Data Science

Capstone Report - Fall 2022

Prolégo

Alejandro Zaccour,
Mingxuan Wu

supervised by
Li Guo

Preface

Translating the real world into numbers has become the usual. Numbers now represent persons with feelings, tendencies, ambitions, and everything that used to make us “human”. Algorithms have never been as powerful as they are now, even knowing what you want before you know it. As more and more humans become data, it is crucial to understand the numbers behind decisions. Therefore, in this project, we embark on a mission to understand what the numbers behind credit delinquency are and how they can be used to predict such action. In order to do so three distinct machine-learning algorithms have been implemented to manifest our findings.

Acknowledgements

An extended thanks to Professor Matthew Zeidenberg for helping us along this journey. His guidance really made an impact on the results of this project. Additionally, thanks to Anna Katz for her multiple contributions regarding the grammatical soundness of the paper. Also, thanks to Professor Guo Li for her concern and advice for our research progress.

Abstract

The report predicts the probability of a customer's future credit card default based on monthly customer data provided by American Express on Kaggle. For legal reasons, the customer's information has been heavily masked. Therefore, this paper first deciphers the original data, and then uses a decision tree classifier, XGBoost classifier, and a multi-layer perceptron. The results show that for the dataset used in this article, XGBoost is the more accurate of the models, having P_2_Last as its most influential feature.

Keywords

Capstone; Data science; NYU Shanghai; Machine learning algorithms; Credit card default prediction

Contents

1 Introduction 5

2 Related Work 6

3 Solution 10

3.1 Unmasking the data . . . 10

3.2 Imputing for Missing Values . . . 10

3.3 Understanding the Data . . . 11

3.4 Decision Tree Classifier . . . 11

3.5 XGBoost . . . 14

3.6 Neural Network . . . 15

3.7 Feature Importance . . . 17

4 Results and Discussion 18

4.1 Experiments Results . . . 18

4.2 Main challenges . . . 19

5 Conclusion 19

1 Introduction

Have you ever applied for a credit card and gotten denied? It's probably because of the classification system the issuer uses. Credit cards are an essential extension of a wallet, especially as the world turns digital by the second. It protects its users from having to carry large sums of cash and helps them acquire their desired purchase with the option of paying it over time. Though the concept of credit is not new, its history began in early Mesopotamia approximately 5000 years ago [1]. Several centuries after, the Medici optimized this version of credit cards throughout the renaissance, and it successfully passed over generations to still be used in our present-day society [1]. Throughout history, creditors have continually sought specific traits that would identify exceptional borrowers. Some traits include age, net worth, payment history, job status, spending habits, and many more. These traits enable creditors to better understand their possible borrowers and formulate a decision regarding their creditworthiness. Copious amounts of companies still use these methods. However, there is yet one that has created a perfect classification system to predict credit defaults. In order to optimize the current systems of classification, American Express launched a Kaggle competition where the sole purpose is to correctly predict if a customer will not repay their debt. The data set given was composed of multiple columns that expressed the particular traits a customer possesses. With these traits, the main goal was to figure out if a customer would pay or not.

The main objective of this project is to understand how to accurately predict if a client will default on its debt or not. Additionally, the dataset provided has been completely masked therefore turning it into useful data will be of the topmost priority. After the dataset has been completely unmasked, the information would still be of a customer will still be on multiple lines. Therefore, by using the 'mean', 'std', 'min', 'max', and 'last' of each customer, a better picture of how the customer behaves can be observed. In the process of finding the best prediction, the use of multiple models will allow for a better understanding of how the data works. The models will be a Random Forest, XGBoost, and multi-layered perceptron. The steps we completed in this process go as follows:

1. Unmask column by column, making sure that the information is of topmost quality, as it's a crucial step to a precise prediction.

2. After doing so, it is essential to correctly input any missing values in the dataset. The KNNImputer will come in handy in this step as it will give more accurate imputations than

just replacing the values with the mean or zeros.

3. Once all of the data has been correctly cleaned, it is time to separate it into the train and test sets that will later be used to evaluate our models.

4. Afterward, the data is ready to input into the first model, the decision tree classifier. This will grant us a good solid base to then adjust for any changes in the data and build our more complex models.

5. Moreover, the use of XGBoost will become the second model as it will test the difference between using one tree against multiple trees.

6. After the XGBoost has been completed, it's empirical that the model is compared with a non-tree classification system. Therefore, the use of a multi-layered perceptron is crucial. This will uncover what method of classification returns a higher accuracy in heterogeneous columns.

7. Once all of the models have been completed to extract further analysis, it is empirical to extract what the most important features & groups features.

8. Finally, a conclusion would be drawn to decide what makes an outstanding predictor and what should be considered when doing such.

2 Related Work

The concept of credit has been around for a long time. The first signs of credit came from ancient Mesopotamia around 5,000 years ago [1]. Historians have been able to retrieve clay tablets that Mesopotamian merchants used to keep track of their trades with neighboring civilizations. These records were the first official known examples of monetary trust agreements between individuals [2]. Credit has been around for an extensive period of time, but the art of whom to lend is why this problem has prevailed through time. Fast forward several centuries, and you stumble across multiple attempts to solve this problem. Some of the most famous cases include the Medici Bank, Retail Credit Company, The Diners Club, American Express, and many more [3]. All of them faced the same question, will they pay me back?

