Process Automation & Data Matching

Project Team:

Griffin Curley - gdcurley@wpi.edu

Chase Goings - wcgoings@wpi.edu

Mirandi McCormick - mlmccormick@wpi.edu

Project Advisors

Professor Wilson Wong

Department of Computer Science

Professor Marcel Blais

Department of Mathematical Sciences

Professor Robert Sarnie

Professor Jim Ryan

WPI Business School

Abstract

For each prospective client engagement, Micronotes employs its Digital Prescreen Acquire process to deliver a comprehensive market breakdown. Previously, this process, being semi-manual, consumed more than a day. Following the successful implementation of this project, the entire procedure is now fully automated, featuring a functional front end and promptly delivered results via email, reducing the time frame to 1-2 hours. Integral to this process is the utilization of anonymous ADS data, necessitating the development of a machine learning algorithm to ensure accurate record matching across different weeks. The successful completion of this initiative has not only enhanced the daily operational efficiency but also established a robust foundation for further advancements in the model.

Executive Summary

Based in Boston, Micronotes was established in 2008 with their business model to enhance overall experience for financial institution customers via dialogue insight and cross-sold products to yield personalized marketing offers. To this end, Micronotes has leveraged machine learning and big data analytics within their processes. The team completed two projects that focused on process automation, data processing, data matching, and machine learning models. Both projects increased efficiency in Micronotes' internal processes.

Both projects used Azure Databricks to write Python scripts that used data stored in Data Lake containers. The first project involved automating the previously semi-manual process used for the company's Digital Prescreen Acquire product. The team finished creating the backend code to calculate rates based on data records specific to a financial institution's locations. These values were then saved to a JSON file to be used in the prescreen engine. After the data was sent to the engine, an output file would be produced with all the details on possible loan offers. This data then needed to be put into an easily readable form. The team created another process to do so based on an existing output file template provided by the company.

In the second project, the primary objective was the development of a machine learning (ML) matching algorithm. This algorithm was designed to analyze and compare two snapshots captured from the same zip code but from different weeks. Employing distinct attribute buckets, the algorithm systematically navigated through the dataset, seeking out approximate matches with precision. Upon completing the matching process, the algorithm generated outcomes as matched-up anonymous IDs. This functionality significantly contributed to the capability of tracking users on a granular week-by-week basis.

The team has a few recommendations for the continuation of both projects. For the Digital Prescreen Acquire, the team recommends the company creates a process to fabricate the in-depth PDF version of the current Excel file summary, as well as add the web app to their main website so customers have direct access to submit a request. For the data matching project, the team recommends implementing a decision tree to select additional features in aiding the matching process. As it stands, the algorithm has matched roughly 22,000 out of approximately 33,000 relevant data points in the test zip code. Different categories of features will aid in matching different people based on the information they possess in the data set.

Acknowledgement

First, we would like to thank our sponsors for such a great experience. Everyone was so friendly and helpful, which made it easy to get help and ask questions. We also got to learn so much more about the fintech world and how big data is handled.

We would also like to thank our WPI advisors: Rob Sarnie, Wilson Wong, Marcel Blais, and Jim Ryan. They were always there to help when we ran into blockers that were out of our control. They also made sure that we all would have a part in the project since we all have different majors.

Lastly, we would like to thank Marc Trudeau for helping all of the teams learn more about Scrum in PQP and during MQP.

Thank you,

Griffin Curley

Chase Goings

Mirandi McCormick

Table of Contents

Abstract....ii
Executive Summary....iii
Acknowledgement.... iv
List of Figures....xiii
List of Tables.... ix
Authorship....x
1. Introduction....1
    1.1 Automation for Prescreen Acquire Sales Prospects....1
    1.2 ADS Data Matching and Propensity Scores to Enhance Conversion Rates....2
2. Research....3
    2.1 Company Background.... 3
    2.2 Company Product....3
    2.3 Existing Prescreen Technology....4
    2.4 Machine Learning Techniques....4
        2.4.1 Multiple Linear Regression....4
        2.4.2 Tree-Based Methods.... 5
        2.4.3 Decision Tree Basics....5
        2.4.4 Improvements on Decision Trees.... 9
        2.4.5 k-Nearest Neighbor Classifier....11
        2.4.6 Locality-Sensitive Hashing....11
        2.4.7 Time Series.... 12
3. Methodology....13
    3.1 Agile Methodology....13
    3.2 What is Scrum?....13
        3.2.1 Roles of Scrum....13
        3.2.2 Epics and Themes.... 13
        3.2.3 User Stories....14
        3.2.4 Scrum Meetings.... 15
    3.3 Nearest Neighbors Search Algorithm.... 16
4. Software Development Environment....17
    4.1 Azure....17
    4.2 Azure Databricks 13.3.... 17
    4.3 Azure Data Lake.... 17
    4.4 Python 3.10.12.... 17
    4.5 Pandas 2.3.1.... 18

4.6 Apache Spark 3.4.1 18
4.7 Project Management and Communication 18
4.8 Experian ADS Data 18
5. Software Requirements 19
5.1 Digital Prescreen Acquire Automation Requirements 19
5.1.1 Themes and Epics 19
5.1.2 User Stories 20
5.2 ADS Data Matching Requirements 21
6. Design 23
6.1 Digital Prescreen Acquire Design 23
6.2 ADS Data Matching 24
7. Software Development 26
7.1 Section Breakdown 26
7.2 Pre-qualifying Project Work 26
7.3 Agile Scrum 27
7.4 Sprints 27
7.4.1 Sprint 0 27
7.4.2 Sprint 1 28
7.4.3 Sprint 2 30
7.4.4 Sprint 3 31
7.4.5 Sprint 4 32
7.4.6 Sprint 5 34
8. ADS Data Matching Results 35
8.1 Manual Matching 35
8.2 Data Cleaning 36
8.3 Feature Selection 37
8.4 Matching Algorithm 38
8.4.1 Matching Algorithm Code 39
9. Digital Prescreen Acquire Discussion 42
9.1 Zip Code Finder Automation 42
9.2 Rate Finder and Calculation Automation 42
9.3 Final Offer Calculations 42
9.4 Digital Prescreen Acquire Front End 43
9.5 Pipeline Connections 44
9.6 Digital Prescreen Acquire Conclusion 44
10. Assessment 45
10.1 Digital Prescreen Acquire 45

10.2 ADS Data Matching....45
10.3 Individual Learnings....46
10.3.1 Mirandi’s Learnings....46
10.3.2 Griffin’s Learnings....46
10.3.3 Chase’s Learnings....47
10.4 Team Learnings....47
11. Business and Risk Management....48
11.1 Innovation Risks....48
11.2 Operational Risks....48
11.3 Reputational Risks....48
11.4 Risk Management....49
11.5 Industry Learnings....49
11.6 Business Learnings....49
11.7 Culture Learnings....49
11.8 Remote Work Learnings....49
11.9 Time/ Value Add....50
12. Future Work....51
12.1 Additional Work for Digital Prescreen Acquire....51
12.2 Additional Work for ADS Data Matching....51
13. Conclusion....52
References....53
Appendix A....55

List of Figures

2.1 Prescreen Approval Example 4
2.2 Decision Tree Example 5
3.1 Scrum Hierarchy 14
3.2 The Big Picture of Scrum 15
6.1 Prescreen Acquire Data Flow Diagram 23
6.2 ADS Data Matching Data Flow Diagram 24
6.3 ADS Data Matching Entity Cluster....................25
7.1 Color Key 26
7.2 Project Burndown Chart 34
8.1 Manual Matches....................36
8.2 Data Match Example....................38
8.3 Feature Selection Code....................39
8.4 Data Scaling Code....................40
8.5 LSH Code....................40
8.6 Distance Computation Code....................40
9.1 Digital Prescreen Acquire Excel Output 43
9.2 Digital Prescreen Acquire Front End 43
9.3 Digital Prescreen Acquire Pipeline 44

List of Tables

5.1 Digital Prescreen Acquire User Stories 20
5.2 ADS Data Matching User Stories 22
7.1 Sprint 0 User Stories........ 27
7.2 Sprint 1 User Stories........ 28
7.3 Sprint 2 User Stories........ 30
7.4 Sprint 3 User Stories........ 31
7.5 Sprint 4 User Stories........ 32
8.1 Data Matching Criteria with Relevant Descriptions........ 35

Authorship

Section	Main Author(s)	Main Editor(s)
Cover Page	All	All
Abstract	Chase Goings	Griffin Curley & Mirandi McCormick
Executive Summary	Chase Goings & Mirandi McCormick	Griffin Curley
1. Introduction	Griffin Curley & Mirandi McCormick	Griffin Curley & Mirandi McCormick
2. Research	Griffin Curley & Mirandi McCormick	Mirandi McCormick
3. Methodology	Griffin Curley & Mirandi McCormick	Mirandi McCormick
4. Software Development Environment	All	Mirandi McCormick
5. Software Requirements	Mirandi McCormick	Griffin Curley
6. Design	All	Griffin Curley
7. Software Development	Griffin Curley & Mirandi McCormick	Mirandi McCormick
8. ADS Data Matching Discussion	Griffin Curley	Mirandi McCormick
9. Assessment	All	Griffin Curley
10. Business and Risk Management	Chase Goings	Mirandi McCormick
11. Future Work	Griffin Curley & Mirandi McCormick	Mirandi McCormick
12. Conclusion	Chase Goings	Griffin Curley & Mirandi McCormick

1. Introduction

The team completed two projects which improved efficiency and effectiveness within the company. The first focused on automating the Prescreen Acquire Sales Prospects process. Originally, this process was semi-manual, which led to human error and lengthy processing times due to data needing to be passed from one person to another. The completion of this project included finishing up the back end code, linking all the back end together, and completing the front end web page which can be accessed by employees to run the process. The second project focused on using anonymous Experian credit and loan data to create a time-series dataset spanning 6 months. The anonymous data was first matched by considering rows that did not change between two weeks to be the same person. For future work, a machine learning model would be used to predict which customers would be most likely to close on certain kinds of loans by assigning “propensity scores” to each person. These projects touched on common topics in the FinTech industry including data analysis, process automation, data matching, and machine learning.

1.1 Automation for Prescreen Acquire Sales Prospects

The goal of this project was to automate the currently semi-manual process that Micronotes has in place to find potential new customers who are preapproved for a certain loan at a financial institution. The original process included multiple employees running code in different environments on different machines and passing the needed information to the next person for the next step. This could possibly lead to errors from mistyping or copying and pasting. This also takes time away from employees who could be working on other projects.

This project was highly important in order to remove potential human error and to remove the need for manual input from employees. Automating the process will also allow financial institutions to input the necessary information into a form on the Micronotes website and receive a PDF report in a few hours after submission. A Micronotes employee will need to approve or deny the request upon submission to initiate the process.

The deliverables for this project included code and documentation. The majority of the engine was already written, however the processes needed to be linked together and some additional data pulling and storing was needed. The front end web app was also completed and linked the entire process together, allowing for anyone with the link to submit a process request. The main future work left for the company is creating a process which produces an in-depth PDF report of the final offers.

1.2 ADS Data Matching and Propensity Scores to Enhance Conversion Rates

The goal of this project was to improve Micronotes' ability to predict how likely customers are to close on a variety of loan types: personal, auto, home equity line of credit (HELOC), etc. The data that Micronotes receives each week is anonymous, so the first step of this project was to find attributes in the data that do not change over time to create an algorithm that will match these data points between snapshots. The data points in question are U.S.-based consumers in the Experian database. With this algorithm, we then created a time series dataset containing 6 months of data. With a more realistic dataset, we could then deploy a machine learning model that would more accurately represent real-time data.

A summer intern worked on a similar project for Micronotes without the time series component. This led to a model that did not fit the real data as well as they would have hoped. Their current conversion rate is about 0.4%, meaning that 0.4% of any kind of customer interaction leads to a loan acquisition. It was estimated that using a more realistic dataset, like a time series, could improve that to upwards of 1%. The tendency to convert people to customers was represented by a "propensity score" assigned by the machine learning model.

2. Research

2.1 Company Background

Micronotes is a privately held FinTech startup based in Boston and was founded by tech entrepreneurs from MIT in 2008. It is a small startup with only twenty employees. As described on its website, “Micronotes.ai delivers cloud-based digital engagement solutions to financial institutions that want to start conversations, develop relationships, and build trust with new and existing customers and members (Micronotes, 2023).” It does this with its “fast, automated, cloud-based solutions that apply machine learning to customer data (Micronotes, 2023),” improving engagement to sell more financial services. The startup has worked with prestigious banks such as Citizens and Bank of America.

Micronotes has a total of $23.3 million in funding, raised over nine funding rounds from five investors. Four of these investors are lead investors including BankTech Ventures, TTV Capital, Experian Ventures, and Harbor Light Capital Partners. Vestigo Ventures is also an investor; however, not a lead investor. Their most recent funding was raised on July 19, 2023 from a Series C round totaling $7.5 million (Crunchbase, 2023) with $2 million invested from BankTech Ventures.

One of their largest advantages over competitors, like Act-On, lies in the sheer amount of data Micronotes has access to. Every week, they get Experian data from 230 million Americans which allows them the opportunity to gain more customer insight than anyone in a similar field. However it is not just their access to this data, but the methodology and analytical techniques they use that allows them to take full advantage of it. A wide variety of data sources like “bank-held data, credit data, property data, location data, behavioral data, and more (Micronotes, 2023)” gives Micronotes the ability “to deliver unprecedented insight to consumers and the institutions that serve them (Micronotes, 2023).” They also leverage machine learning to make predictive models that help their customers close deals by focusing on business while Micronotes focuses on the tech stack.

2.2 Company Product

Micronotes currently has three public products: Micronotes Cross-Sell™, Micronotes Digital Prescreen™, and Micronotes Prescreen Acquire™. Cross-sell helps to raise engagement with banner ads, emails, and SMS messages. Digital Prescreen helps to bring existing account-holders from other competitor institutions, to the customer financial institution. All of Micronotes products have a focus on raising engagement and profit for their financial institution customers. The product that we are closely working with is Micronotes Prescreen Acquire. According to Micronotes, their product is the, “first credit marketing platform for community financial institutions focused on acquiring new accounts in their service area” (“Digital Prescreen: AI for

Financial Institutions" 2023). This is accomplished by their ability to leverage over 200 million consumer credit records in combination with advanced algorithms.

2.3 Existing Prescreen Technology

To fully understand how Micronotes Prescreen Acquire works, an understanding of digital prescreen technology is needed. Traditional prescreen technology is nothing new, and most people who have a bank account have probably come across this – with online banking or even through the mail. Figure 1 to the right shows an example of this sort of prescreen technology. Prescreening, in this context, occurs when companies look into users' credit and pre-approve them for things such as loans or credit cards (Wendel 2023).

Although technology is advancing, some banks may find themselves falling behind in terms of automation. Lenders used to only use manual, paper-based loan approval processes. This caused slower decision times and more work for bankers. Banks then transitioned to manual, digital spreadsheets. However, this still did not speed up the process by much since bankers would still need to manually fill the spreadsheets with the underwriting criteria. According to a survey by Moody's Analytics, over 50% of bankers reported that the manual collection of data and the back and forth between their clients was the most challenging part of the loan initiation process (Peterson 2018).

Figure 2.1 Prescreen Approval Example

2.4 Machine Learning Techniques

This section provides the necessary mathematical background for the data science and machine learning techniques used throughout the project.

