Credit Default Prediction for Bank XYZ using Machine Learning Models:

A Model You Can Take to the Bank

April 15, 2021

The Data Dudes:

Orr Shalev, Antoine Nadaud, Noam Kleinman, Jacob Salomon

Abstract

Based on data collected by bank XYZ, we developed two machine learning models to determine which applicants should be approved for a line of credit using predicted default outcome scores. After fitting, testing, and comparing a logistic regression model and random forest model, the evidence showed that our random forest model was the most effective at helping determine applicant approval to bank XYZ due to high predictive capabilities, more favorable error distribution, resilience to noisy conditions, and sufficient interpretability. Based on our findings, we believe that an implementation/deployment of our random forest model is likely to yield high value for bank XYZ through loss mitigation.

I. Introduction

Financial institutions offer various credit products to consumers and other firms. However, each loan/credit instance requires an application through which the institution gathers data reflecting a variety of factors about the applicant. The data an institution gathers often helps predict important features to take into consideration in the approval process, such as defaulting. Ideally, the use of a predictive model would analyze information about an applicant and tell the bank whether it should grant the loan/credit. In the models our team developed, we focused on the likelihood of the applicant defaulting on the loan or being delinquent for three consecutive months.

We were tasked with developing a machine learning (ML) model based on historical data for a hypothetical bank named bank XYZ. After development, we were to use our model to determine whether or not an account should receive credit based on whether the account was predicted to default or be delinquent for three consecutive months. All information was simulated but treated as real for the purpose of the model.

To develop our model, we explored the resources that were available to the team and determined that logistic regression (LR), random forest (RF), and gradient boosting (GB) had the greatest potential to provide the appropriate results. The team was skeptical about the feedforward neural network (FNN) in this particular application, as it would be difficult to pierce the "black box" nature of a neural network and understand the process underlying the decision made. Because we are representing a bank and transparency to our clients is important to us, we chose to use our aforementioned ML methods.

Our background research showed that there were two Kaggle competitions with very similar premises (UCI Machine Learning, 2016) (Seanny, 2019). The competitions included additional variables such as education, credit scores (such as FICO), and other important financial indicators. Seanny, a user in the second Kaggle competition, was able to build highly accurate predictive models for credit approval using Random Forest and XGBoost, which suggests that our model could be built just as strongly (Seanny, 2020). We also engaged with background domain knowledge of credit services and application processes. The team identified common features in application decisions and compared them to our datasets to discover which of our indicators might be useful to improve model performance. In the event of a direct interaction with bank XYZ (not a hypothetical bank), we would suggest strategies to acquire these additional financial indicators as well to further strengthen our model. Without the potentially useful features that were seen above, we anticipated some noise in our models. This would be problematic for GB, as the data could more easily result in overfitting (Ravashad, 2018). Due to the foreseen limitations with GB and FNN, we chose RF as our second ML model to use.

II. Exploratory Data Analysis

For exploratory data analysis, we first wanted to have rudimentary information about the training dataset, such as shape, data types, and summary statistics to gain a basic understanding of the data we were working with. Each of these three datasets were congruent with the reports in the competition appendix. Upon further probing, the string type variable "States" was the only variable that was not a float

type. The variables of type float were measured on different scales, such as dollar amount, proportion, binary 0/1, years, and quantity of action.

Continuing our exploration of the data, we found that *uti_card_50plus_pct* and *rep_income* contain 2055 and 1570 missing data values respectively. This discrepancy would need to be addressed, but could wait until we began processing the data. We created a table to show key statistics about the training dataset.

Summary Statistics for Training Dataset: Table 1

	tot_credit_debt	**avg_card_debt**	**credit_age**	**credit_good_age**	**card_age**
Mean	94563.70	14088.23	296.69	149.77	268.01
Std Dev	23546.44	9314.49	61.71	34.01	59.36

	non_mtg_12	**non_mtg_6**	**mtg_6**	**credit_past_due**	**inq_12**
Mean	0.11	0.02	0.03	329.287	1.76
Std Dev	0.43	0.17	0.17	2073.89	1.74

	inq_24	**card_36**	**auto_36**	**uti_card**	**uti_50plus_pct**
Mean	3.40	0.16	0.14	0.5031	0.5110
Std Dev	2.92	0.38	0.34	0.1093	0.1134

	uti_max	**uti_card_50**	**ind_XYZ**	**rep_income**	**Default_ind**
Mean	0.5076	0.4895	0.2585	75499.51	0.0793
Std Dev	0.1086	0.1197	0.4378	16361.95	0.2702

There are a few noteworthy observations in **Table 1**, such as a mean reported income of $75,500, which is considerably higher than the 2020 annual mean wage of $56,310 (U.S. BLS, 2021). This distortion was a partial motivation to remove *rep_income*. We were initially unsure if the *rep_income* was a reliable feature to use, since past evidence has suggested unreliability in self-reported income, as was the case when the Federal Reserve found that self-employed individuals in the U.S. substantially underreport income in their tax forms (Hurst & Pugsley, 2011). Another interesting point: slightly over a quarter of the applicants already have an account with bank XYZ prior to application. An ML model that negatively impacts these customers would surely create a rift in public relations with current bank XYZ customers and subsequently may cause loss of business.

