## Introduction

Commercial banks receive a lot of applications for credit cards. Many of them get rejected for many reasons, like high loan balances, low-income levels, or too many inquiries on an individual's credit report, for example. Manually analyzing these applications is mundane, error-prone, and time-consuming (and time is money!). The applications are going to increase with the growing number of digital devices and overall digital transformation across the globe. Credit card companies would need a program to analyze the historical patterns of the data and understand the most impactful factors in an application and decide whether to approve the application or not based on the risk score of the applicant. Credit Analysis involves statistical and qualitative measures to analyze the probability of a customer paying back the loan to the bank in time and predict its default characteristics. Analysis focuses on identifying and reducing the financial risks involved which may otherwise result in the losses incurred by the company while lending. The loss of business risk will also be there by not approving the application of eligible candidates. So, it is important to manage credit risk and handle challenges efficiently for credit decisions as they can have adverse effects on credit management.

Fortunately, this task can be automated with the power of machine learning, and pretty much every commercial bank does so nowadays. In this project, we build an automatic credit card approval predictor using machine learning techniques, just like real banks do.

## Goals

We have three primary goals with our dataset: to predict whether a person will be eligible for a credit card; to predict whether a person will have an overdue payment on a loan/credit card; and to predict an applicant's income, based on demographic and socioeconomic features. The former two goals are classification tasks, while the final objective is a regression-based task. We also want to identify which predictors are significant in determining credit card eligibility, the presence of overdue loans, and income.

## Fetching the Dataset

We found the primary dataset here, and merged it with this accompanying dataset. The former contained demographic features such as family structure, gender, and debt levels, and the latter contained credit history for each applicant. The merged dataset has 23 features: Applicant_Gender, Owned_Car, Owned_Realty, Total_Children, Total_Income, Income_Type, Family_Status, Housing_Type, Owned_Mobile_Phone, Owned_Work_Phone, Owned_Phone, Job_Title, Total_Family_Members, Applicant_Age, Years_of_Working, Total_Bad_Debt, Total_Good_Debt, Status, Months_PaidOff, and Months_Overdue. We are interested in predicting Status, which is a binary variable indicating whether a person was eligible for a credit card; Total_Income, which represents an applicant's reported total income; and an engineered feature Five_Months, which is a binary variable indicating whether a person had a payment overdue by five months or more.

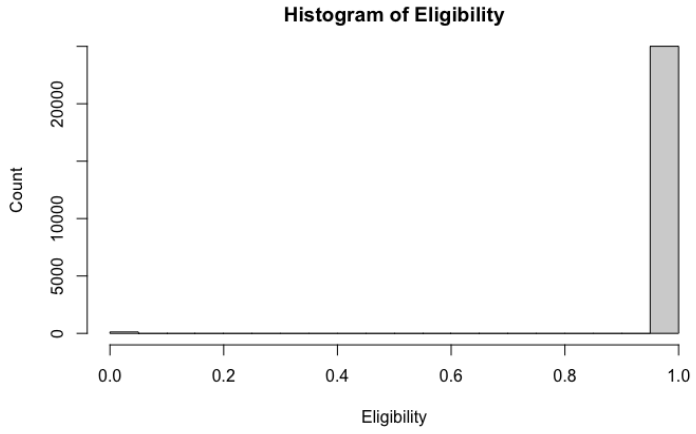
## Preprocessing

Since income type, applicant gender, family status, and housing type have fewer than ten levels per column and do not appear to have a clear ordering, we one-hot encode them. For education type, we see that the mean applicant eligibility does decrease as applicant education level decreases (Education level from highest to lowest goes “Academic degree”, “Higher education”, “Incomplete higher”, “Secondary/secondary special”, and “Lower secondary”. For this reason we employ ordinal label encoding on education type. The dataset with encoded features has 54 columns.

| Education_Type <fctr> | Average <dbl> |
|---|---|
| Academic degree | 1.0000000 |
| Higher education | 0.9949523 |
| Incomplete higher | 0.9939577 |
| Lower secondary | 0.9893048 |
| Secondary / secondary special | 0.9954172 |

Penalized logistic regression and KNN will require feature scaling. In the logistic model, L-2 regularization penalizes large predictors, so scaling is necessary (we use cross-validation to determine the optimal alpha parameter for the penalized logistic model, so the extent to which we regularize based on L-2 norm is not predetermined). The KNN model uses Euclidean distance so features need to be on similar scales. For these models, we scale the non-encoded training and testing columns based on the training mean and standard deviation.

