Improving Employer Outcomes: Supervised Learning to Predict Hiring Success

A Major Qualifying Project by:

Emily Bigwood
Owen Chace
Ranier Gran
Stephanie Martin

Submitted in partial fulfillment
of the requirements for the degree of:

Bachelor of Science

Mathematical Sciences
Worcester Polytechnic Institute
March 17, 2018

Approved by:

Professor Randy C. Paffenroth

Abstract

The Career Development Center (CDC) at Worcester Polytechnic Institute (WPI) is an office which provides students with resources and services to assist students in finding a career path after graduation, which includes connecting employers and students. As such, they wish to understand what activities are most influential to a company's success in hiring talent from WPI. Our team analyzed data provided by the CDC about online job postings, on-campus interviews, information sessions, and career fair attendance along with information about student outcomes after graduation to determine what contributed most significantly to employers' success. Because we wanted to be "fair" to companies in our predictions, we decided to minimize mean squared relative (percent) error, rather than traditional mean squared error. This required us to develop a regression tree completely from scratch to then use in random forest regression.

Acknowledgements

First, we would like to thank our sponsor and Director of Corporate Relations at the Worcester Polytechnic Institute Career Development Center (CDC), Dave Ortendahl. Throughout this project, Dave was incredibly helpful. His initial questions were instrumental in helping us define the scope of our project. In addition, he provided us with feedback during important stages of the project and offered valuable insights.

We would also like to thank Allyson Bernard, Senior Recruiting Coordinator at the CDC. Allyson was one of our main points of contact. She took on the tremendous task of cleaning, organizing, and providing us with relevant CDC datasets. We greatly appreciate her willingness meet with us to answer our questions, especially during the beginning of our project.

This research was performed using computational resources supported by the Academic & Research Computing group at Worcester Polytechnic Institute. The authors would also like to acknowledge the Academic & Research Computing group for providing coding support that contributed to the results reported within this paper. Namely, Spencer Pruitt and James Kingsley from WPI's Academic Technology Center (ATC) provided us with essential support in this area, meeting with us on multiple occasions to offer their experience using the ACE Cluster.

Finally, we would like to thank our MQP advisor Professor Randy Paffenroth. Professor Paffenroth provided us with invaluable advice and ensured our project stayed on task. He helped us overcome a number of obstacles throughout the course of our MQP. His guidance, optimism, and enthusiasm were essential to the success of our project.

Contents

List of Figures

List of Tables

Executive Summary

One of the most important outcomes of a student's college education is internship and post-graduate job prospects. Employment and internship opportunities are a priority of the Worcester Polytechnic Institute (WPI) Career Development Center (CDC). WPI's Career Development Center works with companies of varying sizes and backgrounds to help students obtain both internships and fulfilling, high-paying jobs after graduation. These companies engage in a number of activities at WPI, ranging from on-campus interviews to information sessions to annual career fairs. However, while the CDC records a company's presence on campus, there has not yet been a dedicated effort to analyze company data in depth using supervised machine learning methods. Although the CDC can summarize a company's presence in a given year, additional questions exist regarding steps a company can take in order to most effectively recruit and hire students.

The goal of this project was to analyze data from the 2015-16 school year, provided by the CDC, using supervised machine learning methods. This included datasets with information about student internships, full- and part-time hires, and employer activities (such as attendance at career fairs and preferences in their job posts). First, we had to appropriately encode, merge, and clean these datasets so that we could subsequently consider and test several possible regression methods to fit the data. After doing so, we concluded that decision trees (used in random forest regression) would best suit our purposes. This method produced a greater error than support vector machine (SVM) regression in our tests with built-in scikit-learn functions, but we decided to use it because it allows us to interpret the importance of different features much more easily than we could SVM.

After some consideration, we decided that we did not want our tree to minimize mean-squared error (MSE) as is the convention but mean-squared relative error (MSRE) i.e. squared percent error (the square being important for differentiability). This is because there is a fairly large range in the number of graduates that companies hire. While most companies that hire WPI graduates hire 1 or 2 students, there are many that hire at least 10, and one company with over 25 hires for the 2016 data. For this reason, minimizing raw error would be in a way 'unfair' for companies with fewer hires. For example, if our model adhered to the convention of using MSE, an estimate of 22 for a company with 20 hires would be considered as big an error as estimating 3 for a company that hired only 1 graduate. Clearly, in this context the former is a relatively minor error, while the latter seems drastic. While our modified random forest regression model was built using MSRE, our final error metric remained mean relative error (MRE) as this is much easier to interpret and is simply the root of our minimizing metric. A method with an MRE of 0.3 or 30 percent error would predict values of 1.3 for companies who hired a single employee and predict 13 for a company hiring 10 employees, providing a "fair" metric for companies of all sizes.

Changing the error function over which to minimize a regression tree is not a trivial task, and we were unable to find any existing implementations of a regression tree which minimizes MSRE. For example, the only two error metric options presented in the scikit-learn regression tree are MSE and mean absolute error (MAE). [11] Thus, in order to use MSRE, we needed to write and implement our own regression tree and random forest completely from scratch in Python. This required

several helper functions to be written to perform each of the important steps in creating a tree. One of the most important functions, the one which determined an appropriate estimate for a node/leaf in order to minimize MSRE, was dependent on a derivation that had to be performed by hand first (shown in subsection 6.1). In order to overcome time constraints and reduce runtime with a sufficient amount of data, we made use of WPI's Ace Cluster. The Ace Cluster is a high-performance computing system with multiprocessing capabilities. With the help of WPI staff working for the Academic Technology Center (ATC), we were able to simultaneously run several copies of our existing Jupyter notebook non-interactively on the cluster. Additional code was added before running to write important results from each run to csv files.

After running our code using WPI's Ace Cluster, we cross-validated the results to find the best number of learners (trees) to use in our forest. Additionally, we cross-validated to determine ideal values for minimum and maximum thresholds of error. Trees with more than the maximal threshold were thrown out due to their inaccuracy, and trees with less than the minimum threshold for error on the training data were rejected for overfitting. From this, we concluded that we should accept trees with a training error of at least 24 percent and at most 50 percent. The ideal number of learners was determined to be 116 trees with a 0.3066 mean relative error. However, this is likely only due to our limited amount of data —because of computational time, we only generated 140 trees and did not see a 'spike' in the graph of number of learners vs. error after 116.

The mean relative errors of the models that we considered in this project ranged from 0.2617 for SVM to 0.7407 for Linear Regression, as shown in Table 1 below. We attempted to predict both the unstandardized and standardized number of hires, but the errors when predicting the standardized hires were much higher. Thus, when comparing the models, we looked at the errors for predicting the unstandardized hires.

Model	MRE (unstandardized)	MRE (standardized)
Linear Regression	0.7407	2.526
RFECV with Linear Reg.	0.6977	1.422
SVM	0.2617	1.456
KNN	0.5156	1.461
Original Random Forest	0.5575	1.782
Modified Random Forest	0.3066	2.153

Table 1: Test errors of each model

While SVM had the lowest mean relative error, it was not easily interpretable. Instead we analyzed the results from both recursive feature elimination with cross validation (RFECV) models and both Random Forest models to find the best and worst predictors of hiring success. The first RFECV Linear Regression model had three predictors, Job: Degree Level Master of Physics for Educ, Job: Degree Level Master of Mathematics for Educ, and Job: Total Student Views, but we considered the ten predictors leading up to the final model in our feature importance analysis. The second RFECV model had 12 predictors total, and we considered all twelve as equally important in our analysis.

One predictor, OCI Interviews, stood out as the most important in the original

Random Forest model. The predictor had a feature importance score of 0.389172, while the second most important feature, 10000+ Emp, had a score of 0.0655776. The feature importance scores for all the predictors summed to 1.00 for perspective. Despite this large gap in scores, we still considered the top ten predictors from the model in our feature importance analysis. As for the Modified Random Forest model, we averaged the mean relative errors of the trees using each predictor and considered the ten predictors with the lowest average mean relative errors.

After gathering the top predictors from each model, we found six predictors that were in the top ten for at least two models, shown in Table 2. Two of them, OCI Interviews and 10,000+ Emp, were in the top ten for three of the models. We also observed a strong positive correlation between employer size and number of hires when analyzing the raw data. Companies with 10,000+ employees hired 38.73% of all WPI students hired in 2016 and 45.85% of the students that reported having an internship that year. Location also played a large role in which companies hired students, as more than half of the companies that hired at least one student had an office in Massachusetts.

Predictor	Top 10 Appearances
OCI Interviews	3
10000+ Emp	3
Job Posts	2
2015 Fall	2
Job: Resume Submission Method Accumulate in WPI Job Finder	2
Job: Degree Level Doctor of Philosophy	2

Table 2: Top predictors of hiring success

For future projects, we have a few recommendations. First, in order to allow for easier analysis, the CDC should work towards improving their data collection and data storage methods, such as having consolidated data profiles for each employer and keeping employer names and identifiers consistent between spreadsheets. Second, it would be helpful to have internship/summer plans data that is as comprehensive as the post-graduate survey data to find better predictors of internship hires. Third, we recommend a similar analysis be performed on both previous and future data sets. Previous data sets could confirm our findings, while future data sets could be used to analyze the use of Handshake over Job Finder. Finally, we recommend simplifying some of the features used in this project, such as eliminating the major-specific features in order to better analyze the importance of degree level. Doing so would allow for cleaner and potentially more accurate results/feature importance analysis.

1 Introduction

For many college students, the number one goal upon graduating is getting a job. Students spend their senior year researching companies, going to career fairs, and scouring job boards for entry-level positions. What makes a student more likely to apply to certain companies and what do employers do to get the most hires each year? In this project, we use data analysis and machine learning techniques to find how active employers should be at Worcester Polytechnic Institute (WPI), and in which particular areas, in order to maximize the number of talented students they hire from the university.

Throughout the year, WPI's Career Development Center (CDC) collects post-graduation and hiring data from students and alumni through various online and paper surveys. Information gathered from these forms ranges from internship, co-op, and full-time employment to graduate school and other post-graduation plans. The CDC also keeps track of which employers go to career fairs, hold information sessions, set up tables in the Campus Center, and post jobs on the university's job board/career website, Handshake (formerly Jobfinder). We compiled this information into one main dataset for use in this project.

Between the dataset with employer activity and the post-graduation outcomes data, we had more than 1,500 employers to analyze with machine learning techniques, including a modified random regression forest we wrote ourselves. Using feature importance on these techniques, we were able to give the Career Development Center key information about what employers do to attract the most potential hires to their companies. The CDC can then pass this information along to employers to help them hire the most WPI students possible in the future.

1.1 Measuring Outcomes

This section is about our sponsor, the Career Development Center at WPI, and what they do for the university. It also provides an explanation of where the datasets used come from, and how they relate to the goals of this project. Details on data management and cleaning are further explained in later sections.

The CDC is the main hub for both employers and students at WPI, providing several opportunities for the two groups to interact, including career fairs, information sessions, on-campus interviews, and Handshake. They track how often employers and students interact with each other through these resources, as well as obtaining self-reported data from students and alumni about full-time, internship, co-op, and research opportunities [22].

Each year the CDC publishes the Post-Graduation Report for the most recent graduating class, which contains detailed information on what students are doing after graduation. In this project, we focus on the report for the graduating class of 2016 [16]. WPI successfully granted degrees to 1646 bachelor's, master's, and PhD students in 2016, with an overall success rate of 92.7%, meaning 92.7% of graduates were either employed, going to graduate school, enlisting in the military, or joining a volunteer service after graduation [16]. The knowledge rate for the report was 91.8% overall, as around 135 of the 1646 total graduates did not report their post-graduation plans [16]. If you omit the graduates continuing their education, there were 1,147 students reporting an employer or organization.

The CDC also recorded data on employers' activity at WPI for the 2015-2016 school year, with 1840 employers seeking out students through career fairs, information sessions, resume books, on campus interviews, sponsoring events, job posts, resume searches, educational programs, and sponsoring the CDC. The most common method used to reach out to students was job posts, with almost 5,000 posts recorded, followed by attending an on-campus career fair. These datasets include information for employers seeking students of all class levels, while the Post-Graduation Report only includes information from recent graduates.

These reports and datasets give us a basis for setting our overall project goals. While 92.7% of students had an employer or a graduate school upon graduation, 7.3% did not fulfill the CDC's definition of success [16]. In addition, there were at least 1,840 employers visible to WPI students in one shape or form, but only 1,147 students reported having an employer upon graduation, meaning hundreds of employers were unsuccessful at hiring WPI students. When accounting for the employers who hire multiple students, the number of successful employers decreases even more.

This information leads to the core questions we strive to answer with this project: What do the employers who hire the most students from WPI do to be so successful, and how can employers who do not hire students become successful? Answering these questions will help the CDC widen the line of communication between students and employers, while also potentially lowering the number of students still seeking employment six months after graduation.

2 Literature Review

In the following section we provide a brief review of the existing literature on machine learning techniques to deal with imbalanced data sets using cost sensitive ensembles. A review of the existing literature did not provide any pre-existing methods for using cost sensitive ensembles with regression, all of examples given were of classification. Yong Zhang and Dapeng Wang from the School of Computer and Information Technology at Liaoning Normal University describe a cost-sensitive ensemble method for class-imbalanced data sets.[25] Bartosz Krawczyk and Michał Woźniak discuss a cost-sensitive algorithm for malware detection.[17] Although Krawczyk and Woźniak present an ensemble based on cost-sensitive decision trees, neither study addresses regression trees.

2.1 Cost Sensitive Ensembles

Cost-sensitive ensembles have been used to solve imbalanced classification problems. Class-imbalanced data problems exist in a variety of fields, including medical diagnosis, fraud detection, and a variety of science and engineering problems. When working with imbalanced data, standard classification techniques tend to fail. [13] Most standard techniques tend focus on the larger majority classes in the data and ignore the smaller, minority classes.

When dealing with the class-imbalanced classification problem, it is important to select the appropriate training data. One method for doing so is resampling: a technique for adjusting the size of the training sets. By oversampling minority classes, undersampling majority classes, or using a combination of the two methods,

resampling methods can reduce the extent of the data imbalance. Resampling techniques can be used with a number of methods, including support vector machines (SVM) and Bayes Classifiers.[1] Modified learning algorithms are another solution to deal with imbalanced data effectively. Modified learning algorithms are created by changing existing machine learning algorithms in a way that better suits the data.[25]

Cost-sensitive learning is one such solution, along with feature selection and single-class learning. Cost-sensitive learning is considered an important type of method to handle class imbalance. It addresses the problem of class imbalance by incurring different costs for various classes. However, one difficulty with cost-sensitive learning is that the costs of miscalculation are often unknown. Zhang and Wang create a cost-sensitive ensemble method. This method trains subclassifiers according to the ratio of imbalanced samples. The sub-classifiers are then integrated into a classifier, and cost-sensitive SMV is used to train selected data [25].

2.2 Cost Sensitive Decision Trees

Krawczyk and Woźniak present an ensemble for malware detection based on cost-sensitive decision trees.[17] In this presented solution, individual classifiers are constructed according to an established cost matrix. These classifiers are then trained on random feature subspaces to ensure that they are mutually complimentary. The parameters for the cost matrix are derived using ROC analysis.

One major contribution this paper presents is a new ensemble pruning method that is based on the combination of decision trees trained on different sets of features. [17] Cost-sensitive decision trees were chosen as a base classifier. Decision tree induction is based on misclassification cost rate. At each node of the tree, a local sequential search is performed. This local search assigns a greater cost to a situation when an object of a minority class is misclassified. Thus, the recognition rate of the majority class is boosted.

A random subspace approach is used to create a representative pool of classifiers. In a random subspace approach, the feature space is randomly divided into several subspaces, and individual classifiers are trained in each subspace. [14] This method ensures that the representative pool of classifiers is diverse, and that it contains heterogeneous as opposed to homogeneous classifiers. Krawczyk and Woźniak employ an evolutionary algorithm to select individual classifiers for the ensemble.[17] In this algorithm, each individual data point in the population represents a classifier ensemble $Ch = [W]$. Each component W represents the weights assigned to each of the base classifiers $W = [W_1, W_2, ..., W_L]$ and is a vector with values in $[0; 1]$.

Using experimental analysis on a large malware dataset, Krawczyk and Woźniak prove that cost-sensitive decision trees are capable of outperforming other traditional methods. Thus, cost-sensitive decision trees are an effective solution to dealing with imbalanced malware detection [17].

3 Project Outline

As the previous year's Major Qualifying Project with the Career Development Center focused on analyzing indicators of success among students, the goal of our project

was to use the data provided to us to assess the factors that most influence the success of employers looking to hire talent from WPI. This was achieved through the following steps. First, we used a sample data set to practice relevant data analysis techniques in Python. For this purpose, we used a publicly available student performance data set from the University of California, Irvine [8]. After obtaining the necessary data from the CDC, our next task was to prepare the datasets for analysis. This required cleaning the relevant datasets in Excel, combining them together using Excel and Google Sheets, and finally cleaning the new merged dataset. Once our master dataset was prepared, we applied the methods learned during our first step to the datasets from the CDC to gain the desired insights.

3.1 Test Dataset

Before we received the data collected by the CDC, we began using an openly available dataset on student performance at a Portuguese secondary school to learn concepts that would become important to our project [8]. We used this time to familiarize ourselves with software (e.g. Anaconda, Jupyter notebooks) [10] and Python programming concepts and libraries (pandas, scikit-learn, etc.) [19] [20] that we would use throughout the rest of our project. We began by performing exploratory data analysis, comparing different features of the data. We then encoded the data so that we could run desired algorithms on it. This involved converting categorical data into numerical data so that the relevant functions could make sense of the values. Finally, we used matplotlib to graph the percentage of variance of the data that was explained using different numbers of principle components found through principle component analysis. Because the CDC provided us with their data so early on in our project, further techniques were tested on copies of their data.

3.2 Data Pre-Processing

After the data was provided to us, a considerable amount of time had to be spent cleaning, manipulating, and encoding the different datasets. This involved using Microsoft Excel to locate and delete columns and rows without any meaningful data (including some which were entirely empty), convert binary data values into ones and zeros, and expand certain columns with multiple pieces of information per entry into several distinct binary columns. After this, we combined the manipulated data sets into one large set with all of the data, using the column with names of employers as our common key across the different sets. Due to the fact that some company names were written slightly differently from one dataset to another, there were many instances of one company containing two or three different rows in the merged set. Consequently, we used Google Sheets to allow each of us to work combining duplicate rows. Finally, after loading the data into Python, we replaced blank, or NaN, values with zeros wherever we knew that an absence of information meant that a company had not performed the given action.

3.3 The Curse of Dimensionality

The curse of dimensionality is a well-known concept in data science. As one considers data in higher and higher dimensions, the space this data lives in increases to the

point that the data points become sparse in their dimensional space. In fact, it is found that, as the number of dimensions, n, increases, the required data to make accurate predictions grows by an exponential factor (N^n). Without additional data, this sparsity causes methods that require any real statistical significance to behave very poorly, as shown in Figure 1 below [21].

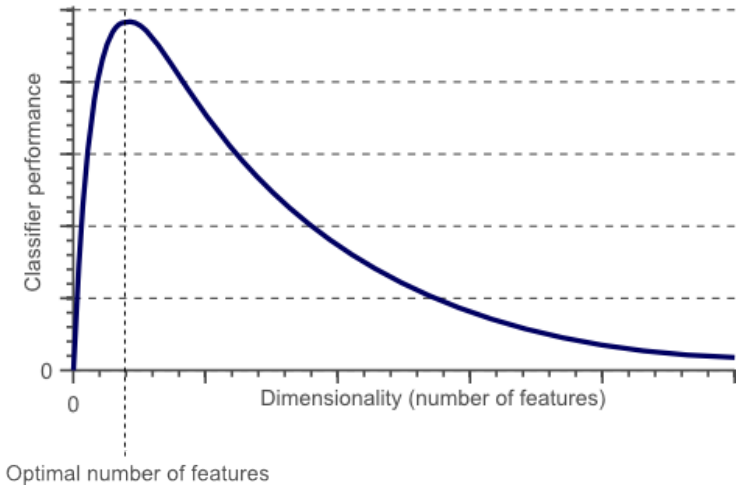

Figure 1: Dimensionality Vs Performance of Common Models (taken from [21])- This generally describes the trade-off of common model performance for added features of data.