Continuing to search for evidence to use in variable selection, we noticed that every utilization variable has a very similar mean (around 50%) and standard deviation (around 10%), suggesting a consistency among utilization variables. This will be explored further in data processing. The standard deviation of *average_card_debt* also gave us some insight into its usefulness. It is relatively large (0.66 as a proportion of the mean), and since there is high variance in average debt accrued by applicants, it has the potential to be a good indicator of whether an applicant will default. Previous defaultings among applicants seems to play an integral role into the decision process, so our next step was to examine

summary statistics of the defaulting and non-defaulting populations separately, as that might give us insight as to where the populations differ most sharply.

Summary Statistics for Defaulting Applicants in Training Dataset: Table 2

	tot_credit_debt	avg_card_debt	credit_age	credit_good_age	card_age
Mean	91915.56	17880.42	275.16	140.53	247.70
Std Dev	29920.36	21195.05	61.98	33.55	58.60

	non_mtg_12	non_mtg_6	mtg_6	credit_past_due	inq_12
Mean	0.70	0.24	0.26	3060.92	2.28
Std Dev	1.03	0.47	0.44	5777.47	1.86

	inq_24	card_36	auto_36	uti_card	uti_50plus_pct
Mean	4.16	0.19	0.14	0.5603	0.5579
Std Dev	3.13	0.41	0.35	0.1107	0.1149

	uti_max	uti_card_50	ind_XYZ	rep_income	Default_ind
Mean	0.5511	0.5443	0.2068	74522.99	1.0
Std Dev	0.1096	0.1230	0.4051	16775.12	0.0

Summary Statistics for Non-Defaulting Applicants in Training Dataset: Table 3

	tot_credit_debt	avg_card_debt	credit_age	credit_good_age	card_age
Mean	94791.79	13761.61	298.552	150.5673	269.7642
Std Dev	22901.39	7363.255	61.33699	33.9392	59.10613

	non_mtg_12	non_mtg_6	mtg_6	credit_past_due	inq_12
Mean	0.06	0.00	0.00	94.01	1.71
Std Dev	0.28	0.09	0.09	1048.87	1.72

	inq_24	card_36	auto_36	uti_card	uti_50plus_pct
Mean	3.34	0.16	0.14	0.4982	0.5069
Std Dev	2.89	0.38	0.34	0.1078	0.1124

	uti_max	uti_card_50	ind_XYZ	rep_income	Default_ind
Mean	0.5038	0.4849	0.2630	75583.34	0.0
Std Dev	0.1077	0.1182	0.4402	16323.78	0.0

Through analysis of **Table 2** and **Table 3**, we were able to extrapolate that the average card debt, credit past due amount, and utilization variables had much higher mean values in the defaulting population than the non-defaulting population. This indicated that these variables may be highly predictive of defaulting. For the variables average card debt and credit past due amount, the defaulting population had a much greater standard deviation, suggesting high variance in the defaulting population. Finally, reported income stats for both populations was very similar, another strike against keeping *rep_income*.

A concern our team had with the data is that there were not enough observations available in the training dataset to make a reliable model. While dataset size is less important for RF and GB, it can severely limit LR and FNN models. Thus, we conducted a rule of ten test. This gave a conservative estimate for how many observations are necessary for a successful regression analysis given probability of a predicted event and number of explanatory variables. We conducted the test as follows:

If $n < \frac{10k}{p}$, there are not enough observations and too many explanatory variables.

- k = explanatory variables = 20
- p = probability that account would default = 0.0793
- n = number of observations = 20,0000

Plugging in the values, we found that the number of required observations is approximately 2522. We have 20,000, much more than that, so the data passes the rule of ten test, and therefore there is an appropriate number of observations for an LR model.

Another concern was that if the prediction variables are too collinear, those features would be weighted too heavily in the LR model. To see how collinear the prediction variables were, we calculated the variance inflation factor (VIF) score of all appropriate prediction variables, which represents the degree of multicollinearity. We also created a correlation matrix with all variables, including the response variable, to get a closer look at how all the variables interact. In order to include all variables in the correlation matrix, we created dummy variables for the states.

