Date: 04 – 29 – 2022

Machine Learning Models to Predict Credit Card Approval Risk

By: Kalide Endale and Donovan Johnson

Abstract

The data utilized for modeling has an extensive set of personal information about each applicant. The dataset comes as two separate datasets, an "application record" and a "credit record." In the "application record" dataset, there are a total of 17 variables, 12 categorical and 5 continuous. In the "credit record" dataset, there are 2 variables, 1 categorical and 1 continuous, both datasets contain an ID key column to tie a candidate's application record with their credit data. The dataset prior to any processing comprises of 537,667.00 observations. ***The objective is to work with this dataset to determine principal factors & build classification models to predict whether a candidate should receive a credit card.*** The team utilized four supervised machine learning models: Gaussian Naïve Bayes, Logistic Regression, AdaBoost, and Random Forest. One of the challenges the team encountered when building models for this project was the imbalance between customers who were classified as risky and not risky. This imbalance made it difficult for any of the models to distinguish the differences between the two classes. To cure the imbalance issue in the training set, the team used an advanced oversampling technique called SMOTE. Out of the total 8 models that were built, base models plus the hyper-tuned models, the best model are the random forest models. This model produced the least number of false negatives, false positives, a high accuracy score, and the best F1 score out of the set. In this paper, the team will discuss the data preparation, the methods used to cure imbalance in data, the math behind the algorithms, results of the models, and discuss how the research could be improved.

Introduction

According to *ibisworld*, credit card issuing is expected to be a $164.80 billion industry in 2022. One of the pain points for this industry is the pre-screening process required for a creditor to determine whether an applicant should or should not be approved. When screening was done manually, it was prone to errors, slow, and costly. However, by virtue of the availability of high-quality financial data, reduction in computation power, and the predictive task being learnable, machine learning (ML) models have become critical in helping credit firms optimize their screening process. In fact, nowadays, most firms across the financial industry utilize ML models for their automation needs. In this project, the goal is to build effective models that contribute toward assessing the risk of an applicant.

Methods

Explanatory Data Analysis

Initial first steps were to examine the data by teasing out some of the relationships that were present through simple exploratory data analysis.

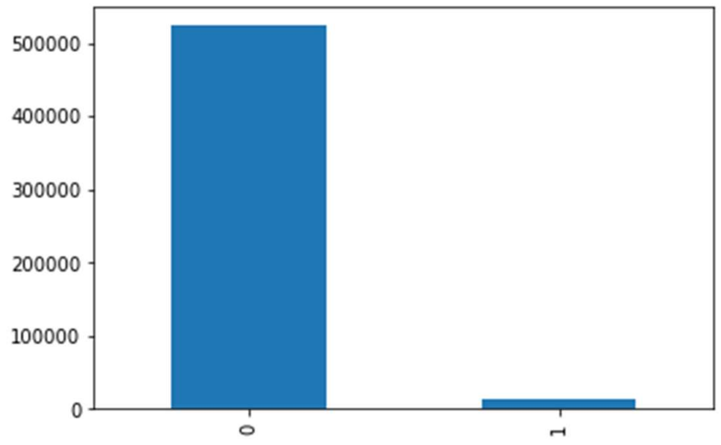

Figure 1

The original dataset does not define who is considered risky (response variable), To begin this project, the team created a custom binary response variable based on an individual's status of credit payment. If a candidate's credit card balance is 60 days past due, they were classified as '1' (risky) and '0' otherwise (non-risky). As it is illustrated in Figure 1, there is a significant imbalance in the response variable. An overwhelming majority of customers are trustworthy payers. In raw count, there are

523,495 ‘0’s and 14,172 ‘1’s in the dataset. In percentage terms, 98.50% of the binary response variable are ‘0’s and the remaining 1.50% are ‘1’s.