Class Imbalance




Note that there is a severe class imbalance. There are 25002 eligible and 121 ineligible applicants in the entire dataset. If our model was to predict each applicant as eligible, we would predict at 99.5% accuracy. Such a model appears impressive at first glance but is obviously not useful to any company wishing to make predictions on eligibility. Our goal is to be able to draw useful insights from both eligible and ineligible applicants, but the class imbalance hinders our

ability to do so. To address this imbalance, we use Synthetic Minority Oversampling Technique (SMOTE) and random under-sampling. The SMOTE algorithm draws a random sample from the minority class and identifies k of its nearest neighbors, then will choose one of the nearest neighbors and calculate the Euclidean distance to it from the sampled point. We multiply the distance vector by a random number on [0,1], then add that to the original randomly sampled observation to get a new minority class observation (He, Haibo, and Yunqian Ma.). Minority observations were synthesized to reach 6669 ineligible observations, and we randomly sampled 10004 eligible observations to create the final balanced training dataset.

**Logistic Model**

We attempt to predict eligibility status using penalized logistic regression. Optimal values of lambda and alpha were selected based on accuracy and found via 5-fold cross-validation and grid search. The logistic model that minimized training accuracy had an alpha of 1 and lambda of 1.54743E-4. An alpha value of 1 yields a pure LASSO model, and the small lambda value tells us that little coefficient shrinking is taking place– that is, the final logistic model strongly resembles an unpenalized logistic model. The final model retained 34 out of the 54 total predictors. Notably, income type was only included for federal workers, the number of months an applicant overdue on their debt was not included, the number of months an applicant paid off debts was not included, and applicant age was not included. The number of months an applicant paid off debts being excluded is surprising, as one might expect that reliability in paying off other debts would conventionally be a strong indicator of an applicant's likelihood to make appropriate payments towards a credit card. The final model was simulated 20 times over varying 80/20 train/test splits to obtain the following metrics:

| mean_accuracy <dbl> | mean_precision <dbl> | mean_sensitivity <dbl> | mean_specificity <dbl> |
|---|---|---|---|
| 99.84282 | 99.99199 | 99.85006 | 98.2651 |

Accuracy on the test set is high, but this is to be expected with the high proportion of eligible status applicants in the dataset. Precision and sensitivity are both high which is also to be expected. The dataset had many original instances of positive (eligible) observations, so the model expectedly learned how to predict the positive cases very well. Specificity is slightly lower than the other metrics but still high. This indicates that the logistic model was able to learn from the minority class very well and consequently predict non-eligible observations effectively. Below is a sample confusion matrix for the final logistic model.

| | Actual | |
|---|---|---|
| Predicted | 0 | 1 |
| 0 | 27 | 7 |
| 1 | 2 | 4990 |

As we can see, there are relatively few misclassifications on the test set. What is especially noteworthy is the comparatively higher number of false negatives in comparison with false positives. From a business perspective, we would rather predict someone is ineligible for a credit card when in fact they are than predict they are eligible when they are not. Doing so minimizes financial risk to the company, as they do not want to allow a person to borrow funds if they do not expect to be paid back.

**Linear Discriminant Analysis (LDA)**

Linear Discriminant Analysis assumes that every class follows a Gaussian distribution. Hence, some data pre-processing was required before we were able to generate the model. For instance, binary variables such as Owned_Car and other variables that did not follow a Gaussian distribution were removed from the dataset and not included in the generation of this model. In the end, we selected 6 predictors: Total_Income, Applicant_Age, Years_of_Working, Total_Bad_Debt, Total_Good_Debt and Months_PaidOff as these features were somewhat normally distributed upon analysis. The balanced dataset is then partitioned into a 80/20 train/test split to generate the model with the following summary below:

```
Call:
lda(Status ~ ., data = lda.train)

Prior probabilities of groups:
       0        1
0.499825 0.500175

Group means:
  Total_Income Applicant_Age Years_of_Working Total_Bad_Debt Total_Good_Debt
0     209038.2      39.66370         5.392207     10.7881813        5.488354
1     195215.0      40.99635         7.700735      0.2826728       21.042831
  Months_PaidOff
0      0.8968552
1      9.0134939

Coefficients of linear discriminants:
                         LD1
Total_Income     -6.646222e-07
Applicant_Age     5.647701e-03
Years_of_Working  1.390711e-02
Total_Bad_Debt   -1.365029e-01
Total_Good_Debt   8.303292e-02
Months_PaidOff   -2.803674e-02
```

As can be seen, the group means varied greatly for Total_Bad_Debt, Total_Good_Debt and Months_Paidoff and less separability is shown for Total_Income, Applicant_age and

Years_of_Working. This was surprising to us as we assumed that income was a big factor to the outcome of a credit card application but as can be seen, predictors related to debt status were far more significant. We then predicted on the test split based on the model generated and obtained the following confusion matrix:

| | Actual | |
|---|---|---|
| Predicted | 0 | 1 |
| 0 | 28 | 259 |
| 1 | 0 | 4739 |

As can be seen, LDA performed really well with an accuracy of 94.8%, sensitivity of 94.8% and specificity of 100%. The number of false negatives is fairly high, but predicting ineligible when actually eligible incurs less risk on the loaner than false positives.

