MATH 485: Final Written Report
Instant Decision for Credit Card Analysis
Group D3

Shitij Seth, Muhammad Salim, Deryck Ma, YuanJea Hew
Department of Mathematics, University of Arizona
Mentor: Dr. Christina Duron
Instructor: Dr. Calvin Zhang-Molina
12 May 2020

1. Introduction

1.1 Introduction to the problem

As the world moves towards automation, cashless payment methods are becoming increasingly popular among people of all ages. Currently, cashless payment methods are the most common form of payments in the US. One of the widely used methods of cashless payments are credit cards which gained popularity in the 1990s with the emergence of electronic banking. Nowadays, cashless payments are widespread in many countries and the volume of transactions has increased significantly over time [1]. The increase in popularity of credit cards is directly related to the amount of applications submitted to credit card companies. A big problem to come out of the increase in popularity and usage of credit cards is credit card debt. According to the Federal Reserve, the total revolving consumer credit debt reached $1.057 trillion in March of 2019 with 43% of credit card holders falling short to pay their monthly debt [2]. Companies have an extensive application screening process, and are discreet about the factors they take into account for applicants. Each application is a potential high risk investment, as companies can incur a loss if a credit card holder is unable to pay off the credit card bill. Therefore, it is critical that credit card companies are very accurate in their acceptance/rejection decisions for each applicant.

The aim of this paper is to demonstrate how various Machine Learning (ML) and Deep Learning techniques can be used in the credit card application decision process that companies face regularly. Although there is a wide-range of ML algorithms, only those deemed most suitable for this application are utilized. As a result, the selected algorithms are Logistic Regression, Support Vector Machines, Artificial Neural Network, Decision Trees & Random Forest, and *k*-Nearest Neighbors.

1.2 How the problem is handled currently

As mentioned earlier, credit card companies are discreet about their method of analyzing credit card applications. Yet, literature research has revealed that each application is analyzed by hand and separately, taking into account the applicant's income, debt-to-income ratio, etc., to determine if they meet underwritten standards. A big challenge for companies is to process the constant backlog of applications and the way of assessing applications by hand one at a time is very time-consuming.

1.3 How the proposed approach is unique

As there is no evidence that credit card companies use any automated methods of approving these applications, this paper proposes the use of a modern, automated way of deciding approval using novel ML algorithms. The aim is to provide a method which can be used by companies in the real world to assess multiple applications instantly. The accuracies and run times for each algorithm are both taken into account in order to identify an algorithm that is the best combination of accuracy and run time.

1.4 Brief summary of the proposed approach

The approach used in this analysis is to compare the six algorithms. The main goal for each algorithm is to maximize its accuracy, whereupon the result is further maximized through the implementation of the GridSearch method. The ensemble process is also explored to find out if any combinations of the six algorithms can yield a higher accuracy. Finally, missing values in the dataset is also handled.

2. Goals

The main objective of this analysis is to create an algorithm that provides the best combination of accuracy and run-time. The scope of the project extends to:

2.1 Enhance results from the reference paper

By enhancing the results from the reference paper, in this project, a similar approach as the reference paper is taken with the aim to improve the accuracy. The reference paper only focuses on two algorithms: Logistic Regression and Decision Trees. The Logistic Regression algorithm has been chosen as the base algorithm in this work.

2.2 Compare the accuracy from different algorithms

Comparing the accuracy is one of the most vital processes in this analysis so that the best algorithm could be chosen for predicting the credit card application approval. For each of the algorithms that are analysed, it is found that each has pros and cons. The challenge here is to compare these algorithms and create an algorithm which can provide the best combination of accuracy and run-time for the credit card application approval.

2.3 Determine the features that most affect card approval

In resolving the process in which credit card applications are approved, various factors should be taken into account. Although there are 16 attributes whose information is requested within each application, the main challenge is to identify how different combinations of attributes contribute to credit card approval.

3. Quantification of Data

The credit card approval dataset is collected from UCI Machine Learning Repository, in which all values are converted to meaningless characters in order to protect the confidentiality of the credit card applicants. To make it easier to work with the data, each value is transformed into integers, as demonstrated in Figure 1 below.

Male: a, b. Married: u, y, l. BankCustomer: g, p. EducationLevel: c, d, i, j, k, m, r, q, w, x, e, aa, ff. Ethnicity: v, h, bb, j, n, z, dd, ff, o. Prior Default: t, f. Employed: t, f. DriversLicense: t, f. Citizen: g, p, s. Label Approval: +,- (class attribute)	This was converted to:- Male: 0, 1. Married: 0, 1, 2. BankCustomer: 0, 1. EducationLevel: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12. Ethnicity: 0, 1, 2, 3, 4, 5, 6, 7, 8. Prior Default: 1, 0. Employed: 1, 0. DriversLicense: 1, 0. Citizen: 0, 1, 2. Label Approval: 1, 0 (class attribute)

Figure 1: The transformation applied to ten of the sixteen attributes within the dataset.

4. Preliminary Data Analysis

This dataset contains 690 data and every data has 16 attributes, 653 are complete data and 37 missing data. Of the 653 completed applications, 450 applicants are male applicants, while 203 are female. Currently, since the dataset only has 37 missing values, removing these values at this stage will not affect the ML models. Therefore, the 653 complete data is used to train and test each of the ML models. In the next stage, an approach to complete the missing data is discussed.