This question was a monumental obstacle for all of these businesses. Bank of America, with its BankAmericard, couldn't manage this obstacle and faced grave consequences. As Frankel states in her Forbes article, "This first attempt ended up being a costly error in judgment, with delinquency rates over 20% and rampant fraud [3]". Therefore, finding a solution that mitigates this risk is essential. In attempts to do so, all data on the debtor is crucial for increasing their debt default prediction accuracy. However, there was yet to be a scalable process that could reach any debtor.

In the 1950s, mathematicians Earl Issac and Bill Fair tackled the credit prediction scalability problem and designed the FICO model. The FICO Model was able to reduce a person's characteristics into quantitative figures, which inherently enabled the creation of a strong database to then use for predictions. In order to assess a successful classification, plenty of variables come into play. These variables fall into five categories: amounts owed, payment history, credit mix, length of credit history, and new credit. Additionally, these five categories hold different weights to signify that different characteristics of a person carry more importance than others when formulating the final decisions. This further helps the model become more "human". Since implementing these methods, we can see a constant decline in the delinquency/default rates on credit cards(see Figure 2) [4]. Proving that as the development of classification models continues, there will be an increase in the accuracy of credit default predictions.

This ignited a series of theories, such as the VaR method, Delta model, Monte Carlo model, Gamma model, and extreme value model applied to credit card risk identification [5] . Since then, solving this problem has been a continuous task amongst scientists. More recently, Ye et al [6] compared K-nearest neighbor classifiers(KNN), Logistic regression(LR), Discriminant analysis(DA), Naive Bayesian classifier(NB), Artificial neural networks(ANNs), and Classification trees(CTs). The six machine learning models brought new proposals for the prediction of default probability. The research results show that, among the above six data mining techniques, artificial neural networks can estimate the actual default probability more accurately, and achieve better performance in such problems. Nonetheless, Bellotti et al. [7] found that SVM is more competitive in the credit card default payment scenario by comparing SVM with traditional statistical analysis methods, and can be used as the basis for feature selection. Additionally, Bellotti et al. [8] applied discrete-time survival models to credit card borrower defaults. The study discovered models that included certain cardholder behavioral data and macroeconomic indicators improved the statistical significance of the models.

More recently, Butaru et al. [9] used machine learning techniques to predict a modified version of delinquency based on a combination of cardholder consumption data, credit status, and macroeconomic indicators from six major banks. The study found that there is large heterogeneity in risk factors and the predictability of inter-bank default. That indicates how no model is suitable for the risk prediction of the six banks, and it points out that a more personalized strategy needs to be adopted through the supervision of financial institutions. This previous research led Schonlau [10] 2020 to conduct in-depth research on the random forest model. This introduced a corresponding new command $rforest$, outlined the random forest algorithm, and explained its usage through two examples: The first is a classification problem that predicts whether credit card holders will default on their debts. The second example is a regression problem predicting the log scale of the share of online news.

After all of our research, we came to an understanding that there are thousands of years into trying to solve this current problem. Therefore, instead of reinventing a new system, we rather understand where it's lacking and build on that. In our efforts to do such, we will analyze the data of one of the first credit card companies, American Express. American Express launched a competition around five months ago where they released anonymous information about their clients. This data included D_* = Delinquency variables , S_* = Spend variables, P_* = Payment variables, B_* = Balance variables, and R_* = Risk variables. Though for anonymity purposes, they don't disclose much about the meaning of the variables, plus it was noted that the columns data had been tampered with for legal reasons. Additionally, after careful examination of the data frame, we identified that the model contained multiple NaN values.

Our first step in tackling this problem is by unmasking the columns and imputing for the missing values. One tool we will use to tackle this problem is by using the k-nearest neighbor with KNNImputer from Sklearn[5]. As defined by Sklearn KNN Imputer takes, "Each sample's missing values are imputed using the mean value from n_neighbors nearest neighbors found in the training set [11]". However, there is a drawback to KNN. KNN is very effective for small data but starts slowing down as the sample sizes enlarge. In our instance, our data set is large, but we still want to try it.

Furthermore, like Ye, we want to try different classification methods to compare our results to better understand the data. The first model we plan on implementing is a decision tree, as it's a simple yet robust tactic to get a sense of how our dataset is performing. Secondly, the use of a gradient-boosting machine will help us understand where the model can improve. Though there are several forms of gradient boosting, we opted to use extreme gradient boosting. Extreme gradient boosting is more efficient on bigger datasets while returning almost the same quality of results as more time-complex methods like extreme gradient boosting. Subsequently, this is the same reason we believe neural networks are also a satisfactory fit for this project. This is backed by neural networks working efficiently with large amounts of data. Therefore, making it our third model of choice. After all of the data has been processed, it's empirical to understand what variables affect the model's decision the most. Therefore extracting this information from the lead model has to be done. Additionally, the groups five groups of data provided will be ranked by most contributed to least. Lastly, we would like to observe if our model resembles the FICO score in any matter by using the information of the group weights and plotting it to see if we can resemble Figure 1.