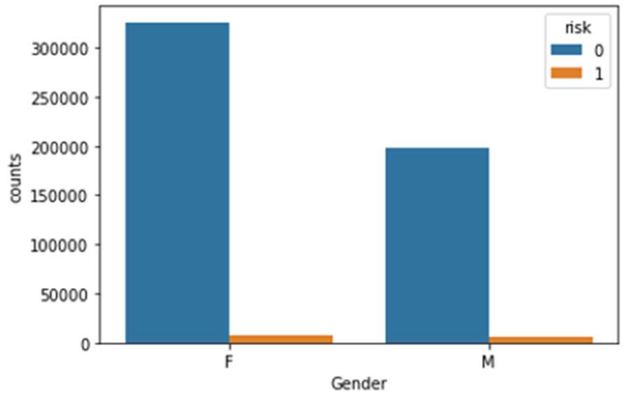

Figure 2

Figure 2 further shows the imbalance of Gender and Risk in the dataset. 66.7% of the observations are classified as female and 33.3% are classified as male. Additionally, there are 8,067 females classified as risky and only 6,105 males classified as risky. This is a bit surprising, since Females are 66.7% of the total dataset but only 56.9% of the applicant are labeled as risky. In proportional terms, males on average pose a higher risk than females for credit firms.

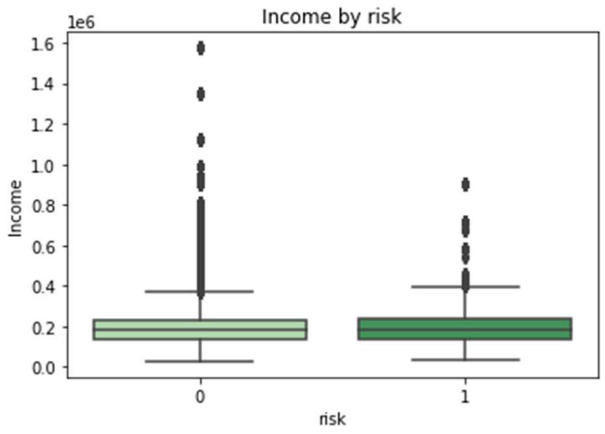

Figure 3

Figure 3 is a boxplot of income for “non-risk” (0) and “risk” (1) applicants. The difference in distribution of income between those labeled as risky and non-risk is evident. The median income between the two groups is similar but the range and quantity of outliers are much greater for non-risk applicants. This is not surprising, since non-risk applicants generally make more money and are able to reliably pay off their credit bill on time.

Data Processing

To begin data preparation, the team inspected the missing values present in the dataset—134,203 rows of missing values. There is a myriad of ways to deal with missing values, e.g., imputation by mean, median, mode, etc. However, since the dataset is particularly large, the best and simplest method of dealing with missing values is to remove them. This is the best option for this project because the classification models do not make any assumptions about the data. Thus, the removal of these variables should not impair the distribution of each respective feature since the consistency principle states that large sample sizes will converge in probability to its population distribution. Additionally, factor columns from the dataset were transformed into dummy variables because machine learning models cannot work on categorical variables in the form of strings.

After the removal of missing values, the team tested the correlation of features to begin feature selection. Feature selection is critical because it allows us to remove variables that are either overlapping with other variables or unnecessary for prediction. This reduces the dimension of the loss function and shrinks the computational power necessary to construct machine learning models.