	Age	Debt	Years Employed	Credit Score	Income
Mean	**31.50**	**4.83**	**2.24**	**2.50**	**1013.76**
Median	**28.42**	**2.84**	**1.00**	**0.00**	**5.00**
Max	**76.75**	**28.00**	**28.50**	**67.00**	**2000.00**
Min	**13.75**	**0.00**	**0.00**	**0.00**	**0.00**

Table 1: The mean, median, maximum, and minimum values of the continuously valued attributes

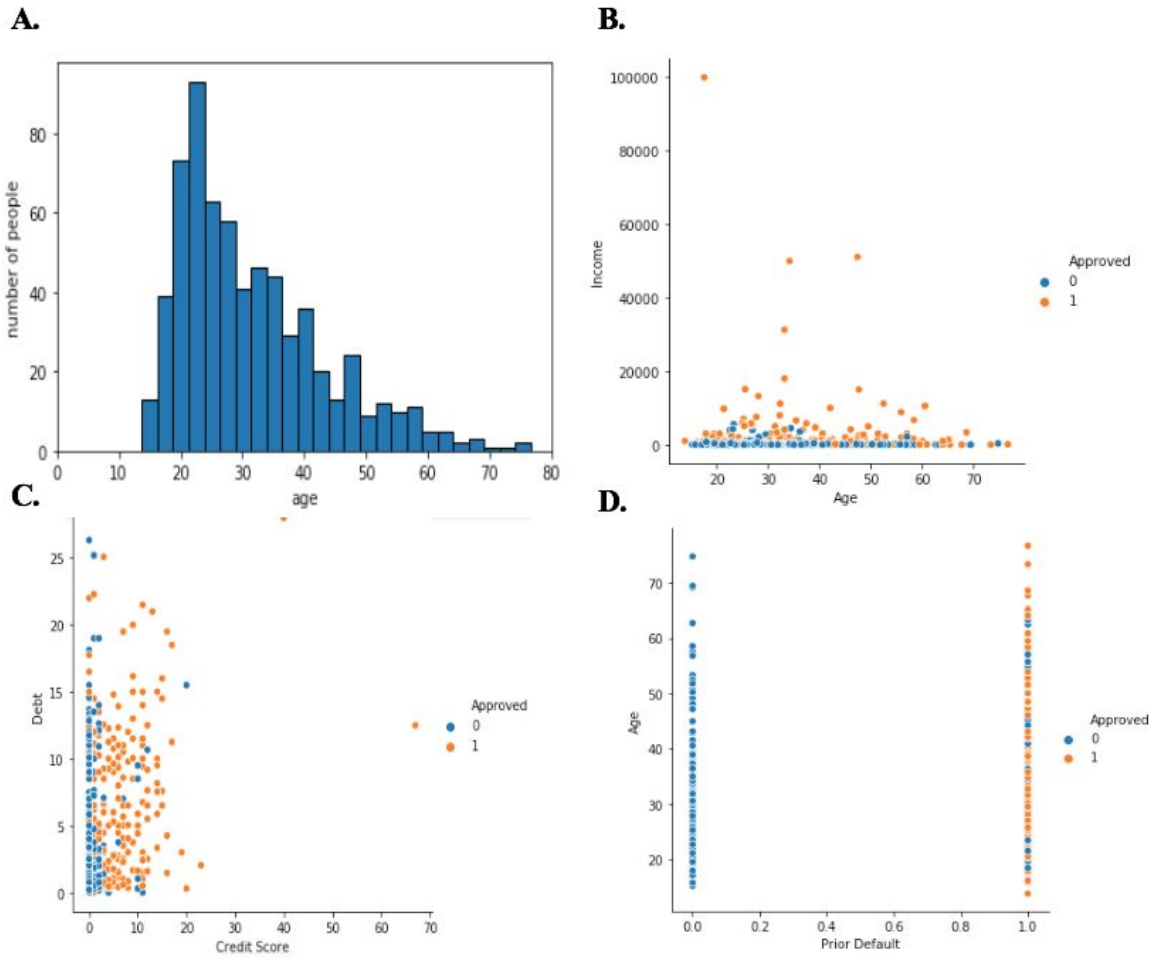

Figure 1: The distributions of the continuously valued attributes.

In Figure 1 (A), the majority of people who apply for a credit card are 10 to 30 years old. Figure 2 (B) suggests that people with higher income are more likely to be approved by the credit card issuer. Both Figures 1 (C) and (D) suggest that banks prefer to issue a credit card to those with a good credit history. In particular, compared to debt and credit score in Figure 1 (C), many people are rejected by credit card issuers because of low credit scores, even if they have good debt conditions. From Figure 1 (D), it is clear that people with prior default never get a credit card.

Based upon this information, a correlation matrix is constructed and displayed in Figure 2. Note that a brighter color and a larger number, taken together, suggests that the two features have a high correlation. As a result, the four

most important features from the correlation matrix is (1) years employed, (2) prior defaults, (3) employed, and (4) credit score.