2.4.1 Multiple Linear Regression

Simple linear regression does well at estimating a response for a single predictor variable. When we have more than one predictor variable it becomes less effective. There is the option of creating multiple simple linear regression models, but this can cause problems if they are correlated. What we do instead is create one model where each predictor has a separate slope coefficient. (James, 2013) So, if we have n predictors the model becomes

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \varepsilon \tag{2.1}$$

where X_j represents the jth predictor and β_j represents the relationship between that variable and the response. You can think of β_j as the average effect on Y of a one unit increase in X_j, holding all else fixed.

Each of the coefficients β_0, β_1, ..., β_n in (2.1) are unknown, so we need a way to estimate them. Now we define new estimates for these coefficients given by $\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_n$, and we make predictions with the formula

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \ldots + \hat{\beta}_n x_n. \tag{2.2}$$

We choose β_0, β_1, ..., β_n to minimize the residual sum of squares

$$RSS = \sum_{i=1}^{k} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{k} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \cdots - \hat{\beta}_n x_{in})^2. \tag{2.3}$$

2.4.2 Tree-Based Methods

This section will provide a background of some of the basics of tree-based methods and a more in depth treatment of the methods we used in this project. Before getting into the math we use to build these diagrams, here is some information their about terminology and structure:

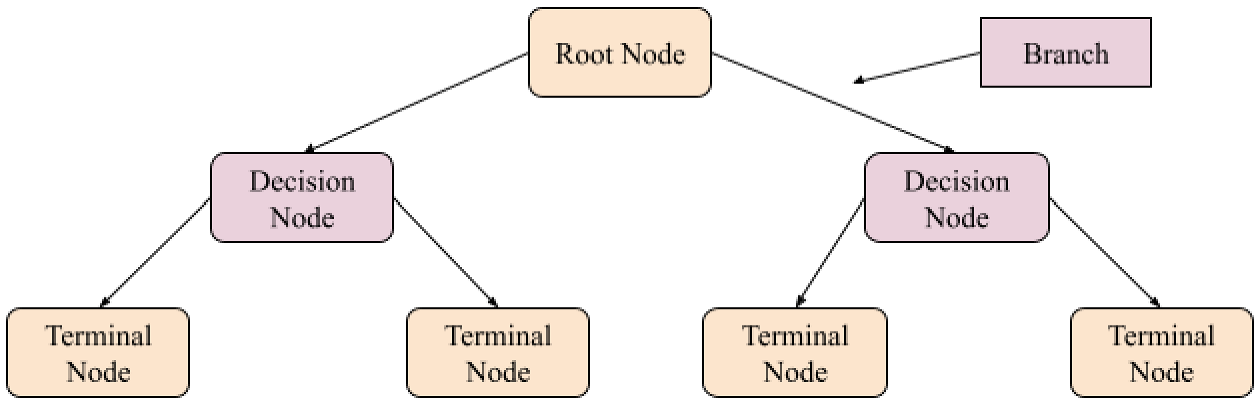


Figure 2.2 Decision Tree Example

Typically, in a decision tree the right-hand branches represent an affirmative or “greater than or equal to” response to the criteria.

2.4.3 Decision Tree Basics

Decision trees can work well with both regression and classification problems. We will first look at regression trees. The construction of a regression tree can be broken up into a simple two step process:

1. Divide the predictor space—the possible values for $X_1, X_2, ..., X_p,$ *where each* X_i *for* $i = 1, 2, 3, ..., p$ is a given observation—into J distinct and non-overlapping regions, $R_1, R_2, ..., R_J$.
2. For every observation that falls into the region R_J we make the same prediction, which is just the mean of the response values for the training observations in R_J. (James, 2013)

The question now becomes, how do we construct the regions $R_1, R_2, ..., R_J$? For convenience, we choose to divide the predictor space into high-dimensional rectangles which we will refer to as *boxes.* The goal is to find such boxes that minimize the residual sum of squares (RSS), given by

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \widehat{y}_{R_j})^2, \tag{2.4}$$

where $\widehat{y}_{R_j}$ is the mean response for the training observations within the jth box. To consider every possible partition of the feature space into J boxes would be too computationally expensive. Instead we take a top-down, *greedy* approach known as *recursive binary splitting*. Where we begin at the top of the tree and then successively split the predictor space in which each split is indicated by two new branches lower on the tree. The process is *greedy* because we make the most optimal split at each step without considering future splits. It is entirely possible that the "best tree" will not make the "best choice" at each step (James, 2013).

In short, to perform recursive binary splitting, we select the predictor X_j and the cutpoint *s* such that splitting the predictor space into the regions $\{X \mid X_j < s\}$ *and* $\{X \mid X_j \geq s\}$ leads to the greatest reduction in RSS. So we consider all predictors $X_1, ..., X_p$, and all possible values of the cutpoint *s* for each predictor, and choose the resulting tree with the lowest RSS. Formally, for any j and s, we define the pair of half-planes

$$R_1(j, s) = \{X \mid X_j < s\} \text{ and } R_2(j, s) = \{X \mid X_j \geq s\}, \tag{2.5}$$

and we find the value of j and s that minimize the equation

$$\sum_{i:\, x_i \in R_1(j,s)} (y_i - \widehat{y}_{R_1})^2 + \sum_{i:\, x_i \in R_2(j,s)} (y_i - \widehat{y}_{R_2})^2, \tag{2.6}$$

where $x \in X$, $\widehat{y}_{R_1}$ is the mean response for the training observations in $R_1(j, s)$, and $\widehat{y}_{R_2}$ is the mean response for the training observations in $R_2(j, s)$.

The next step is to repeat this process with the two previously identified regions; which now gives us three regions. Like before we split one of these three regions so as to minimize the RSS. We continue this process until a stopping criterion is met. This criterion could be anything; for example, we can continue until no region contains more than ten observations.

Recursive binary splitting is likely to have good predictions on the training set, but is likely to overfit the data, leading to poor test set performance. To remedy this, we want to reduce the complexity of the resulting tree through a process called *tree pruning*. This should result in a smaller tree with fewer regions $R_1, R_2, ..., R_J$ that might lead to lower variance and better interpretation at the cost of a little bias. The strategy is grow a large tree T_0, and then *prune* it back to obtain a *subtree*. However, before outlining that strategy, we need to talk about how to find the K-fold cross-validation (CV) estimate. The approach includes dividing the set of observations into k random groups—or folds—of approximately equal size. The first fold is the validation set, and the method is fit on the remaining $k - 1$ folds (Hastie, 2009). The mean squared error–computed with equation (2.4)–(MSE), MSE_1, is then computed on the observations in the excluded folds. Repeat this procedure k times; each time treating a different group of observations as a validation set until you have k estimates of the test error, $MSE_1, MSE_2, ..., MSE_k$. The K-fold CV estimate is written as $CV_{(k)}$.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2, \tag{2.7}$$

where $\hat{f}(x_i)$ is the prediction that $\hat{f}$ gives for the ith observation. In our case, $\hat{f}(x_i) = \hat{y}_{R_j}$.

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i \tag{2.8}$$

With that, we can now talk about *cost complexity pruning* which gives us a way to select a subtree other than considering every possible subtree. We do this by considering a sequence of trees indexed by a nonnegative tuning parameter α (James, 2013). For each value of α there is a corresponding subtree $T \subset T_0$ such that

$$\sum_{m=1}^{|T|}\sum_{i:\, x_i \in R_m}(y_i - \hat{y}_{R_m})^2 + \alpha|T| \tag{2.9}$$

is as small as possible. Here |T| indicates the number of terminal nodes—nodes with no branches below it—of the tree T, R_m is the rectangle corresponding to the mth terminal node, and $\hat{y}_{R_m}$ is the predicted response associated with R_m. This tuning parameter controls a trade-off between the subtree's complexity and its fit to the training data. If we think about (2.6) intuitively, when $\alpha = 0$, the subtrees T and T_0 will be identical because as the $\alpha|T|$-term vanishes (2.6) is just a measure of the training error. Conversely, as α increases, there is a price to pay for having a tree with many terminal nodes (|T|) and so the quantity will tend to be minimized.

As it so happens, as we increase α from zero in (2.6), branches get *pruned* from the tree in a nested and predictable fashion, so finding the entire sequence of subtrees as a function of α is easy. We can select a value of α using a validation set or using CV. We then return to the full data set and obtain the subtree corresponding to α. This process is summarized in the following algorithm:

1. Use recursive binary splitting to grow a large tree on the training data, stopping based on a minimum number of observations criterion.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.
3. Use K-fold CV to choose α. That is, divide the training observations into K folds. For each $k = 1, ..., K$:
    a. Repeat Steps 1 and 2 on all but the kth fold of the training data.
    b. Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

    Average the results for each value of α, and pick α to minimize the average error.
4. Return the subtree from Step 2 that corresponds to the chosen value of α.

The fundamental difference between regression and classification trees is that classification trees are used to predict qualitative responses to criteria. Other differences include using the *most commonly occurring class* for training observations rather than the mean response that we used for regression. The process for growing classification trees is also similar to the process we just described for regression trees. In this case, we also use recursive binary splitting; however, with classification, we cannot use RSS as the criterion for making the splits. Instead we use the *classification error rate*, which is the fraction of the training observations in that region that do not belong to the most common class:

$$E = 1 - max_k(\widehat{p}_{mk}). \tag{2.10}$$

Here $\widehat{p}_{mk}$ represents the proportion of training observations in the m^{th} region that are from the kth class. The problem arises in the fact that this error is linear and therefore not sensitive enough for tree-growing. In practice, there are two alternatives that we prefer known as the *Gini index* and *cross-entropy* given respectively by

$$G = \sum_{k=1}^{K} \widehat{p}_{mk}(1 - \widehat{p}_{mk}), \tag{2.11}$$

$$D =- \sum_{k=1}^{K} \widehat{p}_{mk} log\widehat{p}_{mk}. \tag{2.12}$$

It is clear that the Gini index (2.11) and cross-entropy (2.12) both take on small values if the $\widehat{p}_{mk}$'s are close to zero or one. (James, 2013) This quality allows us to measure the *purity* of the m^{th} node – a small value indicates that a node contains predominantly observations from a single class. When pruning the tree, any of these approaches may be used, however the classification error rate (2.10) is typically used if prediction accuracy of the final pruned is the goal.

2.4.4 Improvements on Decision Trees

The decision trees discussed above in section 2.5 are subject to high variance. For example, if we were to randomly split the training data into two parts and fit a decision tree to both halves, the results could be very different. In this section we will build up some preliminary information to discuss random forests, which help to solve this problem.

Bootstrapping is similar to cross-validation in that they both estimate conditional error Err_τ, however it is especially useful in cases where variance is otherwise hard to obtain. The process involves creating *bootstrapped* data from the original data using replacement. This means that we take n random observations from the original data set where the same observation can be recorded more than once. We call these bootstrapped data sets $Z^{*1}, Z^{*2}, ..., Z^{*B}$ for a large value of B, and each bootstrapped data set is the same size as the original data set. We will depict any quantity that we may want to compute from these bootstrapped data sets as $S(Z^{*1}),\ S(Z^{*2}), ...,\ S(Z^{*B})$. For example, we can look at variance,

$$\widehat{Var}[S(Z)] = \frac{1}{1-B} \sum_{b=1}^{B} (S(Z^{*b}) - \bar{S}^{*})^2, \tag{2.13}$$

where $\bar{S}^{*} = \sum_{b=1}^{B} \frac{(S(Z^{*b})}{B}$ (Hastie, 2009).

The technique of bagging can be thought of as *aggregated bootstrapping* where instead of judging the accuracy of a parameter estimate, we improve upon bootstrapping to enhance the estimate or prediction itself. As the name would suggest, we take the prediction from each bootstrapped data set, and average over all B bootstrap samples. If each prediction is given by $\widehat{f^{*b}}(x)$ for each input x, the bagging estimate is given by

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \widehat{f^{*b}}(x) \tag{2.14}$$

Trees are great models for bagging because if they are grown large enough, they have relatively low bias for how well they can capture complex structures in the data. This leads us to random forests, which is a modification of bagging.

Random forests use an average of de-correlated trees and are easier to train and tune, compared to bootstrapping. One advantage of the trees generated during bagging is that they are identically distributed, meaning that the expectation of an average of B trees is equal to the expectation of any one of them. This implies that bagged trees have the same bias as individual trees, so we need to look at variance reduction. Consider the average of B independent identically distributed random variables, each with variance σ^2, has variance $\frac{1}{B}\sigma^2$ (Hastie, 2009). If the variables are not independent we end up with a positive pairwise correlation ρ, and the variance of the average is now,

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2. \tag{2.15}$$

The goal with random forests is to improve the variance reduction of bagging by reducing the correlation between the trees, without increasing variance too much. We do this by randomly selecting input variables during the tree-growing process. For instance, when growing a tree on a bootstrapped dataset:

Before each split, select $m \leq p$ of the input variables at random as candidates for splitting, where p is the total number of variables. Values for m are usually $1 \leq m \leq \sqrt{p}$. After B such trees $\{T(x; \Theta_b)\}$ for $b = 1, 2, 3, ..., B,$ the random forest predictor is

$$\widehat{f_{rf}^{B}}(x) = \frac{1}{B}\sum_{b=1}^{B} T(x; \Theta_b). \tag{2.16}$$

Here, Θ_b represents the b^{th} random forest tree in terms of split variables, cutpoints at each node, and terminal node values (Hastie, 2009). Reducing m will reduce the correlation between any pair of trees, and by (2.15) will reduce the variance of the average.

2.4.5 k-Nearest Neighbor Classifier

The k-nearest-neighbor classifier is a fairly simple method that does not rely on an underlying machine learning model to make predictions. In essence, the algorithm is based on the idea that similar data points will have similar values or labels. When the algorithm is making predictions it will calculate the distance, typically Euclidean distance, between the input data and the training data. It will then find the *k-nearest-neighbors* based on those distances where k is an integer and varies based on the data. In classification problems, the most frequently occurring class label will be assigned to the input data point. In regression problems, the average or weighted average of the k-neighbors will be used for the value of the input data point. It is common for the data to be *normalized* since it is common for the relevant features to have different ranges and having different magnitudes for their values will throw off the distance measurement, leading to some data points not being weighed as heavily (Hastie, 2009).

2.4.6 Locality-Sensitive Hashing

One way to perform the nearest neighbor search – especially in high dimensions – is with locality-sensitive hashing (LSH). To discuss this process further, some definitions need to be considered.

Definition 2.1 (Randomized c-approximate near-neighbors) (Clarkson, 2006)

Given a set P of points in a d-dimensional space R^d, and parameters R > 0, $\delta > 0$, construct a data structure which, given any query point q, does the following with probability $1 - \delta$: if there is an R-near neighbor of q in P, it reports a cR-near neighbor of q in P.

Definition 2.2 (Locality-sensitive hashing) (Clakson, 2006)

A family H is called (r, cr, P_1, P_2)-sensitive if for any $p, q \in R^d$

-if $||p - q|| \leq R$ then $P_H[h(q) = h(p)] \geq P_1$,

-if $||p - q|| \geq cR$ then $P_H[h(q) = h(p)] \leq P_2$.

Here, h(x) is a hash function that ensures that, for each function, the probability of input values being mapped to the same output values is much higher for points that are close to each other, and P_H represents probabilities (Clarkson, 2006). A hash function is any function that maps data of arbitrary size to fixed-size values that are used to index a fixed-size table called a hash table.