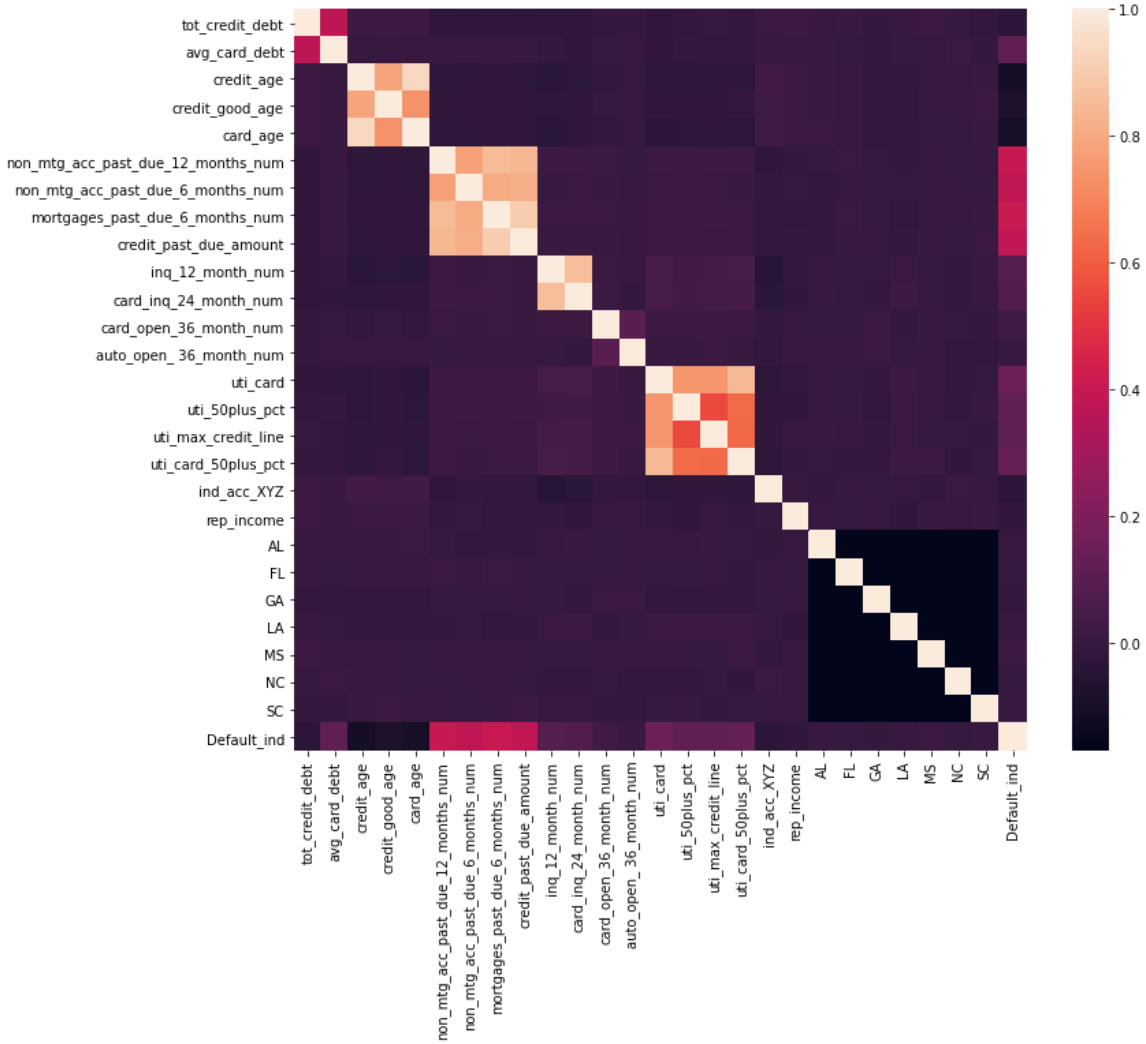
VIF Table Showing Multicollinearity: Table 4

Variables	**VIF**
tot_credit_debt	16.22
avg_card_debt	3.83
credit_age	232.52
credit_good_age	52.88
card_age	174.50
non_mtg_acc_past_due_12_months_num	4.61
non_mtg_acc_past_due_6_months_num	3.46
mortgages_past_due_6_months_num	6.67
credit_past_due_amount	6.57

inq_12_month_num	7.76
card_inq_24_month_num	9.00
card_open_36_month_num	1.18
auto_open_ 36_month_num	1.17
uti_card	78.50
uti_50plus_pct	46.47
uti_max_credit_line	49.13
ind_acc_XYZ	1.34

Table 4 shows VIF values for all explanatory variables. VIF values that are greater than ten are unacceptable for LR models. VIF values between five and ten are also possibly detrimental. With this in mind, we understood that a large amount of feature engineering needed to be done to reduce VIF.

Correlation Map Showing Collinearity: Figure 1



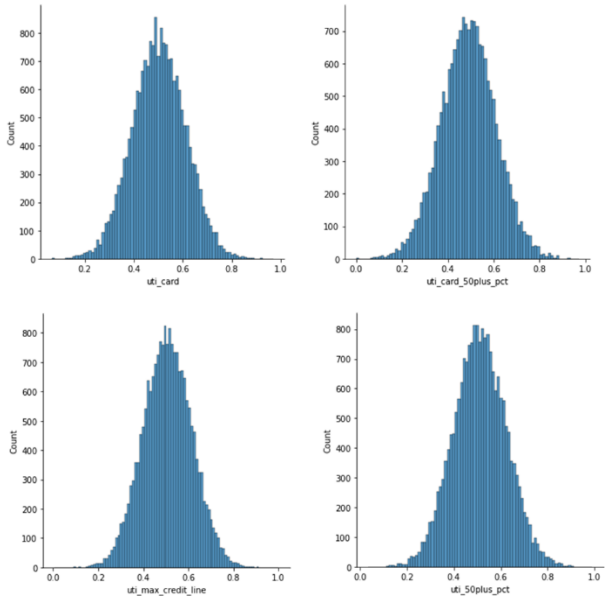
Based on **Figure 1**, the variables *tot_credit_debt*, *avg_card_debt*, *credit_age*, *credit_good_age*, and *card_age* have high levels of collinearity. The latter three variables have almost no correlation to the response variable so we removed them for the LR. There was no easy way to preserve *tot_credit_debt* and *avg_credit_debt* while removing collinearity, so we decided the best option was to remove *tot_credit_debt* for the LR as well.