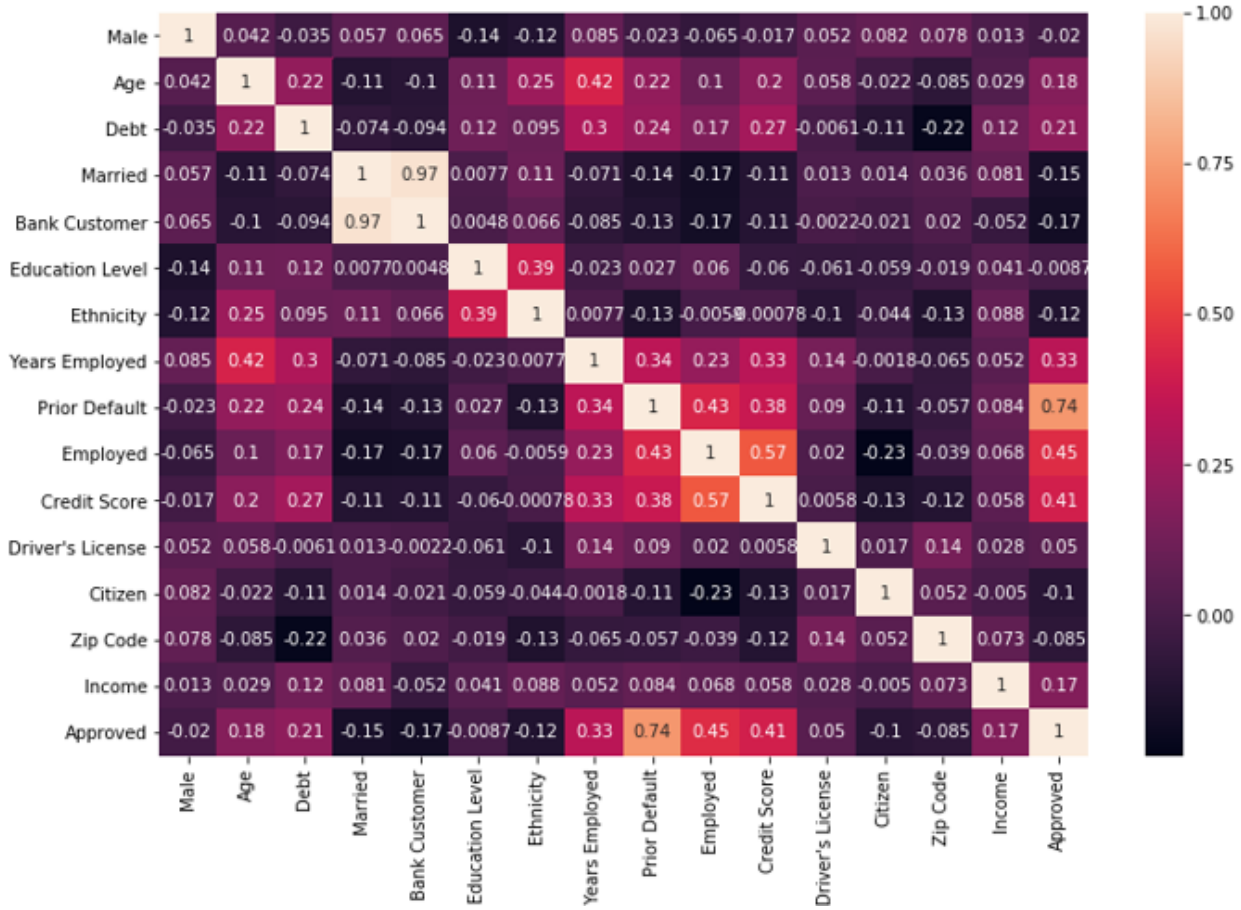

Figure 2: The correlation for each pair of the sixteen attributes. Brighter colors indicate a stronger correlation, while darker colors indicate a weaker correlation. The correlation varies within [-1, 1].

5. Algorithms

5.1 Algorithm Preliminaries

The given data consists of inputs and labels which make the supervised machine learning(ML) algorithms the best tools to solve this problem as these algorithms work on the pretence that the data consists of inputs and labels and the problem is of classification or regression nature. Supervised machine learning algorithms divide the given data set into two datasets: training and test. During the training phase, the model uses the training set to learn patterns and correlations

between the inputs and the labels. After the training phase, the model is tested on the test set to determine the accuracy, or how well the model has "learned". Some supervised ML algorithms, such as ANN, traverse through the training set multiple times to better learn from the data. Due to the similarity in the functionality of the six ML algorithms, some parts of their implementation are similar.

- **Training-test split**: As mentioned above, it is the nature of these algorithms to train on a subset of the data and test on the remaining portion of the dataset. It is a common practice in ML to use a majority of the data to train the model. While implementing these algorithms, the common 75-25 split (75% of data as training set; 25% as test set) is initially used, and then the ratio is optimized for the best results which, in most cases, is found to be the 85-15 split. This 85-15 split is considered a parameter, since it is an optimized value for each algorithm.
- **Grid search**: Grid search is an exhaustive method of hyperparameter tuning to find the optimal values for a given model. It does an exhaustive search over a set of given hyperparameter values and determines which combination produces the best accuracy. This can be pretty significant, as the models highly depend upon the parameter values.
- **Random seed:** A random seed is utilized to ensure that the same training set and test set are used within each algorithm. In particular, the data is loaded from the data (Excel) file in the same order as provided within the file, and then is shuffled once to remove any existing biases.

5.2 Logistic Regression

In this paper, the logistic regression (LR) algorithm is considered the basic algorithm solely due to the paper objectives: to replicate logistic regression results from the reference paper. This algorithm is one of the most commonly used supervised learning algorithms, although there are also linear and multivariate regression algorithms. However, these two regression algorithms are not very useful due to their binary classification. Therefore using logistic regression, the result is not limited to binary classification method only.

$$p = \frac{1}{1 + e^{-(b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_p x_p)}}$$

Before continuing in further details, it is good to know some useful terminology in this algorithm.

- **Logistic**: A logit function which is a common S-shaped curve used to bring non-linearity to predictions.
- **Confusion matrix**: A tabular representation of actual versus predicted values.