The use of a hash function to index a hash table is called $hashing$. For these families to be useful, they must satisfy $P_1 > P_2$. So, given such a family with the parameters defined above, one can pronounce the gap between P_1 and P_2 by linking several functions. In particular, for k and L, we choose L functions $g_j(q) = (h_{1,j}(q), ..., h_{k,j}(q))$, where $h_{t,j}(1 \leq t \leq k, 1 \leq j \leq L)$ are chosen independently and uniformly at random from H. During preprocessing, each input point set $p \in P$ is in the bucket $g_j(p)$, for $j = 1, ..., L$. Since the total number of buckets may be large, only the non empty buckets are retained by hashing the values $g_j(p)$. Finally, to process a query q, all buckets $g_1(q), ..., g_L(q)$ are searched through until either all points are retrieved or after finding the first L points. The benefit of this method is that it does not require searching through every point to find similar points, since similar points are placed in a bucket with high probability.

2.4.7 Time Series

As the name implies, a time series can be used with any variable that changes over time. We can imagine any graph where the x-axis is time for the simplest example. This can be exceptionally useful with large quantities of data because understanding how things have changed over time in the past can help us make predictions in the future. This is called time series forecasting, or just *forecasting,* which uses historical values and associated patterns to predict future activity (Hayes, 2022).

3. Methodology

3.1 Agile Methodology

There are a handful of methodologies which are effective for group work. For the assigned project purpose, Scrum is the agile methodology that was used. With a limited amount of time to complete the project, sprints were set at one week long. Project requirements were also expected to change rapidly at any point. Due to the short sprints and the rapid changes, Scrum was the best fit.

3.2 What is Scrum?

This software development methodology helps teams to self-organize through a set of principles, values, tools, and roles. It is also helpful for teams to learn from their past sprints and adapt changes for future sprints (Jacobson, Sutherland, Kerr, & Buhnova, 2022). The two main Scrum tools are the Product Backlog and the Sprint Backlog. The Product Backlog is a list of requirements and fixes that could be necessary to complete before the project is finished. The Sprint Backlog is the current list of items that are planned to be completed in the current sprint. Each Sprint also has a Sprint Goal which is the overall goal to reach by the end of the sprint. Items can be added to the sprint or carried over to the next sprint to complete the Sprint Goal (Amazon Web Services).

3.2.1 Roles of Scrum

Scrum also includes roles to help sprints run smoothly. The first role is the Product Owner. The main focus of this role is to help the team maximize the value of the product for the business. They help to decide which items to bring into the next sprint and help to fill any gaps between the wants of the business and the knowledge of the team. The second role is the Scrum Master. The main focus of this role is to hold the team accountable on their progress and deliverables. They help to motivate the team and facilitate meetings. The final role is the Scrum development team, which is taken on by everyone on the team. This role includes testers, UX designers, and developers who all work together and help teach each other (Schwaber & Sutherland, 2020).

3.2.2 Epics and Themes

Agile Scrum provides a way to structure development work through a hierarchy which goes from a higher overall strategic focus, down to a more precise technical focus. Figure 3.1 shown below illustrates how the hierarchy is broken down. At the top there are Themes – the current theme being the Product Goal which is the overall goal for the completion of the current product. These are broken down into Epics, which are broken down into User Stories, which can then be broken down into Tasks. This helps teams by starting with a high-level overview and goal that can slowly be chunked into more specific, bite-sized tasks (Aha!, 2023).

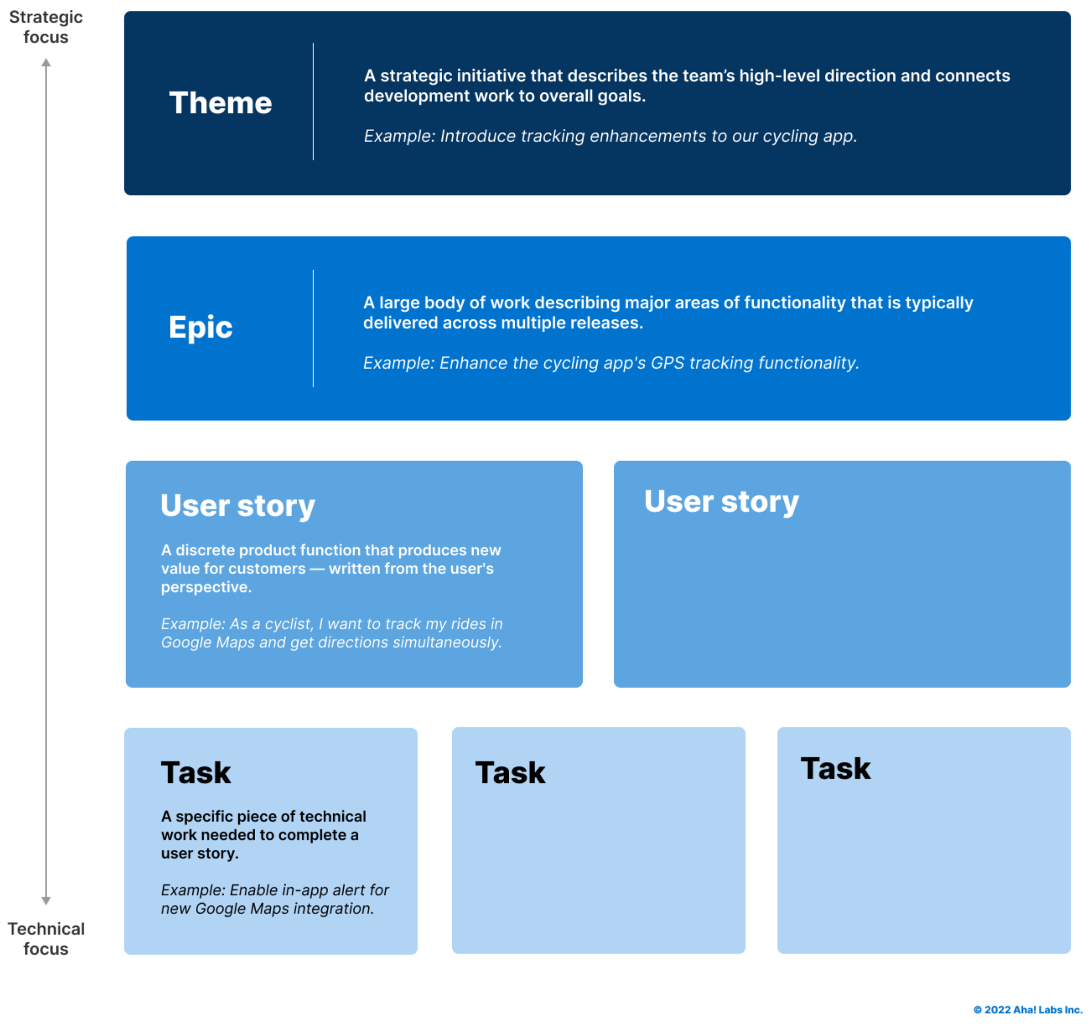

Strategic focus
Theme
A strategic initiative that describes the team's high-level direction and connects development work to overall goals.
Example: Introduce tracking enhancements to our cycling app.
Epic
A large body of work describing major areas of functionality that is typically delivered across multiple releases.
Example: Enhance the cycling app's GPS tracking functionality.
User story
A discrete product function that produces new value for customers — written from the user's perspective.
Example: As a cyclist, I want to track my rides in Google Maps and get directions simultaneously.
User story
Task
A specific piece of technical work needed to complete a user story.
Example: Enable in-app alert for new Google Maps integration.
Task
Task
Technical focus
© 2022 Aha! Labs Inc.

Figure 3.1 Hierarchy of Scrum (Aha!, 2023)

3.2.3 User Stories

User Stories are one of the key concepts in Scrum. They help to describe and define a feature or requirement from the perspective of the product user. During development, members of the team may lose sight of what the users want to get out of the story's completion, by having clearly defined user stories teams can lessen this issue. These stories also help with testing expectations by having acceptance criteria that confirms if the story is completed or not. User stories can also be split into more user stories or sub-tasks. If a user story cannot be completed in one sprint, it should be split into more user stories. If a user story can be completed in one sprint, it should be split into sub-tasks (Kerzner, 2018).

All user stories follow the same template:

- As a < user role | actor | system >,

- I want to < desired goal or outcome >,
- So that I may < anticipated value or benefits >,
- I will know I am done when < all the acceptance criteria >.

This format helps to give a general idea of the overall goal, leaving room for more precise details to be fleshed out later with the help of the product owner (Kerzner, 2018).

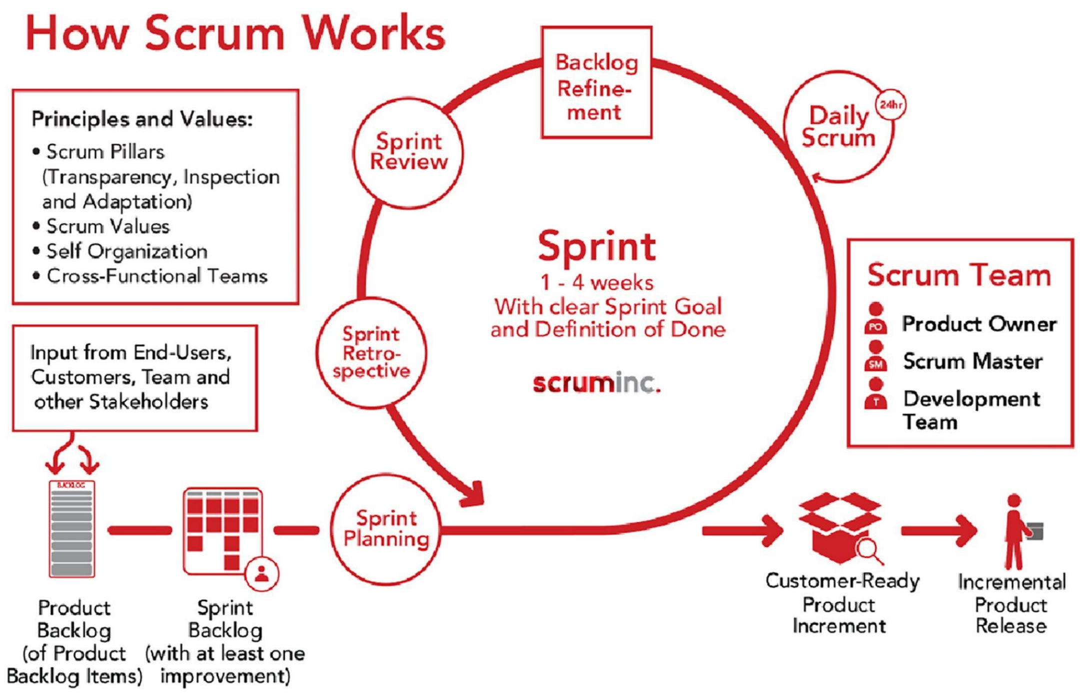

How Scrum Works
Principles and Values:
• Scrum Pillars (Transparency, Inspection and Adaptation)
• Scrum Values
• Self Organization
• Cross-Functional Teams
Input from End-Users, Customers, Team and other Stakeholders
Backlog Refine-ment
Sprint Review
Daily Scrum
24hr
Sprint
1 - 4 weeks
With clear Sprint Goal and Definition of Done
scruminc.
Sprint Retro-spective
Scrum Team
Product Owner
Scrum Master
Development Team
Sprint Planning
Product Backlog (of Product Backlog Items)
Sprint Backlog (with at least one improvement)
Customer-Ready Product Increment
Incremental Product Release

Figure 3.2 The Big Picture of Scrum (Jacobson, Sutherland, Kerr, & Buhnova, 2022)

3.2.4 Scrum Meetings

Throughout a scrum sprint, a team will have five main meetings that continuously cycle, as shown in Figure 3.2. At the very beginning of a project, before sprints can begin, a team must hold two meetings. The first is to create the product backlog, which consists of items to complete based on input from users, customers, stakeholders, and the team. The second meeting is to create the sprint backlog, which consists of the technical items that a team will complete in the sprints until the product is fully finished.

Once a sprint is about to begin, the team will hold the initial Sprint Planning meeting. During this meeting, the team discusses with the Product Owner what items from the backlog need to be pulled into the sprint in order to reach the sprint goal. The Scrum Master will decide based on previous sprints how many points the team can pull in and be expected to successfully finish. Once Sprint Planning is completed, the team begins the sprint.

During the sprint, the team will have Daily Scrum meetings. These meetings take place everyday at a set time and usually are set as a 15 minute meeting. During this time, the team members take turns sharing what they completed the day prior, what they are working on and expect to complete for the day, and discuss any setbacks or issues that the team can help them with. The Scrum Master helps to facilitate these meetings and assign tasks to team members who have less on their plate (Sutherland et al., 2019).

Around mid-sprint, teams will have a Backlog Refinement meeting. At this meeting, the team discusses with the Product Owner which stories should be placed at the top of the backlog to pull into the next sprint. This also includes creating sub-tasks and story pointing. The whole team helps to sub-task and vote on how many points a story is worth. This voting is done blindly and done with the help of a tool such as PlanITpoker. All team members submit what they think a story should be pointed at, and once all members have voted all cards are shown and then discussed to decide on the final point value (Sutherland et al., 2019).

At the end of a Sprint, teams will have a Sprint Review. This meeting involves the stakeholders of the product to see and assess the current status of the product. This meeting is also set to be at most three hours long. Teams will show progress and any possible demos to the stakeholders. Stakeholders can ask questions and give any feedback on what they like or wish to see changed (Sutherland et al., 2019).

The final meetings teams will have is the Sprint Retrospective. The focus of this meeting is for the team to discuss the technicals of what went well and what can be improved on. It is crucial to hold this meeting as soon as possible after the completion of a sprint in which the triumphs and challenges of the sprint are still fresh. These meetings alone are not enough to improve a team, however they provide the space for the team to explore deeper issues. It is also important that teams do not use this time as a complaint session, but focus the energy on possible solutions and improvements for the next sprint that the team can implement (Sutherland et al., 2019).

3.3 Nearest Neighbors Search Algorithm

The algorithm that was developed to automatically match the data points in section 3.3 uses locality-sensitive hashing and the Pyspark function approxSimilarityjoin() to perform a nearest neighbor search. The logic behind this choice was that given any features to be used for comparison, they will not change much, if at all, in the span of one week. A nearest neighbor search would then be an effective way of identifying rows of data that are close in distance as matches. Additionally, this method works well with high dimensional data. Before the search is performed, closely related data points are placed into the same bucket with a high probability, which limits the computational burden of searching through every data point.

4. Software Development Environment

4.1 Azure

Azure, developed by Microsoft, is a comprehensive cloud computing platform encompassing a vast array of servers and networking hardware. This sophisticated infrastructure is meticulously designed to facilitate seamless access to an extensive range of cloud services and resources. As a cloud solution, Azure empowers businesses and individuals to harness the power of scalable computing, storage, and networking capabilities, all provided as services over the Internet (Microsoft, 2023).

4.2 Azure Databricks 13.3

Azure Databricks is used for a multitude of applications including, “building, deploying, sharing, and maintaining enterprise-grade data, analytics, and AI solutions at scale” (Microsoft, 2023). Azure Databricks is very useful in processing Big Data, since it is built on top of the Apache Spark framework. It is able to do this by breaking up the data into different tasks to send to many worker processes to achieve parallelization.

In Azure Databricks, users create a Notebook to write code in. These notebooks are divided into Cells, which the user decides. The user also decides what language they will code in; the choices being Python, SQL, Scala, and R. These notebooks are automatically saved and updated as the user types, which is perfect for coding alongside other people without the need for pushing and pulling code from GitHub (Microsoft, 2023).

4.3 Azure Data Lake

The Azure platform offers a product called Azure Data Lake which is a data storage platform. In better terms, a data lake is, “a centralized repository that ingests and stores large volumes of data in its original form” (Microsoft). Data lakes are also useful for storing any data type. This includes but is not limited to: structured data such as database tables, semi-structured data such as web pages, and unstructured data such as images. This Azure service helps businesses by giving them data storage which is secure, accessible, and central which helps to optimize storage costs. Some major use cases of the data lake service includes: streaming media, finance data, healthcare data, the internet of things, and more (Microsoft).