As we continue to delve into our data from the CDC we must consider this concept to ensure we develop meaningful results from our numerical techniques. Below we describe dimension reduction techniques to reduce the effect of the curse of dimensionality.

3.3.1 Singular Value Decomposition (SVD)

If A $\in \mathbb{R}^{m\times n}$, the singular value decomposition of A is a factorization such that:

$$A = U\Sigma V^T$$

where U $\in \mathbb{R}^{m\times m}$ is the matrix consisting of the normalized eigenvectors of AA^T, V $\in \mathbb{R}^{n\times n}$ is the matrix consisting of the normalized eigenvectors of A^TA, and Σ $\in \mathbb{R}^{m\times n}$ is the matrix with the singular values of A along its main diagonal in descending order, and zeros everywhere else. A singular value of matrix A refers to the square root of the corresponding eigenvalue of A^TA. Also, in SVD, U and V^T are always both unitary. This simply means that UU^T = I, the identity matrix in the same dimension as U, and similarly for V. [2] For our purposes, we use SVD to run future principle component analysis on our data.

3.3.2 Principle Component Analysis (PCA)

Principal component analysis (PCA) is a multivariate technique that analyzes a data table in which observations are described by several inter-correlated quantitative dependent variables. Its goal is to extract the important information from the table, to represent it as a set of new orthogonal variables called principal components, and

to display the pattern of similarity of the observations and of the variables as points in maps. [2]

For our project, we use principle component analysis as our initial method of dimension reduction. By first running singular value decomposition on a data set, we can extract the principle components of our data by taking the dot product of U and the diagonal of Σ, found using SVD. Then, by examining the explained variance of the principle components through Σ, as seen in Figure 2 from the test data set, we hope to see that the majority of the variance of our data is explained by a small amount of principle components. If this is the case, our data should be very predictable by appropriate modeling techniques.

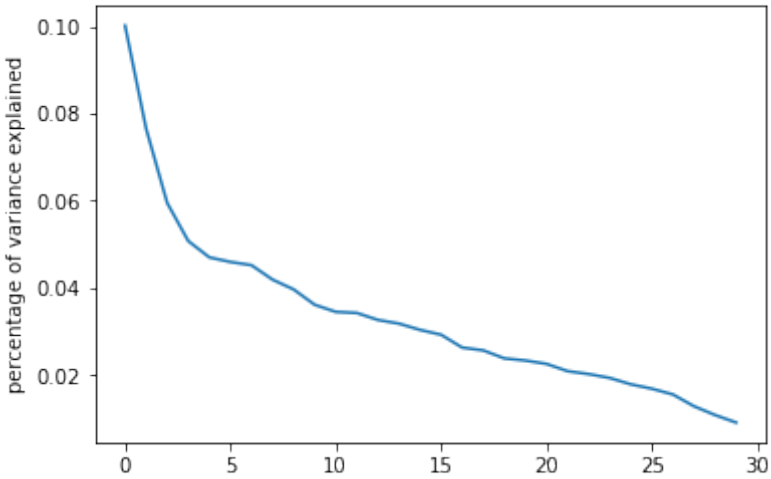

Figure 2: Percent of Variance in Data Explained Vs Number of Principle Components- In one of our first steps in exploring a similar dataset to our own, we analyzed the principle components of a sample student dataset and the variance explained by each component.

Furthermore, we can then use these "most important" components as our new data, thus avoiding the curse of dimensionality.

4 Machine Learning Techniques

In this section, we provide brief descriptions of several supervised machine learning techniques which were considered during the course of our project. As our response variable (number of hires) is continuous, we focus here on regression techniques. Finally, we provide reasoning for whether we decided to use each method.

4.1 K-Nearest Neighbors Regression

K-Nearest Neighbors regression is a non-parametric method that is used to predict a numerical target [23]. Non-parametric methods do not make any assumptions about the data distribution. Thus, it is a helpful method when there is little knowledge about how the data is distributed, as is often the case when working with real data. There are two main types of K-Nearest Neighbors, classification and regression. We

will mostly explain KNN Regression, as this is what we used in our data. However, for clarity we will provide an example using KNN classification.

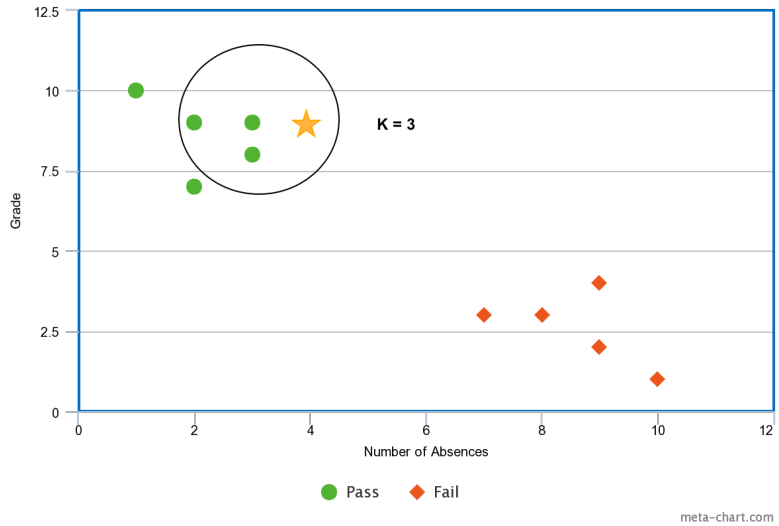

Figure 3: An example of K-Nearest Neighbors Classification. The green and red points represent different classes and the yellow star represents the unknown point to be classified. Adapted from [23].

Figure 3 above shows an example of K-Nearest Neighbors classification with a K value of 3. The red and green data points signify two separate groups, or classes in the data, and the star represents a point whose class is unknown. The 3 nearest points with known values are used to determine which class the unknown data point belongs to. Since all 3 of these points are green, we can assume that the star most likely belongs to the pass group, and not the fail group.

Occasionally, a tie can occur. In our example, this would be an issue if the four nearest points were used to determine which class the data point belongs to, and two were red while the other were green. When a tie occurs, there are a number of tie breaking methods including tie breaking by indices and Stone's tie breaking. In the tie breaking by indices method, if x is equidistant to X_i and X_j, then x is determined closed to X_i if $i < j$. [5] In Stone's Tie Breaking, the number of nearest neighbors in increased until the tie is broken. [12]

Like KNN Classification, KNN regression identifies K number of training observations that are closest to a prediction point, x_0. The predicted value of a point is calculated by finding the average of these K closest training observations using the following equation, where N_0 denotes the K training observations that are closest to x_0, and Y_i is the corresponding value (Pass, Fail, etc.):

$$Pr(Y = j|X = x_0) = \frac{1}{k} \sum_{i \in N_0} I(y_i = j)$$

Choosing the correct value of K is important. In general, a small value of K provides the most flexible fit with a low bias, however the variance is high. In contrast, a large K value will have less variance and a smoother fit, but is more likely to be biased. This is called the bias-variance trade-off. For most datasets, the optimal K value is ten or more.

When the dataset is large, K-Nearest Neighbors is a highly accurate method that is simple to implement and is not sensitive to outliers. It also has a quick training phase and is versatile. However, K-Nearest Neighbors is not ideal for large datasets. All of the training data is stored, and for this reason it is computationally expensive and has a high memory requirement. Based on the size of our dataset, we decided that K-Nearest Neighbors would not be the best option due to runtime concerns.

4.2 Linear Regression

There are two types of linear regression, simple linear regression and multiple linear regression. In simple linear regression, a single predictor variable X is used to predict a quantitative response Y. There is an assumption of an approximately linear relationship between X and Y. This relationship is written as:

$$Y \approx \beta_0 + \beta_1 X$$

where β_0 and β_1 are the model parameters [23]. They are both unknown constants that represent intercept and slope respectively, and can be estimated using the data. Figure 4 is an example of simple linear regression, where there is a clear linear relationship between the predictor and response variables.

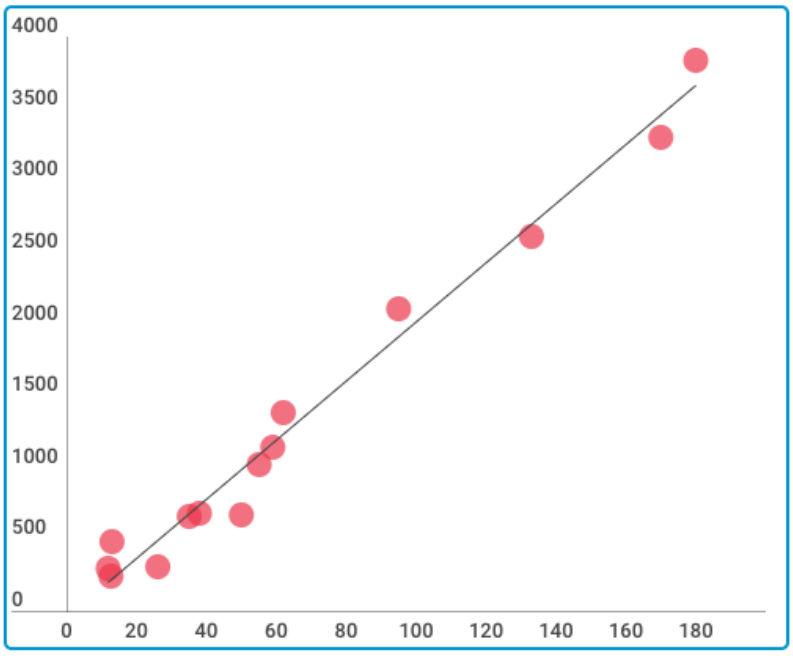

Figure 4: An example of Simple Linear Regression. The x-axis represents the predictor variables, and the y-axis represents the response variables. As is shown, there is a clear linear relationship between the two.

The accuracy of a simple linear regression model can be assessed using residual standard error. Residual standard error is the average amount that the response differs from the actual regression line. The value of the residual standard error will always be between 0 and 1. If the residual standard error is close to 0, the model predictions are very close to the actual values. Thus, the model fits the data well. If the residual standard error is close to 1, the model predictions differ from the actual

values and the model is not a good fit for the data. [3] Residual standard error is calculated using the following formula:

$$RSE = \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

The second type of linear regression, multiple linear regression, is used when there is more than one predictor variable. This was the case with our dataset. Multiple linear regression is similar to simple linear regression, but each predictor has a separate slope coefficient. The multiple linear regression model is represented using the following equation:

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + \epsilon$$

Where X_j is the jth predictor. β_j represents the average effect on Y of an increase by one unit of X_j.

Linear regression models have the ability to determine the influence of one or more predictor variables to the response variable. This type of modeling is also able to identity outliers in the data. However, linear regression models assume that there is a linear relationship between the predictor and response variables. Based on our principle component analysis of the CDC data, it was clear early on that the data does not have a linear relationship. For this reason, we decided not to use a linear regression model, as it would not be able to provide us with accurate predictions.

4.3 Support Vector Machines

Support vector machines are a type of supervised learning model with associated learning algorithms. They can be used for both classification and regression. The SVM model is an extension of the support vector classifier. The support vector classifier is a classification approach used in two-class settings, when the two classes have a linear boundary. [23] The support vector classifier is computed as the inner product of two observations, and can be represented by the following formula:

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle$$

There are n parameters, $\alpha_i, i = 1, ..., n$, or one parameter per training observation, represented by x_i [23].

The support vector machine is initially provided with a set of vectors and their respective labels. In its basic form, SVM is a hyperplane. Each vector has p features represented as $X_1, X_2, \ldots X_p$. Each p-dimensional vector can be separated by a $(p-1)$-dimensional hyperplane. The hyperplane can be represented by the following equation:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p = 0,$$

where $\beta_1 ... \beta_p$ are parameters and $X = (X_1, X_2, ..., X_P)^T$ represents a point in p-dimensional space. A numerical optimization procedure can be used to search for the values of $\beta_1 ... \beta_p$.

The goal is to find the hyperplane in which the data is separated by a maximal margin. The margin is defined as the perpendicular distance between the plane and the closest data points. The larger the margin, the lower the generalization error, and the more accurate the prediction tends to be. Therefore, the best separation is achieved by the hyperplane that is the greatest distance from the surrounding points.

Figure 5 shows what the SVM algorithm is designed to do [15]. In this image, there are two classes: red and blue. In this case, a hyperplane exists that can separate the data into two classes, as is shown by the solid black line in Figure 5. The space between the solid black line and dotted lines are the margins, or the distances of the closest examples from the hyperplane. An infinite number of hyperplanes can exist, however a maximal marginal hyperplane is chosen so that the hyperplane is farthest away from the training data [7].

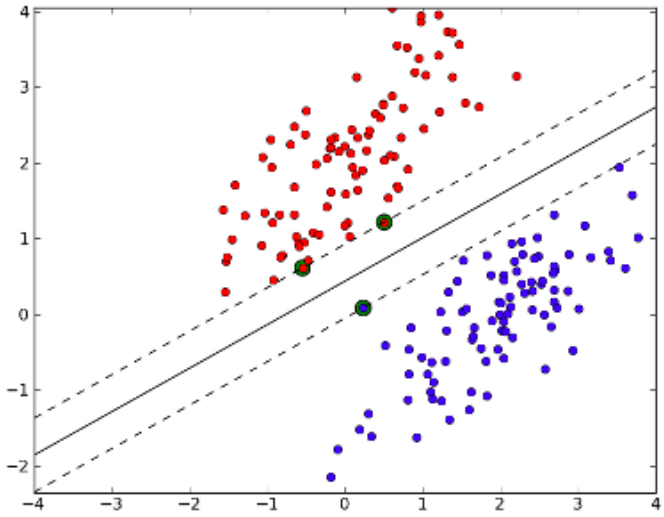

Figure 5: This is an example of a Support Vector Machine. The red and blue data points represent two different classes of data. The two classes of data are separated by a linear boundary, which in this figure is the solid black line. The two dotted black lines are support vectors. The space between the solid black lines and dotted lines are the margins. [15].

However, the support vector classifier performs poorly if the boundary between classes is non-linear. One method of accounting for non-linear class boundaries is to use kernels. A kernel is a function that quantifies the similarity between two different observations. There are various kernel forms including linear and polynomial. The following is an example of one of the simplest forms of a kernel equation:

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

This is an example of a linear kernel equation. The linear kernel equation uses the Pearson correlation to quantify the similarity between a pair of observations. As you can see, $K(x_i, x_{i\prime})$ decreases as $x_{i\prime}$ becomes further away from x_i. The following equation represents the polynomial form of a kernel:

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d \text{ [23]}$$

Although SVM can be used for both classification and regression problems, it generally preforms best in situations of binary classification. In particular, the practice of separating hyperplanes does not work in situations with more than two classes. Also, while SVM works well on small datasets, the training time for large datasets tends to be high. Since our data is large and we used a regression technique, we decided it would not be ideal to use SVM in our situation. SVM is difficult to interpret as well, especially in comparison to other techniques such as decision trees.

4.4 Trees

A decision tree is a supervised learning algorithm that makes predictions on a target variable based on splits in the predictor space. There are two types of decision trees that are defined based on the type of target variable. If the target variable is categorical, classification trees are used. However, if the target variable is continuous, regression trees are used. In classification trees, the terminal value or node is the mode of the values that fall into a certain predictor space, and the tree eventually guesses a class. In regression trees, this value is the mean response and the tree eventually guesses a number [23]. For our datasets, we used regression trees since we are looking to find the number of hires from a given company's activities with WPI's CDC.

Decision trees begin with the root node, or entire data set. The root node is then split into two or more decision nodes based on which variable creates the best homogeneous sets. Decision nodes are subsets that will split into further subsets recursively. Splitting continues until leaves/terminal nodes, or nodes that do not split, are reached. Splits are made based on the split that minimizes the chosen error metric. The lines that connect the leaves and decision nodes are called branches.

Figure 6 is an example of how a decision tree makes splits. This example shows three splits, however in reality a tree can make many different numbers of splits.

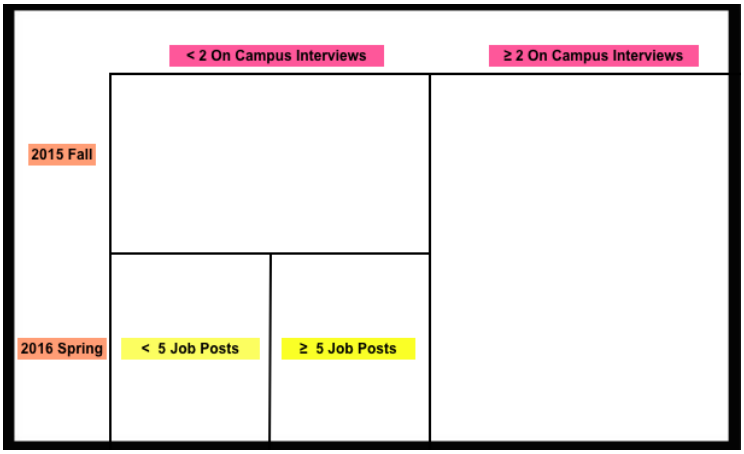

Figure 6: This is an example of a decision tree that makes three cuts using three predictors.

Decision trees are helpful because they are easy to understand. In particular, their graphical representation is intuitive even to people who do not have a mathe-

matical background. It also requires less data cleaning than some other methods, as it tends not to be heavily influenced by missing values and outliers. As mentioned above, decision trees can be used to interpret different types of data as well. However, decision trees can sometimes lose information when categorizing information for continuous variables. There is also the problem of over-fitting, which can be fixed by pruning, the process of removing leaves from the decision tree, or by setting constraints on model parameters.

4.5 Random Forest Regression

Once one has a working decision tree algorithm, random forest regression can be implemented using the tree. This technique reduces variance by making use of a tweak that decorrelates different trees generated by bootstrapping. [23] Bootstrapping is a method of generating many datasets from one by repeatedly sampling random observations from the original dataset with replacement. In other words, a single data point may be selected to appear in a particular bootstrapped sample twice. Each tree only considers splits along a randomly chosen subset of predictors. If p represents the total number of predictors and m the number of predictors considered by the individual trees, the general rule is that $m \approx \sqrt{p}$. A regression forest's prediction is then the average of the predictions of all of its trees.

5 Methodology

Our project followed a structured methodology to maintain organization and understanding as we continued to come to more and more conclusions from our findings. As with any project using real data, we began with a substantial amount of preprocessing of our data. The data provided to us from the CDC was substantial but not all was pertinent and much of the data was in the form of text or other categorical data which we could not use for most analysis techniques. We removed frivolous columns like the contact email addresses for companies and broke up large lists of majors that were searched for by companies into different sets of boolean values (e.g.: a value of one if a company is looking for majors in the math field, zero if not, etc.). Our data also came from many different sources from the CDC, so another step in our pre-processing became consolidating the data by company name. After pre-processing the data we were able to sift through the information to get an idea of how different companies correlate.

With a much better understanding of our data, we moved into the implementation of different modeling techniques. However, at this point, our data contained around 2098 different companies and 67 different predictors, or variables (shown in Appendix C). Having 67 different predictors may sound useful as we would have more data for a given company, but this left us subject to a concept called the curse of dimensionality, discussed in an earlier section. We addressed this by running PCA on the data using SVD. This practice allowed us to use the most important principle components we find to make predictions and build models.