The variables *non_mtg_acc_past_due_12_months_num*, *non_mtg_acc_past_due_6_months_num*, *mortgages_past_due_6_months_num*, and *credit_past_due_amount* have the highest correlation with the response variable but also have high collinearity with each other. This created a dilemma as removing these variables may be detrimental, as they could be highly predictive of the response variable. The team decided to combine *non_mtg _acc_past_due_6_months_num* and *mortgages_past_due_6_months_num* by adding them together into a combined variable because they were measured by the same metric and in the same time frame.The variables *inq_12_month_num* and *card_inq_24_month_num* also have high collinearity. Values in the former are also recorded in the latter, so we decided to drop *inq_12_month_num* for the LR model.

Reported income does not have a high correlation with the response variable. At this point, all the evidence suggested the reported income was unsuitable to be used in the analysis. After we removed *rep_income*, the only variable with missing values was *uti_card_50plus_pct*. However, it seems this variable may be worth keeping due to its relatively high linearity with the response variable.

An important note is that utilization variables have high collinearity and relatively high correlation with the response variable. To deal with missing values while reducing collinearity, we determined the best course of action was to combine all utilization variables into a mean utilization variable which included *uti_card_50plus_pct* only if available for the observation. We already knew that the utilization variables were highly collinear, but we also wanted to make sure that the mean values of utilization was a sufficient metric, so we found the distribution of utilization variables using a visualization, as can be seen in **Figure 2**. With the distribution of utilization variables being approximately normal, we felt confident in our approach.

Distribution of Utilization Variables in Training Dataset: Figure 2



III. Logistic Regression

Data Preprocessing

- Removing the variables *tot_credit_debt, credit_age, credit_good_age, card_age, rep_income, and inq_12_month_num*.
- Creating dummy variables for states.
- Combining utilization variables into mean utilization.
- Combining *non_mtg _acc_past_due_6_months_num* and *mortgages_past_due_6_months_num* by addition.

We created an additional VIF chart and correlation matrix to show what progress was made. The correlation matrix in **Figure A** (see appendix) shows that while *past_due_6_months_num*, *non_mtg_acc_past_due_12_months_num*, and *credit_past_due_amount* have high collinearity with each other, the other variables do not. Based on the high correlation of these variables with the response variable, as well as the low VIF scores seen in **Figure B**, we decided it was appropriate to keep the selected variables.

The final step we took to preprocess the data in preparation of LR was to normalize the data using the following formula:

$$n = \frac{n-min}{max-min} * 20$$

- n = feature value for given observation
- min = feature min
- max = feature max

Model Building

After quickly preprocessing the test and validation datasets the same way we did for the training dataset, we then used the validation dataset for hyperparameter tuning using the following hyperparameters:

- Solvers: newtown conjugate gradient, limited memory bfgs, and liblinear.
- Penalty: L2 (the only penalty that works for all three solvers).
- C values: 100, 10, 1.0, 0.1, 0.01.

Based on the results (a snippet of the entire test can be seen in **Figure C**), the hyperparameters we chose were liblinear as our solver with a C value of 0.1 and the same penalty of L2.

With our hyperparameters identified, we proceeded to train our LR model based on these hyperparameters and using the preprocessed training dataset. We scored our model based on accuracy of prediction in the test dataset, with an accuracy score of 93.58%.

IV. Random Forest

Data Preprocessing

- Removing the variables *credit_age*, *credit_good_age*, and *rep_income* due to very low correlation with the response variables.
- Creating dummy variables for states.
- Imputation of *uti_card_50plus_pct* missing values based on mean strategy with other utilization variables used as predictors due to their high collinearity.

Model Building

We preprocessed the test and validation datasets in the same fashion that we did the training dataset. We then used the validation dataset for hyperparameter tuning using the following hyperparameters:

- Number of trees in the forest: 10, 100, 1000.
- Maximum number of features to consider when looking for best split: square root, $\log_2$.

Based on the results (a snippet of the entire test can be seen in **Figure D**), the hyperparameters that worked best were 100 trees in our forest and $\log_2$ as the maximum number of features when splitting.

We proceeded to train our RF model based on these hyperparameters and using the preprocessed training dataset. We scored our model based on accuracy of prediction in the test dataset, with an accuracy score of 93.66%.

V. Model Comparison

Model accuracy was very similar: 93.58% for LR and 93.66% for RF. It seems that model choice will have to be based upon different aspects since both models are highly accurate for the test dataset. To inform our decision, the team created confusion matrices for both models to show the frequency of correct prediction versus errors.