4.4 Python 3.10.12

In terms of programming languages, Python is a high-level language with built in data structures, dynamic typing, and dynamic binding. Since it is a high-level language, it is intuitive and powerful, and supports procedure-oriented and object-oriented programming. Python also comes with a built-in extensive library, as well as many other packages which can easily be installed (H,

S. C.). For many of the company's projects, they use Python alongside other packages and modules such as Pandas and NumPy.

4.5 Pandas 2.3.1

Pandas is an open-source library built on Python. It is the most popular library for data manipulation and analysis. Pandas improves the capabilities of Python by giving it the ability to interpret spreadsheet style data. It also works with structured data formats like tables and time series data. This makes it easy to use with other scientific libraries for Python (NVIDIA).

The main data structure used in Pandas is called a DataFrame. These are 2D array-like data tables. DataFrames in Pandas are largely used in machine learning. Data manipulation and cleaning is also easily accomplished with Pandas. This includes sorting, joining, plotting, statistics, as well as derived columns and grabbing subsets. After the data is manipulated, Pandas can import and export DataFrames into CSV and JSON files (Pandas, 2023).

4.6 Apache Spark 3.4.1

Apache spark is an open-source, distributed computing system that provides a fast cluster-computing framework for big data processing. It is a "multi-language engine for executing data engineering, data science, and machine learning on single-node machines or clusters" (Apache Spark, 2023). Without it, processing large amounts of data would be infeasible given any reasonable time constraint.

4.7 Project Management and Communication

The primary channel used for intra-team communication was Discord, which provided a dynamic and collaborative space. Outlook and Microsoft Teams meetings were predominantly used for interactions with advisors, ensuring effective communication and updates. Sponsors were updated daily through Teams meetings and direct messages, facilitating seamless coordination and discussions. Jira was used to monitor and track sprint progress, streamline project management, and ensure clarity on tasks and milestones.

4.8 Experian ADS Data

Micronotes gets weekly credit and loan data from Experian. This data is anonymous and contains records from over 230 million Americans. This amount of data was obviously too big to begin using without any cleaning or filtering. The data was spread across many different excel files and parquets in the Data Lakes storage containers. [redacted] In total, there were about 10,000 features across these datasets, spanning many categories like [redacted]

5. Software Requirements

Our sponsor, Micronotes, specifically defined what was in scope, what would be a nice addition, and what was out of scope. For any additional gaps, the sponsors were contacted directly to clarify the requirements.

5.1 Digital Prescreen Acquire Automation Requirements

The main requirements for the digital prescreen acquire automation project were as follows: a web interface for selecting the financial institution type and inputting a FDIC/NCUA number, mile radius, and email. Back end code to automate current processes, from data pull to sending out Excel files via email. Integration with the ADS database to pull the latest rates for all products and data needed.

5.1.1 Themes and Epics

Theme 1: Automate the Digital Prescreen Acquire Process

This theme was focused on the first project which included automating the current workflow: the ability to input if the financial institution is a bank or credit union, the FDIC/NCUA number, and an email address to start the process. This meant that eventually a representative of the financial institution could input the necessary information into a front end form without the need for interaction with an employee of Micronotes. The only part of the process that still has employee interaction is for approving the request for a report. This is just to ensure that only companies they are working with can receive a report.

Epics:

1. Finish and Link Back end Code

   The first Epic of the project focused on finishing up the back end code. The first portion was moving all the Zip Code Finder code into an Azure Databricks Notebook. This way all the code would be in one centralized place. This required modifying the existing code and splitting it into cells for easier understandability.

2. Calculate and Save Rates

   The second Epic of the project focused on doing additional rate calculations, rate pulls, updating settings, and saving all the values to a JSON. This involved writing code to pull data from CSV files, an API, and using dataframes to filter raw results. This tailors the rates specifically to the target demographic of the population near the financial institute.

3. Generate Readable Output

The third Epic focused on the main PDF deliverable that would be sent to the financial institute. Readable output was necessary in order to provide an easy breakdown of the possible loan amounts that a financial institute can offer to those who passed prescreening criteria. This file is the main point of the project since it is what is given to the financial institute in order to show how beneficial using the Micronotes service could be for them.

4. Create User Experience

   The last Epic focused on the front end, user-experience that the representative of the financial institute would interact with to make the initial request. This should include creating a prototype for the UI and the delivery system for the PDF.

5.1.2 User Stories

Based on the given criteria and additional details, the following user stories were created, which were eventually split into smaller sub-stories.

Table 5.1: Digital Prescreen Acquire User Stories

As a Micronotes employee, I want to input the users FDIC/NCUA number and a mile radius, and automatically get back the company branches and the zip codes surrounding the branches within the given mile radius, so that I do not have to manually input and retrieve the information.
As a Micronotes employee, I want to automatically pull necessary data from the ADS database based on the zip code retrieval, so that I do not have to manually pull the data which can create errors.
As a Micronotes employee, I want to retrieve the total number of customers in the Experian database within these zip codes automatically, so that I do not have to manually pull this information.
As a Micronotes employee, I want to pull 40,000 records and apply underwriting criteria to eliminate unfit records, usually leaving about 15-20K records, automatically, so that I do not have to manually pull the information.
As a Micronotes employee, I want to pull the data for the 15-20K records to use in the Digital Prescreen Engine, so that I do not have to manually pull the data.
As a Micronotes employee, I want to automatically put the loan rate and offer values into a JSON file that will be used in the engine.

As a Micronotes employee, I want to have a delivery PDF prototype for the final output of Prescreen Acquire.
As a Micronotes customer, I want to have a web page where I can submit a request for the Prescreen Acquire service and receive a digital report a few hours later.

5.2 ADS Data Matching Requirements

There were three main requirements for the ADS data matching project; the team decided to focus on the first segment, which became the whole project. The development of a matching algorithm aids in the creation of a time series dataset containing six months of data. With this dataset any type of machine learning models used on it would achieve better results when compared to the real-time data. These models would create propensity scores for each data point - person - to represent the likelihood that they would close on a loan. This allows for Micronotes and the financial institutions they work with to target customers as efficiently as possible.

5.2.1 Themes and Epics

Theme 1: Develop Matching Algorithm for Time Series Creation

This theme relates to the second project the team undertook,which involved creating the matching algorithm based on a nearest neighbor search.

Epics:

1. Manually create matches

   The first epic was to manually match users. This was to create a sample of matches to base the matching algorithm.

2. Creating the algorithm

   The last epic was creating the matching algorithm, that takes in two snapshots and matches users.

5.2.2 User Stories

Based on the above criteria and requirements, the following user stories were created to aid in the incremental completion of each task.

Table 5.2: ADS Data Matching User Stories

As a micronotes employee, I want to have attributes that I can trust won’t change over time to verify our matching algorithm.
As a Micronotes employee, I want to have approximately 200 manually matched data points as a reference point for the matching algorithm.
As a Micronotes employee, I want to be able to automatically select features for the algorithm
As a Micronotes employee, I want to be able to automatically match data points between two snapshots.
As a Micronotes employee, I want to have a time series dataset from these matched data points to have a better representation of real-time data.
As a Micronotes employee, I want to fit a machine learning model to this time series data set to help me predict who will close on certain types of loans.
As a Micronotes employee, I want to be able to create propensity scores for each of these people/data points.

6. Design

6.1 Digital Prescreen Acquire Design

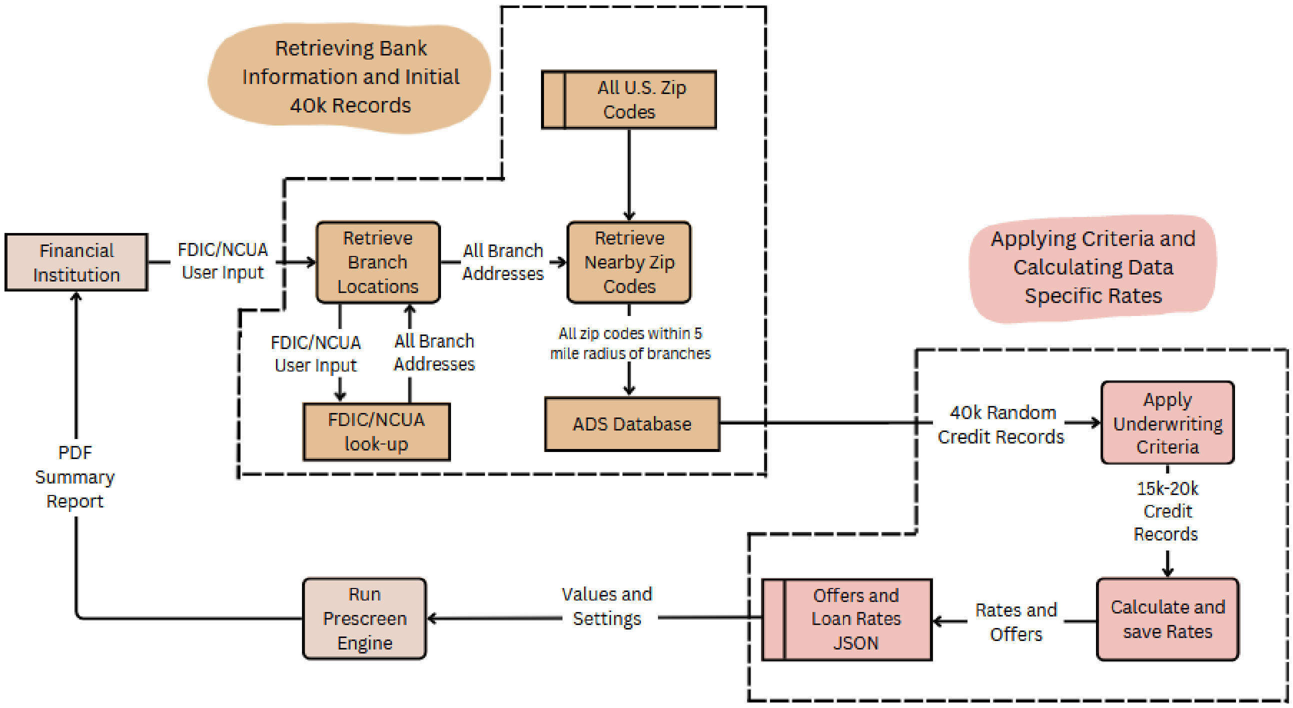

Retrieving Bank Information and Initial 40k Records
All U.S. Zip Codes
Financial Institution
FDIC/NCUA User Input
Retrieve Branch Locations
All Branch Addresses
Retrieve Nearby Zip Codes
Applying Criteria and Calculating Data Specific Rates
FDIC/NCUA User Input
All Branch Addresses
All zip codes within 5 mile radius of branches
FDIC/NCUA look-up
ADS Database
40k Random Credit Records
Apply Underwriting Criteria
15k-20k Credit Records
PDF Summary Report
Run Prescreen Engine
Values and Settings
Offers and Loan Rates JSON
Rates and Offers
Calculate and save Rates

Figure 6.1 Prescreen Acquire Data Flow Diagram

The entire process can be broken down into eight main steps. The first step is reading in the user input which is represented on the graph by the light pink “Financial Institution” outside-actor box. The second step is using this FDIC/NCUA number to query an external database to return all branch locations for the given institution. This is represented on the graph by the orange boxes labeled “Retrieve Branch Locations” which is an internal process, and “FDIC/NCUA look-up” which is an external database. The third step is to obtain all unique zip codes within a five mile radius of all the found branch locations. This is represented on the graph by the orange boxes labeled “All U.S. Zip Codes” which is a CSV file stored internally, and “Retrieve Nearby Zip Codes” which is an internal process. The fourth step is saving all the unique zip codes to a CSV file, which is represented on the graph by the data flow labeled “All zip codes within 5 mile radius of branches”. The fifth step is using the CSV file and retrieving 40,000 random credit records of residents within the zip codes, from the ADS database. This is represented on the graph by the orange box labeled “ADS Database” which is an external database, and the dataflow labeled “40k Random Credit Records”. The sixth step is to apply given underwriting criteria to the 40,000 records and return around 15,000-20,000 which fit the criteria. This is represented on the graph by the pink box labeled “Apply Underwriting Criteria” which is an internal process, and the dataflow labeled “15k-20k Credit Records”. The seventh step is to use the given records to calculate rates and offer limits. This is represented on the graph by the pink box labeled “Calculate and save Rates” which is an internal process, and the pink box labeled “Offers and Loan Rates JSON” which is an internal JSON file. The final step is to use the given records and the JSON file to run the engine, which outputs an output file and is sent to the original user. This is represented on the graph by the light pink box labeled “Run Prescreen Engine” which is an internal process, and the data flow labeled “PDF Summary Report” which links back to the original external-actor labeled “Financial Institution”.

6.2 ADS Data Matching

The flow of data in the data matching project started when Micronotes received Experian’s aggregated U.S. data containing records from approximately 230 million people that are uniquely identified by the “ads_mkt_consumer_id” column. The data was split into many different data sets so it was first filtered by the relevant features for matching. The number of records would then be limited to a single zip code before being matched with the algorithm. The algorithm would then output the matches based on Euclidean distance and create both a delta table in the Data Lakes storage container and a CSV file.

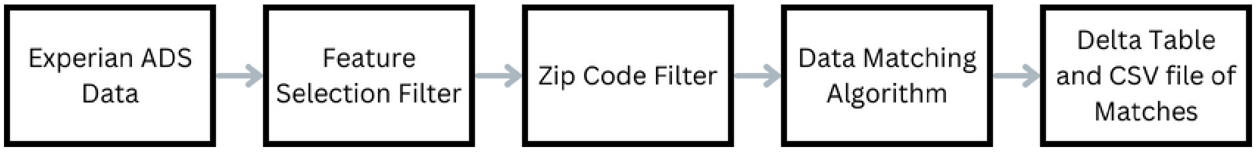


Figure 6.2 ADS Data Matching Data Flow Diagram

6.3 Entity Cluster

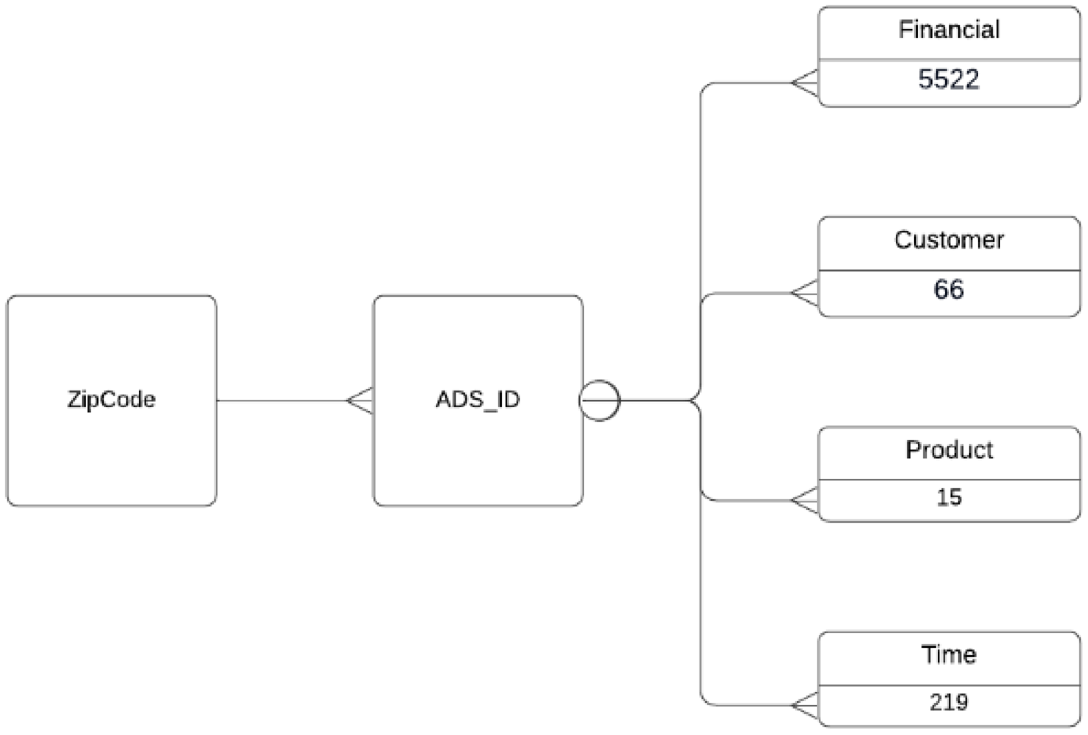


Figure 6.3 ADS Data Matching Entity Cluster

The entity cluster, as shown, has five main clusters. Every zip code has a mandatory one-to-many ads_mkt_consumer_ids. The ads_mkt_consumer_ids have an optional one-to-many relationship with the rest of the attribute clusters. The numbers under the cluster represent the number of attributes in each cluster. See Appendix A for a full breakdown.