Using our data broken into its principle components, we attempted several different modeling techniques, including linear regression and random forest regression. After accessing these models we can find our next steps in the analysis of our data for the CDC to provide our best possible insights.

5.1 Practice Data Testing

Before receiving data from the CDC, we started learning and practicing the implementation of important concepts and methods on an openly-available dataset. The data, provided by UCI, pertained to student performance at a Portuguese secondary school [8]. We used this time to familiarize ourselves with software (e.g. Anaconda, Jupyter notebooks) [10] and Python programming concepts and libraries (pandas, scikit-learn, etc.)[19][20] that we would use throughout the rest of our project. We began by using the pandas library to do exploratory data analysis, comparing different features of the data. We then encoded the data so that we could run Principle Component Analysis (PCA) on it. This involved converting categorical data into numerical data so that the relevant functions could make sense of the values. Finally, we used matplotlib to graph the percentage of variance of the data that was explained using different numbers of principle components as shown in section 3.3.2.

Fortunately, the data to be analyzed for our project was provided to us early on. This meant that some further techniques could be practiced on copies of that data, rather than the aforementioned test dataset, before being fully implemented. These techniques included performing PCA by using Singular Value Decomposition (SVD) rather than using scikit-learn's built-in PCA function, familiarizing ourselves with the Bokeh library for Python to visualize the results of our analysis, and using cross-validation in conjunction with different estimators (e.g. linear regression, forest regression, etc. also explained in later sections) to determine which model would give the best estimate of the data.

5.2 Career Development Center Datasets

After we received the relevant datasets from the CDC, we proceeded through the steps outlined in this section to begin analyzing the data. These steps include cleaning up the data, merging datasets based on their intersections, and encoding the data to prepare it for statistical techniques.

5.2.1 Data Clean Up

Data cleaning is a necessary step in order to make sense of the data. In order for data to be "clean," it needs to be checked for duplicates, inconsistencies, and errors. Irrelevant data needs to be deleted and inaccuracies must be addressed. One of the things we noticed early on was the number of duplicate company names. Because our datasets are drawn from manual entry, they initially included a large amount of human error. Many of the individual company names were misspelled or typed a slightly different way in different datasets, resulting in the same company listed multiple times with slightly different names in our merged dataset. We manually checked each company name in order to avoid errors and redundancy. In addition to combining spelling and formatting differences, we combined companies with various locations. For example, we consolidated Blue Cross Blue Shield of Massachusetts and Blue Cross Blue Shield of New Hampshire as simply Blue Cross Blue Shield. There were also some job postings for which the company name field contained a randomly generated identification string. Because we were unable to obtain a key to tell us what company was associated with each of these postings, we decided not to include them in our analysis.

Next, we combined and renamed some of our categories. We differentiated between majors that are offered at both the graduate and undergraduate level. To decrease our number of columns, we consolidated some of the majors into broader categories. For example, we included computer science and electrical and computer engineering majors in the same category. We also deleted columns that did not have enough information and irrelevant columns. For example, no companies recruited students pursuing a Humanities and Arts master's degree, so we decided to eliminate that column.

We then pulled out only the companies that hired WPI students, or that were actively involved in recruiting students at WPI from 2015-2016. We drew these companies from the Employer Activity, Info Sessions, Job Posts, On Campus Interviews, and Final 2016 Post Grad Outcomes datasets. When we initially merged these datasets, we were left with a large number of unknown values. However, after consulting the Career Development Center we determined that any companies which they did not have data for did not use the corresponding WPI resources. Consequently, we were able to replace many of our unknown values with zeros. After this step, we were left with all of the companies that actively recruited students at WPI, and all associated data from the various datasets. We then needed to decide which categories from the data were useful and relevant, and which were not.

As our project progressed, we decided to add another variable to our dataset, company size. We had to collect the data ourselves as it was not provided to us, but is a very big influence on a company's hiring capacity. The data was mainly sourced from the websites Handshake, LinkedIn, and Glassdoor. Six categories were used to describe size when collecting the data: 1-50 employees, 51-200 employees, 201-1000 employees, 1001-5000 employees, 5001-10000 employees, and 10000+ employees. These categories are a common way to classify employers by size, and are used by LinkedIn and Glassdoor.

5.2.2 Merging Datasets

In order to perform any type of data analysis, the data needed to be in one Excel spreadsheet. After identifying the most relevant data to the project from the available datasets, the cleaned spreadsheets containing employer activity on campus, job posts on Jobfinder, and which employers hired students were merged together. Unfortunately, employer IDs were not provided in each of the spreadsheets from the CDC, so the sheets were merged based on employer name. If the employer names did not perfectly match across spreadsheets, multiple entries were created for the same employer. Manually sifting through the employer list and combining duplicate rows solved the problem but took additional time.

The process of merging the data was as followed. Each worksheet containing relevant data from their respective workbooks was copied over to the master workbook. Each worksheet had the same layout with headers for each column and no blank rows or columns. A fourth worksheet, referred to as the master worksheet, was created to hold the new merged data and was completely blank.

5.2.3 Further Categorization

When looking at columns such as "Job: Position Type", "Job: Majors/Concentrations", "Job: Class Level", "Job: Degree Level", "Job: Resume Submission Method", the

data provided is completely text based. We can see though, that these fields are in the form of different lists. This made it easy enough to separate the elements of the given list into their own boolean columns. For instance in the "Job: Class Level" column, we split this into columns for each class year listed by the employers (Alumni <1 yr graduated, Alumni >1 yr graduated, First Year, Sophomore, Junior, Senior and Graduate Student) and assign a 1 (one) if the employer is looking for that particular class level and a 0 (zero) if they are not looking for someone in that class level. We then repeat this process for "Job: Position Type", Job: Degree Level", and "Job: Resume Submission Method".

However when looking at the "Job: Majors/ Concentrations" column, we find that the list of majors included is considerably long. This would create many new features for our data. In order to avoid this, and to therefore reduce the dimension of our data, we devise a way to group these majors by more general fields, as shown in Appendix A.

Each major/concentration group is split into both bachelors and masters (B and M) with the exception of humanities due to the fact that no company was looking for any humanities graduate students at WPI anyway. The different majors searched for by companies are then filtered into these groups, shown in Appendix B.

5.2.4 Final Predictors and Target Variable

After merging our findings from five datasets into one single dataset, we were left with 67 predictor variables and one target variable, being number of hires. The vast majority of our predictor variables were numerical, looking at factors such as the number of job postings and the number of on campus interviews a company conducted. One variable, compensation type was categorical and thus had to be encoded before a model could understand it. Three variables were one-hot-encoded, meaning we used either 1 or 0 for yes or no answers, respectively. [24] A list explaining each variable, as well as our methods for encoding when relevant are shown in Appendix C.

5.3 Initial Analysis

Upon the completion of the pre-processing of the CDC data, we were able to truly dive into the analysis of the information at hand. We began by looking at the correlation of our data through use of correlation matrices, then moved to singular value decomposition and principle component analysis and finally we began to analyze our data using some common regression methods.

5.3.1 Company Correlation

To begin our analysis we decided to look at our data as a whole. We began by looking at the predictors and how they may correlate. Using the built in corr function in Python we created a correlation matrix from our data and used matplotlib to create a fairly simple and easy to read visual representation of the matrix. After our pre-processing of the data we should see that there is little completely direct correlation between any two predictors as each of our predictors should be unique in nature and each hold use on their own. From the figure below we can see that we were successful in this respect but can still see some blips of similarities from our predictors.

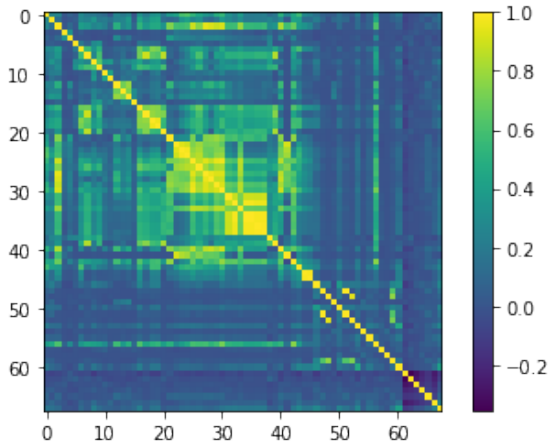

Figure 7: Correlation Matrix of Predictors- To begin our analysis of companies we generated two correlation matrices, this one to identify the correlation of the predictors to one another.

From here we considered a more statistically interesting comparison, the correlation of the companies themselves. Having already compared the predictors for the companies, we now used the transpose of our data to develop a new correlation matrix using the same functions. However, upon initial review we see that there are many blank spaces in our matrix where the companies cannot be correlated at all as seen below.

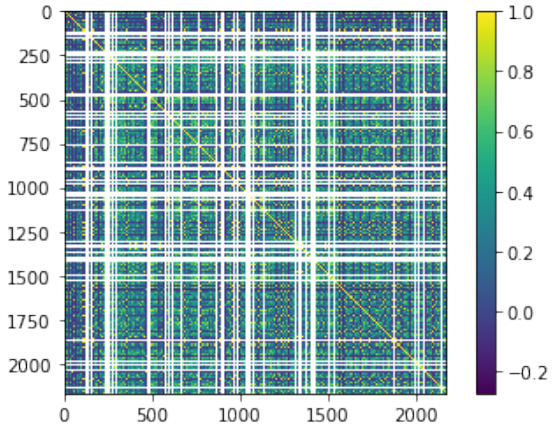

Figure 8: Initial Correlation Matrix of Companies- After analyzing the predictors we try to identify how the different companies correlate to one another using another correlation matrix. However, we see that there are several companies which stand alone, sharing no correlation to their neighbors.

Upon further inspection, we see that these missing sections of the correlation matrix are due to the lack of information in the "Hires" column for many companies. To avoid this confusion in our initial analysis and create a meaningful correlation matrix we used panda's "dropna" function to remove possible correlations which only contain NaN (Not a Number) or blank values. NaN's can occur as a result of reading in data from an Excel sheet with blank cells or attempting an impossible mathematical operation, such as dividing by zero. These correlations are meaning-

less to us in these steps of our work but their data is preserved for use in further steps. This brings us to a very exciting correlation matrix as seen below.

After adding company size as a feature in our dataset, there were no longer any NaN values to remove, meaning company size allowed each company to have some form of correlation regardless of missing "Hires" information. The resulting correlation matrix, shown in Figure 9, has no white space, indicating a lack of NaN values.

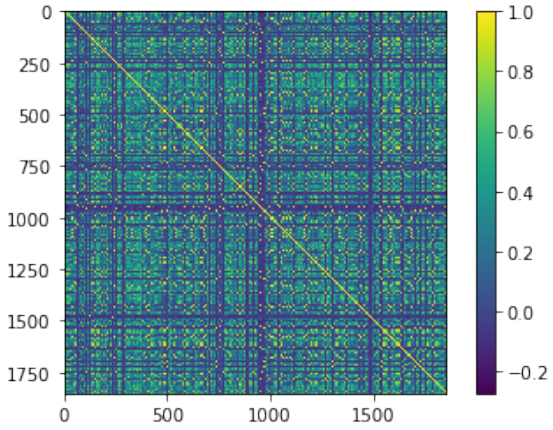

Figure 9: Final Correlation Matrix of Companies- After weeding out ill-measured companies we were able to generate an interesting correlation matrix depicting a fair amount of uniqueness amongst the companies

Given this matrix we can see some interesting and encouraging patterns in our data. While it seems that a large percentage of our data lies around the middle in correlation, we can see that some companies are stark opposites and others still share many similarities with their competitors. This provides insight and fuel to continue further in our analysis of this data.

5.3.2 SVD and PCA of CDC Data

After discovering some interesting insights from our correlation matrices we then looked to further prove our data's potential by using singular value decomposition to run principle component analysis on our data.

Just as we did with the correlation of our data, we began with our predictors. SVD is quite simple to run on our correlation matrix using the numpy.linalg function, "svd". Much like on our test dataset, we were able to plot a graph of the explained variance in our data vs the number of principle components, shown in Figure 10.

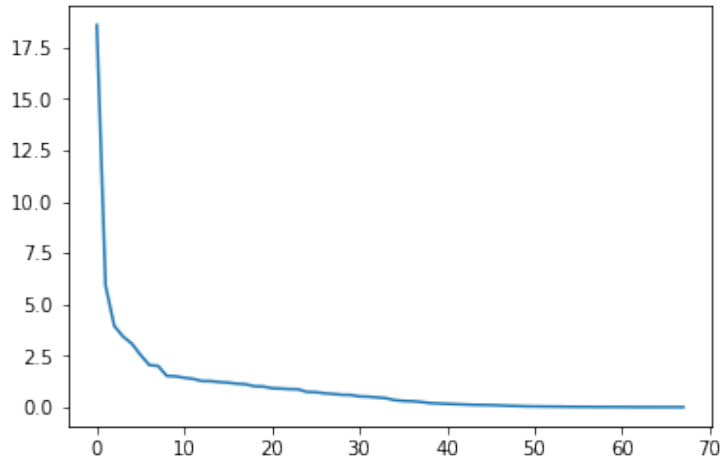

Figure 10: PCA Graph of Predictors- Here we ran principle component analysis on our predictors, demonstrating the fact that the majority of the variance in the predictors can be explained using the first few principle components. This is probably due to the distribution of the hires for our companies.

As we expected, we can see that the majority of the variance from the predictors can be explained using far less than 67 principle components.

We then moved on to our company data. Using the same methods as above, we obtained a similar graph, displayed in Figure 11 this time seeing that even less of the acquired PCA components are necessary the explain the majority of the variance of our data.

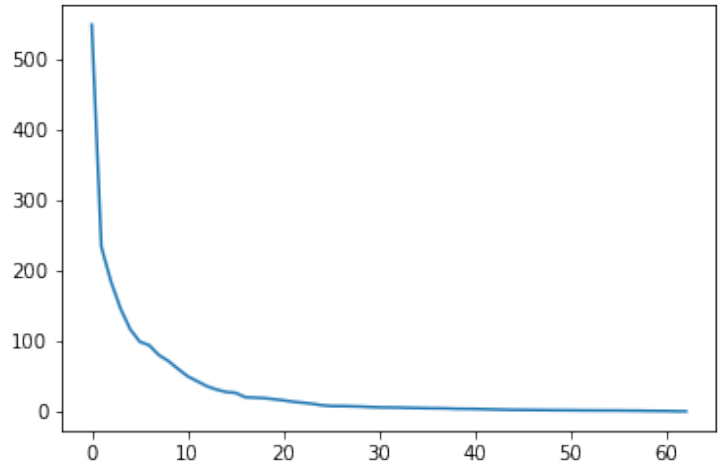

Figure 11: PCA Graph of Companies (63 Components)- Our principle component analysis on our companies, demonstrated a similar phenomenon as the majority of the variance in the companies could be explained using the first few principle components.

Reducing the scale of the graph to just the first ten components, as shown in Figure 12 we can see that even as few as eight PCA components can explain the majority of the variance in our data.

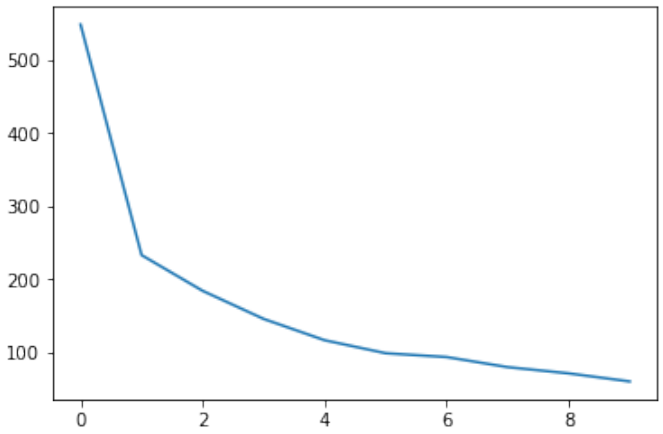

Figure 12: PCA Graph of Companies (First 8 Components)- To dive deeper, we zoom into the first eight points on the previous graph, proving to us that in fact, most of the variance in our data can be explained using only eight principle components.

This is encouraging as this analysis shows that our data should be very predictable and, using these first eight PCA components, we can avoid the curse of dimensionality when modeling our data for insight.

To explore our data further, we decide to graph the first two principle components against each other (those which explain the most variance) using the bokeh plotting tools. This graph is shown below in Figure 13

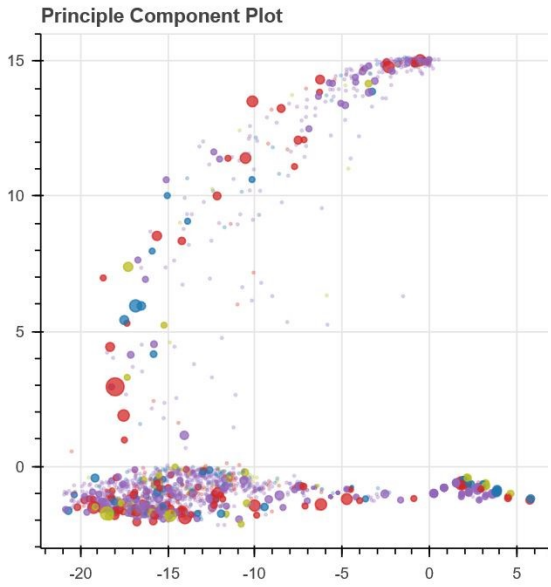

Figure 13: First two principle components graphed against each other, color and size by number of total hires- This plot, using Bokeh gave us a dynamic look at our data as we can see how our companies cluster in the principle component space, specifically the first two components as they accounted for the most variance.

This gives us the ability to make a dynamic graph to show how our data looks in the principle component space while also being able to identify individual companies by mousing over individual points (shown in Figure 14). Each point on the graph

represents a company. The size of the point reflects the number of hires a company has, with a larger circle indicating more hires. The color of the point reflects the size of the company. The scale follows the order of the rainbow, i.e. red, orange, yellow, green, blue, and violet, with red being the largest (10,000+ employees) and violet being the smallest (1-50 employees).

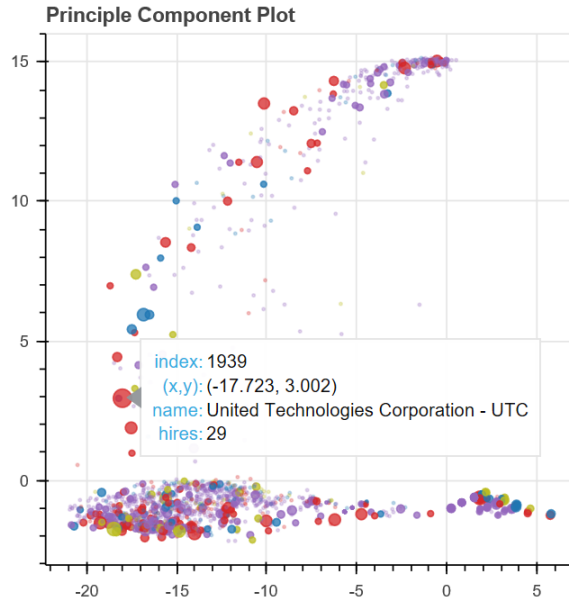

Figure 14: Features the highlighting ability of the bokeh plotting tool- Using the highlighting capabilities of bokeh, we were able to identify individual companies to possibly develop ideas for analysis based on the intrinsic information of the companies themselves.

Through the use of this bokeh plot we can identify individual companies such as UTC, pictured in Figure 14, which has the largest number of WPI hires and 10,000+ employees, and how these companies may relate both in the real world and in the principle component space.

From this graph we can obtain insights on how we can best model our data based on its shape.