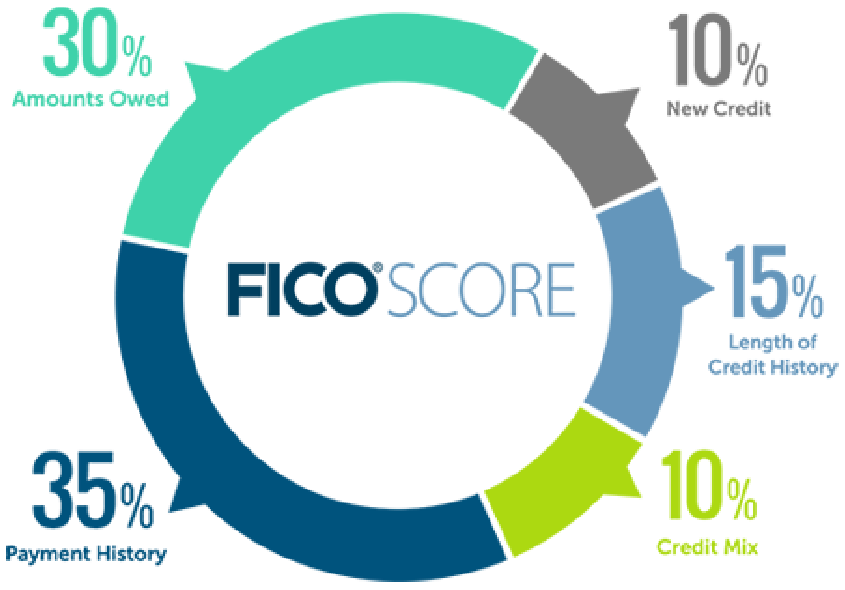

Figure 1: Weights accounted per variable in FICO [12]

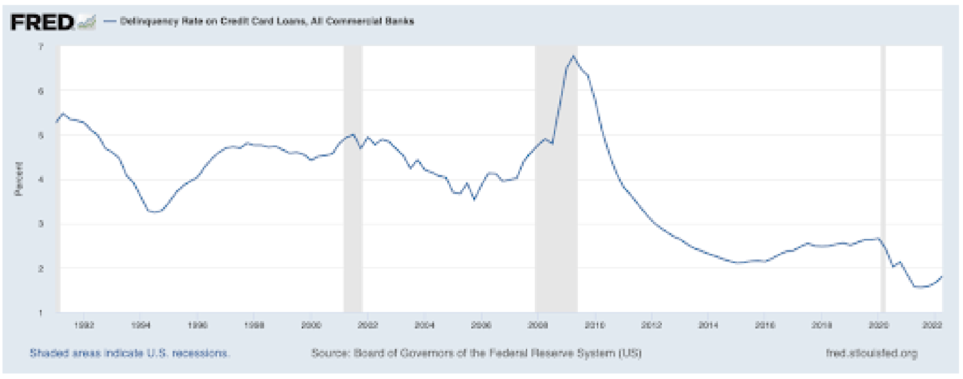

Figure 2: The historical rates of credit delinquency

3 Solution

3.1 Unmasking the data

Handling the data was the first step in our problem's solution. Therefore, feature engineering became the top priority- as good data ultimately leads to good results. Due to that, our dataset was heavily masked; it posed a real challenge to decipher what each column meant. For example, the male and female variables often get translated into 0 and 1, but due to privacy protection, American Express added a random decimal at the end, changing the entire meaning of the variable. That being said, understanding which columns posed a threat of masking misinformation became crucial to obtain better model accuracy. Just like Sherlock Holmes, we embarked on finding out what that was, which was almost indecipherable. The dataset contained 458,913 rows which translated to customer transactions, where the maximum amount of transactions per customer was 13, and the minimum was 2. Additionally, these transactions were described by 188 columns, which were converted into 95 of type $np.int8$ or $np.int16$, and the remaining were of type 93 $np.float32$. Consequently, concluding that if a column was of type float, it meant that it had been masked. Furthermore, the columns ['B_30', 'B_38', 'D_114', 'D_116', 'D_117', 'D_120', 'D_126', 'D_63', 'D_64', 'D_66', 'D_68'] were composed of categorical data. Therefore a quantitative translation was necessary (these columns were included in the 188 mentioned earlier). Plenty of other data scientists also embarked on this arduous endeavor to decipher the variables, but the user Radar could unmask the undecipherable. He correctly addressed the missing information and was kind enough to share his results. Therefore, most of our feature engineering is based on his findings [13]. Additionally, the data that Radar provided was already normalized; hence, no step to do such was required.

3.2 Imputing for Missing Values

Even after the unmasking, our database still possessed thousands of missing data. This resulted in removing columns that were composed of 30% or more missing data, as their contents could expose our model to biased estimates. After the deletion of such columns, plenty of missing values remained, and they had to be dealt with accordingly. In such attempts, we opted to use KNNImputer, as it would compare the missing data with its neighbors, arriving at a more realistic prediction. However, there are some negative factors to using this approach. One factor is computation time of the predictions takes a very long time, and another factor is the accuracy of the predictions is highly dependent on the quality of the data – which we know was not the greatest. Furthermore, our dataset still contains multiple customer transactions. Therefore, to fully process the transactional data, it's essential to extract the 'mean', 'std', 'min', 'max', and 'last' of each.

3.3 Understanding the Data

Following the completion of the dataset cleaning, it was crucial to understand what was going to be predicted. Hence, a deconstruction of the "target" column revealed that the non-defaults accounted for roughly 75% of the dataset, while the defaults accounted for the remaining 25% [Figure 3]. This revealed that the dataset provided was oversampling the number of defaults as the average amount of defaults in credit cards is generally low at 2%. But, 2% of the target is quite a low number of rows to train the models. Therefore, we opted to keep it as it is.