LR and RF Confusion Matrices: Table 5

Logistic Regression

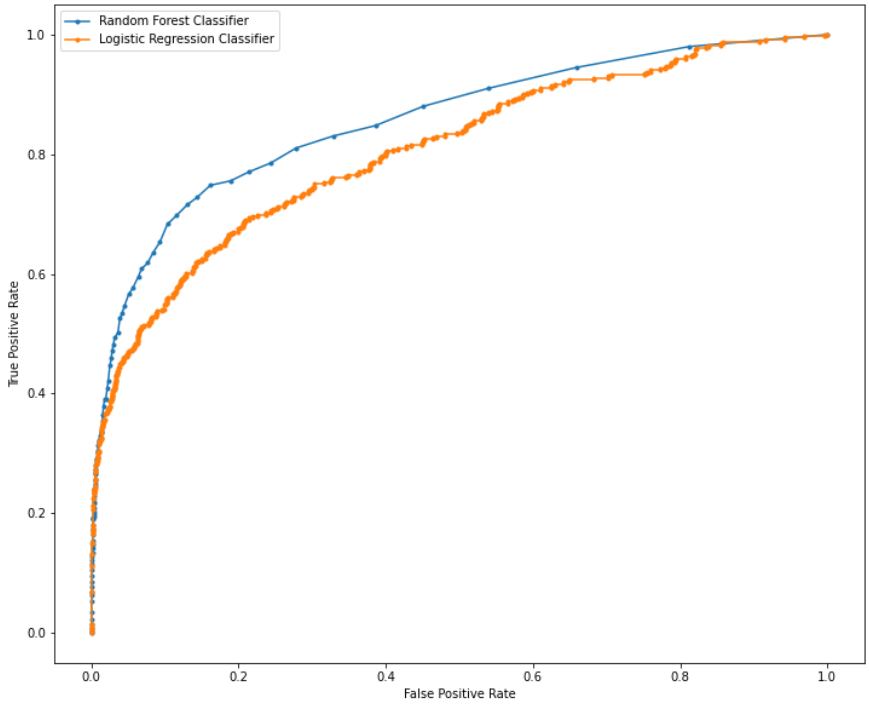
Predicted Values \ Actual Values	Non-default (0)	Default (1)
Non-default (0)	TP 4552 0.9104	FP 47 0.0094
Default (1)	FN 274 0.0548	TN 127 0.0254

Random Forest

Predicted Values \ Actual Values	Non-default (0)	Default (1)
Non-default (0)	TP 4568 0.9136	FP 31 0.0062
Default (1)	FN 286 0.0572	TN 115 0.0230

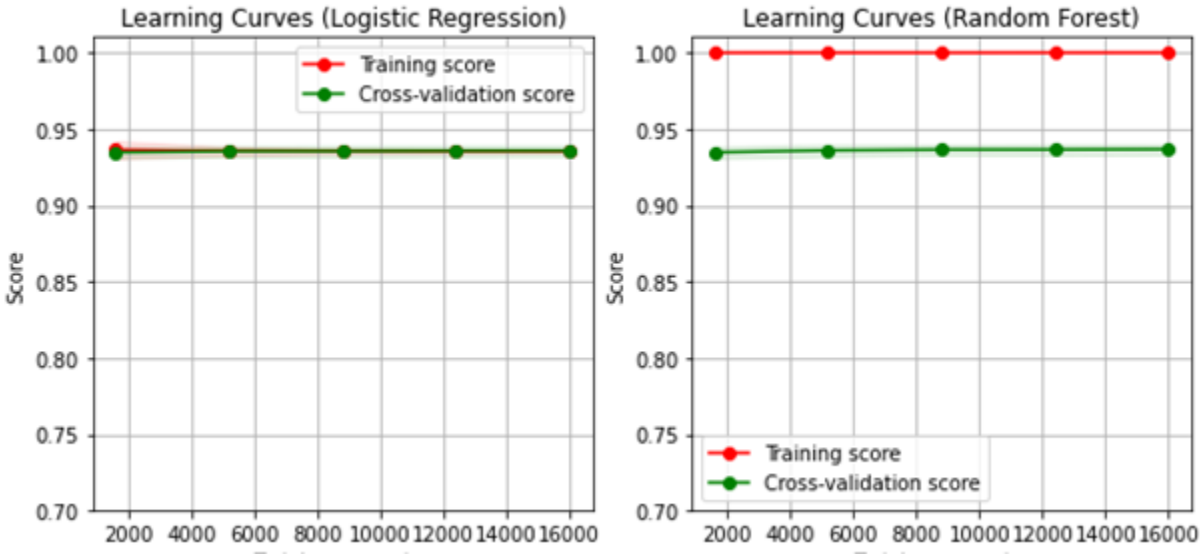
Table 5 showed that both models could be described as relatively "cautious", as the rate of false negatives was much greater than the rate of false positives. Specifically, LR had the upper hand for classifying false negatives and avoiding true negatives, while RF better classified true positives and avoided false positives. Both the LR and RF models predicted 401 (8.02%) applicants would default while 4599 (91.98%) applicants would not. In reality, 127 (2.54%) applicants did default and 4824 (97.46%) applicants did not default. In order to better understand and visualize each model, a receiver operating characteristic (ROC) curve is plotted for each, and their corresponding area under curve (AUC) value is found.

