2210ba8cc4ac (accessed Jun. 08, 2020).
[28] A. Bhande, "What is underfitting and overfitting in machine learning and how to deal with it," *medium.com*, Mar. 11, 2018. https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-howto-deal-with-it-6803a989c76 (accessed Jun. 08, 2020).
[29] T. Fushiki, "Estimation of prediction error by using K-fold cross-validation," *Stat. Comput.*, vol. 21, no. 2, pp. 137–146, Apr. 2011, doi: 10.1007/s11222-009-9153-8.
[30] M. Sanjay, "Why and how to Cross Validate a Model?," *Towards Data Science*, Nov. 12, 2018. towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f (accessed Jun. 04, 2020).
[31] A. Geron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*, 1st ed. O'Reilly, 2017.
[32] L. Breiman, "Random Forests," *Mach. Learn.*, vol. 45, no. October 2001, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.
[33] T. G. Dietterich, "An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization," *Mach. Learn.*, vol. 40, no. August 2000, pp. 139–157, 2000, doi: https://doi.org/10.1023/A:1007607513941.
[34] A. Vezhnevets and O. Barinova, "Avoiding Boosting Overfitting by Removing Confusing Samples," in *Machine Learning: ECML 2007*, Berlin, Heidelberg, 2007, pp. 430–441. ISBN: 978-3-540-74958-5
[35] A. Apsemidis, S. Psarakis, and J. M. Moguerza, "A review of machine learning kernel methods in statistical process monitoring," *Comput. Ind. Eng.*, vol. 142, Apr. 2020, doi: 10.1016/j.cie.2020.106376.

[36] M. Hofmann, “Support Vector Machines — Kernels and the Kernel Trick.” 2006, Accessed: Jul. 05, 2020. [Online]. Available: https://cogsys.uni-bamberg.de/teaching/ss06/hs_svm/slides/SVM_Seminarbericht_Hofmann.pdf.
[37] T. K. Ho, “Random decision forests,” in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, Montreal, Quebec, Canada, 1995, vol. 1, pp. 278–282, doi: 10.1109/ICDAR.1995.598994.
[38] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 785–794, Aug. 2016, doi: 10.1145/2939672.2939785.
[39] R. G. Mantovani, T. Horváth, R. Cerri, J. Vanschoren, and A. C. P. L. F. d. Carvalho, “Hyper-Parameter Tuning of a Decision Tree Induction Algorithm,” in *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*, Oct. 2016, pp. 37–42, doi: 10.1109/BRACIS.2016.018.
[40] D. Krstajic, L. J. Buturovic, D. E. Leahy, and S. Thomas, “Cross-validation pitfalls when selecting and assessing regression and classification models,” *J. Cheminformatics*, vol. 6, no. 1, p. 10, Dec. 2014, doi: 10.1186/1758-2946-6-10.
[41] L. Garber and A. B. Draper, “The Effects of Process Variables on the Internal Quality of Aluminum Die Castings,” *NADCA Trans. T79-022*, 1979, [Online]. Available: http://www.diecasting.org/archive/transactions/T79-022.
[42] B. M. Asquith, “The Use of Process Monitoring to Minimize Scrap in the Die Casting Process,” *NADCA Trans. T97-063*, 1997, Accessed: May 25, 2020. [Online]. Available: http://www.diecasting.org/archive/transactions/T97-063.pdf.
[43] S. L. dos Santos, R. A. Antunes, and S. F. Santos, “Influence of injection temperature and pressure on the microstructure, mechanical and corrosion properties of a AlSiCu alloy processed by HPDC,” *Mater. Des.*, vol. 88, pp. 1071–1081, Dec. 2015, doi: 10.1016/j.matdes.2015.09.095.
[44] H. Cao, M. Hao, C. Shen, and P. Liang, “The influence of different vacuum degree on the porosity and mechanical properties of aluminum die casting,” *Vacuum*, vol. 146, pp. 278–281, Dec. 2017, doi: 10.1016/j.vacuum.2017.09.048.
[45] I. Outmani, L. Fouilland-Paille, J. Isselin, and M. El Mansori, “Effect of Si, Cu and processing parameters on Al-Si-Cu HPDC castings,” *J. Mater. Process. Technol.*, vol. 249, pp. 559–569, Nov. 2017, doi: 10.1016/j.jmatprotec.2017.06.043.
[46] L. N. Smith, “A disciplined approach to neural network hyper-parameters: Part 1 -- learning rate, batch size, momentum, and weight decay,” *ArXiv180309820 Cs Stat*, Apr. 2018, Accessed: Jun. 08, 2020. [Online]. Available: http://arxiv.org/abs/1803.09820.
[47] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *ArXiv14126980 Cs*, Jan. 2017, Accessed: Jun. 23, 2020. [Online]. Available: http://arxiv.org/abs/1412.6980.
[48] Wen Jin, Zhao Jia Li, Luo Si Wei, and Han Zhen, “The improvements of BP neural network learning algorithm,” in *WCC 2000 - ICSP 2000. 2000 5th International Conference on Signal Processing Proceedings. 16th World Computer Congress 2000*, Beijing, China, 2000, vol. 3, pp. 1647–1649, doi: 10.1109/ICOSP.2000.893417.
[49] J. de Villiers and E. Barnard, “Backpropagation neural nets with one and two hidden layers,” *IEEE Trans. Neural Netw.*, vol. 4, no. 1, pp. 136–141, Jan. 1993, doi: 10.1109/72.182704.
[50] R. A. Miller, “Multi-time Scale Systems and Quasi Equilibrium,” *NADCA Trans. T16-082*, 2016, [Online]. Available: https://www.diecasting.org/archive/transactions/T16-082.pdf.
[51] S. Shahane, N. Aluru, P. Ferreira, S. G. Kapoor, and S. P. Vanka, “Optimization of solidification in die casting using numerical simulations and machine learning,” *J. Manuf. Process.*, vol. 51, pp. 130–141, Mar. 2020, doi: 10.1016/j.jmapro.2020.01.016.
[52] W. Sequeira, S. Sikorski, and M. Brown, “Application of Simulation As A Front-End Design Tool In Die Cast Product Development And For The Optimization Of The Die Casting Process,” *NADCA Trans. T02-012*, 2002, [Online]. Available: https://www.diecasting.org/archive/transactions/T02-012.pdf.