**Quadratic Discriminant Analysis (QDA)**

Quadratic Discriminant Analysis also has the same assumption that every class follows the Gaussian distribution. Hence, the predictors selected for QDA will be the same as for LDA. The following model was generated on a 80/20 train/test split:

```
Call:
qda(Status ~ ., data = qda.train)

Prior probabilities of groups:
       0        1
0.499825 0.500175

Group means:
  Total_Income Applicant_Age Years_of_Working Total_Bad_Debt Total_Good_Debt
0     209038.2      39.66370         5.392207     10.7881813        5.488354
1     195215.0      40.99635         7.700735      0.2826728       21.042831
  Months_PaidOff
0      0.8968552
1      9.0134939
```

The QDA model is fairly similar to LDA and so it generated a similar insight on the dataset. We then predicted on the test split based on the model generated and obtained the following results:

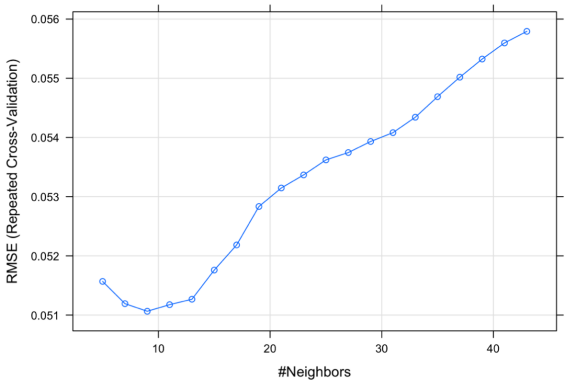
| | Actual | |
|---|---|---|
| Predicted | 0 | 1 |
| 0 | 28 | 634 |
| 1 | 0 | 4364 |

Based on the confusion matrix above, we see that the model has an accuracy of 87.4%, sensitivity of 87.3% and specificity of 100%. We can also conclude that QDA has performed

slightly worse than LDA in terms of accuracy and sensitivity. Like LDA, QDA has many false negatives and zero false positives.

**K-Nearest Neighbors (KNN)**

In order to select the best value for the variable K in K-Nearest Neighbors classification, we decided to do a repeated 10-fold cross validation on the dataset on various values of K to select the optimal value based on the smallest resultant root mean square error (RMSE).




Based on the figure above, we see an expected "U-shaped" graph where the RMSE tends to decrease before increasing again as the K value increases. Based on the results, we select 9 to be our K value for the model.

| | Actual | |
|---|---|---|
| Predicted | 0 | 1 |
| 0 | 28 | 0 |
| 1 | 0 | 4998 |

As we can see from the confusion matrix above, the K-Nearest Neighbor classification where K=9 has done very well with accuracy, sensitivity and precision of 100%.

**Naive Bayes**

We also constructed a Naïve Bayes model in order to predict eligibility status. Naïve Bayes works on the principle of conditional probability and uses Bayes Theorem in order to predict whether a data point belongs to a certain class.

| | y.test | |
|---|---|---|
| | Approved | Denied |
| Approved | 3055 | 4 |
| Denied | 1943 | 24 |

Naïve Bayes struggled at predicting the right status, as seen in the figure above, the overall accuracy was a lot lower than the other models at 61.26%. However, the model was able to predict denied applicants pretty well with a specificity of 85.71%, but struggled with predicting

approved applicants with a sensitivity of 61.12%. The reason why this model performed so poorly is most likely due to Naïve Bayes' reliance on the assumption that the features are conditionally independent, and since our dataset contains a lot of features on demographics, the variables are bound to be correlated somewhat.

**Random Forest**

We then tried using a random forest model to predict eligibility status. We decided to use 500 trees because the dataset was very large and computing that many trees wouldn't take too long to run. Both the regular random forest classifier method and the bagging method were used in order to find the best model. First, we tried to do the random forest classifier. For this method we needed to find the square root of all the predictors in the dataset to use in our model, since we had 53 predictors, the model used 8.

```
Call:
 randomForest(x = x.train, y = y.train, ntree = 500, mtry = 8,      importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 8

        OOB estimate of  error rate: 0.08%
Confusion matrix:
         Approved Denied  class.error
Approved    20003      6 0.0002998651
Denied         27  19968 0.0013503376
```

After running the model and getting good results on the training data, we then ran the model on the test data and compared results.

```
           y.test
yhat.rf     Approved Denied
  Approved     4996     14
  Denied          2     14
```

Looking at the figure above, the overall accuracy was 99.68%. The sensitivity was also very high with 99.96% of approved applicants being predicted correctly. However, the specificity was far lower at 50% with only half of the rejected applicants being predicted correctly. This was seen as problematic as approving denied applicants increases the financial risk for the companies.