LR is useful for predicting continuous variables. Instead of having a binary classification, which can only produce two output levels such as 0 or 1, LR can produce an output that varies between 0 or 1. For the splitting part, LR uses 85-25 split instead of 75-25, as this provides the highest percentage of accuracy for this algorithm. In the first step, LR tries to determine the relationship between the target class of dependent variables (DV) and the 15 attributes of the independent variables (IV). After that, in the second step, a classifier is created by using calculated logits. These calculated logits are then used to predict the target class. In the third and final step, a confusion matrix is used to determine the accuracy. Additional details about the confusion matrix are included below.

After completing the three steps, the model is evaluated based upon the results of the confusion matrix, which determines the number of individuals that are classified as “True” and “False”.

	FALSE	TRUE
0	55	9
1	3	33

Figure 3: An example of a confusion matrix.

A confusion matrix makes a classification of correct and incorrect predictions such that the percentage of accuracy can be computed as follows by considering the toy confusion matrix in Figure 4. First, the top left number is the number of observations correctly predicted as denied credit, while the bottom right is the number of observations correctly predicted as approved credit. The other two are the number of falsely predicted numbers. From the total number of observations of 100 (55 + 3 + 9 + 33), there are 88 (55 + 33) correctly predicted observations such that 88% is considered the accuracy.

$$\boldsymbol{\beta^1 = \beta^0 + [X^T W X]^{-1}.X^T(y - \mu)}$$

$\boldsymbol{\beta}$ *is a vector of the logistic regression coefficients.*

$\boldsymbol{W}$ *is a square matrix of order N with elements* $n_i\pi_i(1-\pi_i)$ *on the diagonal and zeros everywhere else.*

$\boldsymbol{\mu}$ *is a vector of length N with elements* $\mu_i = n_i\pi_i$.

5.3 Support Vector Machine

SVM or Support Vector Machine, is another supervised learning. Different from LR, SVM builds a classifier that separates data points using a hyperplane. Before continuing in further details, it is good to know some terminology pertinent to this algorithm.

- **Hyperplane**: A decision plane, separate dataset accordingly to different class memberships.
- **Support vectors**: The closest data points to the hyperplane.
- **Margin**: The distance between the line and support vectors.

The SVM algorithm is a machine learning algorithm that can handle both linear and nonlinear classification. The SVM algorithm will try to separate the dataset into 15 different classes by creating a hyperplane. This hyperplane is formed by finding the best (i.e., ideal) seperating lines by computing the margin, where the best separating lines are those with the highest margin. Some essential parts are completed before the SVM algorithm model evaluation takes place; refer to Section 5.1 for further details. For the splitting part, SVM uses 85-25 split instead of 75-25, as this provides the highest percentage of accuracy for this algorithm.

In following the above steps, an SVM model is created by importing the SVM module. A support vector classifier object is created by passing the argument kernel as the linear kernel. The model evaluation for the SVM algorithm is quite simple. The percentage of accuracy is achieved by comparing the actual test set values and predicted values. In two dimensions, it is called a line, in three dimensions, it is called a plane, while in more dimensions, it is called a hyperplane.

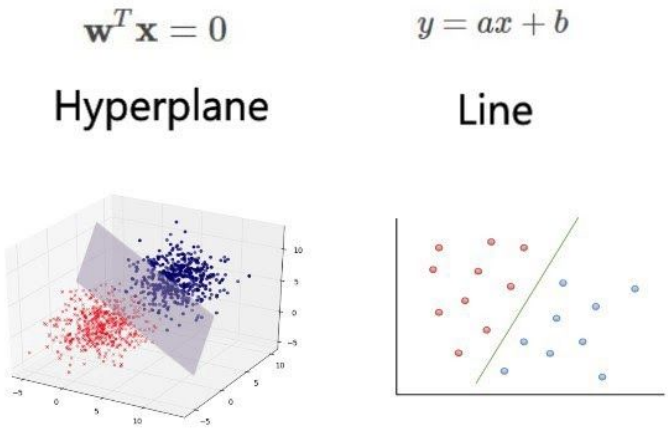

Figure 4: An example of a hyperplane versus a line.

5.4 Artificial Neural Networks

Artificial Neural Networks in another supervised machine learning algorithm which is inspired by neurons in animal brains. Each neuron takes in a set of inputs and predicts an output. These neurons are divided up in layers and every neuron in each layer is interlinked to several surrounding neurons which allows them to learn and find patterns in the data. Say each neuron is given a set of inputs as matrix *x*. The neuron calculates the weighted average of the input matrix based on the weight matrix *w* and adds bias *b*.

$$z = w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_nx_n = \mathbf{w}^T \cdot \mathbf{x}$$

The neuron then uses a pre-defined nonlinear activation function to calculate the predicted value. Activation functions are one of the key elements of the neural network. Without them, the neural network would become a combination of linear functions, or simply, a linear function itself. The ANN model would have limited expansiveness, yet nothing greater than the logistic regression. The non-linearity element allows for greater flexibility and creation of complex functions during the learning process. The activation function also has a significant impact on the speed of learning, which is one of the main criteria for their selection. In the implementation of ANN for this project, it is found that the Logistic function gives the best results for this data set. A logistic function or logistic curve is a common S-shaped curve (sigmoid curve) with equation

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}},$$

where

e = the natural logarithm base (also known as Euler's number),

x_0 = the x value of the sigmoid's midpoint,

L = the curve's maximum value, and

k = the logistic growth rate or steepness of the curve.