6 Building Our Own Decision Tree

When using decision trees to analyze our data, we wanted to use our own error metric that could take the number of hires into account. For instance, a prediction error of one employee impacts a company with two hires much more than a company with 100 hires. The error metric options presented in the scikit-learn regression tree function did not meet our needs, as the only two options provided were mean squared error and mean absolute error. Consequently, we decided to write our own decision tree from scratch in order to use our own error metric that accounts for company size. This article by Jason Brownlee was very helpful in guiding us through the process of creating our own tree from scratch. [6]

We first separate the dataset into two lists of rows based on two components. The first is the index of an attribute. The second is a split value for that attribute. We can then use our error metric (relative squared error) to evaluate the cost of the

split. The tree will make the split that has the lowest cost. Our code does this by going through each row of the data and comparing the attribute value to the split value. If the attribute is below the split value, it is assigned to the left group. If it is above the split value, we assign it to the right group.

We then find the best split we can make, or the split with the lowest cost. We do this by going through the data and checking every value on each attribute. The final best split is returned after all of the checks are made and used as a node in our decision tree. Our name for this function is "cut()".

We build a tree by calling the "cut()" function for the entire data set. We can divide the tree building process into three parts: terminal nodes, recursive splitting, and building a tree. The first, terminal nodes, allows us to decide when to stop growing a tree by setting a maximum tree depth. This limits the number of layers, or nodes from the root, that a tree can have. Setting a maximum tree depth can help prevent overfitting.

The second, recursive splitting, is an essential part of building a tree. After a node is created, further nodes (called child nodes) can be created by splitting the parent node recursively. Each node can have up to two child nodes. First, the node is split into two groups of data. We then check if either group is empty or if we have reached the maximum tree depth. If either of the previous statements are true, we create a terminal node. Otherwise, we process the left child. If the group of rows is too small, we create a terminal node, but if not we continue to make splits until the maximum depth is reached. We then process the right side in the same way.

Finally, building the tree involves creating the root node and then making splits recursively. We provide the tree-building function with our dataset, the maximum tree depth, and the minimum tree size. We first use the cut() function to find the root node. Then, we call the split() function to make splits recursively. After the decision tree is built, it can be used to make predictions with the data.

6.1 Relative Error

Because our data for number of hires varied from 1 to 29, we wanted to minimize our relative, or percent, error rather than the squared error or absolute error metrics, which are more common in random forest regression (particularly the former). Searching for an existing implementation of such a tree yielded no useful results, and one cannot simply pass a new criterion (error function to minimize over) to the existing scikit-learn functions in order to achieve this. Therefore, we had to write and implement our own tree from scratch in Python.

In order to decide on what value, α, to assign to a group after making a cut, we wanted a value that would minimize mean squared relative error. Similarly to the reasons behind common use of MSE, we used the square of relative error because it is a smooth differentiable function and more heavily "punishes" outliers. Thus, we had:

$$\min_{\alpha} \sum_{i=0}^{n-1} \frac{(\alpha - x_i)^2}{x_i^2}$$

Where x_i is the number of hires for data point i, and n is our total number of data points. Notice that if we were to expand each term in this sum, our α^2 term would have a coefficient of $\frac{1}{x_i^2}$. Because our data for hires are obviously nonnegative, the

equation that we wish to find the minimum for is simply the equation of an upward-facing parabola. Therefore, to find said minimum, we just need to set its derivative equal to zero and solve. So,

$$\sum_{i=0}^{n-1} \frac{2(\alpha - x_i)}{x_i^2} = 0 \implies \frac{\alpha - x_0}{x_0^2} + \frac{\alpha - x_1}{x_1^2} + \cdots + \frac{\alpha - x_{n-1}}{x_{n-1}^2} = 0$$

Multiplying by $x_0^2 x_1^2 \dots x_{n-1}^2$, we obtain:

$$(\alpha - x_0) \prod_{i=1}^{n-1} x_i^2 + (\alpha - x_1) \prod_{\substack{i=0 \\ i \neq 1}}^{n-1} x_i^2 + \cdots + (\alpha - x_{n-1}) \prod_{i=0}^{n-2} x_i^2 = 0$$

Finally, by distributing the $(\alpha - x_i)$ terms and factoring the resulting α terms, we obtain:

$$\alpha \left(\sum_{i=0}^{n-1} \prod_{\substack{j=0 \\ j \neq i}}^{n-1} x_j^2 \right) - \sum_{i=0}^{n-1} \left(x_i \prod_{\substack{j=0 \\ j \neq i}}^{n-1} x_j^2 \right) = 0 \implies \alpha = \frac{\displaystyle\sum_{i=0}^{n-1} \left(x_i \prod_{\substack{j=0 \\ j \neq i}}^{n-1} x_j^2 \right)}{\displaystyle\sum_{i=0}^{n-1} \prod_{\substack{j=0 \\ j \neq i}}^{n-1} x_j^2}$$

6.2 Implementing Random Forest Regression with Our Tree

To implement random forest regression using our own tree, we generate a bootstrapped training set for each decision tree within the forest. Each tree is also given a random sample of predictors over which splits are considered. The convention is that, given a total of p predictors, one should use random predictor samples of size $m \approx \sqrt{p}$. In our case, because we had 67 total predictors, we used a random sample size of 8. This made sense intuitively, as our principal component analysis had previously implied that our data could be well-represented by 8 features (i.e. the additional variance explained by principal components beyond 8 was very small).

We then build a forest using our own decision tree from above, given the relevant data, number of trees to be created, size of bootstrapped samples, maximum depth of the trees, and minimum size of leaves in the trees. While building the forest, we use a series of lists to keep track of what predictors each tree was built upon, the bootstrapped set of data points that each tree used to learn and the trees themselves which were generated by the forest. Namely, our manager, oracle and forest, held the set of predictors, data points and trees respectively. Next, for each tree in the forest, we compare the actual number of hires for each data point given to the tree to learn, (kept in our oracle) to the predicted values of tree, and compute the mean relative error for each individual tree. This will be used later to identify feature importance and in our cross validation methods to construct the ideal ensemble of trees. Finally, we determine the mean relative errors of all trees in the forest. Ultimately, our forest was run with bootstrapped samples of size 260, and we were able to produce 140 trees for our analysis.

Unfortunately, due to time limitations and our limited knowledge of how to optimize computational efficiency in our code, our modified tree had a notably long

runtime. This was particularly noticeable as we had thus far been running code on our own personal computers. This obviously led to an unreasonable runtime when trying to use the tree for random forest regression (approximately four hours to run a forest with only 10 trees of 260 data points each). We first tried to address the issue using the Numba's JIT Compiler [9], which attempts to "Compile the decorated function on-the-fly to produce efficient machine code." Unfortunately, when we attempted this method, we received errors that we were unable to resolve. As a result, we had to make use of WPI's Ace Cluster (a high-performance computing system) in order to run our code.

6.3 Ace Cluster Implementation

After gaining access to WPI's Ace Cluster, we looked to utilize its multiprocessing capabilities to run multiple iterations of our forest function simultaneously. We began by attempting to simply run multiple copies of our code at the same time, generating 10 trees per run. For future analysis, we collected the relevant results for each forest (predictors, predictions on testing and training data points, and mean relative errors of each tree in the forest) into several csv files and compiled them into a single forest after a given run.

However, this method did not properly use the cluster to its full potential and only barely increased our runtime. From there, we sought out help from Spencer and James of WPI's Academic Technology Center (ATC). With their help, we were able to create a script which ran our forests on 10 separate CPUs, decreasing our run time exponentially. Even still, we were limited to generating forests of only seven trees each totaling an approximate 40 hour run time. After completing this process twice, we were able to muster our 140 total trees for our ensemble learning methods.

7 Overview and Parameters

In this section we give an overview of each method we used to model the data and the parameters we set for each to achieve our results. For all the methods listed below, except for our modified random forest, we used the functions from scikit-learn.

7.1 Linear Regression

Our first attempt with linear regression used the PCA components as predictors. Cross-validation did not seem useful for error reduction due to the nature of PCA components and how high the MRE was compared to other methods. Instead, we attempted cross-validation using the recursive feature elimination cross-validation (RFECV) function in scikit-learn with all 68 predictors. The method uses recursive feature elimination, but also utilizes cross-validation selection to determine the best number of features. The only parameter chosen was the cv parameter, which was set to 5, representing 5-fold cross-validation. All other parameters were the scikit-learn defaults. We first ran the method using standardized data with the exception of the Hires data, i.e. the Hires column in the data was the raw number of hires for a company, but all other columns were z-transformed numbers. Having the data in

that format yielded the lowest MRE. For feature importance comparison, we also ran the method with unstandardized data.

7.2 KNN

The best number of neighbors for the K-nearest neighbors method, chosen through cross-validation, was 37 (Figure 15). The number of neighbors tested ranged from 1 to 50, with mean relative error (MRE) as the accuracy metric. All other parameters were the default values set by scikit-learn. The data used for the KNN model was the same as for the RFECV model, as it also yielded the lowest MRE for KNN.

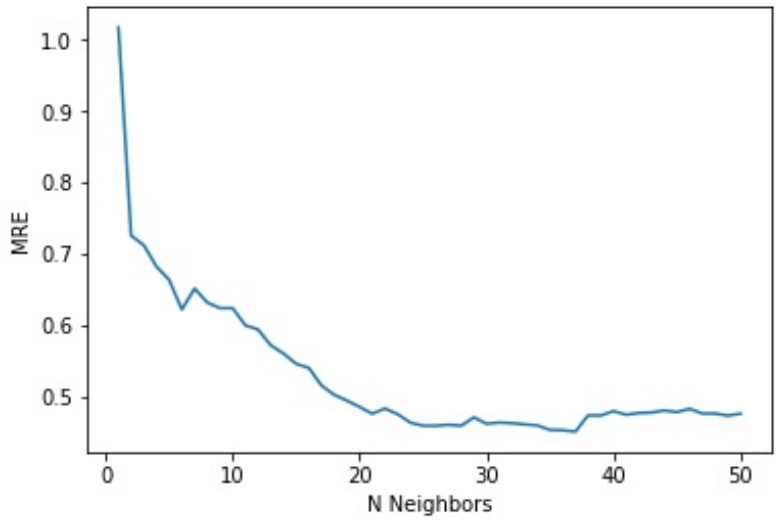

Figure 15: Cross-validation for KNN - number of neighbors from 1 to 50

7.3 SVM

The support vector machine kernel that provided the best fit for the data was the Radial Basis Function kernel, using the PCA components as predictors. Once we determined this, we used cross-validation to find the optimal values for C and gamma.

To find the best value for C, we tested values ranging from 0.01 to 10. Figure 16 shows the MRE's for the different values of C when gamma was set to the scikit-learn default of 0.1. The error dropped as the value of C decreased, but stopped decreasing by significant values after 0.01, so we chose that as the optimal C value.

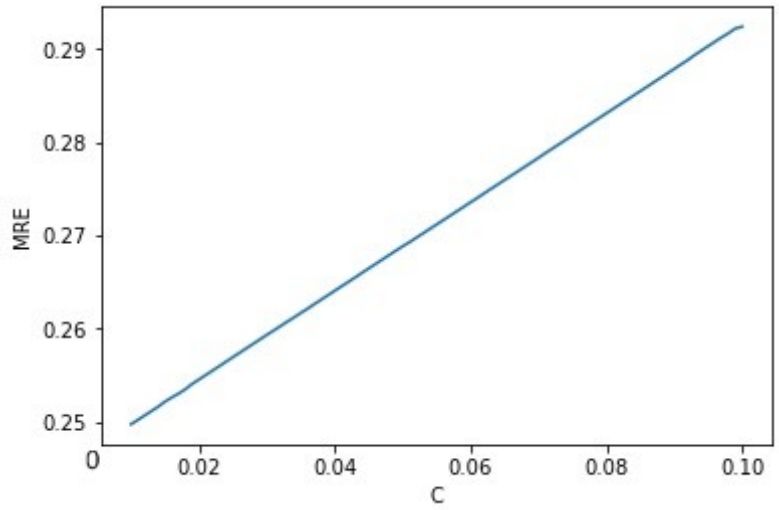

Figure 16: Cross-validation for SVM - Zoomed in to show error for C values between 0.01 and 0.10

Using the newfound value for C, we then used cross-validation to find the best value for gamma, testing values ranging from 0.01 to 10. Figure 17 shows an interesting pattern. The MRE peaked at 0.09, and decreased in both directions from that point until reaching a minimum at about 3.8. Lowering the gamma value led to an even lower MRE, but at the cost of higher variation in the model. In order to avoid over fitting, we chose 0.01 as our optimal gamma instead of 3.8, but they did both lead to similar MRE's.

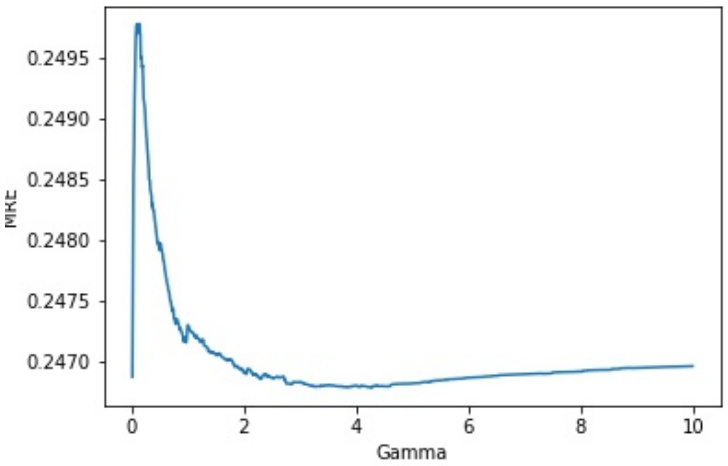

Figure 17: Cross-validation for SVM - showing error for gamma values between 0.01 and 10

7.4 Original Random Forest

For the built-in random forest regression, we used cross-validation to choose the best number of learners, maximum tree length, and minimum leaf size. The random

state of the tree was set to zero. The accuracy metric to find the best parameters was mean relative error. We used the default setting for the maximum features per tree, which was all 68 predictors. There was no guarantee all predictors would be used in a tree, nor the forest as a whole. The data used to build the forest was not standardized, as standardizing led to a higher MRE.

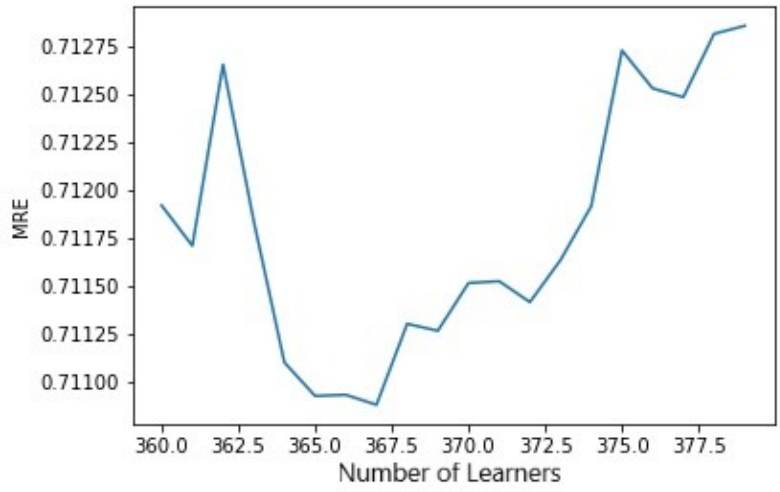

Figure 18: Cross-validation for Original Random Forest - number of learners (trees) in the forest

To find the optimal number of learners for the forest, we tested values between 1 and 500. Figure 18 shows the MRE's for 360 to 375 learners, as that was minimum of the full cross-validation run was fell within that range. Based on the graph, we chose 367 as the number of learners for the model.

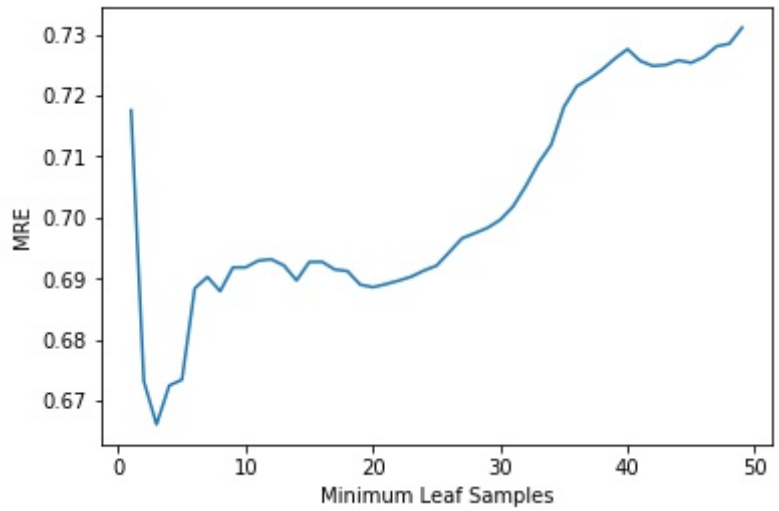

Figure 19: Cross-validation for Original Random Forest - minimum samples in a leaf

To optimize the minimum number of sample per leaf, we cross-validated using values between 1 and 50. As shown in Figure 19, having 3 samples per leaf at minimum resulted in the lowest MRE. The same was done to find the optimal

maximum depth for a tree, and based on Figure 20 the depth that minimized the MRE was 5.

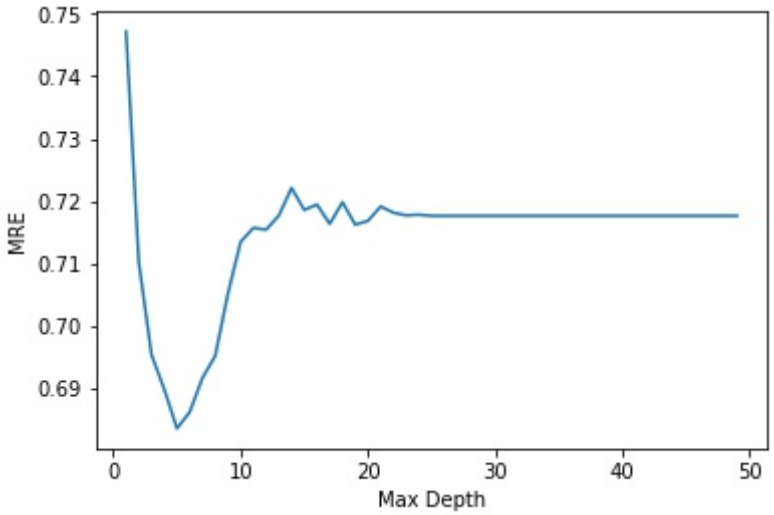

Figure 20: Cross-validation for Original Random Forest - maximum tree depth

7.5 Modified Random Forest

As with the built-in random forest, we used cross-validation to find the best number of learners. However, due to our computational limitations, we were not able to cross-validate on the maximum tree depth and minimum leaf size. Instead, we used a base maximum depth of 25 with a minimum leaf size of 5 and found that this assumption was accurate enough for our purposes. From there, we developed a forest pruning method, similar to a global threshold pruning where the optimal subforest is calculated by pruning subtrees [4]. In our case, we selected the "best" trees for our model by implementing a threshold on the training errors of individual trees. In its simplest sense, if a tree in our forest was highly inaccurate on the training data, then we would throw it away and just the same, if a tree was too accurate on the training set, it would be considered over-fitting and would be thrown away. This allowed us to cross-validate on these two threshold values (the upper and lower bound) to see how removing ill fitting trees affected the overall testing error of our forest.

Let us first take a look at the training errors for all of our trees that we were able to generate.