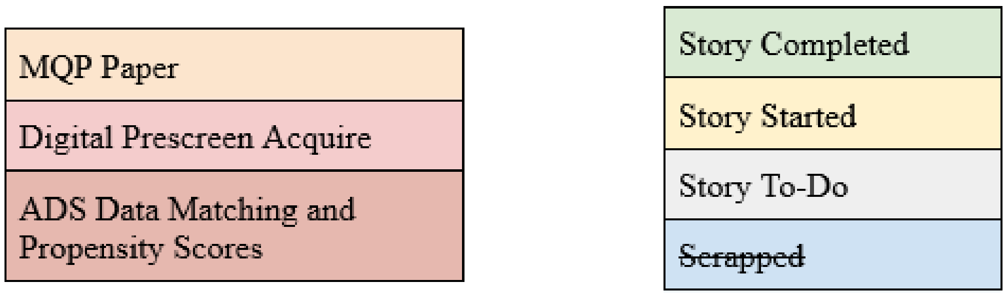
7. Software Development

7.1 Section Breakdown

This section discusses work done during the pre-qualifying project term, how the team used scrum, and all the sprints completed during the project term. The sprint section breaks down the six sprints the team completed. Each sprint lists the user stories that were included in the sprint, how many sprint points were completed, an overall review of the sprint, as well as what went well and where improvements could be made.

In order to easily understand which stories pertain to the MQP paper, Digital Prescreen Acquire, and ADS Data Matching and Propensity Scores, a color key was created. There is also a color key for the completion of each story. Both are shown in Figure 7.1. If the story was fully completed in the sprint, it is colored green. If the story was started during the sprint but not finished, it is colored green. If the story was planned to be done in the sprint, but was not started at all, it is colored gray. Lastly, if the story was in the sprint but plans changed and it was scrapped, it is colored blue and striked out.

Figure 7.1 Color Key



MQP Paper
Digital Prescreen Acquire
ADS Data Matching and Propensity Scores

Story Completed
Story Started
Story To-Do
~~Scrapped~~

7.2 Pre-qualifying Project Work

Before the beginning of the project term, background research about the company was done. An introductory meeting with one of the sponsors took place, but no details about the actual project were shared until the start of B-term, due to NDA's. The necessary background checks and NDA's for the team were completed before the term began so that the project could proceed with no delays. Without much detail on the project the team focused on learning about software tools that were expected to be used, Azure and Python.

7.3 Agile Scrum

To ensure that questions were answered in a timely manner, and to ensure that the project was staying on track, the sponsors set up daily meetings with the team. Each weekday morning at 9 AM a standup meeting took place to inform them what was completed on the previous day and what was planned to be done for the day. By the end of each week, a report was posted in the project documentation pages to help the sponsors keep track of what was accomplished. The last standup meeting of each week was used as a sprint review where the team discussed all the progress that was made, what was completed, and anything that would need to be continued the next week.

Outside of daily scrums and weekly reviews with the sponsors, the team would meet on Monday mornings to review the previous week and plan for the upcoming week. The scrum leader would prompt each member to discuss what they finished, what testing was done, any blockers or issues that needed additional help. The product owner would then quantify the total progress that was made and how beneficial it was to the company. This helped the team to understand the significance of the work that was done.

7.4 Sprints

7.4.1 Sprint 0

This sprint Table 7.1 shows that the team focused on writing the base of the paper and important pre-sprint tasks such as defining the project goal and creating stories based on it. The team also got familiar with the existing code and the software environment the company uses. All Story points were able to be completed during the sprint.

Table 7.1 Sprint 0 User Stories

Review Preexisting Code and Data For Prescreen Acquire.	2 points
As a WPI student, I want to write the Background on Company Technology section of the paper.	2 points
As a WPI student, I want to complete research on decision trees for section two of the paper.	2 points
As a WPI student, I want to write the project goal in story form.	2 points
As a WPI student, I want to complete the definition of done for the project.	2 points
As a WPI student, I want to complete the story map backbone.	2 points

As a WPI student, I want to complete the self-facilitated backlog workshop.	2 points
Story Points Completed	**12/12 Points**

Sprint Review:

This week the team received the details on the project, but then realized the scope of the project did not fully work with the team's skills. This concern was escalated to the advisors and sponsors so they could bridge the gaps of understanding. Because of this slight setback, the team worked on the report. More information was found for research, methodology, and software development environment sections. The team also spent some time looking at the preexisting code and understanding the current workflow they use.

What went well:

- Daily meetings with sponsors to keep us on track, and discuss any concerns.
- Not much onboarding, had to wait on access to azure and the VM's.
- The employees we are working with are very nice, understanding, and ready to help us.
- Able to escalate and express concerns when needed.

What we can improve on:

- Did not use Jira much because of the blocker to getting started.
- Need to set up our user stories together as we get more information.
- None of us have experience in Databricks Notebooks for coding, may have a bit of a learning curve.

7.4.2 Sprint 1

This sprint Table 7.2 shows that the team focused on both projects. For the Digital Prescreen Acquire project, progress was made on taking user input, pulling data, and obtaining qualifying records. One story was started but not completed and was carried over into the next sprint.

Table 7.2 Sprint 1 User Stories

As a micronotes employee, I want to input the users FDIC/NCUA number and a mile radius, and automatically get back the company branches and	2 points

the zip codes surrounding the branches within the given mile radius, so that I do not have to manually input and retrieve the information.	
As a micronotes employee, I want to automatically pull necessary data from the ADS database based on the zip code retrieval, so that I do not have to manually pull the data which can create errors.	2 points
As a micronotes employee, I want to retrieve the total number of customers in the Experian database within these zip codes automatically, so that I do not have to manually pull this information.	2 points
As a micronotes employee, I want to pull 40,000 records and apply underwriting criteria to eliminate unfit records, usually leaving about 15-20K records, automatically, so that I do not have to manually pull the information.	2 points
As a Micronotes employee, I want to have approximately 200 manually matched data points as a reference point for the matching algorithm.	2 points
As a Micronotes employee, I want to be able to automatically match data points between two snapshots.	3 points
Story Points Completed	**10/13 Points**

Sprint Review:

This week the team got rolling on the projects. The sponsors decided on two separate projects in order to cover the skills for all of the team's majors. The team then started to work on linking the code between the zip code finder and the intermediate file generator. The team also started to manually match ~200 anonymous user data records using fields that cannot change (e.g. initial loan amount). After the initial findings the team worked on writing the machine learning algorithm to match the anonymous user data records.

What went well:

- Completed a good amount of work on the projects.
- Learned more about the tools we were using.
- Got help from other employees easily.
- No major setbacks.

What we can improve on:

- Not using Jira to its fullest potential.
- Need to set up our user stories before we start working.
- Estimating points and how many user stories we can complete.

7.4.3 Sprint 2

This sprint Table 7.3 shows that the team made progress on all three themes. The story which was carried over from last sprint was not finished this sprint either, but progress was made. There was one other small story which was started this sprint but not finished.

Table 7.3 Sprint 2 User Stories

As a Micronotes employee, I want to automatically pull rates from FRED and FreddieMac to use in the engine.	2 points
As a Micronotes employee, I want to automatically calculate HELOC prime rate, Mortgage rate, Auto Loan Rates, Personal Loan Rates, HELOAN loan rates, HELOC loan rates, and the Mortgage MNP Minimum Offer Amount to use in the engine.	4 points
As a Micronotes employee, I want to automatically put the loan rate and offer values into a JSON file that will be used in the engine.	1 point
As a micronotes employee, I want to have attributes that I can trust will not change over time to verify our matching algorithm.	2 points
As a micronotes employee, I want to be able to automatically match data points between different snapshots to create a time series dataset.	3 points
As a WPI student, I want to have a background section written on all relevant machine learning techniques	3 points
Story Points Completed	**11/15 Points**

Story points completed: 11/15

Sprint Review:

This week the team made a large amount of progress in both projects. A small story had to be carried over since there was some clarification needed regarding the JSON file, which did not get answered until the following Monday. The back end pulling and storing of data was almost completed. After this is finished, the Prescreen Engine will be ready for a full run through of the newly automated process. For the second project, the team completed a version of the matching algorithm that will be iterated upon and improved in the coming week. As is, someone can match a decent number of datapoints, however it is limited by the manual selection of features. Many data points only have default values for these features and are treated as matches. Research sections on machine learning techniques were up to date at this point and progress was made on chapter nine.

What went well:

- Completed more story points than last week
- Only 4 points of carry-over
- Comfortable with the language and environment we are working in

What we can improve on:

- Remembering to prioritize the paper in order to not fall behind

7.4.4 Sprint 3

This sprint Table 7.4 shows that the team made progress in all main themes. The small story that was carried over from the last sprint was completed this sprint. The story which was carried over the last two sprints was not completed this sprint, but was resized as challenges surfaced. One story that was expected to be completed this sprint was not started and carried over to the next sprint.

Table 7.4 Sprint 3 User Stories

As a Micronotes employee, I want to have documentation on the Prescreen Acquire code to know how to start the process.	2 Points
As a Micronotes employee, I want to automatically put the loan rate and offer values into a JSON file that will be used in the engine.	1 Point
As a Micronotes employee, I want to have a delivery PDF prototype for the final output of Prescreen Acquire.	2 Points
As a WPI student, I want to have more information on similar prescreen technology.	1 Point
As a WPI student, I want to have a detailed list of tables and figures at the beginning of our paper.	1 Point
As a WPI student, I want to have a more detailed company background section.	1 Point
As a Micronotes employee, I want to have a matching algorithm that can	5 Points

match anonymous data points between two different snapshots a week apart.	
Story Points Completed	**6/13 Points**

This week both projects started to lose pace. The back end of the Prescreen Acquire project was fully finished, however feedback from our sponsors was needed to ensure the output aligned with what was expected. There were a few errors that needed to be fixed and were easily solved. However, this project lost pace a bit due to the time waiting for feedback in order to make more progress. Feedback took longer than expected – typically the next day – so more focus went into the paper during the downtime. Because of this delay in feedback, the next story that the team was hoping to start and finish that week had to be carried over. The ADS data matching project hit a large bump. The approaches that were taken did not work, this caused the team to reassess and strategize on how to approach the problem differently. The problem was much harder than originally thought, and in turn needed to be broken down into smaller increments. In the meantime, some progress was made on small parts of the paper.

What went well:

- Finished the MQP paper draft
- Completed second round CATME survey

What we can improve on:

- Pushing a bit harder to get more timely feedback due to our short timeline
- Reaching out sooner when things are looking more difficult
- Consistently follow up to gather feedback and enable continuous improvement

7.4.5 Sprint 4

This sprint Table 7.5 shows the team made progress in all aspects. For the Digital Prescreen Acquire project, the story carried over from the last sprint was scrapped and replaced as the priorities of the sponsors changed. The other story which was carried over was completed this sprint. One story was started and carried over to the next sprint.

Table 7.5 Sprint 4 User Stories

As a Micronotes employee, I want Prescreen Acquire results to be in a readable excel file, to give a breakdown to the customer.	3 Points
~~As a Micronotes employee, I want to have a delivery PDF prototype for~~	~~2 Points~~

~~the final output of Prescreen Acquire.~~	
As a Micronotes employee I want a proof of concept for the front end of the Digital Prescreen Acquire process, so that eventually non-engineer employees can run the process.	3 Points.
As a WPI student, I want to complete the abstract, executive summary, and introduction paragraph portions of our paper.	2 Points
As a WPI student, I want to complete the assessment, Future work, and conclusion sections of our paper.	3 Points
As a Micronotes employee, I want to be able to drop rows from the dataset that cause duplicate matches.	2 Points
As a Micronotes employee, I want to have features that contain enough information to be useful for matching.	3 Points
As a Micronotes employee, I want to have a concrete subset of matches that I can be confident in.	4 Points
Story Points Completed	**17/20 Points**

This sprint was slightly longer than normal since we decided to combine the two day work week due to Thanksgiving break, with the following full work week. Originally, the sponsors wanted the team to work on the delivery PDF prototype for the Digital Prescreen, however after further discussion, they decided it would be better to focus on the front end prototype. This was due to the realization that there is no way for non-engineers to start the progress, and there is already a readable output file showing the results. This meant the team had to switch gears after the decision was made and so one user story was scrapped and replaced with another that had to be carried over. As for the data matching project, significant progress was made. Further data preprocessing was done which helped eliminate the issue of duplicated -or more- matches being made by the algorithm. Then features were chosen based on a slightly different criteria -large standard deviation relative to mean and range values- which ended in much better results with the matching algorithm. The algorithm was able to match roughly 16,000/35,000 data points using a fairly small subset of features.

What went well:

- Achieved concrete results in the matching project which was at a dead end for the past few weeks
- Was able to pivot strategies quickly on the matching project thanks to Agile and frequent meetings

- Able to regroup and push forward when plans changed on what the sponsors wanted

What we can improve on:

- Allocating time more efficiently between the project and the paper

7.4.6 Sprint 5

This sprint Table 7.6 shows all the remaining items the team completed to wrap up the projects.