In a nutshell, the input layer takes the input matrix. Each input is linked to other inputs through a weight matrix. The weight determines the influence of one input on another. The model learns by constantly changing the weights and creating a feed-forward propagation function. It adjusts the weights during backpropagation using the error found during forward propagation. The model learns until it finds the (supposed) global minimum in the loss function by going through the data several times, each called an "epoch". In this work, a binary cross entropy is used, but depending upon the problem, different functions can be applied. After each epoch, it is expected that the loss function J decreases and the accuracy function L increases.

$$J(W,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$

$$L(\hat{y}, y) = -(y \log \hat{y} + (1-y)\log(1-\hat{y}))$$

The output layer then uses an activation function to map the input variable to the output variable.

5.5 Decision Tree

Decision Tree is a tree-structured Supervised Machine Learning algorithm that is commonly used in classification problems, whose goal is to construct a model that predicts the value of a target variable by learning basic rules of decision derived from data characteristics [4]. Before continuing in further details, it is good to know some terminology pertinent to this algorithm.

- **Root node**: This represents the entire dataset and gets divided into two or more homogeneous sets.
- **Decision node**: This splits itself into further sub-nodes
- **Leaf/Terminal node**: This is a node that does not split
- **Splitting**: This is the process of dividing a node into two or more sub-nodes
- **Pruning**: This is the process of removing sub-nodes of a decision node and is the opposite process of splitting

Most decision trees use the ID3 algorithm, which creates decision trees across the space of possible branches without backtracking by using a top-down greedy search method.As the name suggests, a greedy algorithm always makes the choices which seems to be the best at the moment [6]. Below is an ID3 decision tree algorithm. Note that the attribute selection criteria is discussed in the next section.

- Set the original dataset as the root node.
- For each attribute
  - Partition all data at the current node by the value of the attribute.
  - Compute the entropy or the information gain
- Select the attribute with the greatest information gain or smallest entropy.
- Set this attribute to be the splitting criterion at the current node.
  - If the best information gain is 0, tag the current node as a leaf and return.
- Partition all instances according to the selected attribute.
- For each child node
  - If the child node has instances from only one class, tag it as a leaf and return.
  - If not set the child node as the current node and recurse to step 2.

If the decision tree fully extends, it may lose some generalization capability, which is called overfitting. To prevent overfitting, the decision tree is pruned.

5.6 Random Forest

Random Forest is a supervised Machine Learning algorithm. Random forest, as its name indicates, is made up of a large number of individual decision trees that act as an ensemble. Each tree spits out a class prediction in the random forest, and the class with the most votes becomes the prediction of the model [6].

One big advantage of the random forest algorithm is the low correlation between trees. Each tree branch in the algorithm allows their individual errors to be separated from one another. Some trees may have the wrong prediction, while other trees have the correct prediction. Grouping these trees together hides the errors and enhances the qualities of the individual trees.

Attribute Selection Criteria

- Entropy

Entropy is a measure of the randomness in the information being processed[6]. To calculate entropy for an attribute use this formula.

$$E(S) = \sum_{i=1}^{n} - p_i log_2 p_i$$

S is the current attribute, p_i is the probability of a case i of attribute S.

Take male the attribute as an example, there are 450 male, 203 females.

$$E(male) = -(0.69 log_2 0.69) - (0.31 log_2 0.31) = 0.8931$$

5.7 *k*-Nearest Neighbors (K-NN)

The *k*-Nearest Neighbors algorithm is a supervised machine learning algorithm that uses a neighbor based classification scheme for instance-based learning [1]. Since the chosen label is the approval of credit cards, this limits the model to have a binary prediction value. Therefore, the implementation of K-NN is relatively simple.

Default implementation of K-NN begins with a query point being assigned to the dataset expressed with *x*-list and *y*-list. Each data point's distance with respect to the query point is measured using the Minkowski metric system [1]. Based on the distances, K-NN then selects the *k* number of data points nearest to the query point as its nearest neighbors [1]. The classification process only takes into account the *k* number of nearest neighbors. With that, the algorithm takes a single majority vote among the nearest neighbors and concludes the highest representation to be the predictive outcome of the label [1]. This methodology of finding the label is run with every datapoint in the training set. Then, with the same procedures, K-NN is applied to the testing set to see how well the algorithm has performed. Using the *predict_proba* method of K-NN, the percentage accuracy is calculated to analyze how well the prediction is made by the algorithm.

The type of metric system and the *k* value for nearest numbers utilized are some of the hyperparameters of the algorithm, which means that they can be manually tuned to other options. GridSearch is used to find the optimal hyperparameter setting that outputs the highest percentage accuracy.

7.Ensembling

Ensembling is a way of combining two or more algorithms in order to achieve a higher accuracy than any single algorithm. The idea behind it is that different algorithms have different strengths and weaknesses and ensembling helps

to hide the weaknesses of individual algorithms. The two main ensembling methods are:

- **Max Voting:** This is a common approach in classification problems. In this approach, multiple algorithms make a prediction on the same data point. Each prediction is considered a "vote". The label with the most votes is chosen as the output.
- **Averaging:** Similar to the max voting technique, multiple predictions are made for each data point in averaging. In this method, an average of the predictions from all the models is used to make the final prediction. Averaging can be used for making predictions in regression problems or while calculating probabilities for classification problems.

Both of the ensembling methods are attempted, and all possible combinations of the six algorithms used in this project are tested. Using the same train-test split and same test set, the accuracies and run time for each are recorded and compared. The goal for using the ensembling method is to find an algorithm which does better than any of the individual algorithms. Due to the way that the ensemble method combines different algorithms, it hides the negative features of the algorithms and enhances the positive features, thereby making it a promising candidate to yield better results.

8.Missing Data

Based on the credit card dataset, the data points inscribed as "*?*" are considered to be missing data. By only considering the available data points at hand, one common approach to handling the missing data is to replace the missing values with the average. However, this approach may not yield the most accurate data.