So, we then tried a random forest bagging model using all 53 predictors. The model had similar accuracy and specificity at 99.92% and 99.96% respectively, but had a much better sensitivity, 92.86%, than the regular random forest model.

```
            y.test
yhat.bag     Approved Denied
  Approved     4996      2
  Denied          2     26
```

**Eligibility Models Comparison**

| | Accuracy | Sensitivity | Specificity |
|---|---|---|---|
| Logistic | 99.84282 | 99.85006 | 98.2651 |
| LDA | 94.80000 | 94.80000 | 100.0000 |
| QDA | 87.40000 | 87.30000 | 100.0000 |
| KNN | 100.00000 | 100.00000 | 100.0000 |
| NaiveBayes | 61.26000 | 85.71000 | 61.1200 |
| RandomForest_bagged | 99.96000 | 99.92000 | 92.8600 |

The above chart lists the accuracy, sensitivity, and specificity for each classification technique. Clearly KNN stands out as the best classifier, touting 100% accuracy. Logistic regression, LDA, QDA, KNN, and bagged random forest models unsurprisingly had high predictive accuracy on the majority class. These models also performed well at classifying non-eligible applicants, indicating that these models were effectively able to learn the minority class. The Naive Bayes model had the lowest accuracy and sensitivity, and second-lowest specificity.

**Other Predictions**

In addition to predicting credit card eligibility, we also decided to predict whether an applicant had a loan or overdue credit card payment by over five months. This is because credit card companies typically stop reporting overdue payments after about seven years. The idea behind predicting this was for it to possibly act as a "red flag" in order to possibly detect applicants who had an overdue payment for over 5 months even though it may not show up on the credit report. In order to be able to make actually useful predictions, we tried to predict whether the applicant had an overdue payment for over five months without consulting the credit history (and thus making the prediction useless). We started by removing features such as good and bad debt that would be on the credit history. Then, we used techniques such as logistic regression, Naive Bayes, Decision Trees, and K-nearest neighbors. All of the techniques we used predicted that zero applicants had a payment overdue by 5+ months besides k-nearest neighbors, which had an average accuracy of around 99% and a precision and recall of about 20% on average after twenty attempts. This is still much better than a random guess, since if our model predicts a person had a payment overdue by more than 5 months, there is about a 20% chance it is correct; meanwhile, if you randomly guess a person had an overdue payment for over 5 months, you would only have a 1% chance of being right due to class imbalance.

Along with credit card eligibility and overdue payments, we decided to predict an applicant's income as well. Fraud detection is an important aspect of many financial institutions as fraud can damage an institution's reputation, expend unnecessary resources, and overall cause the company to lose money. Since income is self-reported on many credit card applications it can serve as a signal or flag that indicates something is wrong. Income that doesn't match what should be reported based on demographics and history can be a tell that either someone's identity was stolen and they are trying to apply without knowing a person's income or someone might be boosting their income in order to get approved for a loan or credit card without actually being eligible. In order to predict income, three regression models were used: ridge, lasso, and random

forest. Random forest with 500 trees and using all predictors had the best result on the test data and had a root mean squared error of about 44382. Looking at this model we hypothesized that this model would work better at detecting fraud for people with higher income as their salary information is less available and people are more likely to over/underestimate by 44 thousand dollars if they made more money. We also looked at the importance of the variables in the model and found that Job_Title, Education, Applicant_Gender, Years_of_Working, and Applicant_Age were the 5 most important predictors in predicting income.

## Future Directions

One future direction we would like to explore is how we can make predicting loan eligibility more holistic. We had previously mentioned how we favored false negatives in our binary classification models. Conversely, favoring false positives would increase the total number of eligible applicants, albeit at the risk of the lender. We would also want a more expansive dataset that includes factors like race and geographic location, then observing features based on those groups. This could inform us about disparities in loaning and what steps should be taken toward building a fair loaning model.

## References

He, Haibo, and Yunqian Ma. "Imbalanced learning: Foundations, algorithms, and applications." Imbalanced Learning: Foundations, Algorithms, and Applications, (2013): 1-210. doi:10.1002/9781118646106.