[53] D. Blondheim, "Unsupervised Machine Learning and Statistical Anomaly Detection Applied to Thermal Images," *NADCA Trans. T18-071*, 2018, [Online]. Available: http://www.diecasting.org/transactions/T18-071.
[54] B. Kosec, G. Kosec, and M. Soković, "Case of temperature field and failure analysis of die-casting die," *J. Achiev. Mater. Manuf. Eng.*, vol. 20, pp. 471–474, 2007, doi: 10.1.1.526.5501.
[55] W. Bishenden and R. Bhola, "Die Temperature Control," *NADCA Trans. T99-051*, 1999, [Online]. Available: https://www.diecasting.org/archive/transactions/T99-051.pdf.
[56] D. Schwam, "Additive Manufacturing of Cores with Conformal Cooling Lines," *NADCA Trans. T16-041*, 2016, [Online]. Available: http://www.diecasting.org/transactions/T16-041.

VI. Acknowledgements

The authors of this paper would like to thank the membership of the ACRC consortium at WPI and UCI. We extend our thanks to FCA, especially Corey Vian and the Kokomo Casting Plant, for providing the data for this project. FCA is a long-term member of the ACRC consortium.

On behalf of all authors, the corresponding author states that there is no conflict of interest.

APPENDIX A: Processing Data

Table A-I. HPDC process data used in the analyses.

Variable Name	Description
MachineID	Die casting machine on which the part was cast. One hot encoding expands this one column into 12 columns, one per machine.
SerialNumb	Unique identifier given to the casting when it is made. Used to merge process and tensile testing datasets.
AvgFastHeadPressure	Average pressure reading on the head side of the shot cylinder during fast shot.
AvgFastRodPressure	Average pressure reading on the rod side of the shot cylinder during fast shot.
AvgIntermediateHeadPressure	Average pressure reading on the head side of the shot cylinder during intermediate shot.
AvgIntermediateRodPressure	Average pressure reading on the rod side of the shot cylinder during intermediate shot.
AvgSlowHeadPressure	Average pressure reading on the head side of the shot cylinder during slow shot.
AvgSlowRodPressure	Average pressure reading on the rod side of the shot cylinder during slow shot.
BiscuitLength	The thickness of the biscuit calculated based on the end of stroke position of the shot rod.
CavityFillTime	The time taken to fill the part geometry cavity in the die. Calculated from CavityFillTimeWinStartPos until the end of the shot velocity is detected.
DieCloseTankLevel	Level of the hydraulic fluid reservoir
DieCloseTankTemp	Temperature of the hydraulic fluid reservoir.
EndofShotPosition	Position where fast shot velocity decelerates to the end of shot velocity.
FastShotVelAve	Calculated average shot velocity at which the plunger moved forward during fast shot.
FinalIntensifierPressure	Maximum pressure applied to the biscuit during intensification phase.
IntensificationStroke	Amount of plunger forward movement after intensification is initiated.
IntensPressRiseTime	Time measured to reach the programmed intensification pressure.
IntensVelRiseTime	Calculated average velocity at which the plunger moved forward during intensification rise window.
IntermediateVelAve	Calculated average shot velocity at which the plunger moved forward during intermediate shot.
MetalTemp	The temperature of the molten alloy in the holding furnace at the die cast cell.
SlowShotVelAve	Calculated average shot velocity at which the plunger moved forward during slow shot.
TieBarTon1	Tons of force measured by the load cell on tie bar #1 when the die is closed and locked.
TieBarTon2	Tons of force measured by the load cell on tie bar #2.
TieBarTon3	Tons of force measured by the load cell on tie bar #3.

TieBarTon4	Tons of force measured by the load cell on tie bar #4.
TieBarTonTotal	Sum of the tonnage of all four tie bars.
TipLubeTimePre	Programmed time for which tip lube is applied to the plunger tip.
CycleTime	Elapsed time for the entire process to produce one piece.
Dwell Time	Elapsed time between end of shot and die open.
Die Open Time	Elapsed time to open the die.
Extract Robot Time	Elapsed time for the extract robot to complete its full cycle.
Spray Robot Time	Elapsed time for the spray robot to complete its full cycle.
Liner Load Time	Elapsed time to load cast in liners into the die.
Core Insert Time	Elapsed time to insert core feature into the die.
Die Close Time	Elapsed time to close the die.
Ladle Pour Time	Elapsed time for the ladle to pour molten alloy into the cold chamber.
Shot Delay Time	Elapsed time between pour complete and shot forward.
VacuumPressDuringShot	Measured vacuum pressure.
VacuumPurgeResult	Measured pressure when clearing the vacuum chill block of debris.
deltatime	Time elapsed between cycle start to the next cycle start.
Shots Since Last Warm Up	Number of cycles since the last warm-up shot.
Cavity	The identification number of the die cavity. One hot encoding expands this one column into 20 columns, one per cavity.
Model	The identification of the block casting model. One hot encoding expands this one column into 4 columns, one per model.