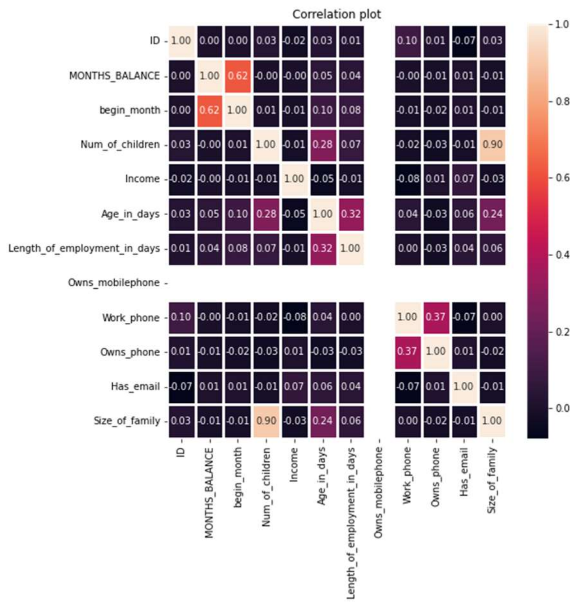

Figure 4

Figure 4 shows the correlation plot of all the features in the dataset, with brighter colors representing a higher degree of correlation. From the correlation plot, it is evident that most of the features are uncorrelated except for "size of family" and "number of children". Considering this, the team decided to remove "number of children" as a step toward dimension reduction. Furthermore, every observation had a mobile phone, so the entire column is a list of 1s. Thus, the variance is 0 for this feature, and correlation with each variable results in an undefined correlation. This resulted in the removal of the feature as it is not useful for prediction.

Following feature selection, the dataset was partitioned at random into a training and test set. 80% of the data was assigned to training and the remaining 20% reserved for testing the trained model. Additionally, standardization was utilized for features with extremely large range differences. For example, "Income" has a range of [3,600:1,575,000] and "Length_of_employment_days" has a range of [17:15,713]. Feature scaling technique is essential because machine learning algorithms are sensitive to the distances between data. If the data is not properly scaled, the features with a higher value range (Income) will dominate when calculating distances. This is because machine learning algorithms only see numbers. If some features have extremely large values, the algorithm assumes these values have some superiority and will attempt to train the model across these large values.

After feature selection, partitioning, and standardization, the next challenge was to deal with the imbalance that is present in the response variable (Figure 1). The team addressed this obstacle by utilizing an advanced oversampling technique called: Synthetic Minority Oversampling Technique (SMOTE). Traditionally, oversampling creates duplicates of the minority class by random repetition of the original dataset until the training set is balanced. However, instead of making exact duplicates, the SMOTE algorithm augments the original data points by adding small perturbations to the original dataset. In technical terms, SMOTE generates these synthetic points by K-nearest neighbors, a type of linear interpolation. Linear interpolation is a method of curve fitting using linear polynomials to create new data points within the range of a discrete set of known data points. SMOTE sets a minority class A, for each $x \in A$, it's K-nearest neighbors are found through the calculation of Euclidean distance between x and all other sets of A. The sampling rate variable N is set based on the amount of new data needed to be generated. For example, if there are 50 more observations for the majority class compared to the minority class, then N=50. For each $x \in A$, N examples are randomly chosen from K-nearest-neighbors and A1 is created. For each $xk \in A1$ where k is 1 through N. You use the following formula to create a new observation: $x' = x + rand(0,1) * |x - xk|$, where $rand(0,1)$ is a random number between 0 and 1.

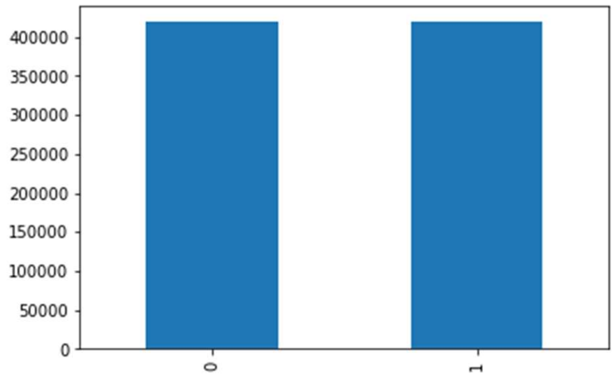

Figure 5

Figure 5 shows the training set after utilizing SMOTE. The new training set has an equal number of applicants who are labeled as "risky" (1) and "non-risky" (0). The training set has 418,721 '0's and 418,721 '1's.