LR and RF ROC Curves: Figure 3



Based on **Figure 3**, our calculated ROC AUC values were .806 for the LR model and .852 for the RF model. With our calculations and visualizations, at a glance we can tell that the RF performed more accurately at classifying true positives, but to further confirm our observations we compared the learning curves of both models.

LR and RF Learning Curves: Figure 4



Because of the nature of specificity in the random forest model, the training score was nearly perfect. However, cross-validation shows the models to be highly comparable. The learning curves in **Figure 4** do not change greatly over time, suggesting that not much would change given more observations. This is consistent with our previous analysis. Before deciding on our final model, we compared them below:

Logistic Regression	**Random Forest**
Deals well with high variance in explanatory and noise variables (Kirasich, Smith, & Sadler, 2018)	Better false positive rate when importance of noise variables is large (Kirasich, Smith, & Sadler, 2018)
Higher true negative and lower false positive rate	Higher true positive and lower false negative rate
High interpretability	Medium interpretability
Requires greater amount of data preprocessing	Requires less data preprocessing

VI. Chosen Model – Random Forest

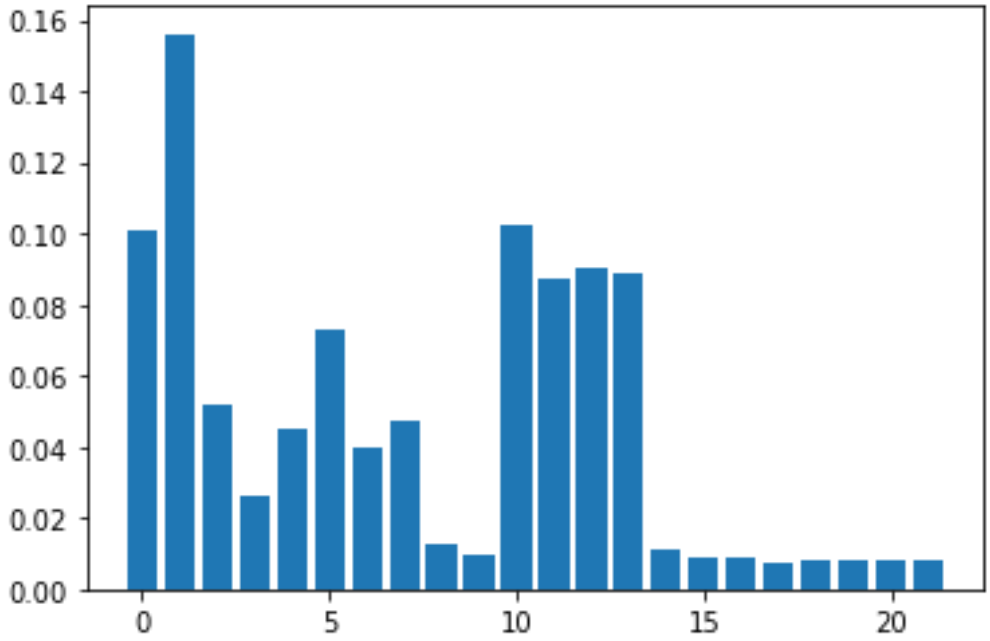
Although both models did well and have comparable efficacy, we chose RF as our preferred model for a few reasons. The first is that we believe there is likely to be a large amount of noise, such as credit scores, which are often considered in credit applications; RH handles this noise better. Under both models, 4599 applicants would be accepted and 401 would be rejected from the test dataset. That being the case, bank XYZ would benefit the most from maximizing the number of accepted non-defaulting applicants and minimizing the amount of accepted defaulting applicants. This is where the RF model exceeds, as can be seen by the confusion matrix and ROC curve. Lastly, while the interpretability of RF is lower than LR, the reduction in data processing helps justify the results of RF since feature selection and ranking is assigned to the algorithm.

A neat graph, such as a decision tree, cannot be drawn in the case of RF to specify how the algorithm makes decisions. However, we can gain insight into the importance the RF model placed on the predictive variables using **Table 6**, and the corresponding graph, **Figure 5**:

RF Model Feature Importance Scores (Truncated after Fourth Decimal): Table 6

Feature	Importance Score	Feature	Importance Score
tot_credit_debt	0.1011	uti_50plus_pct	0.0869
avg_card_debt	0.1561	uti_max_credit_line	0.0904
non_mtg_acc_past_due_12_months_num	0.0515	uti_card_50plus_pct	0.0891
non_mtg_acc_past_due_6_months_num	0.0261	ind_acc_XYZ	0.0113
mortgages_past_due_6_months_num	0.0447	AL	0.0085
credit_past_due_amount	0.0727	FL	0.0089
inq_12_month_num	0.0401	GA	0.0073
card_inq_24_month_num	0.0476	LA	0.0081
card_open_36_month_num	0.0125	MS	0.0080
auto_open_ 36_month_num	0.0093	NC	0.0082
uti_card	0.1027	SC	0.0080

Visualization of Feature Importance Scores: Figure 5



- Total credit debt and average card debt play the most important role in determining default predictions.
- Utilization variables and credit past due amount play a large role.
- Credit cards and auto loans opened in the last 36 months, prior XYZ account, and state variables play a very minor role.