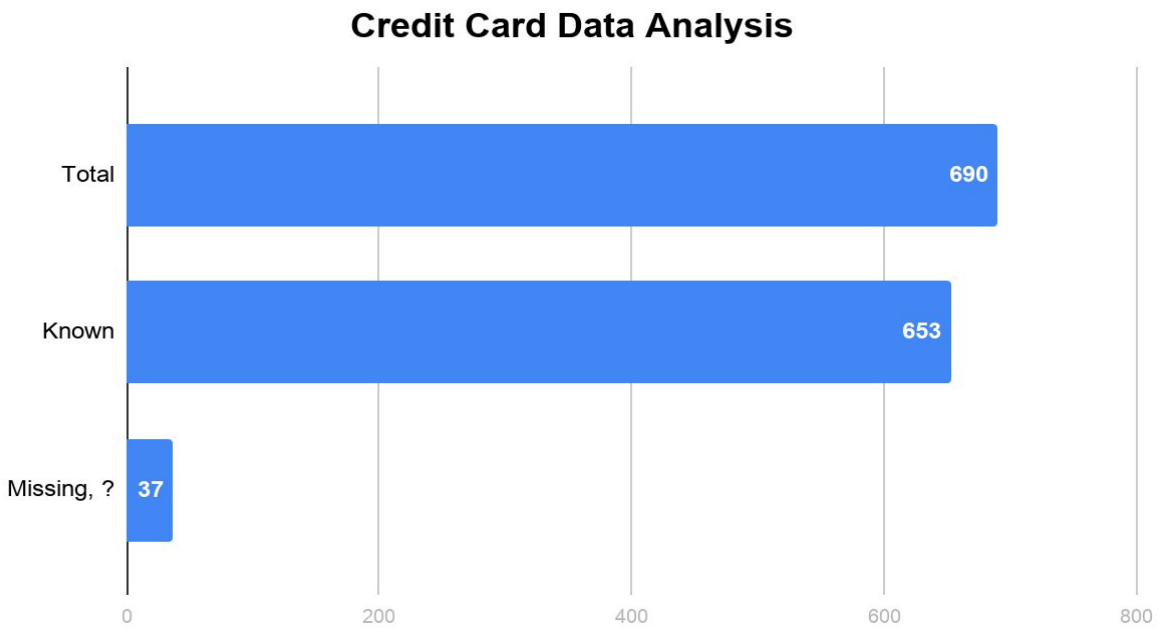

Figure 5: A bar chart showing the number of missing data points of 37 as compared to the number of data points known of 653 and altogether of 690 [2].

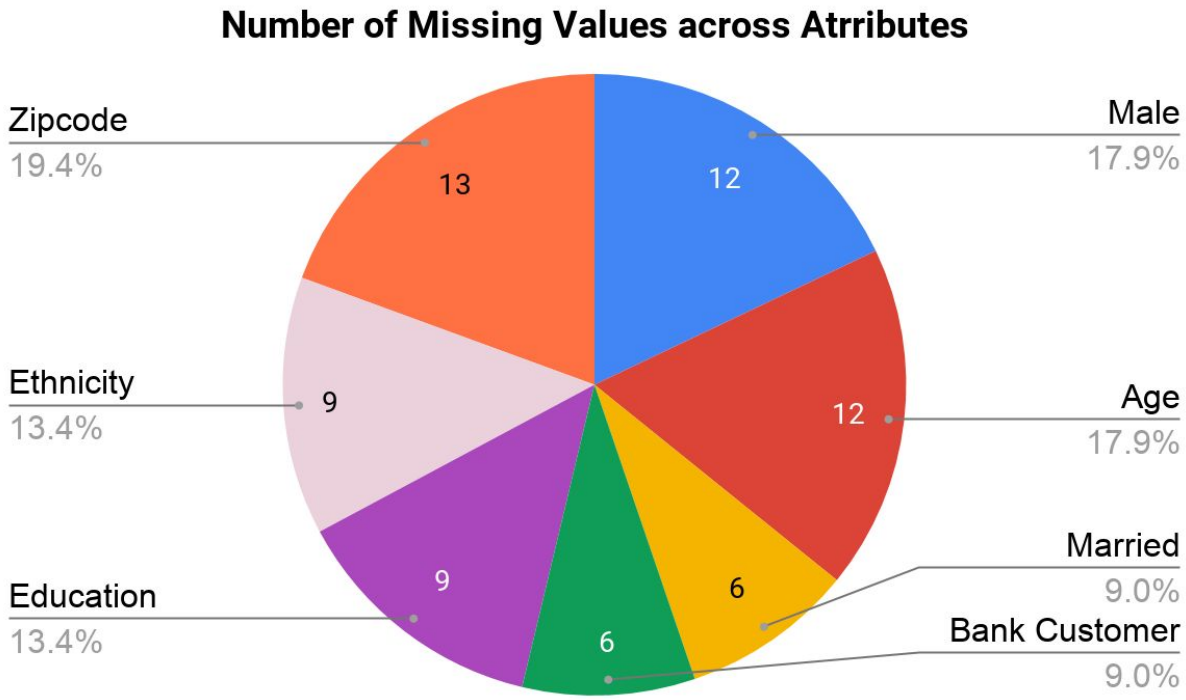

Figure 6: A pie chart illustrating the distribution of missing values across all the attributes. Notice only 7 out of the 16 attributes contain missing values. A total of 67 missing values are embedded in those attributes [2].