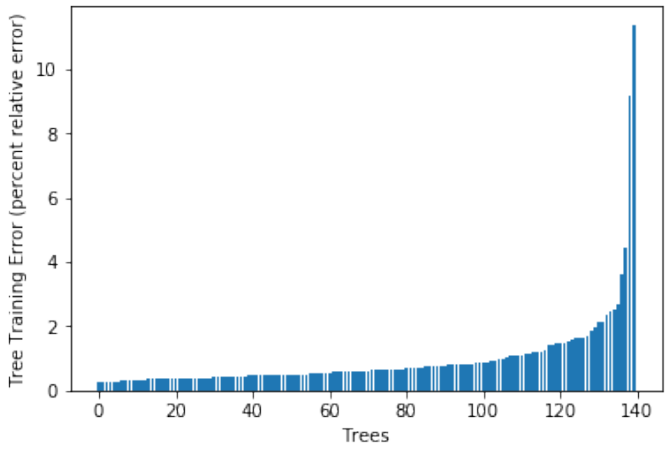

Figure 21: Training Accuracy of Each Tree Generated- We explored the training accuracies of all of the trees sorted by their error that we generated.

We can see from Figure 21 that the majority of our trees have a mean relative error of below one while only a few have exceptionally high errors, for our purposes we will completely disregard those far-right outliers.

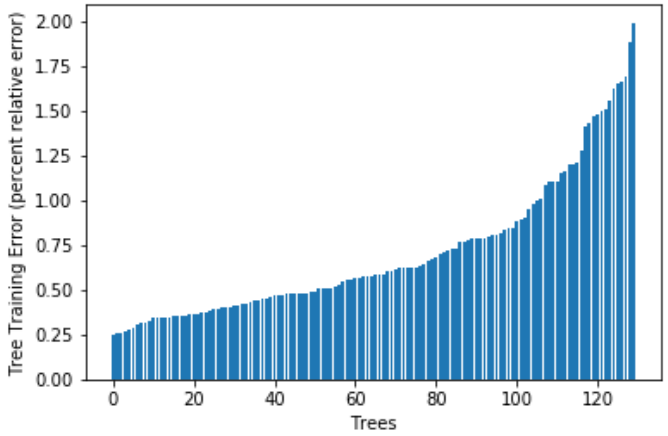

Figure 22: Training Accuracy of Each Tree Generated Without Outliers- We removed the outlier trees entirely as their high inaccuracies would lead to poor results regardless of our cross-validation methods.

In Figure 22 we can more easily discern the distribution of the relative error for our trees. Now, with a graphical representation of our trees' error metric, we may implement our threshold process.

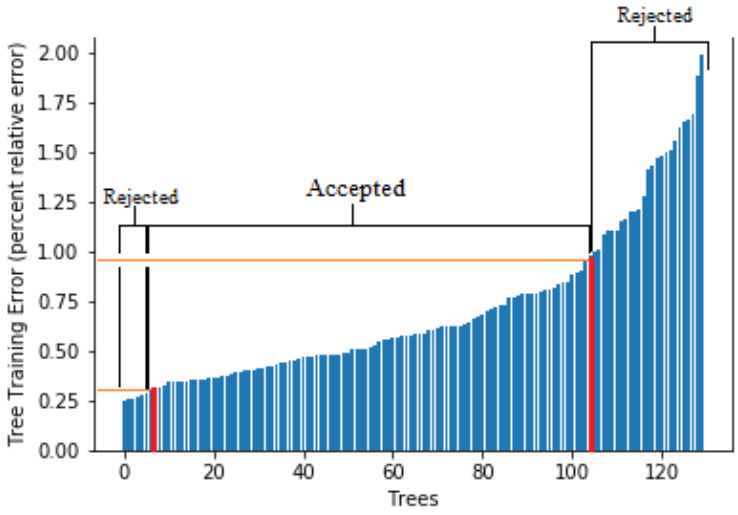

Figure 23: Example of Threshold Methodology for Cross-Validation- Using this example, we showed how our cross validation method was implemented. Any trees to the right of the upper bound (red, right) was rejected, and any trees to the left of the lower bound (red, left) were also rejected, leaving a range of accepted trees in between which our model will compute with.

Figure 23 demonstrates an example bounds which we may put on our trees. In this example, our bounds are from 30 percent relative error to 90 percent relative error (shown in orange). These bounds are represented by the thick red line on the trees' themselves. By our cross-validation method, we will disregard any trees which do not fall within these bounds (Rejected) and only evaluate our model using those trees which satisfy this bound (Accepted).

We began our process by first cross-validating on the upper bound of our threshold. These were the trees that would be deemed too inaccurate and would not be included in our final model. This would be represented in Figure 23 by moving the right red line. For this test we kept our lower bound of the threshold at 0 percent error, allowing trees that are completely over-fitting to be included in the model. From here we compared the overall testing error of the forest with an upper bound of training errors for each tree from 30 to 100 percent error as seen in Figures 24 and 25.

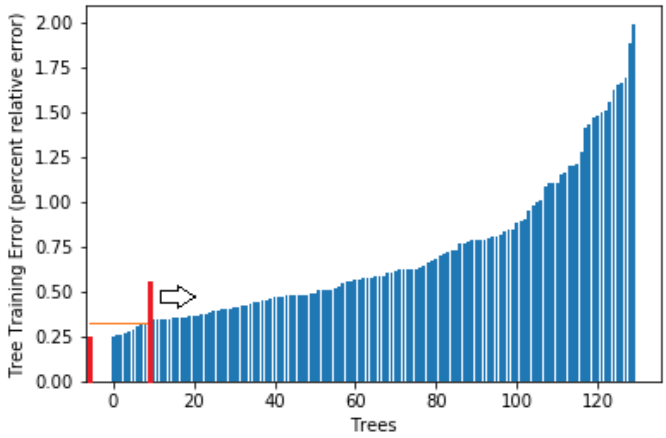

Figure 24: Upper Bound Threshold Methodology for Cross-Validation- Our cross-validation for the upper bound began at 30 percent error with a lower bound of 0 percent error, then progresses the upper bound to the right (indicated by arrow).

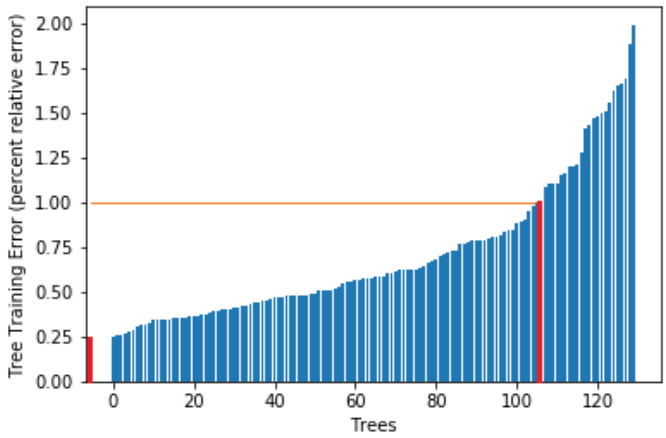

Figure 25: Upper Bound Threshold Methodology for Cross-Validation- Our cross-validation for the upper bound ended at 100 percent error with a lower bound of 0 percent error, represented by these bounds.

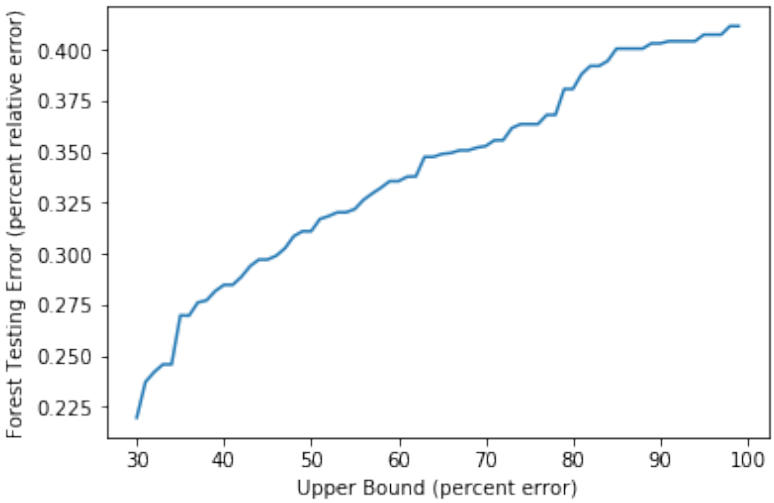

Figure 26: Cross-validation for Modified Random Forest upper bound- We cross-validated our model on the lower bound, starting with at lower bound at zero percent error and finishing at 45 percent error.

From the graph in Figure 26 we can see that by removing as many of the "worst" trees as possible, we can reduce our testing error substantially for our model.

From here we then cross-validate on the lower bound of our threshold, the "over-fitting" trees. Through this process we looked to eliminate trees who's training error is too low and thus may have been over-fitting to the training data, giving poor testing results. Here, we kept the upper bound of the threshold at 45 percent error, as we deemed from our previous graph this was a reasonable cutoff point for inaccurate trees. We then compared the testing error of the forest with a lower bound of training errors for each tree from 0 to 45 percent error. The result is illustrated in Figures 27 and 28.

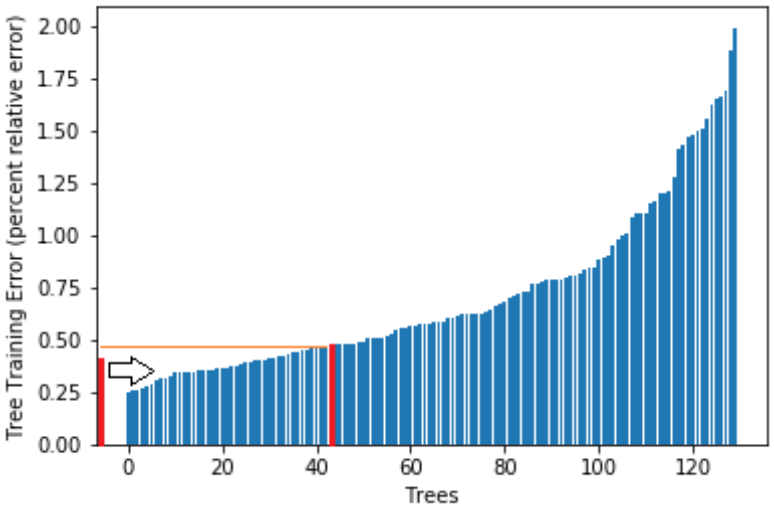

Figure 27: Lower Bound Threshold Methodology for Cross-Validation- Our cross-validation for the lower bound began at zero percent error with an upper bound of 45 percent error, then progresses the lower bound to the right (indicated by arrow).

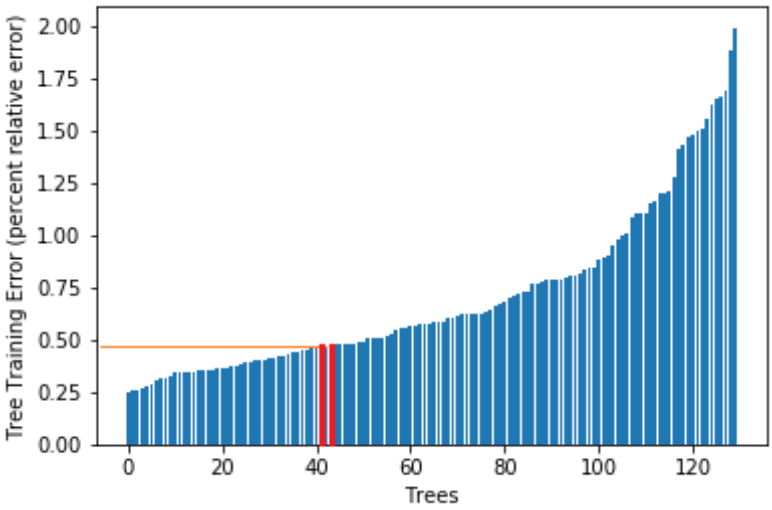

Figure 28: Lower Bound Threshold Methodology for Cross-Validation- Our cross-validation for the lower bound ended at 45 percent error with the upper bound of 45 percent error, represented by these bounds (contains only one tree at this point).

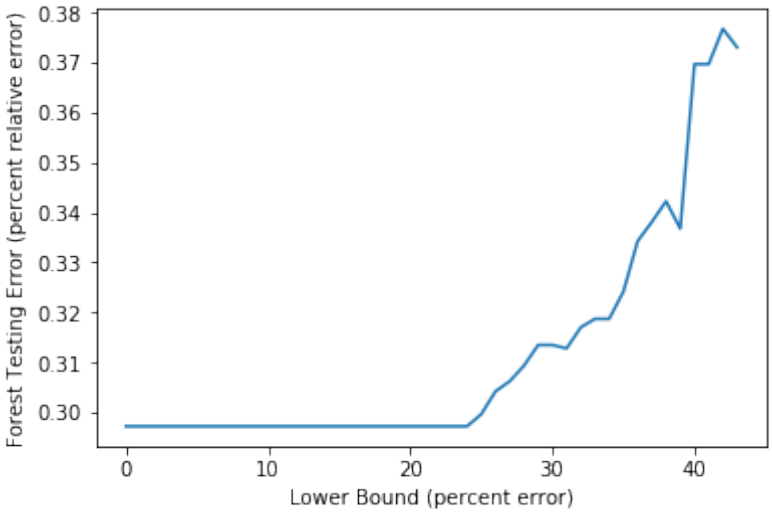

Figure 29: Cross-validation for Modified Random Forest lower bound- We cross-validated our model on the lower bound, starting with at lower bound at zero percent error and finishing at 45 percent error.

In Figure 29 we can see that our testing results are actually best when allowing as many accurate trees as possible in our model. It seems that, based off of this, none of our trees are completely over-fitting as we see the straight line starting at 0 and ending at 24 percent error. This means that the lowest error tree we had generated still had a training error of 24 percent and is far from over-fitting.

To test this further, we cross-validated both of these values simultaneously, shifting the lower bound iteratively for each time that we shifted the upper bound. For our lower bound, we tested values from 24 (our most accurate tree) to 50 percent error and for our upper bound we tested from 50 to 100 percent error. This gave us a much better idea of how these thresholds interacted and, given the graph generated in Figure 30, we are better able to select the inputs for our final model.

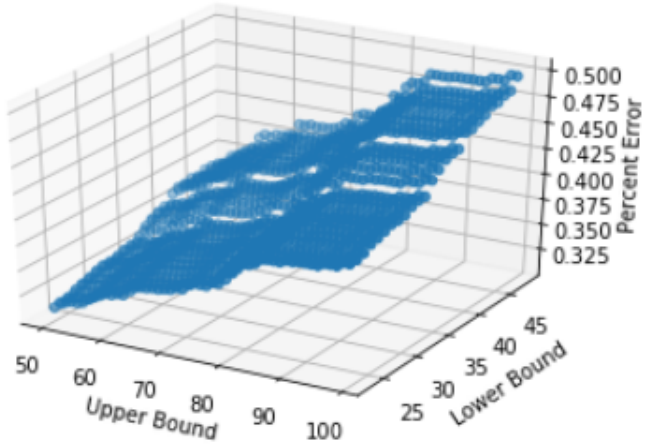

Figure 30: Cross-validation for Modified Random Forest simultaneous upper and lower bound- To thoroughly evaluate our model using this method, we cross-validated on both thresholds simultaneously.

To add to this analysis, we imported plotly to duplicate this 3D plot and color scale our resultant forest testing error. Plotly also enabled us to rotate and zoom on the graph to gain a better understanding of how our thresholds were affecting the results as seen in Figure 31.

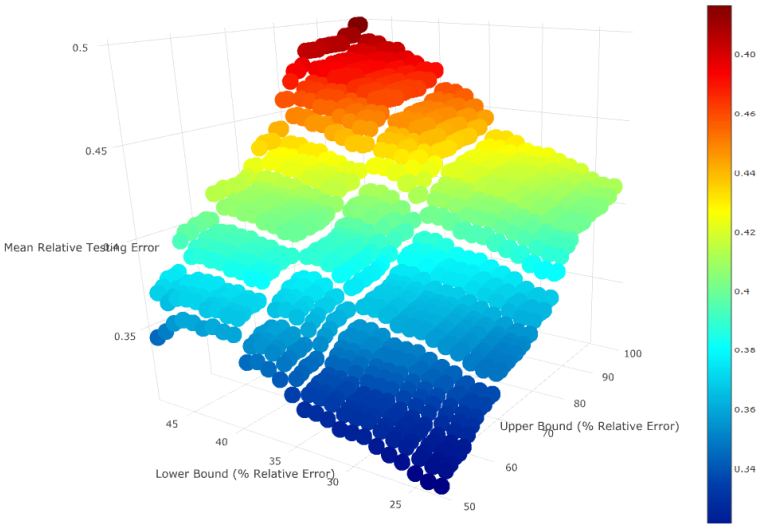

Figure 31: Cross-validation for Modified Random Forest plotly- We used the plotly tool to generate this dynamic representation of our cross-validation method.

From this graph and the color bar on the side, those bounds which result in darker blue colored points correlate to the lowest testing error for our forest while those in the darker red region correlate to higher testing error for our final model.

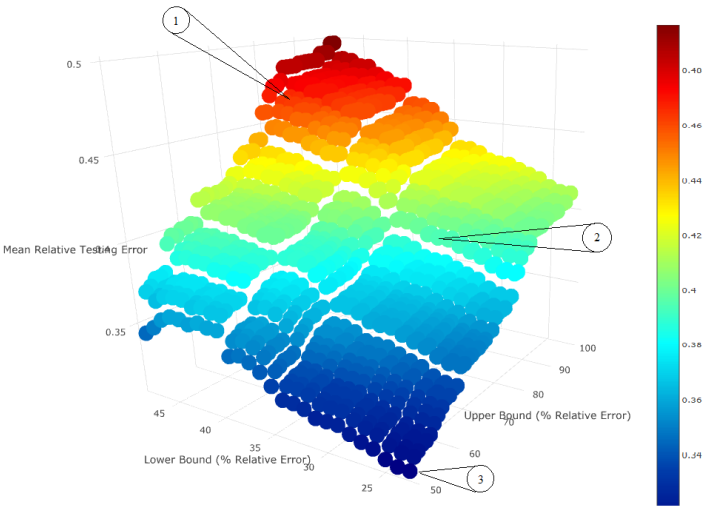

Figure 32: Callout Markers to Explore Specific Thresholds- We picked a few, distributed points to observe more closely. Callout 1 represents a threshold that results in exceptionally high mean relative error. Callout 2 represents a threshold that results in a middle of the pack relative error. Callout 3 represents the threshold that corresponds the lowest mean relative error.

To explore further still, we can look at the thresholds at some of these individual points. Here we sampled a few instances from Figure 32 to identify the thresholds each are using.

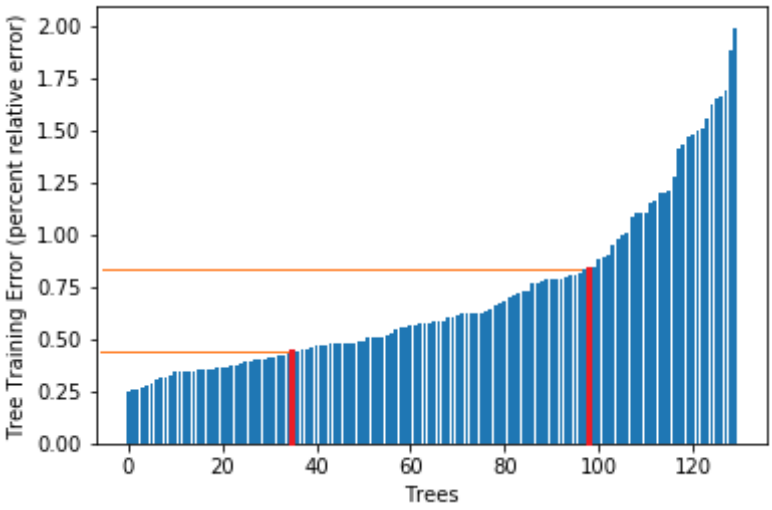

Figure 33: Callout 1 Threshold Methodology for Cross-Validation- At this specific point in our cross-validation, we have a lower threshold of 45 percent error and an upper threshold of 82 percent error. This corresponds to an overall 46 percent testing error for our forest.