Table 7.6 Sprint 5 User Stories

As a Micronotes employee I want a proof of concept for the front end of the Digital Prescreen Acquire process, so that eventually non-engineer employees can run the process.	3 Points
As a Micronotes employee I want a fully functioning web app where I can submit a Digital Prescreen Acquire request and receive an Excel output report by email a few hours later.	5 Points
As a WPI student, I want to make final edits to our paper.	5 Points
As a Micronotes employee, I want to increase the number of records that can be matched with the algorithm to over 50% of the data set.	4 Points
Story Points Completed	**17/17 Points**

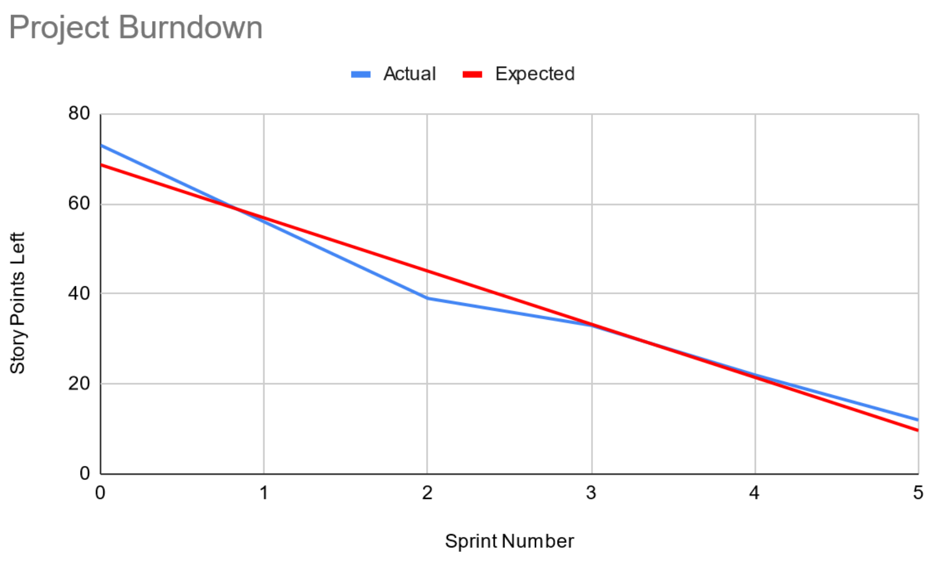


Figure 7.1 Project Burndown Graph

8. ADS Data Matching Results

Using features from one of the many datasets at a time was successful in matching a subset of users. However, because of the nature of financial data, not every data point will have usable information for each feature chosen for matching. Most of the work for this project therefore revolved around feature selection.

8.1 Manual Matching

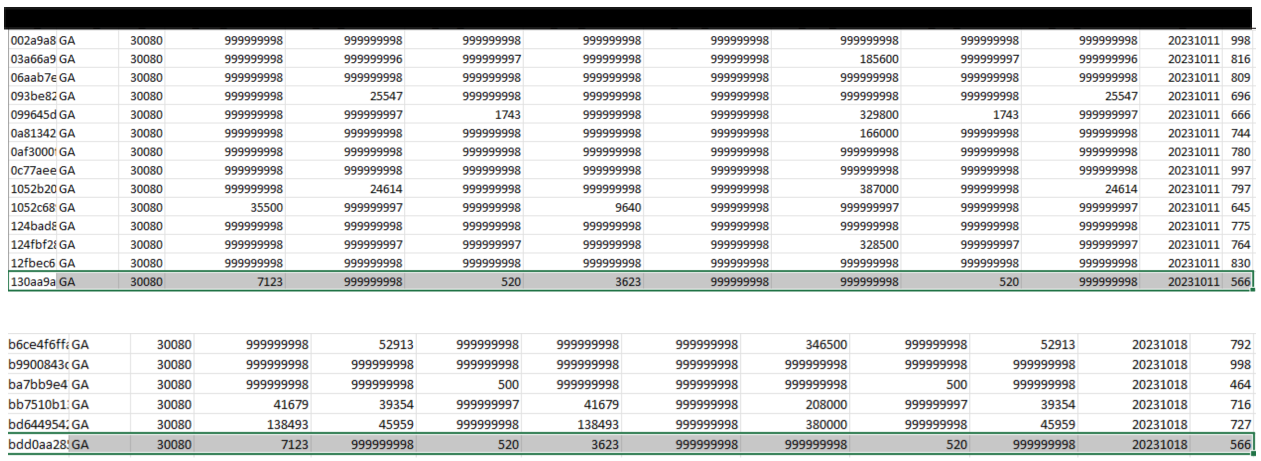
Each week, Micronotes receives anonymized data from Experian. This helped to define the first goal of this project: develop an algorithm to match data points between snapshots based on selected criteria. The criteria that was selected had to be relatively constant over time in order to verify that the data points were in fact the same people. The easiest and most obvious choice for this criteria was [redacted]. Information for [redacted] was used: [redacted]. There were also four different kinds of default values for [redacted]. Using [redacted] helped differentiate between people with the same value [redacted] as each other. For example, if two people [redacted] [redacted], it might be hard to tell the difference between them in the data; however, it would be highly unlikely for those same people to have a [redacted] at the same price. Other criteria, like [redacted], that change by a predictably low amount week-to-week helped to make these distinctions. Table 8.1 contains the [redacted] used for manual matching and their descriptions.

Table 8.1 Data Matching Criteria with Relevant Descriptions

Feature	Description
[redacted]	[redacted]
[redacted]	[redacted]
[redacted]	[redacted]
[redacted]	[redacted]
[redacted]	[redacted]

	[redacted]
[redacted]	[redacted]
[redacted]	[redacted]

Before the algorithm could be developed, manual matching of about two hundred records was required to have a basepoint to verify users. It was also useful to gain some intuition about the data considering its large volume. The following screenshots are an example of two matching rows between the data they received on October 11th and October 18th.



[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
002a9a8	GA	30080	999999998	999999998	999999998	999999998	999999998	999999998	999999998	999999998	20231011	998
03a66a9	GA	30080	999999998	999999996	999999997	999999998	999999998	185600	999999997	999999996	20231011	816
06aab7e	GA	30080	999999998	999999998	999999998	999999998	999999998	999999998	999999998	999999998	20231011	809
093be82	GA	30080	999999998	25547	999999998	999999998	999999998	999999998	999999998	25547	20231011	696
099645d	GA	30080	999999998	999999997	1743	999999998	999999998	329800	1743	999999997	20231011	666
0a81342	GA	30080	999999998	999999998	999999998	999999998	999999998	166000	999999998	999999998	20231011	744
0af3000	GA	30080	999999998	999999998	999999998	999999998	999999998	999999998	999999998	999999998	20231011	780
0c77aee	GA	30080	999999998	999999998	999999998	999999998	999999998	999999998	999999998	999999998	20231011	997
1052b20	GA	30080	999999998	24614	999999998	999999998	999999998	387000	999999998	24614	20231011	797
1052c68	GA	30080	35500	999999997	999999998	9640	999999998	999999997	999999998	999999997	20231011	645
124bad8	GA	30080	999999998	999999998	999999998	999999998	999999998	999999998	999999998	999999998	20231011	775
124fbf28	GA	30080	999999998	999999997	999999997	999999998	999999998	328500	999999997	999999997	20231011	764
12fbec6	GA	30080	999999998	999999998	999999998	999999998	999999998	999999998	999999998	999999998	20231011	830
130aa9a	GA	30080	7123	999999998	520	3623	999999998	999999998	520	999999998	20231011	566

b6ce4f6ff	GA	30080	999999998	52913	999999998	999999998	999999998	346500	999999998	52913	20231018	792
b9900843	GA	30080	999999998	999999998	999999998	999999998	999999998	999999998	999999998	999999998	20231018	998
ba7bb9e4	GA	30080	999999998	999999998	500	999999998	999999998	999999998	500	999999998	20231018	464
bb7510b1	GA	30080	41679	39354	999999997	41679	999999998	208000	999999997	39354	20231018	716
bd644954	GA	30080	138493	45959	999999998	138493	999999998	380000	999999998	45959	20231018	727
bdd0aa28	GA	30080	7123	999999998	520	3623	999999998	999999998	520	999999998	20231018	566

Figure 8.1 Manual Matches

As shown above, the values for each [redacted] are identical, including the default values for [redacted] this person does not possess. In this case the values are specific, like 7,123 and 3,623, which made the process of manually matching these much easier than if someone had [redacted]. However, even in that case, the person could still be identified using the default values for [redacted] and a relative neighborhood for their [redacted].

8.2 Data Cleaning

To help with the burden of working with so much data, the initial sample size used was about 40,000 people from one zip code. This was done with a function that filtered through the large dataset(s) and only pulled records from the specified zip code. This was only used at first when matching pairs manually; once the creation of the algorithm started, the .select() feature in

Pyspark was used instead. One of the biggest problems with the data was the amount of null values present. This was more prevalent while trying to combine the different datasets into a single dataframe to analyze. This makes intuitive sense, since financial data is not going to apply to every person for every kind of measure there is. To help with this, general information that would apply to most people were pulled from multiple datasets, like [redacted]. From this point, one dataset would be selected to pull primary features from. The types of categories these datasets fell under include [redacted]. Each dataset had a unique identifier for each data point called "ads_mkt_consumer_key," which changed from week to week. This allowed for the columns from other datasets to be joined on the correct rows.

The problem with the data was duplicate rows. In any given zip code there would be people without credit scores or income and in the data those people were identical. Since they would also be unlikely to be Micronotes clients, dropping them from the data set using .dropDuplicates() was the obvious choice. This brought the number of records down to about 35,000 and took care of the problem of duplicate matches.

8.3 Feature Selection

The feature selection logic was initially similar to how features were chosen for manual comparison. The standard deviation was found for the same features between datasets; the features with a standard deviation of zero were selected as they would not change over time. Choosing all of the unchanging features seemed like the most logical choice to differentiate between anonymous data points because there are upwards of ten thousand features in the datasets. The probability that two people have the exact same values for each of these features is incredibly low. However, there were multiple reasons that the logic for feature selection was changed.

The first problem encountered with this method of feature selection, was that the reason many features had a low standard deviation between snapshots was because they contained mostly default values. The features chosen this way were very ineffective at matching. Since there were such a large number of features, implementing machine learning in the feature selection process was taken into consideration. The problem with this approach was that it requires a substantial amount of training data to already be matched. The method settled on was a slight variation on the initial standard deviation criteria. The idea was that any given feature would not change significantly,or at all, in the span of a single week. So instead of choosing features based on small standard deviations between snapshots, features were chosen based on a high standard deviation within a single snapshot relative to their mean and range of values. This would indicate that the selected features have a large number of valid inputs, meaning that these features result in the highest number of rows containing useful information. The specific features chosen were from the dataset labeled [redacted] and the [redacted] feature came from [redacted]. The various features selected were [redacted]

[redacted].

8.4 Matching Algorithm

The portion of the code designed for the matching is compact and straightforward. The method discussed in Chapter 2.8.1, locality-sensitive hashing (LSH), was used for the nearest neighbor search to sort similar rows into buckets before matching. This was instrumental in lowering the computational cost that comes with working with a large volume of data. Having sorted the rows into buckets allows for more efficient matching since not all data points need to be compared. The matching itself was then done with the Pyspark function .approxSimilarityjoin() which takes four inputs: two datasets, a threshold, and the output column which tracks the Euclidean distance between rows. With this method, and only a small subset of features, approximately 17,000 matches were made of the roughly 35,000 records. Below is an example of two data points that are matched where the second consumer key was cut off for ease of readability.

```
"ads_mkt_consumer_key": "083cac400e9da28413b0caae7ccb10bcbdfc1fbe5fa617152e08874a9d5e7c7c",        "ads_mkt_consumer_key": "3acced
```

[redacted]

Figure 8.2 Data Match Example

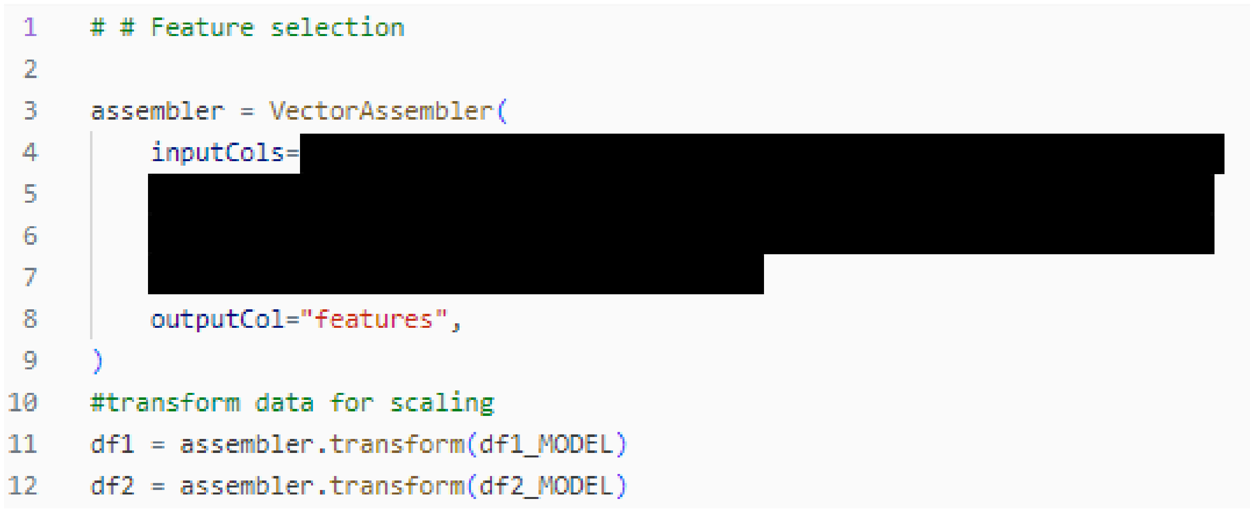
The left hand side represents this person's information on October 11th, 2023 and the right hand side represents the same information on October 18th, 2023. Each time different features were used in the matching algorithm, the resulting data frame would be saved as a delta table in the Delta Lakes storage container. This allowed for easy comparison of the effectiveness of specific features.

The largest improvements made in terms of having a concrete number of records matched came from not considering the entire dataset. Not every person from a zip code was going to be a suitable candidate for loans, so figuring out a way to drop them from the data set had become the main task. For the most part, these were people without credit scores, income, and other basic financial information that applied to most people. These were not represented by null values in the data, but by error codes. This caused many duplicate rows for people with missing information, making it hard to distinguish genuine matches from those who just lacked significant data. The issue of duplicate rows was also an issue for those with a large amount of information; however, the probability of this happening was very small considering the number of features and the wide range of values they took. From this point, the idea of finding features that do not change over time became a possibility again. The problem before was that, for example, people without [redacted] would show up as duplicates in the data which causes issues for the matching algorithm.

The [redacted] from Table 8.1 were used as stable features combined with several different income models and resulted in the largest number of matched data points. The purpose of the income models was to help eliminate duplicate rows without the need to drop them from the dataset. [redacted] was chosen as a feature because, like [redacted], it was likely not to change over a short span of time. After filtering the data based on these features, approximately 33,000 of the original 38,000 records remained and 22,000 records were matched.

8.4.1 Matching Algorithm Code

Once the data had been sufficiently cleaned and preprocessed, the features being used were input to the VectorAssembler() Pyspark function so that they would be a valid input for the models chosen. The data was then transformed to have the data type “ArrayType” so it could, again, be input to the various models being applied. In this following example, the features shown were used to match 17,000 records.