In this paper, the missing data problem is handled by using a K-NN inspired approach. Although K-NN is a supervised machine learning algorithm, the same principles can be applied to achieve better estimates of the missing data. The proposed idea is to set the data labels as the attributes with missing values instead of approval. In other words, the new *y*-list will have 7 elements. This way K-NN predicts upon the 690 data values per corresponding attributes listed on Table 2. With that, using the supervised machine learning nature of K-NN, a training and testing split ratio (as before, 85-15) is applied onto the 653 known data points across all attributes for tuning the hyperparameters of the algorithm. Once the optimal hyperparameter setting is justified through GridSearch, the total number of datapoints of 690 will now be set for the algorithm training. Then, using the optimized K-NN hyperparameters setting, the machine learning algorithm generates a prediction to all 690 data values per missing valued attribute such that the new predicted data values fill in the *?* labelled data. Regardless of the approach used to handle missing data, the accuracy of these new data values can be questioned. Therefore, one way to address this directly is to analyze the newly predicted known data values with the original known data values. Quantifying how much both values differ can help to determine how well the missing data is being filled.

9.Results

One of the goals of this project is that the final algorithm can be used by a credit card company for their purposes. Therefore, a conscious effort is made to take into account every possible constraint or standards these companies have to abide by in their operations. From literature, the United States Federal Trade Commission advocates the Equal Credit Opportunity Act (ECOA) which forbids companies to use several personal details of the applicants to be factored into credit card approvals namely race, color, religion, national origin, sex, marital status and age. Some of the features listed above are part of the data set. Therefore, these specified features are left out during the analysis within this paper. Furthermore, the results of the correlation matrix plot (Figure 3) indicate that all the prohibited attributes show low or negative correlation to approval, except for age [16], which suggests that the data set abides by ECOA laws. The subsequent analysis is entirely based on the assumption that the acquired data set very accurately labels applicants.

Also, by extrapolating from the correlation matrix plot, four features show very high correlation to the approval compared to others. These four features are (1) years employed, (2) prior default, (3) employed, and (4) credit score. As a result, these four features are used within the analysis, as these four seemingly are the most important with respect to the final approval. The fifth highest factor within the correlation matrix is not used within the analysis, as there is a significant difference between the correlation of the fourth highest feature and the fifth highest feature. Utilizing four of the fifteen possible features allows both the dimensionality of the data set and the overall algorithmic run time to be reduced, while also improving the accuracy of these algorithms. It is noted that none of the four features contained any missing data values for any applicant and are thus considered "complete".

All the algorithms mentioned in Section 5 are optimized and tuned to perform to the best of their ability. Some common (as mentioned in Section 5.1) and some specific (mentioned in each algorithm's description) tools are used to optimize the train-test split and the hyperparameters. The results for each algorithm are presented in Table 2.

Algorithm	Percentage (%)	Run Time (s)
ANN	**92.96**	**0.035865**
LR	**93**	**0.004508**
SVM	**88**	**0.010412**
***K*-NN**	**93**	**0.004109**
DT	**91**	**0.222498**
RF	**90.56**	**0.001875**

Table 2: The measured percentage accuracy and run time in seconds comparison for each supervised machine learning algorithm using Grid Search.

Based on Table 2, K-NN provides the best combination of accuracy and run time with 93% accuracy and a run time of about 0.004 seconds per application. These results are for the individual algorithms, whereas the ultimate goal is to find one single algorithm, or an ensemble of algorithms, which yield the best accuracy. To be able to compare these models or ensemble them, the models need to be

comparable. In order for them to be comparable, some of the features like the train-test split, random seed, etc. are kept the same. In the implementation of the ensembling method, it is found that the Max Voting approach gave better results. All 63 possible combinations of the individual algorithms are tested. The results from the ensembling method are displayed in Table 3.

Algorithm	Percentage (%)	Run Time (s)
LR + DT + RF	**94**	**0.05390**
ANN + DT + RF	**93.63**	**0.04508**
ANN + DT	**93.4**	**0.10413**
***K*-NN + DT + LR**	**93.31**	**0.04103**
DT + LR	**93.31**	**0.22249**
ANN + *K*-NN + LR	**93.26**	**0.01879**

Table 3: The measured percentage accuracy and run time in seconds comparison for top 5 combinations of algorithms in terms of accuracy.

Table 3 shows that an ensemble of LR, DT and RF gives the best accuracy (94%) and a run time (0.05390). Therefore, the ensemble of LR, DT and RF is deemed the final (best) algorithm, as the algorithm achieves a higher accuracy than that used in the reference paper, which reported 84% accuracy using an ensemble of LR and DT. It is noted that the reference does not provide an analysis of the run time.

In addition, the final algorithm is compared to a few baselines. One of these baselines is the "Data Baseline", which is the most amount of data classified in a single label in the data set. This is 55% for the provided data set, which means that 55% of the applicant's in the data set are classified into the single label "Approved". The final algorithm should beat this baseline because if one randomly picks a label for an applicant, then there is a 55% chance that it will be classified correctly, and the complex ML algorithms should at least beat random selection. A second baseline is the "Default Parameter Baseline", where the default parameters as offered by *scikit-learn* package are used, and the parameters are not optimized by Grid Search. Table 4 displays the results of the baseline comparisons.

	Accuracy (%)	Run Time (s)
Data Baseline	**55.5**	**N/A**
Default Parameter Baseline	**86.52**	**0.0628**
Final Model	**94**	**0.0539**

Table 4: The final models percentage accuracy and run time in seconds comparison from baselines.