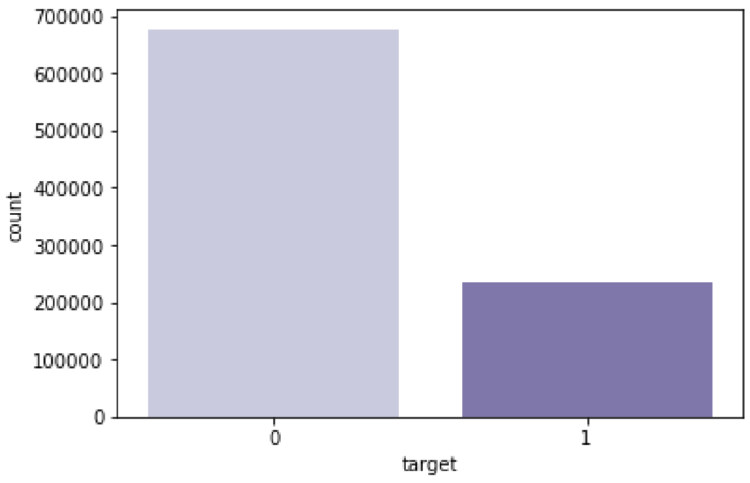

Figure 3: Graphing the number of defaults(1) and non-defaults(0)

3.4 Decision Tree Classifier

After succeeding in the cleaning and analysis of the database, it was time to process our findings through the selected classification models. This step will be crucial in deciphering how much of our target feature we can predict. Firstly, a decision tree was used as it allowed us to create nodes that represent a feature, a branch that represents the rules applied to the feature, and a leaf that represents the outcome. A clearer representation can be observed in Figure 4 [14]. Furthermore, this model was chosen due to its high efficiency in mimicking a real-life human thinking process. Therefore, making the model easily understandable hence creating a solid foundation for our future steps. Before running the model, it is empirical to understand the metrics used to calculate its accuracy, precision, and recall. Accuracy is the percentage of all correctly classified examples in our test set. The precision score is the percentage of predicted defaults that we classify correctly. The recall score is the percentage of defaults that we classify correctly. Figure 5 [15] shows clarification on how these values were calculated and their relationship with the confusion matrix. Thus, the data extracted from the decision tree can be observed in the confusion matrix of Figure 6. The results extracted demonstrated that the model had an accuracy of 84.4%, a recall of 69.7%, and a precision of 70%. This tells us that the model was able to correctly calculate 70% of the defaults in the target. Additionally, it was crucial to plot the ROC curve and then extract the area under the curve(AUC) to understand how the true positive rate(recall) behaved when compared to the false positive rate. The false positive rate is calculated by dividing the false positives by the sum of false positives and true negatives. After doing such calculations, Figure 7 was plotted, demonstrating that the model had an area under the curve of .8. This .8 demonstrates that the model could capture 80% of correct predictions.

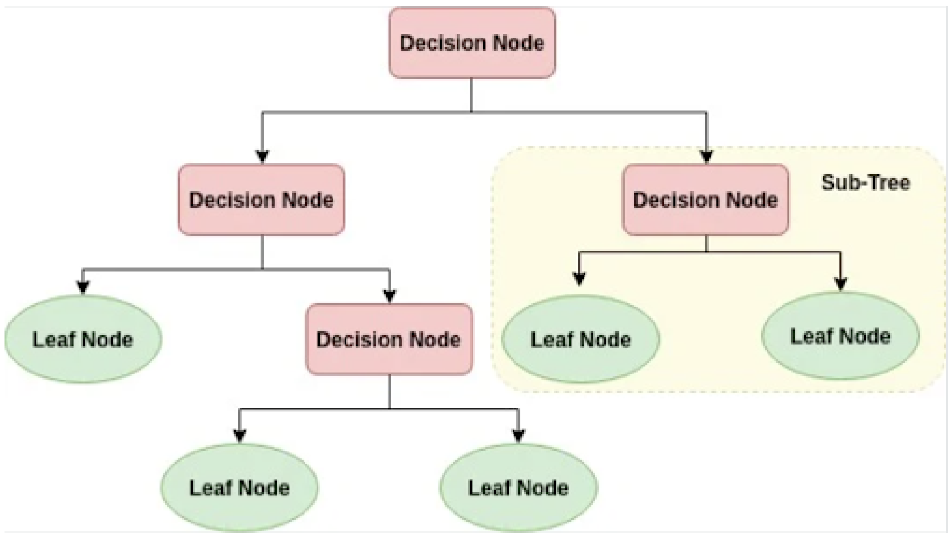

Figure 4: Decision Tree Classification Flow

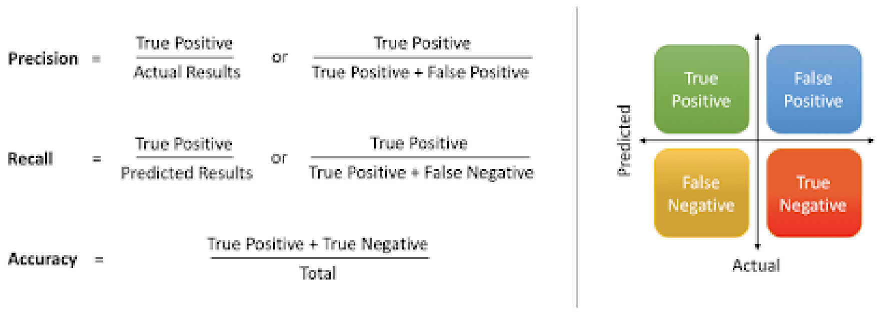
Precision = True Positive / Actual Results or True Positive / True Positive + False Positive
Recall = True Positive / Predicted Results or True Positive / True Positive + False Negative
Accuracy = True Positive + True Negative / Total
True Positive
False Positive
False Negative
True Negative
Predicted
Actual