BUILDING A DATA SCIENCE MODEL

The objective is to determine principal factors & build a classification model to predict whether a candidate should receive a credit card

The project will utilize four popular supervised machine learning classifiers to determine which candidates should be rejected for a credit card. Logistic Regression, Gaussian Naïve Bayes, Random Forest, and AdaBoost. To optimize the hyperparameters used in each model, a hyperparameter tuning method called Grid Search Cross Validation (CV) will be implemented to each base model. Grid Search CV works by creating a "grid" of hyperparameters. After the grid has been made, the algorithm will perform k-fold cross validation on every possible combination inside of the grid and return the combination of hyperparameters that produce the best score. Scoring metric for this project will be F1 score with a k-fold of 5. F1 score was selected as the evaluation scoring metric because the metric elegantly combines the predictive performance of a model by joining two otherwise competing metrics—precision and recall (Type I and Type II errors). Precision measures the extent of errors caused by false positives whereas recall measures the extent of error caused by false negatives. In business sense, a high false negative model creates a systemic risk for a company (FN) whereas a high false positive model diminishes a company's competitive edge (FP).

MODEL: Gaussian Naive Bayes

The algorithm for Gaussian Naive Bayes: draws influence from Bayes Theorem: $P(y|X) = \frac{P(X|y)*P(y)}{P(X)}$

The classifier assumes each predictor makes an equal and independent contribution to the output class. First, calculate the posterior probability for the two hypotheses, applicant is approved (0) & applicant is denied (1). Afterwards, assume features are independent, therefore:

$$P(X|y) = P(x_1|y) * P(x_2|y) * \ldots * P(x_n|y)$$

The probability of X is a constant so calculate proportionality and find the maximum probability:

$$y = argmax_y[P(y) * \prod_{i=1}^{n} P(x_i|y)]$$

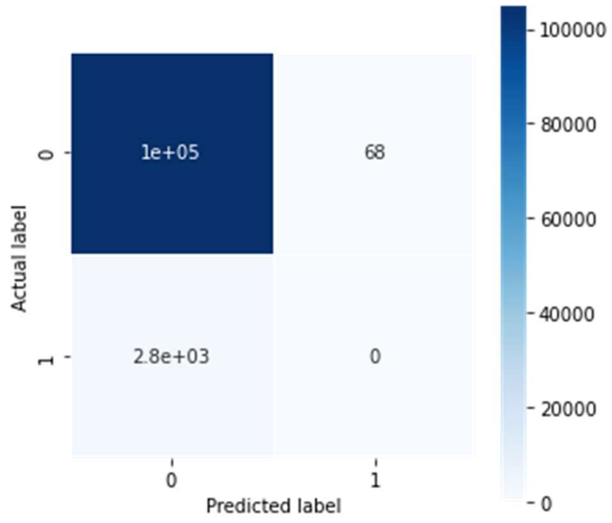

Figure 6

Figure 6 is the confusion matrix for the Gaussian Naive Bayes with default hyperparameters. The accuracy of this model was 0.9737 and F1 score was 0. This model does a poor job at predicting the minority class and is unable to distinguish the two classes appropriately. However, since the data is imbalanced the accuracy score is very high but because it's inadequate at measuring precision the F1 score is 0.

GAUSSIAN NAÏVE BAYES AFTER HYPER-PARAMETER TUNING

Grid Search CV hyperparameter:

1. **Var_smoothing** - A stability calculation to widen (or smooth) the curve and therefore account for more samples that are further away from the distribution mean: [np.logspace(0, -9, num=100)]

The best hyperparameters combination for Gri d Search CV was var_smoothing=0.0002