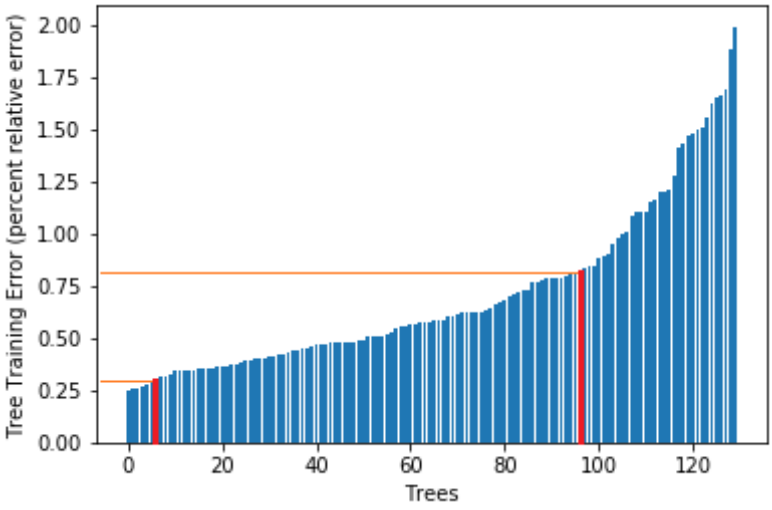

Figure 34: Callout 2 Threshold Methodology for Cross-Validation- At this specific point in our cross-validation, we have a lower threshold of 30 percent error and an upper threshold of 79 percent error. This corresponds to an overall 39 percent testing error for our forest.

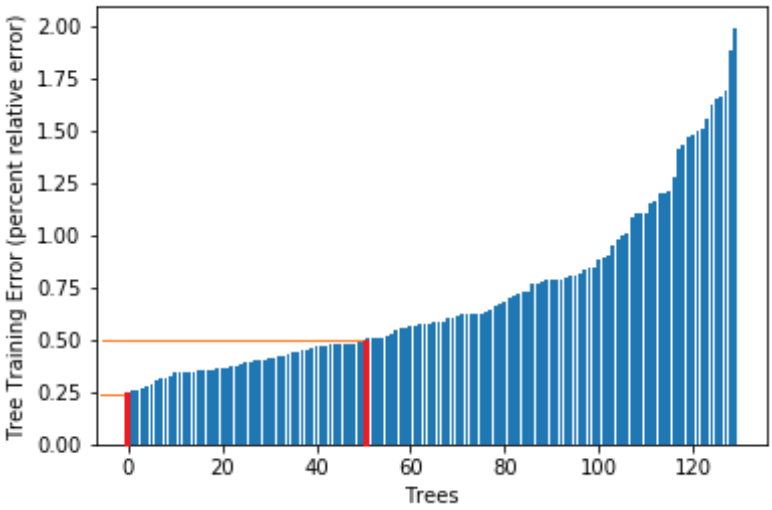

Figure 35: Callout 3 Threshold Methodology for Cross-Validation- At this specific point in our cross-validation, we have a lower threshold of 24 percent error and an upper threshold of 50 percent error. This corresponds to an overall 31 percent testing error for our forest.

After observing some of the results from our cross-validation in Figures 33, 34, and 35 We were able to determine an optimal threshold for our trees' training error to fall under. Selecting the point with the lowest error from Figure 31, which corresponds to the thresholds in the callout 3, Figure 35, we see that our threshold will accept trees who's training accuracies fall between 24 and 50 percent, yielding a 31 percent relative error.

To finalize our methods, we cross validated on the number of learners, or trees in our forest. For this, we began with a meager 10 trees and tested up to our maximum 140, each pulled at random, without replacement and maintained our thresholds from the previous method, generating the cross-validation graph in Figure 36.

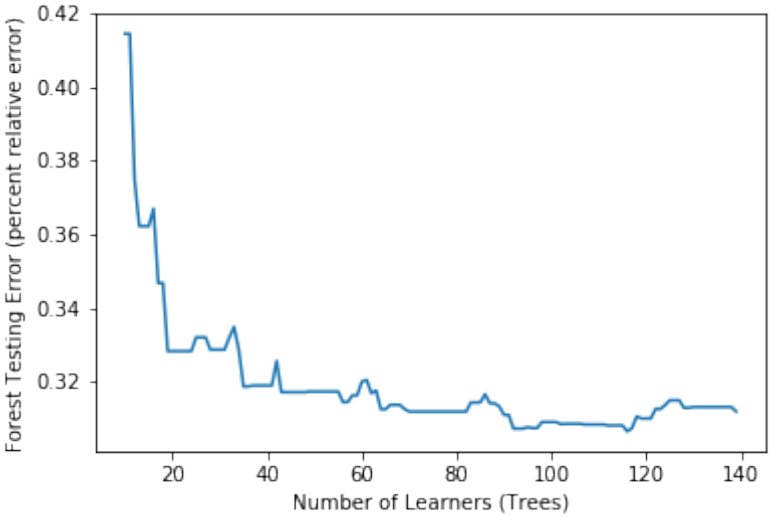

Figure 36: Cross-validation for Modified Random Forest number of learners- We cross-validated our model on the number of trees given to the forest while maintaining a constant, finalized value for our thresholds from the previous cross-validation method. This allowed us to identify how many trees should be fed into our forest and gave us our final model results.

Given these results we can see that our best accuracy comes when using 116 trees of our 140 with a resultant 0.3066 mean relative error. While we would expect this graph to be more variable in its results, we assume that, given more computational time to generate more trees, we would see a graph more close to that of figure 18, as the original random forest was able to generate up to 500 trees.

Regardless of our computational difficulties, based off of our cross validation results, our final model would accept trees of training error inside of our threshold: [0.24, 0.5] and use 116 of our generated trees resulting in our final mean relative error of 0.3066 or 30.66 percent. As this is a relative error, our final model would, in theory, be able to predict within 30 percent of company hires for both large and small companies, culminating in a model that is fair for companies of all sizes.

8 Method Results

8.1 Linear Regression

The RFECV linear regression model with the lowest MRE had only three predictors, Job: Degree Level Master of Physics for Education, Job: Degree Level Master of Mathematics for Education, and Job: Total Student Views. The MRE for the method was 0.6977 when estimating the unstandardized number of hires. The highest possible prediction produced from the model was 6.3017 hires for the company with 21 hires. While this may seem like the model was off by a lot, in reality it results in a relative error of 0.699. In comparison, the model predicted 1.7248 for the majority of companies with one hire, which is a 0.7248 relative error.

In addition to the most accurate linear regression model, we also analyzed a model with a total of 12 predictors produced using the RFECV method on the unstandardized data. The MRE for the method was 0.7748, which is higher than the

one-variable model, but offers more insight on feature importance. The model predicted 11.4257 for the company with 21 hires, and 1.1011 for many of the one hire companies. While these results seem good, both models faltered in predicting companies of all sizes leading to larger MREs when compared to the other models used. However, the linear regression models seemed to differentiate between companies with one hire and companies with more than one hire, especially the 12 predictor model, leading us to consider these models in our feature importance analysis below.

8.2 KNN

The final K-Nearest Neighbors model had an MRE of 0.5156 when estimating the unstandardized number of hires. The model predicted 4.2432 for the company with 21 hires, while the predictions for companies with one hire mainly fell between 1.0811 and 1.2973. The prediction of 4.2432 was the largest for the model, while all other predictions fell below 3. This means the model was able to cluster some of the companies with more hires together, but not all, as some of the predictions for companies with more than one hire but less than 21 hires did not show the same trend.

8.3 SVM

The final Support Vector Machine model had an MRE of 0.2617 when predicting the unstandardized number of hires. Table 3 shows some of the test predictions and the actual number of hires for the corresponding companies, e.g. (prediction, actual number of hires). The model predicted every company as having between 1.0999 and 1.1349 hires, meaning it did not predict hires for companies with more than one hire accurately. Instead, the model achieved such a low MRE by predicting the majority every time, with the majority being one hire.

Prediction	True Value	Prediction	True Value	Prediction	True Value
1.1003	1	1.1006	2	1.1318	2
1.1007	1	1.0999	2	1.1001	1
1.1229	2	1.0982	1	1.0999	1
1.1113	3	1.1003	1	1.1022	1
1.1323	1	1.0999	1	1.1001	1
1.1003	3	1.1004	2	1.1142	5
1.1009	2	1.1008	1	1.1349	21
1.1065	14	1.1321	13	1.1190	1

Table 3: Predictions versus the true value for the best SVM model

8.4 Original Random Forest

The final original Random Forest model had an MRE of 0.5575 when estimating the unstandardized number of hires. In terms of the scope of the project, this is the MRE we aimed to reduce using our modified Random Forest model. The model predicted 12.8538 for the company with 21 hires, while the majority of the

predictions for companies with one hire ranged between 1.2275 and 1.5879. The model also predicted 10.0339 for one of the companies with 14 hires but 2.0639 for the other company with 14 hires. Despite having low relative errors for predicting companies with one and 21 hires, the model is clearly not consistent in its success. The relative error for predicting companies with one hire was also higher than all but one other model.

8.5 Modified Random Forest

The final modified Random Forest model had an MRE of 0.3066 when estimating the unstandardized number of hires. The model predicted 1.0475 for almost every company with one hire, which was the most consistent and accurate of any of the models we used. The model did struggle to produce higher-valued predictions, but the highest predictions still mainly corresponded to the companies with the most hires. This indicates that the model was able to predict both when a company only hired one student and when a company hired multiple students.

8.6 Feature Importance

Feature importance provides insight on the features that predict a company's number of hires the best. Linear regression and random forest are both interpretable methods that allow for the implementation of feature importance. Here we provide the results of feature importance for Linear Regression, the original Random Forest, and our modified Random Forest.

8.6.1 Linear Regression

Even though the Linear Regression model with the lowest MRE only had three predictors, the feature selection ranking generated from the RFECV method, shown in Table 4, could still provide some insight on predicting hires. It is important to note this ranking does not mean that the best predictor of hires is if a company is looking for students with a Master of Physics Education degree. Instead we consider the top 10 features from the model to compare with the top features selected by the other models. In the tables below, a predictor with rank 1 indicates it was used in the model. The predictor with the lowest rating was the first to be removed from the model by the RFECV method, while the predictor with rank 2 was the last predictor to be removed from the model before RFECV found the best model.

Predictor	Importance Rating
Job: Degree Level Master of Physics for Educ	1
Job: Degree Level Master of Mathematics for Educ	1
Job: Total Student Views	1
Job: Unique Student Views	2
Job: Degree Level Master of Engineering	3
Job: Degree Level Graduate Student	4
Job: Compensation Type	5
Job Posts	6
OCI Interviews	7
Job: Class Level Junior	8

Table 4: Linear Regression feature importance - best features

In order to achieve a more relevant feature importance list, we analyzed the predictors of the Linear Regression model trained with the unstandardized data. The model had 12 predictors total, as shown in Table 5. None of the features with rank 1 are more important than the others of the same rank. Going to the career fairs, being a silver, event, and/or office sponsor, and having 10,000+ employees seemed to be the major themes in the 12 predictors used in the model.

Predictor	Importance Rating
Job: Position Type General/Non-Professional Job	1
Job: Majors/Concentrations HumanitiesB	1
2015 Fall	1
Silver Sponsor (F)	1
2016 Life Sciences	1
2016 Spring	1
Silver Sponsor (S)	1
Info Sessions	1
Educational programs	1
Count of Event Sponsor	1
Count of Office Sponsor	1
10000+ Emp	1
Job: Position Type Contract Professional	2
Resume Book	3
Job: Position Type Summer Internship	4
Job: Degree Level Bachelor of Science	5
Job: Position Type Full Time-Professional	6
....	.....
5000-1000 Emp	47
Job: Majors/Concentrations ChemM	48
Job: Unique Student Views	49
Job: Total Student Views	50
Job: Resume Submission Method Other (enter below)	51
Job: Student Favorite Count	52
51-200 Emp	53
Bronze Sponsor (S)	54
Bronze Sponsor (F)	55
Job: Class Level Alumni ($<$ 1 yr graduated)	56

Table 5: CV Linear Regression feature importance - best and worst features

8.6.2 Original Random Forest

To perform feature importance on the original Random Forest model we used scikit-learn's RandomForestRegressor feature importance attribute. The method uses a "mean decrease impurity" to rank the features, calculated as the average decrease in node impurity over all the trees in the forest (Further explained in [18]). The decrease in node impurity is weighted by the probability of reaching that node. The probability is approximated by totaling all the samples directed to a decision node associated with a feature and dividing that by the total number of samples in the training set. Essentially, a feature used to make an early split will involve more samples and will likely have a higher importance weight, and the lower the decrease in impurity, the less important the feature is to the model. Table 6 shows a normalized (sum of ranks equals 1) ranking of the predictors produced using scikit-learn.

Predictor	Importance Rating
OCI Interviews	0.3891720
10000+ Emp	0.0655776
Job: Majors/Concentrations CS/ECEB	0.0475571
2015 Fall	0.0374671
Job: Majors/Concentrations MathB	0.0364330
Job: Resume Submission Method Accumulate in WPI Job Finder	0.0363132
Job: Student Favorite Count	0.0319309
Job Posts	0.0252511
Job: Degree Level Doctor of Philosophy	0.0172753
Job: Class Level First Year (Undergraduate)	0.0169964
Job: Class Level Graduate Student	0.0152993
Info Sessions	0.0148919
Job: Degree Level Master of Science	0.0146426
Job: Majors/Concentrations EngineeringB	0.0129859
Job: Position Type Full Time-Professional	0.0124963
Job: Class Level Junior	0.0124112
2016 Spring	0.0109016
Job: Unique Student Views	0.0107605
Job: Class Level Senior	0.0101961
....	.....
Job: Majors/Concentrations ChemM	0.000652621
Resume Book	0.000410857
51-200 Emp	0.000392524
2016 Life Sciences	9.56791e-05
1-50 Emp	3.19474e-06
Job: Position Type General/Non-Professional Job	0
Job: Position Type Contract Professional	0
Silver Sponsor (F)	0
Silver Sponsor (S)	0
Count of Event Sponsor	0

Table 6: Original Random Forest feature importance - best and worst features

Only one feature had an importance rating above 0.1 (OCI Interviews), which is a commonly used cut-off for feature importance using this ranking system. However, due to the large number of predictors in our model, we shifted the cut-off to 0.01. Any feature with a rating of 0 was not used by any tree in the forest, meaning even when a tree was given it as one of the 8 randomly selected predictors, the tree itself did not use it to make a cut. With 367 trees in the forest, it is very unlikely that the predictors with 0 rating were never seen by a tree.

8.6.3 Modified Random Forest

Due to time limitations, we were unable to properly develop a feature importance metric analogous to the those of the built in scikit-learn functions. However, we were able to analyze the features of each of our ensembled trees and their relative "importance". Two aspects were considered when determining the top predictors of

hiring success, number of occurrences and accuracy.

We first observed the number of trees that used each predictor. For this consideration, only decision trees with mean relative error greater than 0.24 and less than or equal to 0.50 were considered. This cutoff was used in order to eliminate trees with extremely high rates of relative error, so we were not looking at predictors used in the most inaccurate trees. Table 7 lists the top 13 most frequent predictors (not including number of hires) used in decision trees with mean relative errors between 0.24 ad 0.50, as well as the number of trees in which each occurred:

Top 13 Most Frequently Used Predictors	Number of Occurrences
1-50 Emp	11
Job: Degree Level Doctor of Philosophy	11
CivilB	11
Job: Majors/Concentrations HumanitiesB	10
Job: Position Type Intern/Co-op (4-8 months)	10
Resume Searches	9
Job: Resume Submission Method Accumulate in Job Finder	9
Job: Position Type General/Non-Professional Job	9
Resume Book	8
Silver Sponsor (S)	8
Job: Class Level Graduate Student	8
Job: Majors/Concentrations All Majors	8
Job: Majors/Concentrations BUS/MNGM	8
Job: Position Type Summer Internship	8

Table 7: The 13 most frequently used predictors by Modified Random Forest

As is listed in the above table, the top three most frequently used predictors were companies of size 1-50 employees, the number of postings on Job Finder accepting students with a Doctorate degree of Philosophy, and number of jobs per company open to students pursing their Bachelors degree in Civil Engineering. Table 8 shows the 5 least frequently used predictors:

5 Least Used Predictors	Number of Occurrences
Job: Class Level Junior	1
Job: Majors/Concentrations BioB	2
Job: Degree Level Master of Business Administration	3
2016 Spring	3
51-200 Emp	3

Table 8: The 5 least frequently used predictors by Modified Random Forest

As is shown in the above table, the number of job postings for undergraduate juniors was the least used predictor, as it was only used in 1 decision tree. Other predictors used in three or less trees were the number of job postings for biology majors pursuing a bachelor's degree, the number of job postings for students pursing a master's degree in Business Administration, if a company attended the 2016 Spring Career Fair, and if a company was of size 51-200 employees.

Although it was helpful to get an idea of the most and least frequently used predictors, this method for determining feature importance has a number of drawbacks. The predictors were chosen randomly, thus it is possible that some predictors were selected more than others simply based on luck. Also, while features used in decision trees with a mean relative error greater than 0.50 were not considered, simply considering the number of occurrences of each feature does not take into account the mean relative error of each tree in which the features are used. For example, a feature that appears 8 times on trees with an average mean relative error of 0.49 is considered equally successful as a feature that appeared 8 times on trees with an average mean relative error of 0.30. For this reason, the average mean relative error rates of the trees with each predictor were calculated. As we collected the error of each tree that we generated, and the predictors used to build each, we were able to compute the mean error relating to each predictor. To do this, we identified the mean relative error (MRE) for each tree that a given predictor was used in and took the mean of these instances to find the average overall MRE for each predictor. Table 9 shows the predictors ordered from lowest to highest overall MRE.

Predictor	Average MRE
Job: Resume Submission Method Accumulate in WPI Job Finder	0.4403236
OCI Interviews	0.4619576
1-50 Emp	0.5138745
CivilB	0.5450084
Job: Majors/Concentrations ChemB	0.5457903
Resume Book	0.5523481
Job: Degree Level Doctor of Philosophy	0.5603279
Bronze Sponsor (S)	0.5793731
Job: Majors/Concentrations BioB	0.5809165
10000+ Emp	0.5810108
Job: Position Type Intern/Co-op (4-8 months)	0.5830088
Job: Class Level Senior	0.6077643
Job: Degree Level Master of Physics for Educ	0.6079236
Job: Position Type Contract Professional	0.6119266
Job: Onestop Job	0.6184994
Job: Majors/Concentrations BUS/MNGM	0.6453953
Job: Position Type Summer Internship	0.6572316
Job: Position Type General/Non-Professional Job	0.6591620
....	.....
2016 Spring	0.7938544
Count of Office Sponsor	0.7969152
Job: Majors/Concentrations CS/ECEM	0.8137167
Job: Unique Student Views	0.8184280
Job: Degree Level Master of Engineering	0.8194426
1001-5000 Emp	0.8301961
Job: Class Level First Year (Undergraduate)	0.8553419
Job: Degree Level Bachelor of Science	0.8828907
201-1000 Emp	0.9180624
Job: Class Level Junior	1.2208132

Table 9: Modified Random Forest "feature importance" - lowest to highest average errors from trees w/ given feature

8.7 Data Observations

8.7.1 Employer Location

Based on the data received from the CDC, WPI students in the class of 2016 were hired in at least 35 states in the U.S. and at least 13 other countries. Figure 37 shows a detailed view of the locations based on the number of students hired. The location data does not include students hired for internships or co-ops unless specified.