Figure 5: The relationship between Accuracy, Recall, and Precision with the Confusion Matrix

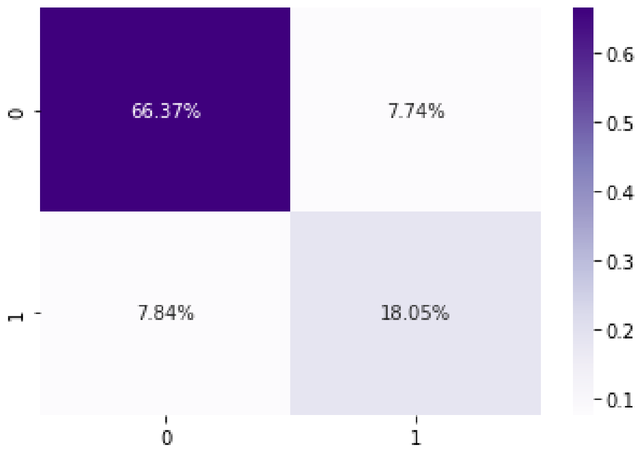

Figure 6: Confusion Matrix of Decision Tree Classifier

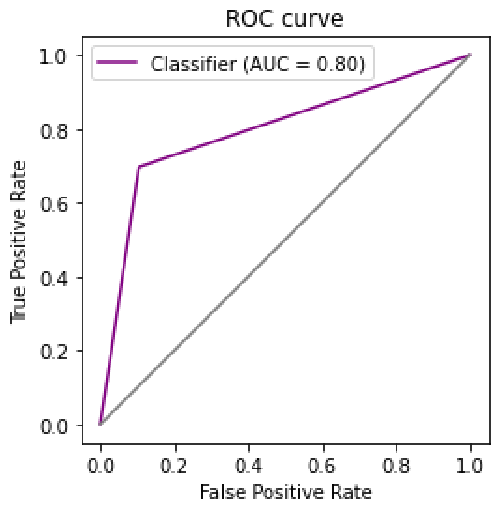

Figure 7: AUC of Decision Tree Classifier

3.5 XGBoost

Secondly, after creating a base model, it was crucial to test new methods to observe how the database reacted in different environments. Therefore, the second model tested was the XGBoost classifier. XGBoost can be seen as the extension of the decision tree's model Figure 8 [16] gives a very accurate, yet humorous, way of evolution. This model was due to many of its outstanding features, such as parallelized tree building, tree pruning, cache awareness, regularization to avoid overfitting, efficient handling of missing data, and in-built cross-validation capability [17]. These features allow the model to learn in a quick and "extremely" efficient fashion. The confusion matrix from processing the results of the model can be observed in Figure 9. From the results received, the accuracy of the model resulted in 89.7%, the recall 79.4%, and the precision 80.6%. Jumping by almost 10% in each category from the previous model. Additionally, the AUC of the model also increased to a value of .864, meaning that XGBoosting could predict 6.4% more accurately than a decision tree. Figure 10 shows the graph for the AUC curve.