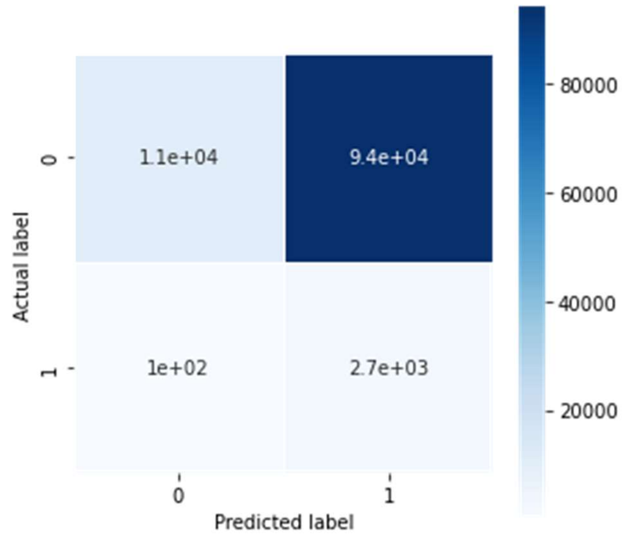

Figure 7

The accuracy of this model was 0.1244 and F1 score was 0.0534. Grid Search CV did not have any meaningful influence on the base Gaussian Naive Bayes model. The F1 score improved by 0.0534 with a .8493 drop in accuracy. Although there was an increase in F1 score, the magnitude in which accuracy decreased is alarming. In fact, both iterations of Gaussian Naive Bayes were poor models for appropriately classifying a risky applicant.

Model: Logistic Regression

The algorithm for Logistic Regression: first, calculate the likelihood of an applicant being approved (0) or denied (1). Using linear regression, model the relationship between the outcome and features. However, since this is a classification problem, output should be in probabilities, bounded between 0 and 1. To do this, create a linear function, p(x), and wrap the equation into a logit transformation. After solving for p(x) you are left with:

$$p(x) = \frac{e^{a_o+a}}{(e^{a_o+a}+1)}$$

Afterwards, MLE is applied since logistic regression uses probabilities. This will estimate the parameters of probability distribution by maximizing the likelihood function:

$$\frac{dl}{da_j} = \frac{1}{k}\sum_{i=0}^{n}(yi - p(x_i; a_o, a))x_{ij}$$

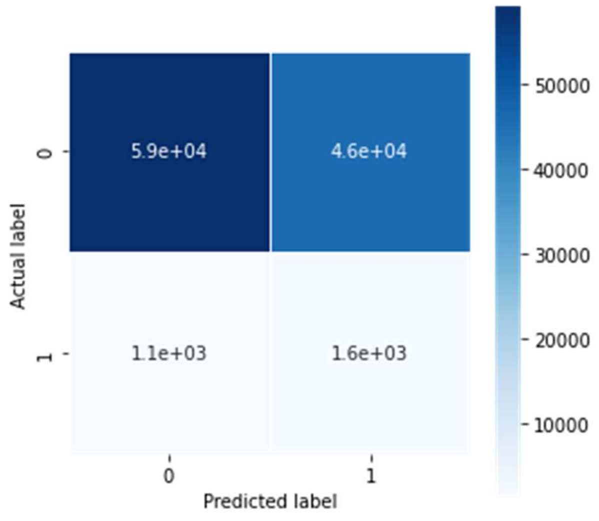

Figure 4

The training data was used to fit the logistic model and the fitted model was used to predict on test data. Figure 4 is the confusion matrix for the logistic regression model with default h yperparameters. The accuracy of this model w as 0.5658 and F1 score was 0.0650.

Logistic Regression After Hyper-Parameter Tuning

Grid Search CV hyperparameters:

1. **Penalty** - Specify the norm of the penalty: ['none', 'l1', 'l2', 'elasticnet']
2. **C**- Controls the penalty strength: [.01,0.1, 0.5, 1,10,15]
3. **Solver** - Different solvers which try to find the parameter weights that minimize a cost function: ['newton-cg', 'lbfgs', 'liblinear']

The best hyperparameters combination for Grid Search CV was C=0.01, penalty='l1', solver='liblinear'. We once again fit training set and test the results of new model.