VII. Model Implementation

The best implementation of the RF model that we have created is to make decisions on credit card applications. The model, which has already been trained, can be easily deployed to predict whether an applicant will default or not in real time. It can leverage these predictions to accept those that are predicted not to default while rejecting those that it predicts will default. However, if bank XYZ would like to go for a more hybrid approach, it could, like many other credit application processes, instantly accept and reject some applicants while sending others to review. If bank XYZ desires to implement this system, our model is easily adaptable as it can predict relative probabilities that an applicant will default or not default. If this is a path that bank XYZ is interested in, further inquiry could be done by our data team to determine which probabilities would be ideal for instant decision and which would be better for review.

Unfortunately, due to the limited interpretability of RF, we cannot describe with certainty whether applicants with accounts at bank XYZ get favorable treatment, despite having the importance of variables available. We do know the importance of having an account at XYZ is very low for the model, but we do not know if having an account makes one more likely to be predicted as a defaulter or non-defaulter. However, based on the low linearity with the response variable and low importance in the model, it is fair to say that it is highly unlikely that having an account in XYZ will change bank XYZ's decision. While it would likely encourage brand loyalty to give applicants with XYZ accounts favorable treatment, the data simply does not justify taking this step if the goal is to make default-based decisions.

If a credit card application is rejected using our model, and the applicant asks us to provide an explanation for why it was rejected, we would explain that unfortunately, the applicant hasn't met the necessary criteria to qualify for an XYZ credit card at this time. While we want to be as transparent as possible with bank XYZ about the process and decision-making involved in the development and implementation of our model, it is ultimately in the hands of bank XYZ as to whether they want to give details about their decision-making process to their customers.

In conclusion, we feel that our RF model is optimal for identifying potential defaulters with high accuracy. Bank XYZ will surely be better off with the value that is added by implementing our algorithm in their daily operations.