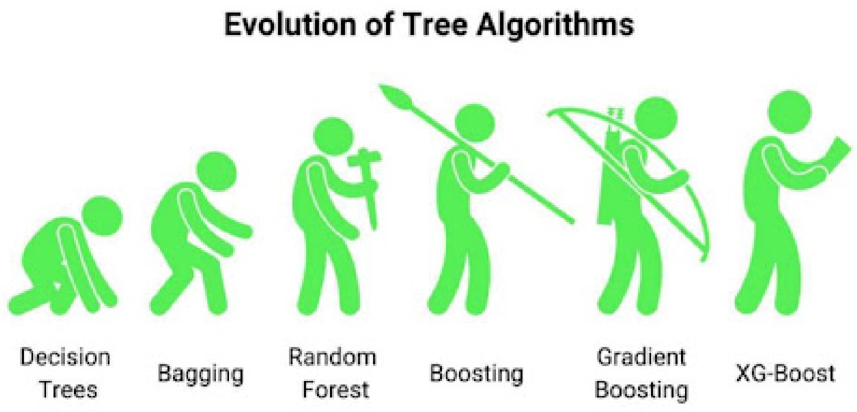

Figure 8: Evolution of Tree Algorithms

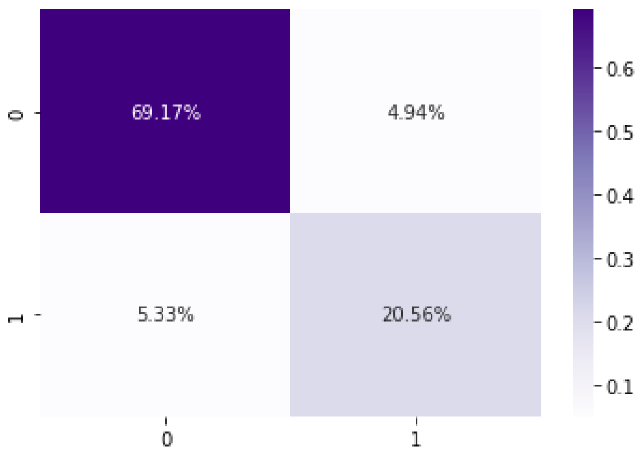

Figure 9: Confusion Matrix of XGBoost

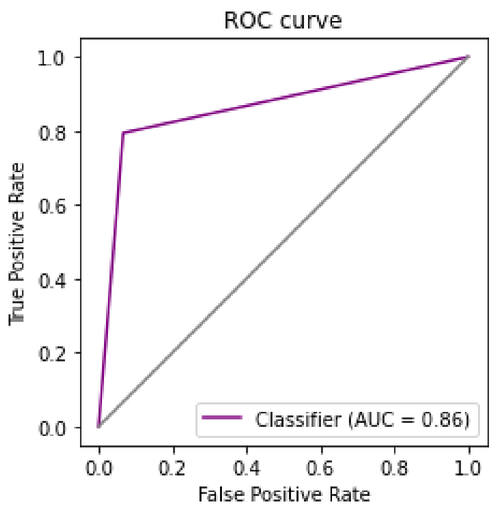

Figure 10: AUC of XGBoost

3.6 Neural Network

Lastly, the third model used was a neural network. This model was chosen in order to compare the efficiency of multi-layer perceptron(MLP) models against tree-based models. In order to formulate a conclusive argument, it's essential to understand what is a neural network. Neural networks are computing systems inspired by the biological neural network that constitutes the human brain. This interconnected network is based on a collection of nodes and connections, as shown in Figure 11 [18]. The nodes resemble the neurons in the brain, and the connections which transmit the information between nodes resemble the brain's synapses. Furthermore, perceptron models can learn weight coefficients which inevitably helps the models classify inputs. For this reason, MLPs are highly efficient when it comes to arranging and categorizing inputs, which will become a humongous advantage in predicting who will default. In order to keep our model computationally simple, it was decided to use only one hidden layer, and even by implementing this parameter, the calculation took over two hours, mainly because of the size of the data. Therefore, the results of this model included an accuracy of 88.4%, a recall of 77.8%, and a precision of 77.4%. The confusion matrix for the calculations can be observed in Figure 12. These results are similar to the XGBoost, but still not quite as accurate or efficient. Additionally, the AUC of the model was .849, signifying that almost 85% of the data was correctly predicted. Figure 13 demonstrates the plotting of the model's AUC.