14. Conclusion

As seen in Section 10, the highest accuracy achieved is 94% from an ensemble of DT, RF and LR, which is much higher than the "Data Baseline" (55.5%) and also significantly higher than the accuracy reported in the reference paper (84%). The difference in accuracy between the best individual algorithm and the final model is not very statistically significant with the final algorithm having 1% higher accuracy, but in an actual scenario even 1% misclassification could cost the company a big financial loss, especially for a large volume of applicants.

As it turned out, there was an initial plan to deal with the missing data values (section 8) but as mentioned in section 13, we used just the top 4 features for which there were no missing data, therefore, we did not have to address the problem. Although, the method suggested was tested and seems to work well.

All the code files, data files, etc. may be accessed in the Box folder (https://arizona.box.com/s/86iqqsksxj549i13b8i17k7q5ube6w2).

Each member has contributed equally to this project. Until now, a stage-by-stage approach has been taken, and the following is a report of each member's contribution at each stage:

Stage 1

Shitij Seth - Coding and Analysis
YuanJea Hew - Literature study and Graphics
Muhammad Salim - Coding and Analysis
Deryck Ma - Literature study and Graphics

Stage 2

Shitij Seth - Artificial Neural Networks
YuanJea Hew - K-Nearest Neighbors
Muhammad Salim - Support Vector Machines
Deryck Ma - Decision Trees/Random Forests

Stage 3

Shitij Seth - Model comparison analysis and Ensembling
YuanJea Hew - Missing data
Muhammad Salim - Logistic Regression
Deryck Ma - Graphics

As the end of the semester approaches, there are a number of ways to carry forward into uncharted aspects of the model. These are the possible trajectories that this project could take, including some areas that require further enhancements, which are detailed below:

- Another way to compare models is on the basis of some Bayesian statistics. Statistics like the AIC score, BIC score, Maximum log-likelihood, etc. can be explored as a way to weigh them as each statistic measures a different aspect of the model.
- While there are numerous techniques to handle missing data, the procedure extracted from the K-NN algorithm is a promising option to tackle such an issue. Other alternatives, such as taking the mean of the existing data, can also be considered if the current proposed approach of K-NN does not produce promising results.
- We settled on a set of ML algorithms for our project but we would like to point out that there could be several other algorithms which might perform as well or even better.
- Other methods for dimensionality reduction such as Principal Component Analysis (PCA) can be used to further reduce the number of features used in the model.

13. Acknowledgements

This project is mentored by Dr. Christina Duron, whose help in code troubleshooting and literature review is acknowledged with great appreciation. The project would not have been successful without her constant guidance and supervision. Support from the class instructor, Dr. Calvin Zhang-Molina is also gratefully acknowledged. The time and effort spent on both Dr. Christina Duron and Dr. Zhang-Molina during the unprecedented time of Covid-19 is immensely in appreciation and gratitude.

14. References

1) Scikit Learn, Nearest Neighbors
   Retrieved from https://scikit-learn.org/stable/modules/neighbors.html
2) Kuhn, R. (n.d.). Analysis of Credit Approval Data. Retrieved from http://rstudio-pubs-static.s3.amazonaws.com/73039_9946de135c0a49daa7a0a9eda4a67a72.html
3) Brownlee, J. (2019, August 12). Supervised and Unsupervised Machine Learning Algorithms. Retrieved from https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/
4) Scikit learn, Decision Tree
   https://scikit-learn.org/stable/modules/tree.html
5) Chauhan,N.S(2019,Dec 24) Decision Tree Algorithm-- explained
   https://towardsdatascience.com/decision-tree-algorithm-explained-83beb6e78ef4
6) Yiu,T(2019, Jun 12) Understanding Random Forest
   https://towardsdatascience.com/understanding-random-forest-58381e0602d2
7) Wilson, A. (2019, October 1). A Brief Introduction to Supervised Learning. Retrieved from
   https://towardsdatascience.com/a-brief-introduction-to-supervised-learning-54a3e3932590
8) Krishni. (2019, January 5). An introduction to Grid search. Retrieved from https://medium.com/datadriveninvestor/an-introduction-to-grid-search-ff57adcc0998
9) Allibhai, E.(2018, November 19). Ensemble Learning Using Scikit-learn. Retrievedfrom

https://towardsdatascience.com/ensemble-learning-using-scikit-learn-85c4531ff86a
10) Singh, A. (2020, March 22). A Comprehensive Guide to Ensemble Learning (with Python codes). Retrieved from https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/
11) Brid, R. S. (2018, October 17). Logistic Regression. Retrieved from https://medium.com/greyatom/logistic-regression-89e496433063
12) Scikit Learn, Support Vector Machine, Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
13) Scikit Learn, Logistic Regression, Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html?highlight=logistic%20regression#sklearn.linear_model.LogisticRegression
14) Navlani, A. (2019, December 27). Support Vector Machines (SVM) in Scikit-learn. (n.d.). Retrieved from https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python
15) Sharma, N. (2020, February 11). Understanding the Mathematics behind Support Vector Machines. Retrieved from https://heartbeat.fritz.ai/understanding-the-mathematics-behind-support-vector-machines-5e20243d64d5
16) Your Equal Credit Opportunity Rights. (2013, January). Retrieved from https://www.consumer.ftc.gov/articles/0347-your-equal-credit-opportunity-rights
17) Szmigiera, M. (2019, December 16). Cashless payments in the United States - Statistics & Facts . Retrieved from https://www.statista.com/topics/4586/cashless-payments-in-the-united-states
18) 2020 Average Credit Card Debt Statistics in the U.S. (2020, January 4). Retrieved from https://www.lexingtonlaw.com/blog/credit-cards/average-credit-card-debt-statistics.html#general-statistics