Figure 37: Global map of WPI student employer locations

There were four countries with more than one student hired, as is shown in Table 10. The United States was by far the most popular country for WPI students to seek employment. It is also important to note that not all students reported an employer location, so it is possible the true number of both national and international hires is different from what is reported below.

Country	Hires
United States	977
Saudi Arabia	5
China	4
Singapore	2

Table 10: Top 4 countries for employers

The area of the United States with highest concentration of hires was New England, as shown in Figure 38. More than half of the hiring locations were in Massachusetts alone.

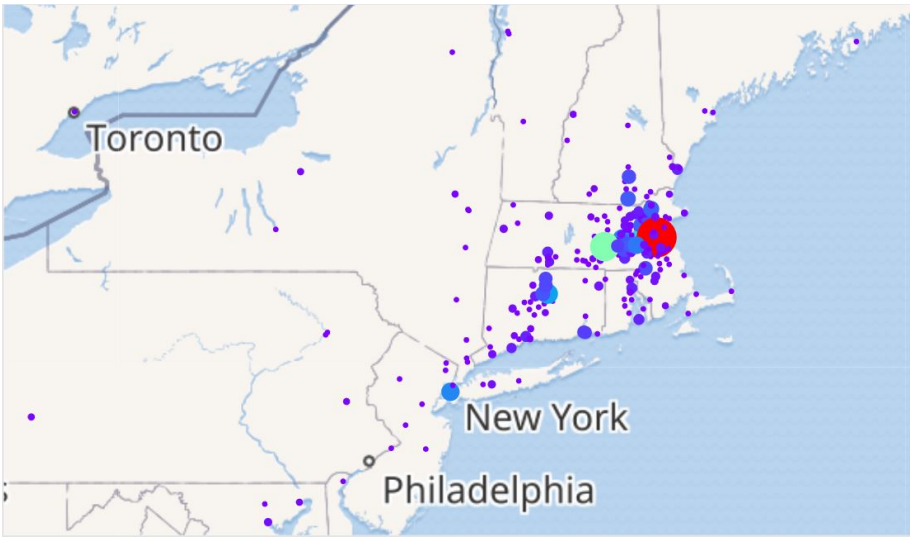

Figure 38: Map of employer locations in New England

As seen in Table 11, California was the only state in the top five for hires not on the east coast.

State	Hires
Massachusetts	563
Connecticut	126
California	55
New Hampshire	46
New York	37

Table 11: Top 5 states in the U.S. for employers

As for cities, Boston reigned supreme, and East Hartford was the only city outside of Massachusetts to make the top five. Some notable west coast cities not shown in Table 12 were Seattle, WA with 12 hires and Mountain View, CA with 14 hires.

City	Hires
Boston	101
Worcester	52
Cambridge	35
Waltham	33
East Hartford	23

Table 12: Top 5 cities in the U.S. for employers

8.7.2 Employer Size

The size of a company can play a significant role in hiring capacity. A larger company is more likely to have several available jobs as opposed to a smaller company that may only need to hire a couple people a year. Based on this logic, we collected the size of each employer in our dataset and used it to help predict the number of students an employer hires from WPI. Table 13 shows the 10 most successful

employers, and 8 out of 10 have over 10,000 employees, and all of them have at least 1,000. The numbers below also include summer employment data, such as internships, co-ops, and research programs, alongside full-time employment data. Of WPI's 24 hires listed below, only five were post-grad jobs; the rest were mainly for research programs over the summer.

Employer	Size	Hires
United Technologies Corporation (UTC)	10001+	29
Pratt and Whitney	10001+	25
Worcester Polytechnic Institute	5001-10000	24
General Electric	10001+	21
Wayfair	1001-5000	19
Raytheon	10001+	17
National Grid	10001+	15
UnitedHealth Group	10001+	15
Analog Devices	10001+	14
Cimpress (formerly Vistaprint)	10001+	14

Table 13: Top 10 employers and their sizes

Companies of a variety of sizes interacted with WPI and the CDC, as shown in Figure 39. Small companies were the most prevalent, and the number of companies decreased as size increased, except for companies with more than 10,000 employees. Overall, this trend matches the way businesses work; there are more small businesses than there are large ones as it takes time and success to grow into a large corporation. Companies with more than 10,000 employees often have multiple office locations, allowing them to reach a wider range of students. They also tend to be more well known, increasing their chances of having a presence in categories such as job postings and career fairs.

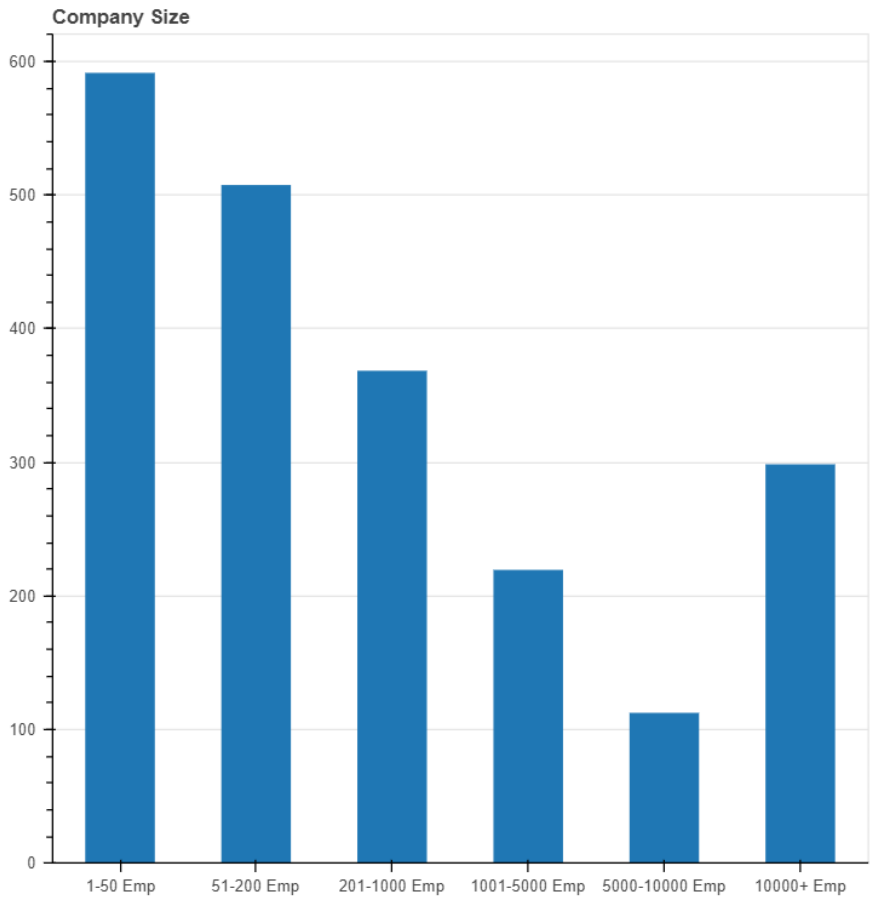

Figure 39: Number of employers based on size

Looking further into the effect of company size on the number of hires reveals two notable trends, which can be seen in Figure 40 below. The red bar in the graph represents the number of companies that hired at least one WPI student, the blue bar represents the companies where we do not have a number of hires, and the green bar represents the total number of students hired by companies of the specified size. As the company size increased, the percentage that hired students rose in proportion to the total number of companies of the same size. Also, companies with more than 10,000 employees hired significantly more students overall.

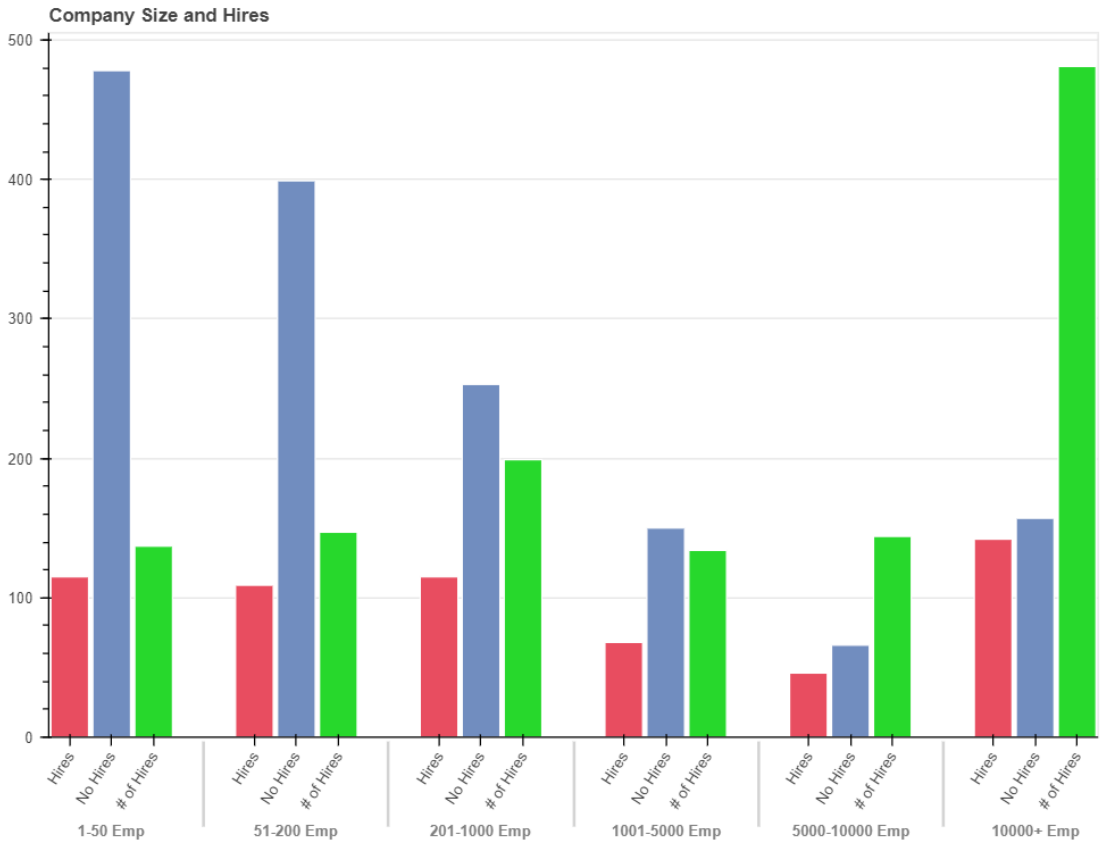

Figure 40: Number of hires based on employer size. Red indicates the number of companies with 1+ hires, blue the number without a hire, and green the total number of hires for that size.

Table 14 shows the percentage of companies of a certain size that hired at least one student. Approximately 19.39% of companies with 1-50 employees hired at least one WPI student, while 47.79% of companies with more than 10,000 employees hired at least one WPI student. This indicates that almost half of the largest companies recruiting WPI students ended up hiring one, and the turnout for larger companies was more than for smaller companies.

Company Size	Percentage with Hires
1-50 Emp	19.39%
51-200 Emp	21.46%
201-1000 Emp	31.25%
1001-5000 Emp	31.19%
5001-10000 Emp	41.07%
10000+ Emp	47.79%

Table 14: Percentage of companies with hires by size

As shown in Table 15, companies with more than 10,000 employees hired 38.73% of all WPI students that reported getting hired. Meanwhile, the percent of total hires for companies with 1-50, 51-200, 1,001-5,000, and 5,001-10,000 employees fell between 10% and 12%. This suggests that not only did a higher proportion of companies with 10,000 or more employees hire students, but they also hired more

students than smaller companies. Also, as company size increased, the average number of students hired by a company increased, with companies with 1-50 employees hiring 1.19 students on average versus companies with more than 10,000 employees hiring 3.39 students on average.

Company Size	Percentage of Total Hires	Ave. Number of Hires per Company
1-50 Emp	11.03%	1.19
51-200 Emp	11.84%	1.35
201-1000 Emp	16.02%	1.73
1001-5000 Emp	10.79%	1.97
5001-10000 Emp	11.59%	3.13
10000+ Emp	38.73%	3.39

Table 15: Percentage of total hires by company size

8.7.3 Internship Hires

Many companies of differing sizes recruit seasonal interns as well as full-time hires. As shown in Figure 41, as company size increased the number of interns hired tended to increase as well.

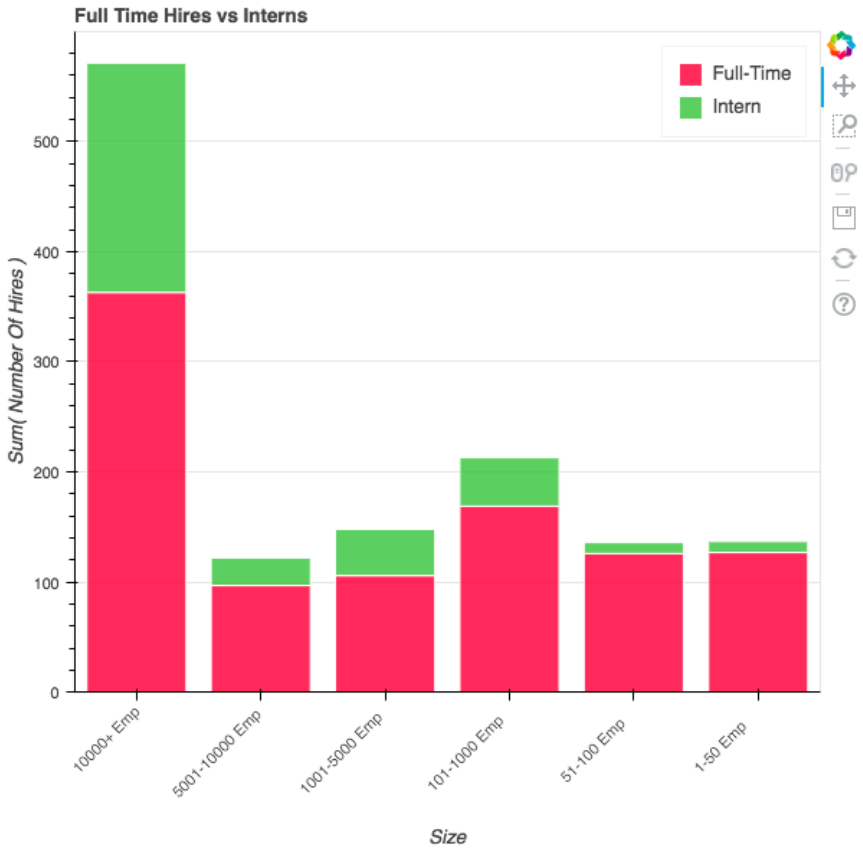

Figure 41: Interns vs. full-time hires by company size

According to the reported data, companies with 10,000 employees or more hired just over 45% of interns from WPI, which was by far the largest amount (see Table

16). Companies with 5001-10000 employees hired the second largest amount of interns from WPI, at 18.58%. Meanwhile, the percentage of total internship hires for companies with 1001-5000, 201-1000, 51-200, and 1-50 employees ranged from 11.46% to 3.95%. It is important to note that while there were more companies with 1-50 employees that interacted with WPI than companies of any other size category, companies with 1-50 employees hired the least amount of interns from WPI.

Company Size	Percentage of Internship Hires
1-50 Emp	3.95%
51-200 Emp	8.30%
201-1000 Emp	11.46%
1001-5000 Emp	11.06%
5001-10000 Emp	18.58%
10000+ Emp	45.85%

Table 16: Percentage of total internship hires by company size

9 Discussion

9.1 Model Accuracy

The mean relative errors of the models used in this project ranged from 0.2617 for SVM to 0.7407 for Linear Regression, as shown in Table 17. We attempted to predict both the unstandardized and standardized number of hires, but the errors when predicting the standardized hires were much higher. Thus, when comparing the models we looked at the errors for predicting the unstandardized hires.

Model	MRE (unstandardized)	MRE (standardized)
Linear Regression	0.7407	2.526
RFECV with Linear Reg.	0.6977	1.422
SVM	0.2617	1.456
KNN	0.5156	1.461
Original Random Forest	0.5575	1.782
Modified Random Forest	0.3066	2.153

Table 17: Test error comparison of each model

The model with the lowest MRE was SVM at 0.2617, meaning the model was off by an average of 26.17% for its predictions. The model with the highest MRE of 0.7407 was Linear Regression using all 68 predictors, listed as just Linear Regression in the table. Our modified Random Forest model had the second lowest MRE at 0.3066, and was 15% lower than the traditional Random Forest's MRE (0.5575).

Although SVM achieved the lowest MRE, it was only through predicting one hire for every company. SVM was also not easy to interpret when trying to find which predictors were the most significant, which led us to put more emphasis on optimizing the Random Forest Regression model instead. Both Random Forest models had their strengths and weaknesses; the original Random Forest had a lower MRE when

making predictions for companies with a larger numbers of hires, while our modified Random Forest had a lower MRE when making predictions for companies with a smaller number of hires. However, both were successful in distinguishing between companies with one hire and multiple hires.

9.2 Predictors of Hiring Success

In order to determine the most important predictors of hiring success, we analyzed the features used in four different models: two linear regression models using RFECV, the original Random Forest model, and our modified Random Forest model. In order to compare the predictors determined to be most important by each model, we looked at the top 10 and bottom 10 predictors from each model, with the exception of the Linear Regression model which used 12 predictors.

Predictor	Top 10 Appearances
OCI Interviews	3
10000+ Emp	3
Job Posts	2
2015 Fall	2
Job: Resume Submission Method Accumulate in WPI Job Finder	2
Job: Degree Level Doctor of Philosophy	2

Table 18: Predictors appearing in the top 10 for multiple models

Two predictors, OCI Interviews and 10000+ Emp were in the top 10 predictors for 3 out of 4 of the models. This indicates that companies who conduct on-campus interviews and/or have 10,000+ employees are likely to hire WPI students. Four predictors overlapped in the original and modified Random Forests' top 10 features: OCI Interviews, 10,000+ Emp, Job: Resume Submission Method Accumulate in WPI Job Finder, and Job: Degree Level Doctor of Philosophy. In addition, none of the six predictors listed in Table 18 were in the bottom 10 predictors of any model, indicating a higher probability in truly being good predictors of hiring success.

Three predictors were in the bottom 10 predictors for multiple models: Job: Majors/Concentrations ChemM, 51-200 Emp, and Job: Unique Student Views. One of those, Job: Unique Student Views, was in the top 10 list for the Linear Regression model with three parameters, so we could not necessarily rule it as a good or bad predictor. Other predictors that were in the bottom 10 of a model that were not also in the top 10 of a model include Job: Resume Submission Method Other (enter below), Bronze Sponsor (F), Job: Class Level Alumni ($<$ 1 yr graduated), Job: Position Type Contract Professional, Job: Majors/Concentrations CS/ECEM, 1001-5000 Emp, and 201-1000 Emp.

The Linear Regression model with 12 predictors and the original Random Forest model disagreed on which predictors were most important, as 5 of the predictors in the Linear Regression model were in the bottom 10 of the Random Forest model. Four predictors, Silver Sponsor (F), Silver Sponsor (S), Count of Event Sponsor, and Job: Position Type General/Non-Professional Job, were not used by the Random Forest at all, while the other one, 2016 Life Sciences, had a very low rating. In addition, one of the predictors in the bottom 10 of the Linear Regression model,

Job: Student Favorite Count, was in the top 10 for the Random Forest model. Given the higher accuracy of the Random Forest model, we put more weight on that model's feature importance results, making it less likely that those predictors were good indicators of hiring success.