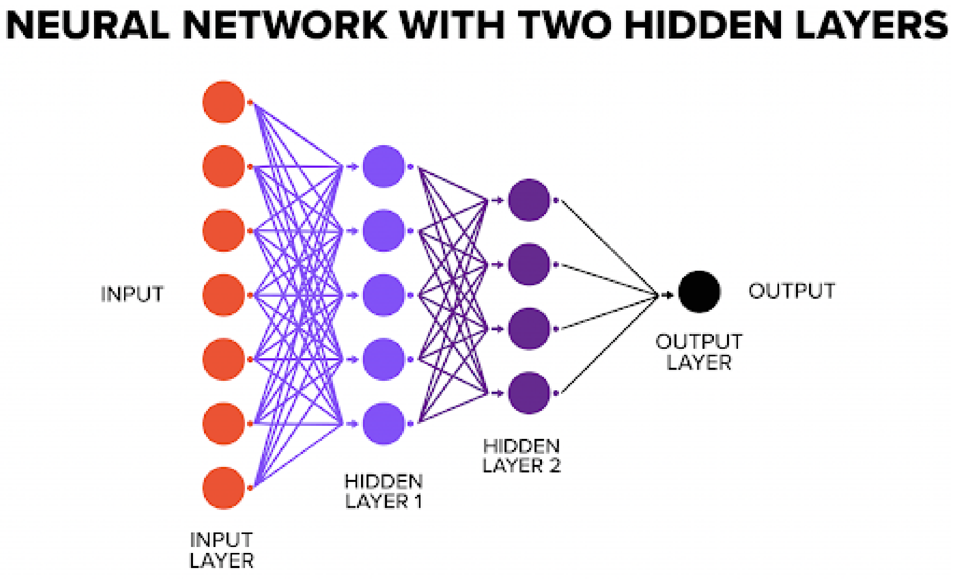

Figure 11: Multi-Layer Perceptron Model Visualization

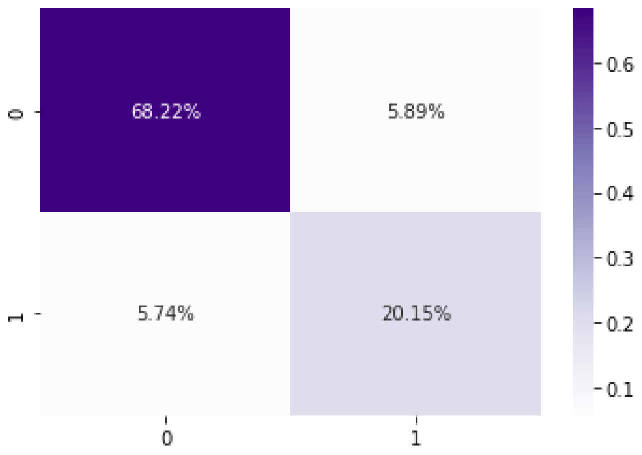

Figure 12: Confusion Matrix of MLP

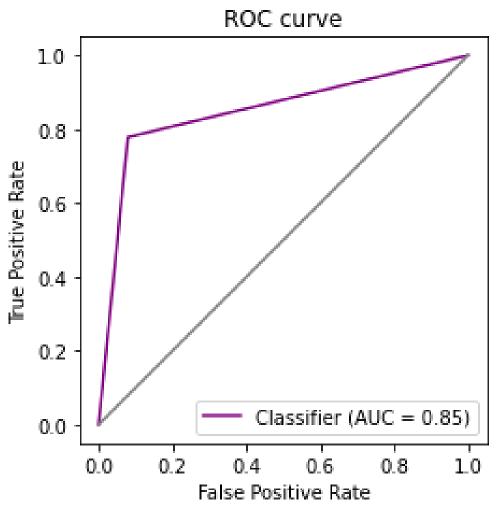

Figure 13: AUC of MLP

3.7 Feature Importance

Last but not least, the calculations proved that the XGBoost returns the most accurate and fastest results. Additionally, it's crucial to understand what variable was the most important in the predictions. Therefore, by extracting the individual results of each variable, it was concluded that 'P_2_Last' was the most important feature (Figure 14). On the contrary, when observing the aggregate group importance of the D = Delinquency variables, S = Spend variables, P = Payment variables, B = Balance variables, and R = Risk variables, it was calculated that the P group had the least contribution. The total feature group importance contribution was composed of 51% in group D, 21% in group B, 15% in group R, 11% in group S, and 2% in group P. For a clearer view, please refer to Figure 15. It is important to note that not all groups possess the same amount of variables. Therefore, group D, which has more variables than P, will return a higher percentage of the contribution.

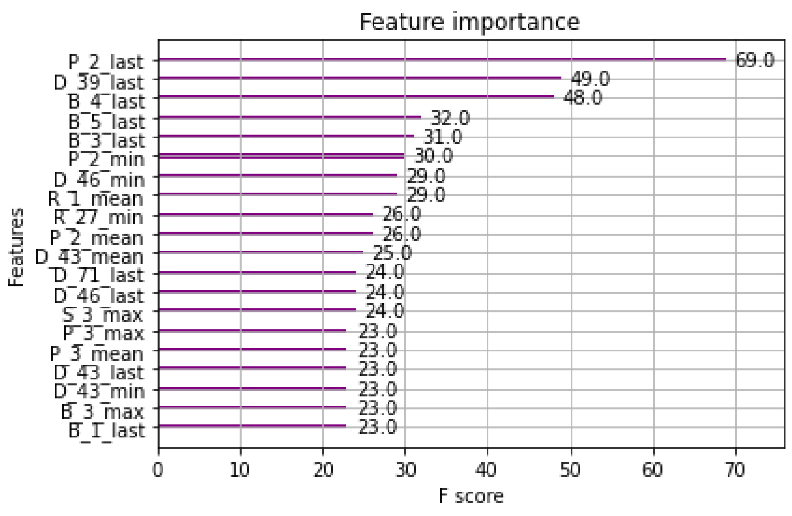

Figure 14: Feature Importance Calculation

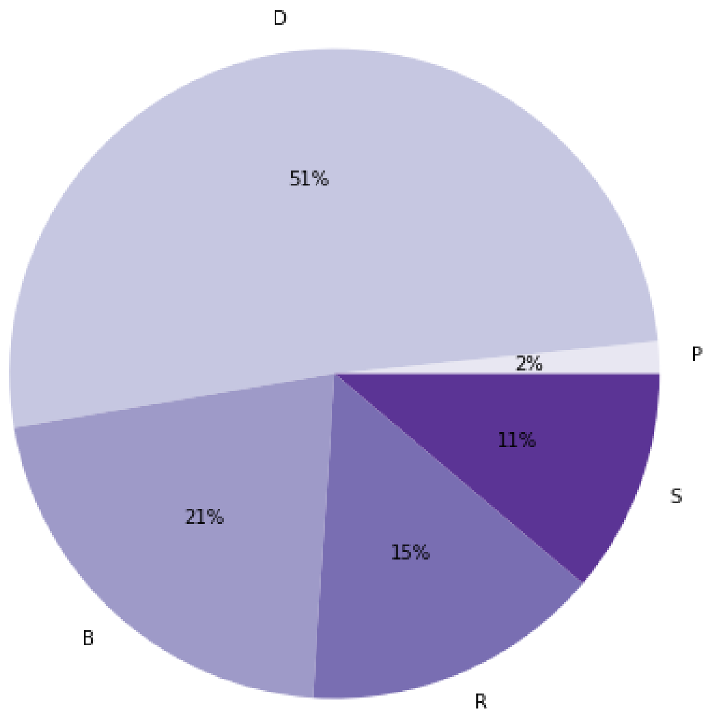

Figure 15: Aggregate group Feature Importance

4 Results and Discussion

4.1 Experiments Results

In the end, XGBoost returned the best result with an AUC of .86. Correctly predicting 86% of the true positives and negatives. Though multi-layered perceptrons did not fall behind with an outstanding score of .85. Therefore, tree-based models are more adequate when predicting data composed of heterogeneous columns. However, it does not mean that multi-layered perceptrons are inadequate models. If tuned to the most effective epoch and the most accurate weights, then it can be a really powerful tool. Some of the top five solutions for the competition included neural networks, but the winners all included either XGBoost or LGBoost.

Additionally, we can observe that the distribution of the 5 groups in figure 14 sort of resembles the FICO score distribution presented at the beginning of the paper. As they both have five categories, and their category weights are sort of similar. This might suggest that the FICO score is indeed an outstanding metric of evaluation, and that future credit delinquency models should follow its weights carefully.