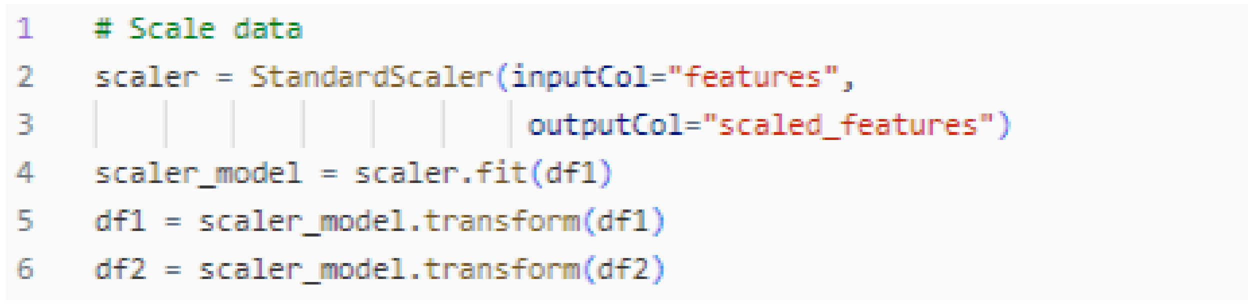


```
# # Feature selection

assembler = VectorAssembler(
    inputCols=[redacted]
    outputCol="features",
)
#transform data for scaling
df1 = assembler.transform(df1_MODEL)
df2 = assembler.transform(df2_MODEL)
```

Figure 8.3 Feature Selection Code

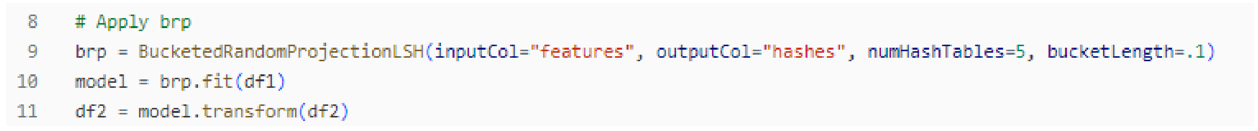
Once the data had been transformed, it would be normalized with StandardScaler() and then transformed again, since the various features have different ranges of valid values. For example, if one feature had mostly values in the hundreds of thousands and another feature had values ranging from zero to one thousand, then the first feature would dominate in importance for the distance calculation. However, in the most successful matches only exact matches were considered, so this would not play a role. When considering approximate matches it is more important to scale the data.



```python
# Scale data
scaler = StandardScaler(inputCol="features",
                        outputCol="scaled_features")
scaler_model = scaler.fit(df1)
df1 = scaler_model.transform(df1)
df2 = scaler_model.transform(df2)
```

Figure 8.4 Data Scaling Code

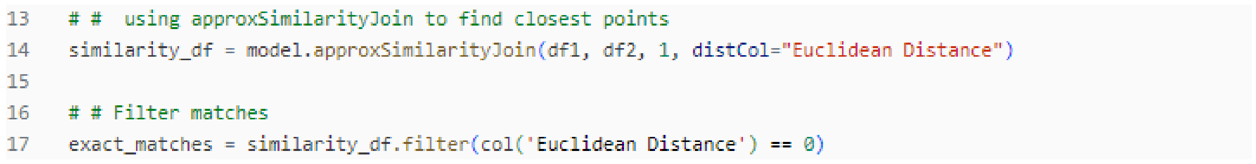
At this point the data would be ready for LSH as discussed in chapter 2.8.1. The Pyspark function used takes four different arguments: the input and output columns, the number of hash tables, and the bucket length. Here, increasing the number of hash tables decreases the number of false negatives and decreasing it improves performance and lower bucket lengths increase precision while higher ones increase performance. These are analogous to the parameters L and k discussed in chapter 2.8.1.



```python
# Apply brp
brp = BucketedRandomProjectionLSH(inputCol="features", outputCol="hashes", numHashTables=5, bucketLength=.1)
model = brp.fit(df1)
df2 = model.transform(df2)
```

Figure 8.5 LSH Code

Depending on the specific use-case, “features” or “scaled_features” can be used. This depends on whether an exact or approximate match is the goal. Once the model was fit to the data frames, everything was ready for the nearest neighbor search. This was done with the previously mentioned approxSimilarityjoin() Pyspark function that takes two data frames and a threshold as arguments and appends a distance column to the data frame it outputs.



```python
# #  using approxSimilarityJoin to find closest points
similarity_df = model.approxSimilarityJoin(df1, df2, 1, distCol="Euclidean Distance")

# # Filter matches
exact_matches = similarity_df.filter(col('Euclidean Distance') == 0)
```

Figure 8.6 Distance Computation Code

A small threshold would be set and depending on if an exact or approximate match was the goal, and a filter would be applied to limit the resulting rows of the data frame. In this case the Euclidean distance is computed for each data point and compared to other data points within the same bucket. Points with a distance of zero between them were matched. Depending on the number of features, the code would take approximately five to ten minutes to run, not including the time it would take to display the information as a table in Databricks.

9. Digital Prescreen Acquire Discussion

This project involved moving local code into the back end, finishing up code, tying the back end together, extrapolating results, and building the front end. This section discusses how all of this was implemented.

9.1 Zip Code Finder Automation

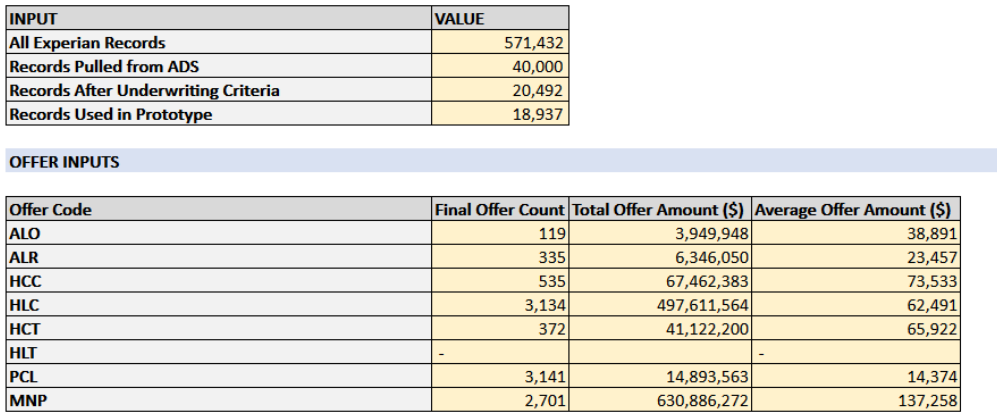
The first step in the process is finding all nearby zip codes within five miles of the given financial institution; this is applied to each branch location. Originally, this code was being run on a local machine by an employee, this code was moved into Databricks Notebook where all of the other back end code was located. This helped to centralize the location of all the processes code. This code takes in three fields: the financial institution type (bank or credit union), the corresponding FDIC or NCUA number, and the email address to send the output file to. When run, this code collects all locations of the given financial institution, and uses a Google Maps API and document containing all US zip codes in order to find all zip codes within five miles based on latitude and longitude. The output for this code consists of a file of all unique zip codes within the five mile radius. This information is then sent to the intermediate file generator code. The process then finds 40,000 random credit records located in the zip codes and outputs 15,000-20,000 records that fit underwriting criteria. Those records are then sent to the grab rates and update rates process.

9.2 Rate Finder and Calculation Automation

This process calculates rates that are specific to the 15,000-20,000 records retrieved. There are also two rates which are standard set rates: the first being the WSJ Prime Rate, the second being the latest 30 year mortgage rate from Freddie Mac. The team wrote Python scripts using Pandas and Pyspark to collect these rates, and calculate the other specific rates based on the data. The calculated values are then saved into a JSON rate sheet template. The process is then handed off to the Prescreen Engine.

9.3 Final Offer Calculations

After the process passes through the engine, the output is sent to the final offer calculation code. Here all the offers are totaled up and the median is taken. This is done through Python script using Pyspark and Openpyxl. All offer sums are taken in a specific order based on criteria given by the company where all loan offers have a priority order. All of the calculated averages are then saved to a readable excel file which can be presented to the financial institution.



INPUT	VALUE
All Experian Records	571,432
Records Pulled from ADS	40,000
Records After Underwriting Criteria	20,492
Records Used in Prototype	18,937

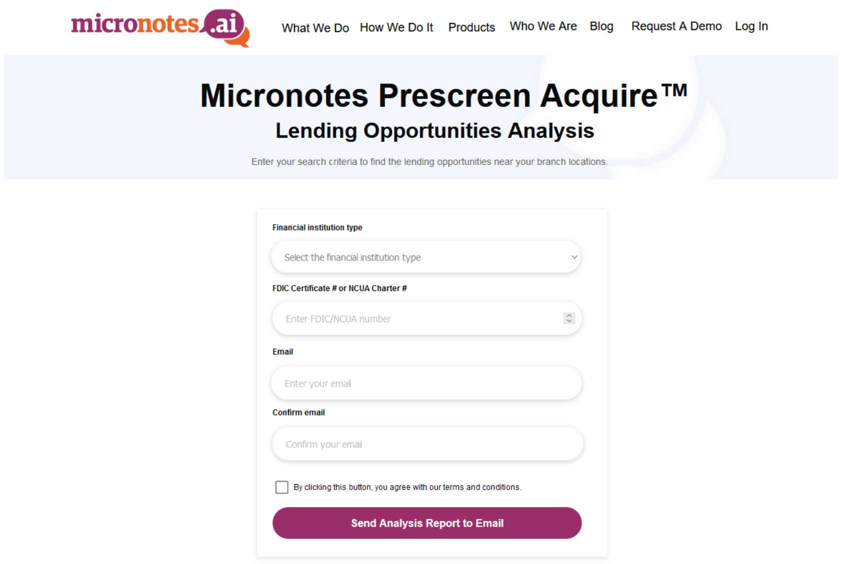
OFFER INPUTS

Offer Code	Final Offer Count	Total Offer Amount ($)	Average Offer Amount ($)
ALO	119	3,949,948	38,891
ALR	335	6,346,050	23,457
HCC	535	67,462,383	73,533
HLC	3,134	497,611,564	62,491
HCT	372	41,122,200	65,922
HLT	-		-
PCL	3,141	14,893,563	14,374
MNP	2,701	630,886,272	137,258

Figure 9.1 Digital Prescreen Acquire Excel Output

9.4 Digital Prescreen Acquire Front End

The last portion of the project completed was the front end interface. This was prioritized since non-engineering employees who submit the requests on behalf of financial institutions cannot run the code in the back end. They would have to submit a request ticket and have an engineer enter the required fields and run the process in Databricks. By having a front end connected to the back end, non-engineers can easily enter in the information and submit a request without going to Databricks. To create the front end, the company had a UI mockup on UIzard. This site allows the mockup to be generated into HTML code. This code was used as a base for the UI. All of the generated code was simple style HTML, so the code was revised to have usable text fields.



Micronotes Prescreen Acquire™

Lending Opportunities Analysis

Enter your search criteria to find the lending opportunities near your branch locations.

Financial institution type

Select the financial institution type

FDIC Certificate # or NCUA Charter #

Enter FDIC/NCUA number

Email

Enter your email

Confirm email

Confirm your email

☐ By clicking this button, you agree with our terms and conditions.

Figure 9.2 Digital Prescreen Acquire Front End

9.5 Pipeline Connections

In order to connect the front end to the back end, the team worked closely with an employee in order to request the necessary resources and properly make the connections. All of the front end code was moved from a local device to an Azure App Service to host the app on a private link. The app sends a stringified JSON of the parameters submitted by the user to an azure logic app. This logic app then parses the JSON and feeds the parameters into the first process labeled "ZipCodeFinder" in Figure 9.3, which is a process discussed earlier. This pipeline connects all the processes used and also sends the final email with the output file which is shown as "Send Analysis" in Figure 9.3.

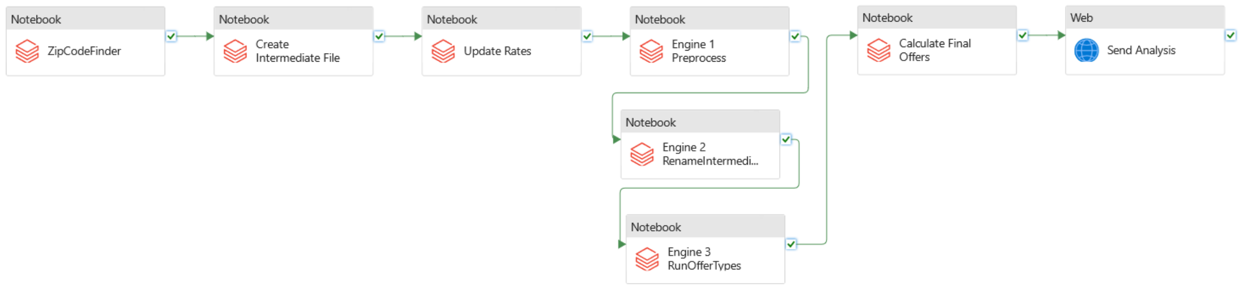


Figure 9.3 Digital Prescreen Acquire Pipeline

9.6 Digital Prescreen Acquire Conclusion

During the creation of the process automation, a few things were switched around priority wise, however the overall project stayed true to the initial vision. Originally the front end was planned to be integrated into the company's existing webpage. This changed to be a private, separate Azure web page that only those with the link can access. The other change made was keeping the output file simple. Instead of focusing on having a detailed and in-depth PDF output, the company put more focus on the front end in order to have the process be completely automated. That said, the excel file produced still gives a broad overview of the current market breakdown and the possible total offers that could be given.

10. Assessment

10.1 Digital Prescreen Acquire

The original goal for this project was to write code to automate the previously semi-manual process of the Digital Prescreen Acquire. This included completing back end code, creating documentation for easy upkeep, and creating the web interface to make the process usable by non-engineering employees.

In the end the main portion of the original goal was completed. Getting started was a bit slow with getting access to the databases and necessary code notebooks. There was also a learning curve with using Databricks notebooks and learning how to use different Python libraries and packages. Thankfully online resources expedited the learning process for Python and Databricks Notebooks. This project also provided experience in using Pandas, PySpark, API's, and large datasets.

In terms of non-technical learnings, this project generally provided real-life work experience. The team attended daily stand-ups with the sponsors which provided a space to give important updates, information on any blockers, and ask questions. Since the company only has 20 employees, this presented the opportunity to interact with a large portion of the company employees. There was also plenty of experience with getting feedback and using the iterative process where progress was presented and feedback was received on things to change or additional value parameters that were not originally noted.

10.2 ADS Data Matching

The original goal for this project was to develop an algorithm to match anonymous data points from two datasets spanning one week apart. The algorithm would then be used to create a time series data set containing five or six months of data. The company was not able to do this prior because the data they receive from Experian is anonymized before they receive it.. The time series data set would allow the team to create models for a much more realistic depiction of the data. The next step for Micronotes would be to assign propensity scores to each person as to how likely they would be to close on a certain type of loan.

The first part of the project was to manually find features that would help distinguish data points between the different snapshots. The purpose was to have a subset of matches that the team could be confident in, and use as a reference point for when the matching algorithm was finished. The main bump in the project is that the development of this matching algorithm was a much harder task than initially thought. Not having worked with such a large volume of data before, nearly 10,000 features spread across many different datasets and a third of the country's population worth of data points, was one of the main challenges the team faced.

The first major hurdle of the project was the matching algorithm. The learning curve of working with such a large amount of data, as well as getting used to their systems and databricks, made the task much more challenging than expected. For multiple sprints, the team was stuck on getting any form of reliable matches from the algorithm. Problems included debugging to get any results in the first place, and returning to data cleaning when the number of matches exceeded the number of records in the data set.

Once the team was able to reevaluate the approach to the problem, progress was made at a more consistent pace. Having achieved about 22,000 matches out of approximately 35,000 records leaves Micronotes with a concrete foundation to continue the work on this project. The other main success was the opportunity to work with such a large volume of data, which was incomparable to anything the team had worked with before. This coupled with the daily interactions with Micronotes' employees gave the team a chance to experience daily remote work life while solving a challenging problem.

10.3 Individual Learnings

Throughout our MQP, we all engaged in diverse learning experiences. This section is a culmination of our learnings. Each team member contributed unique insights and skills to the project, and this section encapsulates the culmination of those efforts.

10.3.1 Mirandi's Learnings

Over the course of the term, I gained extensive knowledge in software technologies which I had not previously worked with. The first major learning was using Azure. I had dealt with big data before, but not to the scale which Micronotes handles. I used my existing knowledge of filtering data, making queries, and reporting the output and applied it to the project. Although the syntax of Python and the libraries was different from what I had done previously, the same concepts still applied. I learned how to link the code from different notebooks together, and the coding standards that the company uses.