Appendix

Figure A: Correlation Matrix for Preprocessed LR Training Dataset

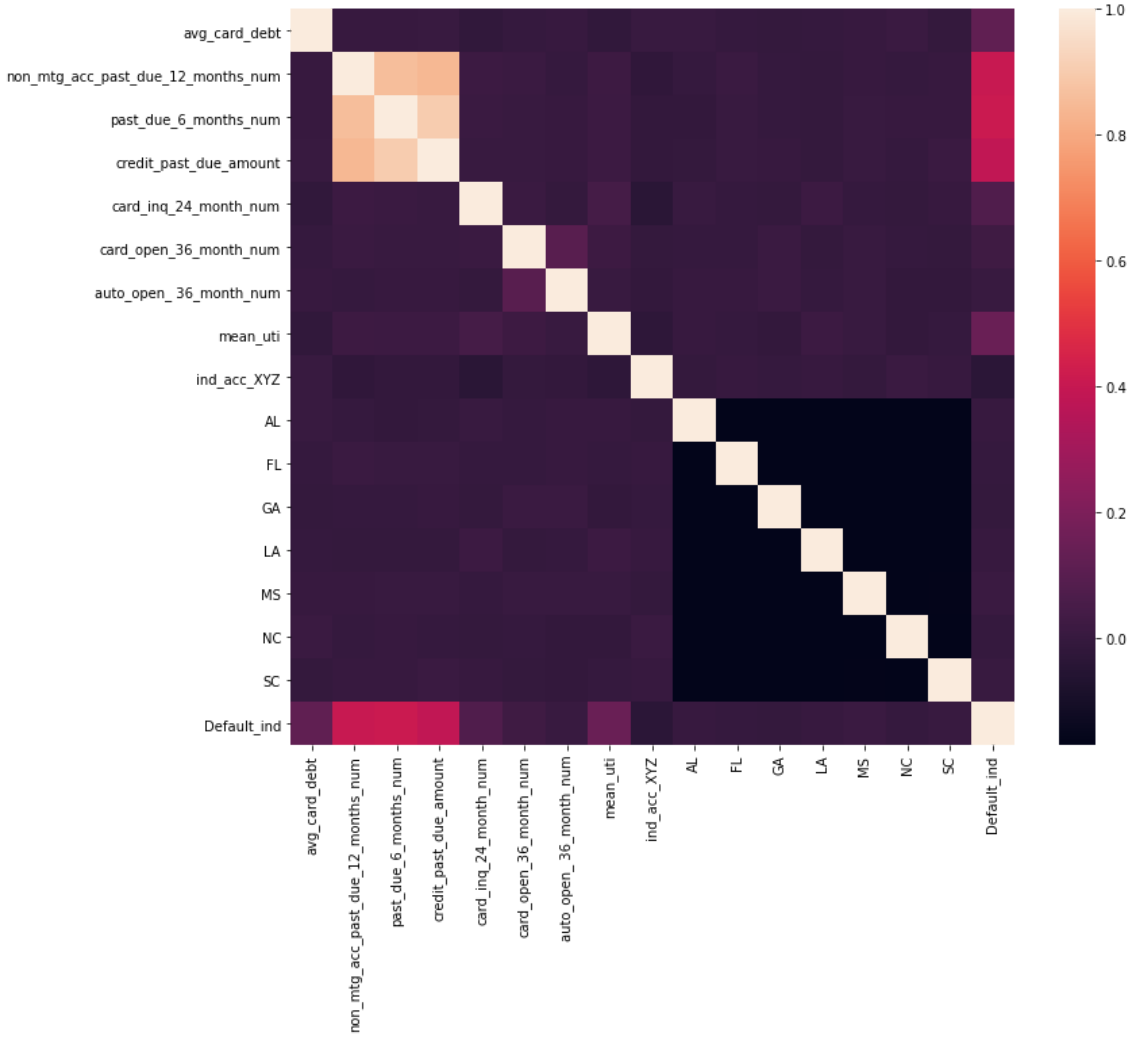


Figure B: VIF Table for Preprocessed LR Training Dataset

Variables	VIF
avg_card_debt	1.00
non_mtg_acc_past_due_12_months_num	4.24
past_due_6_months_num	6.52
credit_past_due_amount	5.94
card_inq_24_month_num	1.00
card_open_36_month_num	1.01
auto_open_ 36_month_num	1.00
mean_uti	1.00
ind_acc_XYZ	1.00
AL	5.38
FL	5.30
GA	5.26
LA	5.36
MS	5.28
NC	5.34
SC	5.24

Figure C: Accuracy of LR model using Different Hyperparameters

```
Best: 0.937778 using {'C': 0.1, 'penalty': 'l2', 'solver': 'liblinear'}
0.937667 (0.009195) with: {'C': 100, 'penalty': 'l2', 'solver': 'newton-cg'}
0.937667 (0.009195) with: {'C': 100, 'penalty': 'l2', 'solver': 'lbfgs'}
0.937667 (0.009195) with: {'C': 100, 'penalty': 'l2', 'solver': 'liblinear'}
0.937667 (0.009195) with: {'C': 10, 'penalty': 'l2', 'solver': 'newton-cg'}
0.937667 (0.009195) with: {'C': 10, 'penalty': 'l2', 'solver': 'lbfgs'}
0.937667 (0.009195) with: {'C': 10, 'penalty': 'l2', 'solver': 'liblinear'}
0.937667 (0.009195) with: {'C': 1.0, 'penalty': 'l2', 'solver': 'newton-cg'}
0.937667 (0.009195) with: {'C': 1.0, 'penalty': 'l2', 'solver': 'lbfgs'}
0.937667 (0.009195) with: {'C': 1.0, 'penalty': 'l2', 'solver': 'liblinear'}
0.937667 (0.009195) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'newton-cg'}
0.937667 (0.009195) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'lbfgs'}
0.937778 (0.009081) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'liblinear'}
0.937778 (0.008664) with: {'C': 0.01, 'penalty': 'l2', 'solver': 'newton-cg'}
0.937778 (0.008664) with: {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
0.936667 (0.008300) with: {'C': 0.01, 'penalty': 'l2', 'solver': 'liblinear'}
```

Figure D: Accuracy of RF model using different Hyperparameters

```
Best: 0.941111 using {'max_features': 'log2', 'n_estimators': 100}
0.936222 (0.009574) with: {'max_features': 'sqrt', 'n_estimators': 10}
0.940667 (0.010306) with: {'max_features': 'sqrt', 'n_estimators': 100}
0.940556 (0.009929) with: {'max_features': 'sqrt', 'n_estimators': 1000}
0.937778 (0.009480) with: {'max_features': 'log2', 'n_estimators': 10}
0.941111 (0.010830) with: {'max_features': 'log2', 'n_estimators': 100}
0.940444 (0.010532) with: {'max_features': 'log2', 'n_estimators': 1000}
```

References

Hurst, E., Li, G., & Pugsley, B. (2011). Are Household Surveys Like Tax Forms: Evidence from Income Underreporting of the Self-Employed. *Board of Governors of the Federal Reserve System*, from https://www.federalreserve.gov/pubs/feds/2011/201106/201106pap.pdf

Kirasich, K., Smith, T., & Sadler, B. (2018). Random Forest vs Logistic Regression: Binary Classification for Heterogeneous Datasets. *SMU Date Science Review, 1*(3), from https://scholar.smu.edu/cgi/viewcontent.cgi?article=1041&context=datasciencereview#:~:text=In%20general%2C%20logistic%20regression%20performs,variables%20increases%20in%20a%20dataset.

Ravanshad, A. (2018, April 27). Gradient Boosting vs Random Forest. Retrieved April 14, 2021, from https://medium.com/@aravanshad/gradient-boosting-versus-random-forest-cfa3fa8f0d80

Seanny. (2019, November 26). Credit Card Approval Prediction. Retrieved April 14, 2021, from https://www.kaggle.com/rikdifos/credit-card-approval-prediction

Seanny. (2020, November 14). Credit Card Approval prediction Using ML. Retrieved April 14, 2021, from https://www.kaggle.com/rikdifos/credit-card-approval-prediction-using-ml#Algorithms

UCI Machine Learning. (2016, November 03). Default of Credit Card Clients Dataset. Retrieved April 14, 2021, from https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset

U.S. Bureau of Labor Statistics. (2021, March 31). May 2020 National Occupational Employment and Wage Estimates. Retrieved April 14, 2021, from https://www.bls.gov/oes/current/oes_nat.htm