4.2 Main challenges

Completing this project was no easy task. By far, the most complicated aspect was deciphering how to unmask the data. In addition to the unmasking of the data, the file was composed of 458 thousand rows and 190 columns, making computations take ages in the desktop. Though the insights from Radar helped enormously, dealing with value types and NaNs. Additionally, we were not really familiar with KNNImputer. That being said, it took quite some time to decipher what organized rows return better imputations. If the columns weren't in order, the values imputed would become misleading and destroy all of our previous efforts. Additionally, it is empirical we mention the possibility of overfitting, and though we tried to mitigate this possibility by choosing the correct model and techniques, there is always the possibility of such. If we had to start the project again from zero, we would dedicate more time to generating the best possible dataset to work with. Before Using the help from Radar, my models averaged an accuracy of 50%. However, after using the unmasked versions of the dataset, my accuracy increased almost to 90% on some models. Therefore, proving that the quality of the input makes the entirety of difference.

5 Conclusion

From the study conducted, it's proven that credit card delinquency can be predicted. However, we are still far away from a model that has 100% certainty of if a customer will default on its debt. By far, it's been proven that the data imputed is the most important factor when calculating predictions. Additionally, XGBoost proved to be extremely efficient and precise when it came to processing the data. Furthermore, the multi-layered perceptron also proved to be quite effective, falling a little bit short in terms of accuracy behind XGBoost. Moreover, when dissecting the results of the XGBoost it was found that the last payment done by a customer was the variable that contributed the most towards the predictions. Therefore, checking for immediate spikes in credit usage is critical to prevent credit delinquency.

There are several aspects of this project, that if given more time, could have been more fine-tuned. One of the most important aspects would be dissecting if the oversampling of the model negatively affected the results. As mentioned previously, the sample set was conformed of 75% non-delinquent statements and 25% delinquent statements. In reality, the distribution of the data should be closer to 98% non-delinquent and 2% delinquent. Another aspect that should be considered is using a random forest to check how multiple trees come to a prediction and compare that to the XGBoost. My guess would be that the result would lie between that of the XGBoost and the decision tree. In terms of the multi-layered perceptron, the model was very basic and therefore, there is a lot of room for improvement. The first approach to create a better network would be to calculate the optimal number of epochs by brute force with a variance in hidden layers adjusting to the requirements.

References

[1] "A brief history-and future-of credit scores." [Online]. Available: https://www.economist.com/international/2019/07/06/a-brief-history-and-future-of-credit-scores

[2] Rachel, "Cuneiform tablets reveal secrets of mesopotamian payroll," Sep 2021. [Online]. Available: https://rethinkq.adp.com/cuneiform-tablets-secrets-mesopotamian-payroll/

[3] R. S. Frankel, "When were credit cards invented: The history of credit cards," Sep 2022. [Online]. Available: https://www.forbes.com/advisor/credit-cards/history-of-credit-cards/

[4] "Delinquency rate on credit card loans, all commercial banks," Nov 2022. [Online]. Available: https://fred.stlouisfed.org/series/DRCCLACBS

[5] F. Derrien, A. Kecskés, and S. A. Mansi, "Information asymmetry, the cost of debt, and credit events: Evidence from quasi-random analyst disappearances," *Journal of Corporate Finance*, vol. 39, p. 295–311, 2016.

[6] I.-C. Yeh and C.-h. Lien, "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients," *Expert Systems with Applications*, vol. 36, no. 2, p. 2473–2480, 2009.

[7] T. Bellotti and J. Crook, "Support vector machines for credit scoring and discovery of significant features," *Expert Systems with Applications*, vol. 36, no. 2, p. 3302–3308, 2009.

[8] ——, "Forecasting and stress testing credit card default using dynamic models," *International Journal of Forecasting*, vol. 29, no. 4, p. 563–574, 2013.

[9] F. Butaru, Q. Chen, B. Clark, S. Das, A. W. Lo, and A. Siddique, "Risk and risk management in the credit card industry," *Journal of Banking amp; Finance*, vol. 72, p. 218–239, 2016.

[10] M. Schonlau and R. Y. Zou, "The random forest algorithm for statistical learning," *The Stata Journal: Promoting communications on statistics and Stata*, vol. 20, no. 1, p. 3–29, 2020.

[11] "Sklearn.impute.knnimputer." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html

[12] "How are fico scores calculated?" Feb 2022. [Online]. Available: https://www.myfico.com/credit-education/whats-in-your-credit-score

[13] R. Raddar, "Amex data int types - train," Jun 2022. [Online]. Available: https://www.kaggle.com/code/raddar/amex-data-int-types-train

[14] A. Navlani, "Python decision tree classification tutorial: Scikit-learn decision-treeclassifier," Dec 2018. [Online]. Available: https://www.datacamp.com/tutorial/decision-tree-classification-python

[15] Richard, "Contingency table. true positive, false positive, false negative, and ..." [Online]. Available: https://www.researchgate.net/figure/Contingency-table-True-Positive-False-Positive-False-Negative-and-True-Negatives-are_fig5_280535795

[16] R. Raj, "Xg-boost (extreme gradient boosting) algorithm in ml." [Online]. Available: https://www.enjoyalgorithms.com/blog/xg-boost-algorithm-in-ml

[17] V. Morde, "Xgboost algorithm: Long may she reign!" Apr 2019. [Online]. Available: https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

[18] M. Grindle, “Deep learning vs neural network: What’s the difference?” Nov 2020. [Online]. Available: https://smartboost.com/blog/deep-learning-vs-neural-network/