There were also a plethora of non-technical learnings. The first major learning was with communication and clarification. Even when tasks are very descriptive and laid out, there may still be some questions that were not originally thought of and planned for. When that would happen, sending questions right away was the highest priority. This also taught me about how other people have busy schedules and may not be able to fully answer until the next day. I would then look for other tasks to do which are not blocked by my questions in order to stay productive.

10.3.2 Griffin's Learnings

My two largest technical learnings from the project were my general coding skills and working with big data. Prior to this project, I had had some experience with Python; however, most of my experience with algorithms was in MATLAB. Expanding my comfortability with another language, along with going through the process of learning something on the job was very valuable. As for working with big data, the jump from sophomore level data science projects to working with the financial data for a third of the country was a steep one. However, because of these challenges I am satisfied with everything I have learned and am happy with the results.

Arguably the most important thing I learned from this project was to reach out and ask for help. This has always been something I have struggled with, whether in classes or in general. There was a point in the project where everything felt too big and impossible to complete, but the stark difference in weight on my shoulders after going to Professor Blais for help made it very hard not to learn that lesson. A majority of the tangible progress that will be useful to Micronotes was made in the week following that meeting.

10.3.3 Chase's Learnings

My primary focus was dealing with big data and improving my coding skills. The challenges posed by working with vast amounts of data pushed the boundaries of my abilities and knowledge. While I had some prior familiarity with Azure and data robots, the process of creating algorithms from the ground up provided a distinctive learning experience.

My professional development and my technical growth coincided with my interpersonal experiences. From the outset, I gained insights into a company's inner workings. This encompassed understanding the intricacies of documentation reporting and delving into the behind-the-scenes processes that drive successful outcomes.

10.4 Team Learnings

Although our team was split working on two separate projects for the majority of the term, we had many shared learnings. Due to the small employee size of Micronotes, we as a team got to meet and interact with the majority of the engineers and executives. This gave us an inside look at the importance of the work we were doing. Everyone we interacted with was friendly and happy to answer any questions we had. This helped us to bring up any concerns quickly which preserved the limited time we had.

Our major experiences improved our coding, working within the Azure environment, constant and daily communication with our sponsors, interim communication, weekly communication with our advisors, and software development life cycle skills. These were all shared learning that allowed the team to develop ourselves.

11. Business and Risk Management

Upon the conclusion of these projects, Micronotes will experience an efficiency and overall profitability enhancement. The goal is to strategically automate labor-intensive processes, resulting in various benefits, including time savings for employees and the automation of manual tasks. This enables employees to focus on activities that align more significantly to the company's objectives.

11.1 Innovation Risks

Regarding innovation risk, our project played a pivotal role in Micronotes' growth strategy, enhancing efficiency by freeing up significant hours of workforce and labor, thereby boosting profitability. Inadequate completion of these projects could have impeded Micronotes' growth potential as a company, particularly in automating the ADS Propensity score. Successful automation not only liberated specialized labor hours but enabled the simultaneous creation of more propensity scores, allowing the company to address multiple clients concurrently while ensuring ads matched the automation. Failure in this endeavor would have hindered the company's ability to reduce marketing costs for both Micronotes and their clients, subsequently limiting the improvement of conversion rates from offer to closed loans.

11.2 Operational Risks

Executing the automation projects had the potential to positively impact Micronotes, enhancing operational efficiency, customer satisfaction, and overall business performance. However, the benefits came with notable operational risks. The reliability of machine learning models, particularly in the data matching project, posed ongoing risks to decision-making quality. Scalability issues could arise as data volumes increased, potentially impacting performance. Feature selection complexities and potential documentation gaps presented persistent risks that needed vigilant attention. Managing these risks effectively was crucial to ensuring the continued success of those projects.

11.3 Reputational Risks

The inherent reputational risk lay in completing the assigned projects to an unsatisfactory level, impacting various aspects such as our team's reputation and our school's standing. Failing to deliver proficient and practical results could have impeded the school's capacity to collaborate on future projects with the organization, affecting our reputation and potential partnerships with other entities affiliated with Micronauts. Additionally, Micronotes might have suffered reputational damage if inefficiencies persisted, potentially harming client relationships and hindering their overall capabilities. These risks were all linked to the prospect of incomplete or unsatisfactory project outcomes.

11.4 Risk Management

Our risk control strategy included scheduling and timelines. Implementing a tight schedule allowed the team time to identify and tackle problems as they arose proactively. This structured timeline served as a preventive measure and allowed for a comprehensive risk assessment at various stages of the project. By adhering to a well-defined schedule, the team established a systematic approach to risk management, enabling the team to address challenges promptly and make informed decisions that significantly contributed to the project's overall success.

11.5 Industry Learnings

The team gained valuable insights and experiences in various aspects of business and project management. One notable growth area was leadership skills; each team member learned how to effectively lead and manage their time, fostering a collaborative team environment that proved essential to the team's success.

11.6 Business Learnings

Regarding business acumen, the team's deep dive into market dynamics, particularly customer retention in the banking sector, played a crucial role in the project's success. Understanding the industry empowered the team to make informed decisions. Individual learning revolved around software technologies, big data, and communication skills. The team adapted to new coding languages, handled vast data scales, and recognized the value of seeking help when needed. These skills directly contributed to the project's accomplishments, aligning with the industry's demand for tech proficiency and effective communication.

11.7 Culture Learnings

The MQP journey made the team keenly aware of the intricacies of smaller companies' hierarchical structures. The team discovered how these organizations maintain focus and coordination to achieve their objectives by navigating the various levels and roles. This knowledge proved invaluable, shedding light on the interconnectedness of different components and how they harmonize to drive success.

A significant learning curve involved saying "no" when necessary. Recognizing that diverse perspectives and ideas are inherent to any project, the team improved their skills of discernment, understanding when to embrace innovation, and streamlining processes for prompt project completion. This ability to make decisive choices contributed to the team's overall efficiency and success of the endeavors.

11.8 Remote Work Learnings

Additionally, the challenges of remote work became apparent, underscoring the importance of practical collaboration tools. Overcoming obstacles, such as coordinating with team members in different time zones and leveraging technologies like Teams for seamless collaboration, became integral to the workflow. The contrast between remote and in-person work illuminated the need for proactive communication and scheduled interactions, emphasizing the importance of planning even for the most minor in-person conversations.

11.9 Benefits Gained

Both projects have several benefits to the company, with the Digital Prescreen Acquire process, the initial procedure required 3-5 labor hours and the involvement of at least three individuals for analysis. The sales team would start the process by notifying the product team of the request. Subsequently, the product team would extract branch and zip code information, utilizing existing code, and upload it to Jira to generate a ticket. The engineering team would then process these zip codes through the Credit Bureau database, obtain raw data from a sample population, and transmit the data back to the product team. The product team would conduct analytics, perform calculations, and use the data to generate the final deliverable file for the potential client. Given the frequency of such requests, the product and engineering teams invested significant weekly time, resulting in a notable opportunity cost as they could have utilized this time for other pending tasks. The revised process completes the entire operation within 1-2 hours and is fully automated.

The completion of the matching project provided Micronotes the ability to establish a system for generating a time series for all records. Subsequently, this system streamlined the development of propensity scores, assigning each individual a probability of accepting offers. This strategic approach enabled Micronotes to focus on customers with higher propensity scores, yielding several advantages: a decrease in marketing costs for both Microphones and their clients, an increase in conversion rates from offer to closed loans. Furthermore, by utilizing this solution to improve new customer acquisition for partner banks, such as extending credit offers to non-customers, Mictronotes' partners could achieve a negative customer acquisition cost, marking a departure from the historically positive costs associated with such endeavors.

12. Future Work

This section discusses how the projects were finished, and additional projects that the company may pursue.

12.1 Additional Work for Digital Prescreen Acquire

The project was concluded by finishing the back end automation for the prescreen acquire process. The mockup for a rudimentary front end web page was also completed and handed off to Micronotes for continuation. As a future project, the web page could be integrated with their existing web pages and connected to the back end to submit a process request. After this, a process could be created to automatically send the request to the proper person for approval; this would link together the initial user request to the automated process. After approval, the automated process would then proceed and eventually output the final PDF report and automatically send it to the email specified by the user. The projects mentioned would fully automate the process with the exception of the necessary request approval. The current algorithms could also be improved to reduce running time. The current automated process takes around 1.5-2 hours to fully finish. This is a major improvement from the original semi-manual process which could take multiple days due to sending output to the next person to continue the process.

12.2 Additional Work for ADS Data Matching

Many projects that were reliant on a data matching algorithm can now be started. The development of the algorithm was the first step in a project designed to improve Micronotes' conversion rates for different types of loans. Our team was able to confidently match 22,000 rows of data based on the features chosen. On that front, there is more work to be done to get closer to matching each person between snapshots. Ideas for how to approach this involve using decision trees for feature selection. It is unlikely that there is one group of features that will successfully match all data points given the nature of financial data. Using a decision tree would allow for subsets of the data to be matched based on different categories of features. For example, using loan data is useful for matching people that have loans taken out, but not for others. Once the first set of people with loans are matched, they can be discarded from the dataset and a new category of features can be used to capture a different demographic of people. Additionally, now that there is a significant amount of data matched to be used as training data, the problem could be boiled down to a classification problem. The other projects that can be completed include using the matching algorithm to create a time series data set spanning six months. Once this is completed, further analysis can be done on the dataset to extract the likelihood for each person closing on different kinds of loans.

13. Conclusion

For each prospective client engagement, Micronotes employs its Digital Prescreen Acquire process to deliver a market breakdown. Before the completion of this project, the semi-manual process could take over a day. This process also uses anonymous ADS data, necessitating the development of a machine learning algorithm to ensure accurate record matching across different weeks. The projects focused on process automation, data processing, data matching, and machine learning models. The team effectively used the agile method to deliver the projects. Sprints were a week long and involved daily scrum meetings, weekly sprint reviews, retrospectives, and backlog refinements. This helped the team adapt quickly and adjust approaches to problems from week to week.

The first project automated Micronotes' existing Digital Prescreen Acquire process, removing human error and lengthy processing times. Originally, a ticket had to be created to start the process. Someone would then manually enter the information and get an output file which would be passed to another person. This process involved handing data from one person to another and running processes on multiple devices, including local processes. Now, this process only needs the required user input and the click of a button to have the process run from start to finish. This was completed by writing Python scripts within Databricks Notebooks using libraries such as Pandas and Pyspark. This technology was used to process the data and keep entries which met requirements. Openpyxl was also utilized to help create and save a readable Excel file of extrapolated results.

The second project aimed to develop a matching algorithm to analyze Experian data over time. Machine-learning techniques were utilized in order to match anonymized user records. This was accomplished through the use of distinct attribute categories for features which would not change from week to week. Creating a realistic time-series dataset would allow for more accurate propensity-scoring models. Significant challenges were faced in feature selection and algorithm development due to the scale and complexity of the data. Overall, this functionality significantly contributed to the capability of tracking users on a granular week-by-week basis.

Key learnings included working with large datasets, Azure environments, Agile methodologies, and communicating effectively with stakeholders. Technical skills like Python, Pandas, and Spark were strengthened. Non-technical growth areas included leadership, business acumen, and adapting to different work cultures.

Overall, both projects came to a successful conclusion. The Digital Prescreen Acquire project will be used in day to day activity, providing a fully automated way to retrieve a market breakdown for a specific financial institution before meeting with them as a prospect. The ADS Data Matching project model will be built on to produce propensity scores for successfully matched ADS records.

References

Aha! (2023, September 29). *Themes, epics, stories, and tasks.* Aha! software. https://www.aha.io/roadmapping/guide/agile/themes-vs-epics-vs-stories-vs-tasks

Amazon Web Services. (n.d.). *What is Scrum?*. AWS. https://aws.amazon.com/ what-is/scrum/

Apache Spark. (n.d.). *Apache SparkTM - Unified Engine for Large-Scale Data Analytics*. Apache SparkTM - Unified Engine for Large-Scale Data Analytics. spark.apache.org/

Clarkson, K. et al. (2006). *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice.* MIT Press.

Crunchbase (2023). *Micronotes - Funding, Financials, Valuation & Investors*. *Crunchbase*, www.crunchbase.com/ organization/micronotes/company_financials

First American. (n.d.) *What Is Title Insurance?* First American - Title Insurance, Specialty Insurance, and Real Estate-Related Services. www.firstam.com/index.html

Hastie, T. et al. (2009). *The Elements of Statistical Learning Data Mining, Inference, and Prediction*. Springer Texts in Statistics

Hayes, A. (2022, 12 June). *What Is a Time Series and How Is It Used to Analyze Data?* Investopedia. www.investopedia.com/terms/t/timeseries.asp#:~:text= A%20time%20series%20is%20a%20data%20set%20that%20tracks%20a,economic%20v ariable%20changes%20over%20time.

H, S. C. (n.d.). *About Python*. HonKit. https://python.swaroopch.com/about_python.html

Jacobson, I., Sutherland, J., Kerr, B., & Buhnova, B. (2022). Better Scrum through Essence. *Software: Practice and Experience*, *52*(6), 1531–1540. https://doi.org/10.1002/spe.3070

James, G. et al. (2013). *An Introduction to Statistical Learning with Applications in R*. Springer Texts in Statistics

Kerzner, H. (2018). Project management best practices: achieving global excellence (Fourth edition.). Wiley.

Microsoft. (2023, October 10). *What is Azure Databricks?* Microsoft Learn. https://learn.microsoft.com/en-us/azure/databricks/introduction/

Microsoft. (n.d.). *What is a Data Lake?* Microsoft Azure. https://learn.microsoft.com/ en-us/azure/databricks/delta/

Micronotes. (2023, September 21). *Digital Engagement for Banks + Credit Unions*. Micronotes. https://micronotes.ai/

Micronotes. (2023, October 20). *Digital Prescreen: AI for Financial Institutions*. Micronotes. https://micronotes.ai/products/

NVIDIA. (n.d.). *What is pandas?*. NVIDIA Data Science Glossary. https://www.nvidia.com/en-us/glossary/data-science/pandas-python/

Pandas. (2023). *Package overview*. Package overview - pandas 2.1.3 documentation. https://pandas.pydata.org/pandas-docs/stable/getting_started/overview.html

Peterson, D. (2018, November). *Maximize Efficiency: How Automation Can Improve Your Loan Origination Process* . Moody's Analytics. https://www.moodysanalytics.com/articles/2018/maximize-efficiency-how-automation-can-improve-your-loan-origination-process

Schwaber, K., & Sutherland, J. (2020). *Scrum Guide*. Scrum.org. https://scrumguides.org/scrum-guide.html

Sutherland, J., Coplien, J. O., Heasman, L., Hollander, M. den, & Oliveira Ramos, C. (2019). *A Scrum book: the spirit of the game*. The Pragmatic Bookshelf.

Wendel, S. (2023, September 8). *What is prescreen and prequalification?*. Experian. https://www.experian.com/blogs/insights/what-is-prescreen/

Wrike. (n.d.). *What is Agile Methodology in project management?*. Wrike. https://www.wrike.com/project-management-guide/faq/what-is-agile-methodology-in-project-management/

Appendix A

In order to provide a comprehensive overview of the key entities discussed in this paper and their interrelationships, we have included an entity cluster in the appendix. https://docs.google.com/document/d/1f5vqGDVLwrCt-OYnf6--5NkKvq25tJ5UFdjeK3suAqM/edit?usp=sharing