Overall, the best predictors of hiring success were OCI Interviews, 10,000+ Emp, Job: Resume Submission Method Accumulate in WPI Job Finder, Job Posts, 2015 Fall (career fair) and Job: Degree Level Doctor of Philosophy. The worst predictors were Job: Majors/Concentrations ChemM and 51-200 Emp. In addition, we observed hiring trends in both employer location and size. More than half of WPI students hired in 2016 ended up at companies with offices in Massachusetts, with companies in Boston hiring the most at 101 students. Companies with 10,000+ employees hired 38.73% of all WPI students hired in 2016 and 45.85% of the students that reported having an internship. Also, 47.79% of companies that size involved in recruiting on campus hired at least one student, further supporting the feature importance results.

10 Limitations and Recommendations

10.1 Project Limitations

The main roadblocks of the project were time, data quality and quantity, unbalanced data, and computing power. The timeline for the project was set at three academic terms since the beginning, requiring us to set our project goals according to what can be feasibly completed in the timeframe. Rather than to try answering all of the CDC's questions, we prioritized the most important and mathematically significant questions over the others. The main goal of the project was to find out how involved employers that hire WPI students are on campus, so we put predicting an employer's number of hire based on campus involvement at the top of the list. However, while collecting the data used in our analysis, we were able to look into some of the smaller questions, such as how the location and size of the employers plays a role in the number of WPI students hired.

While the data we received from the CDC was plentiful, there were many holes we needed to fill in order to get a working dataset. Data consolidation and cleaning took a large amount of time, including verifying if employers listed by students in the Final Outcomes survey actually existed. As we progressed with the project we also realized the size of an employer could be an important factor when determining how many students are hired. We were not given this data and had to gather it ourselves using sources such as Handshake and LinkedIn. Even with the additional data, our project was limited to the data we were given and/or collected. If all students reported their post-graduation status, our hire predictions would be much more conclusive. There were some students who reported a full or part-time job, but did not state any employer information, rendering the data points useless. There could also be other factors that play into the number of students a company hires, such as number of alumni currently working for a company or the number of interns hired from WPI.

Having unbalanced data created multiple hurdles for the project. The first issue was poor prediction accuracy, as discussed above, which we solved by creating a new error metric to calculate the relative error when building trees/forests. However, the

new error metric required us to write our own Random Forest Regression code, which turned out to be a big detour for the project and led to another issue, inefficiency.

A lack of computing power was the biggest limitation for the project. Since we had to write our own Random Forest Regression code, it was significantly less efficient than using the scikit-learn packages. Creating a regression tree with just 50% of the data (approximately 250 observations) took over an hour to run on our code, while creating an entire random forest with scikit-learn took seconds. Our temporary fix was to limit the training set to just 20% of the data, but that sacrificed accuracy of the model.

Unfortunately, due to the limited amount of time that we had, we were unable to implement a full analysis of feature importance. Although, as detailed above, we were still able to produce an ad hoc analysis using our regression forest.

10.2 Recommendations for Future Projects

The data used in the project was from the class of 2016, so future studies could use data from subsequent years. In particular, WPI switched from Job Finder to Handshake in the academic year of 2016-2017. Thus, it would be valuable to use data from the class of 2017 in order to determine the effectiveness of Handshake as opposed to Job Finder. It would be in the CDC's best interest to complete comparative studies of data from different years, as covering multiple class years would offer additional insights.

It would also be in the best interest of the CDC to further improve its data clean up and survey gathering methods. The most time-consuming part of the project was cleaning and merging various datasets. In particular, data cleaning would be a much faster process if the CDC took steps to ensure that data from various years is recorded in a consistent form. It would also be helpful if the CDC took measures to increase the number of students that provide information about their summer internships. While the CDC received information regarding post-graduation plans from over 95% of students in the class of 2016, internship information is reported at a much lower rate. Internship information would be helpful to have, especially for companies that use their resources primarily to recruit and hire interns.

Our feature importance showed that on-campus interviews were a top predictor for the Class of 2016 Dataset. In the future, the CDC could keep track of the number of on-campus interviews a company holds in each term, to determine if there is a specific time of the year that on-campus interviews are most effective. The CDC currently has a similar practice in place for keeping track of career fairs, as it notes which companies attend career fair in the spring and the fall instead of just listing career fairs as one general category.

Finally, we recommend that future teams reduce the number of subcategories in order to have cleaner runs. For example, any major-specific features such as Job: Majors/Concentrations MathB and Job: Majors/Concentrations CS/ECEB can be eliminated, leaving only the broader degree level features such as Job: Degree Level Bachelor of Science. As is previously mentioned, we had 67 predictor variables. Decreasing this amount could produce cleaner, more accurate results.

Appendices

A Major Groups

1. Job: Majors/Concentrations MathB
2. Job: Majors/Concentrations MathM
3. Job: Majors/Concentrations EngineeringB
4. Job: Majors/Concentrations EngineeringM
5. Job: Majors/Concentrations CivilB
6. Job: Majors/Concentrations CivilM
7. Job: Majors/Concentrations BioB
8. Job: Majors/Concentrations BioM
9. Job: Majors/Concentrations ChemB
10. Job: Majors/Concentrations ChemM
11. Job: Majors/Concentrations CS/ECEB
12. Job: Majors/Concentrations CS/ECEM
13. Job: Majors/Concentrations BUS/MGB
14. Job: Majors/Concentrations BUS/MGM
15. Job: Majors/Concentrations HumanitiesB
16. Job: Majors/Concentrations All Majors

B Group Contents

- MathB
  - Actuarial Mathematics
  - Mathematical Science
  - Physics
- MathM
  - Applied Mathematics
  - Data Science
  - Applied Statistics
  - Financial Mathematics
  - Industrial Mathematics
  - Mathematics for Educators
- EngineeringB
  - Aerospace Engineering
  - Electrical Engineering
  - Engineering Physics
  - Industrial Engineering
  - Mechanical Engineering
  - Robotics Engineering
- EngineeringM
  - Manufacturing Engineering
  - Materials Process Engineering
  - Materials Science & Engineering
  - Power Systems Engineering
  - Systems Engineering
- CivilB
  - Architectural Engineering
  - Civil Engineering
  - Construction Project Management
  - Environmental & Sustainable Studies
  - Environmental Engineering
- CivilM
  - Fire Protection Engineering

- BioB
  - Biochemistry
  - Biomedical Engineering
- BioM
  - Bioinformatics & Computational Biology
  - Biology & Biotechnology
- ChemB
  - Chemical Engineering
  - Chemistry
- ChemM
  - Materials Process Engineering
  - Materials Science & Engineering
- CS/ECEB
  - Computer Science
  - Computers with Applications
  - Electrical & Computer Engineering
  - Interactive Media & Game Development
- CS/ECEM
  - Information Technology
  - Data Science
- BUS/MNGB
  - Economic Science
  - Management Engineering
  - Management Information Systems
  - Management Science & Engr.
- BUS/MNGM
  - Business Administration
  - Financial Mathematics
  - MBA
  - Management
  - Marketing & Innovation
  - Operations Analytics & Management
  - Power Systems Management

  - System Dynamics & Innovation Management
  - Learning Sciences & Technology
- HumanitiesB
  - Humanities & Arts
  - Interdisciplinary
  - International Studies
  - Liberal Arts & Engineering
  - Professional Writing
  - Psychological Science
  - Society, Technology & Policy
- All Majors
  - Companies That Listed All Majors

C Predictor Variables

1. Job: Position Type Summer Internship: This predictor showed the number of summer internship positions a company posted.
2. Job: Position Type Intern/Co-op (4-8 months): This predictor showed the number of 4-8 month internship positions and co-ops a company posted.
3. Job: Position Type Full Time-Professional: This predictor showed the number of full time positions a company posted.
4. Job: Position Type General/Non-Professional Job: This predictor showed the number of non-professional positions a company posted.
5. Job: Position Type Part Time Job: This predictor showed the number of part time positions a company posted.
6. Job: Position Type Contract Professional: This predictor showed the number of contract professional positions a company posted.
7. Job: Majors/Concentrations MathB: This predictor showed the number of positions a company posted that were open to Math Majors pursuing a bachelor's degree. The majors that went into this category were Actuarial Mathematics and Mathematical Sciences.
8. Job: Majors/Concentrations MathM: This predictor showed the number of positions a company posted that were open to Math Majors pursuing a master's degree. The majors that went into this category were Applied Mathematics, Applied Statistics, Data Science, Financial Mathematics, Industrial Mathematics, and Mathematics for Educators.

9. Job: Majors/Concentrations EngineeringB: This predictor showed the number of positions a company posted that were open to engineering majors pursuing a bachelor's degree. The majors that went into this category were Aerospace Engineering, Biomedical Engineering, Chemical Engineering, Industrial Engineering and Robotics Engineering.

10. Job: Majors/Concentrations EngineeringM: This predictor showed the number of positions a company posted that were open to engineering majors pursuing a master's degree. The majors that went into this category were Aerospace Engineering, Biomedical Engineering, Chemical Engineering, Fire Protection Engineering, Manufacturing Engineering, Materials Process Engineering, Materials Science and Engineering, Power Systems Engineering, Robotics Engineering, and Systems Engineering.

11. Job: Majors/Concentrations CivilB: This predictor showed the number of positions a company posted that were open to civil engineering and environmental engineering majors pursuing a bachelor's degree. We grouped these two majors together because they are offered in the same department at WPI.

12. Job: Majors/Concentrations CivilM: This predictor showed the number of positions a company posted that were open to civil and environmental engineering majors pursuing a master's degree. We grouped these two majors together because they are offered in the same department at WPI.

13. Job: Majors/Concentrations BioB: This predictor showed the number of positions a company posted that were open to Biology and Biotechnology and Bioinformatics and Computational Biology majors pursuing a bachelor's degree.

14. Job: Majors/Concentrations BioM: This predictor showed the number of positions a company posted that were open to Biology and Biotechnology and Bioinformatics and Computational Biology majors pursuing a master's degree.

15. Job: Majors/Concentrations ChemB: This predictor showed the number of positions a company posted that were open to Chemistry and Biochemistry majors pursuing a bachelor's degree. We grouped these two majors together because they are offered in the same department at WPI.

16. Job: Majors/Concentrations ChemM: This predictor showed the number of positions a company posted that were open to Chemistry and Biochemistry majors pursuing a master's degree. We grouped these two majors together because they are offered in the same department at WPI.

17. Job: Majors/Concentrations CS/ECEB: This predictor showed the number of positions a company posted that were open to Computer Science, Electrical and Computer Engineering, and Interactive Media and Game Development Majors pursuing a bachelor's degree.

18. Job: Majors/Concentrations CS/ECEM: This predictor showed the number of positions a company posted that were open to Computer Science, Electrical and Computer Engineering, Computer Security, Information Technology, and Systems Modeling majors pursuing a master's degree.

19. Job: Majors/Concentrations BUS/MGB: This predictor showed the number of positions a company posted that were open to Business, Management Engineering, and Management Information Systems majors pursuing a bachelor's degree.

20. Job: Majors/Concentrations BUS/MGM: This predictor showed the number of positions a company posted that were open to Business Administration, Construction Project Management, Management, Marketing and Innovation, Operations Analytics and Management, Power Systems Management, Supply Chain Management, and System Dynamics and Innovation Management majors pursuing a master's degree.

21. Job: Majors/Concentrations HumanitiesB: This predictor showed the number of positions a company posted that were open to Humanities Majors pursuing a bachelor's degree. The majors that went into this category were Environmental and Sustainability Studies, Humanities and Arts, International and Global Studies, and Society, Technology, and Policy.

22. Job: Majors/Concentrations All Majors: This predictor showed the number of positions a company posted that were open to all majors.

23. Job: Class Level First Year (Undergraduate): This predictor showed the number of positions a company posted that were open to undergraduate first year students.

24. Job: Class Level Sophomore: This predictor showed the number of positions a company posted that were open to undergraduate sophomore year students.

25. Job: Class Level Junior: This predictor showed the number of positions a company posted that were open to undergraduate junior year students.

26. Job: Class Level Senior: This predictor showed the number of positions a company posted that were open to senior first year students.

27. Job: Class Level Graduate Student: This predictor showed the number of positions a company posted that were open to graduate students.

28. Job: Class Level Alumni ($<$ 1 yr graduated): This predictor showed the number of positions a company posted that were open to alumni of less than one year.

29. Job: Class Level First Year (Undergraduate): This predictor showed the number of positions a company posted that were open to alumni of one year or more.

30. Job: Degree Level Bachelor of Arts: This predictor showed the number of positions a company posted that were open to students pursuing a Bachelor of Arts degree.

31. Job: Degree Level Bachelor of Science: This predictor showed the number of positions a company posted that were open to students pursuing a Bachelor of Science degree.

32. Job: Degree Level Doctor of Philosophy: This predictor showed the number of positions a company posted that were open to students pursuing a Doctorate of Philosophy degree.

33. Job: Degree Level Master of Business Administration: This predictor showed the number of positions a company posted that were open to students pursuing a Master's of Business Administration.

34. Job: Degree Level Master of Engineering: This predictor showed the number of positions a company posted that were open to students pursuing a Master's of Engineering.

35. Job: Degree Level Master of Mathematics for Edu: This predictor showed the number of positions a company posted that were open to students pursuing a Master's of Mathematics for Education.

36. Job: Degree Level Master of Physics for Edu: This predictor showed the number of positions a company posted that were open to students pursuing a Master's of Physics for Education.

37. Job: Degree Level Master of Science: This predictor showed the number of positions a company posted that were open to students pursuing a Master's of Science.

38. Job: Degree Level Post Doc: This predictor showed the number of positions a company posted that were open to students pursuing a Post Doctorate Degree.

39. *Job: Resume Submission Method E-mail: This predictor showed the number of positions a company posted that accepted resume submissions by email.

40. Job: Resume Submission Method Accumulate in WPI Job Finder: This predictor showed the number of positions a company posted that accumulated resumes in WPI's Job Finder.

41. Job: Resume Submission Method Other: This predictor showed the number of positions a company posted that accepted resume submissions by some form other than the two listed above.

42. Job: Onestop Job: This predictor showed the number of positions a company posted that were Onestop Jobs.

43. Compensation Type: Indicates the number of positions a company posted that were paid

44. Job: Student Favorite Count: This predictor showed the number of times a student favorited one of the jobs a company posted.

45. Job: Unique Student Views: This predictor showed the number of unique student views one of the jobs posted by a company received.

46. Job: Total Student Views: This predictor showed the number of total student views one of the jobs posted by a company received.

47. 2015 Fall: Indicates whether a company attended the 2015 Fall Career Fair
48. Silver Sponsor (F): This predictor showed if a company was or was not a Silver Sponsor of the Fall 2015 Career Fair
49. Bronze Sponsor (F): This predictor showed if a company was or was not a Bronze Sponsor of the Fall 2015 Career Fair
50. 2016 Life Sciences: Indicates whether a company attended the 2016 Life Sciences Career Fair
51. 2016 Spring: Indicates whether a company attended the 2016 Spring Career Fair
52. Silver Sponsor (S): This predictor showed if a company was or was not a Silver Sponsor of the Spring 2016 Career Fair
53. Bronze Sponsor (S): This predictor showed if a company was or was not a Bronze Sponsor of the Spring 2016 Career Fair
54. Info Sessions: This predictor showed the number of on campus information sessions a company held.
55. Resume Book: This predictor showed the number of positions for which a company accepted resumes through resume book submissions.
56. Educational Programs: This predictor showed the number of educational programs that a company ran.
57. Job Posts: This predictor showed the number of job posts that a company placed using Job Finder.
58. Resume Searches: This predictor showed the number of resume searches that a company conducted.
59. Event Sponsor: Indicates whether a company sponsored an event other than a career fair
60. Office Sponsor: Indicates whether a company was a sponsor of the CDC office for the '15-'16 academic year
61. OCI Interviews: This predictor showed the number of on campus interviews than a company conducted.
62. 10000+ Emp: Indicates a company with more than 10,000 employees
63. 5001-10000 Emp: Indicates a company with 5,001-10,000 employees
64. 1001-5000 Emp: Indicates a company with 1,001-5,000 employees
65. 201-1000 Emp: Indicates a company with 201-1,000 employees
66. 51-200 Emp: Indicates a company with 51-200 employees
67. 1-50 Emp: Indicates a company with 1-50 employees

References

[1] V. García A. I. Marqués and J. S. Sánchez. On the suitability of resampling techniques for the class imbalance problem in credit scoring. *Journal of the Operational Research Society*, 64:1060–1070, 2013.

[2] Hervé Abdi and Lynne J. Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459, 2010.

[3] George Seber Allen Lee. *Linear Regression Analysis*. John Wiley and Sons, Hoboken, New Jersey, 2003.

[4] J. Serra B Ravi Kiran. Cost-compleixty pruning with out-of-bag samples. https://beedotkiran.github.io/forest.html, 2017.

[5] Gleb Beliakov and Gang Li. Improving the speed and stability of the k-nearest neighbors method. *Pattern Recognition Letters*, 33:1296–1301, 2012.

[6] Jason Brownlee. How to implement the decision tree algorithm from scratch in python. https://machinelearningmastery.com/implement-decision-tree-algorithm-scratch-python/, 2016.

[7] Corinna Cortes and Vladimir Vapnik. *Machine Learning*. Springer. Kluwer Academic Publishers, New York, 1995.

[8] Paulo Cortez and Alice Silva. Using data mining to predict secondary school student performance. In Antonio Brito and J. Teixeira, editors, *Proceedings of 5th FUture BUsiness TEChnology Conference*, pages 5–12, April 2008.

[9] Numba Developers. 1.3. compiling python code with @jit. http://numba.pydata.org/numba-doc/0.37.0/user/jit.html, 2012.

[10] Plotly Developers. Jupyter notebook tutorial in python. https://plot.ly/python/ipython-notebook-tutorial/, 2015.

[11] Scikit-Learn Developers. 1.10. decision trees. http://scikit-learn.org/stable/modules/tree.html, 2017.

[12] Luc Devroye and Laszlo Gyorfi. On the strong universal consistency of nearest neighbor regression function estimates. *The Annals of Statistics*, 22(3):1371–1385, 1994.

[13] Haibo He and Edwardo Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge*, 21:1263–1284, 2009.

[14] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:832–844, 1998.

[15] Laura Kahn. Machine learning for beginners-support vector machine. https://thedatalass.com/2018/02/26/support-vector-machine/, 2018.

[16] Steve Koppi. Post-graduation report class of 2016. Technical report, WPI, 2016.

[17] Bartosz Krawczyk and Michał Woźniak. Evolutionary cost-sensitive ensemble for malware detection. *International Joint Conference SOCO'14-CISIS'14-ICEUTE'14*, 14(978):433–442, 2014.

[18] Richard A. Olshen Leo Breiman, Jerome H. Friedman and Charles J. Stone. *Classification and Regression Trees*. Brooks/Cole Publishing, Monterey, 1984.

[19] Wes McKinney. pandas: Python data analysis library. http://pandas.pydata.org/, 2017.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[21] Vincent Spruyt. About the curse of dimensionality. https://www.datasciencecentral.com/profiles/blogs/about-the-curse-of-dimensionality, 2014.

[22] The WPI Career Development Center Staff. About the cdc. http://wp.wpi.edu/cdc/about/, 2018.

[23] Daniela Witten Trevor Hastie, Gareth James and Robert Tibshirani. *An Introduction to Statistical Learning*. Springer Texts in Statistics. Springer, New York, 2013.

[24] Rakshith Vasudev. What is one hot encoding? why and when do you have to use it? https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f, 2017.

[25] Young Zhang and Dapeng Wang. A cost-sensitive ensemble method for class-imbalanced datasets. *Abstract and Applied Analysis*, 2013(196256):1–6, 2013.