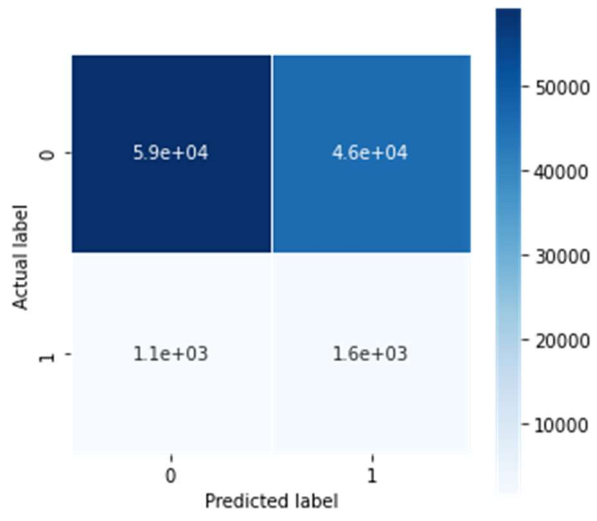

Figure 5

The accuracy of this model was 0.5654 and F1 score was 0.0652. Grid Search CV did not have any meaningful influence on the base logistic regression model. The F1 score improved by 0.0002 and the accuracy dropped by .0004. The hyper-tuned logit model is the better model of the two, but the difference is trivial.

Model: AdaBoost

AdaBoost is a technique that consists in sequentially fitting several weak learners in an adaptive way: each model in the series is fitted giving more weight to observations in the dataset that were misclassified by the previous models in the series. Intuitively, each new model stresses its efforts on learning the most difficult observations to fit, so that at the end of the process, a strong learner with a lower bias and a reduced variance is assembled. In summary, with AdaBoost, weights of well classified observations decrease relatively to weights of misclassified observations. Models that perform better have higher weights in the final ensemble model. The advantage of this models is that it is resistant to overfitting, but the model's downside is that it can be computationally expensive to execute.

The algorithm for AdaBoost: first calculate sample weights: $w(xi, yi,) = \frac{1}{N}, i = 1,2 \ldots n$
Where N is the total number of data points. A decision stump will be created for each feature, the best decision stump is chosen based on a Gini or Entropy score. The stump with the least value will be the first base learner. Formula for calculating performance of a stump:

$$\frac{1}{2} * \log \left(\frac{1 - Total\ Error}{Total\ Error}\right)$$

After finding the base learner, train a sequential model on a training set that increases the frequency of misclassified observations from the previous model. For incorrectly classified records, the formula for updating weights is:

$$New\ sample\ weight = old\ weight * e^{\mp Amount\ of\ say(\alpha)}$$

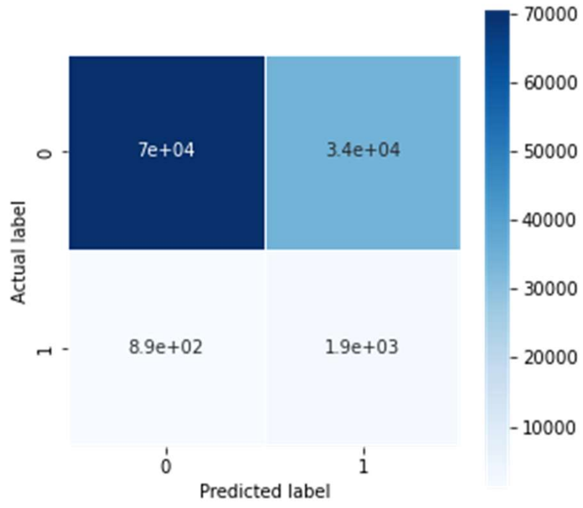

Figure 10

The training data was used to fit the AdaBoost model and the fitted model was used to predict on test data. Figure 10 is the confusion matrix for AdaBoost model with default hyperparameters. The accuracy of this model was 0.6727 and F1 score was 0.0961.

AdaBoost After Hyper-Parameter Tuning

Grid Search CV hyperparameter:

1. **Learning_rate** - The rate at which we are adjusting the weights of our model with respect to the loss gradient: [50, 100]

2. **n_estimators** - The maximum number of estimators at which boosting is terminated. In case of perfect fit, the learning procedure is stopped early: [0.001, 0.1, 1]

The best hyperparameters combination for Grid Search CV is learning_rate=1 and n_estimators=100.

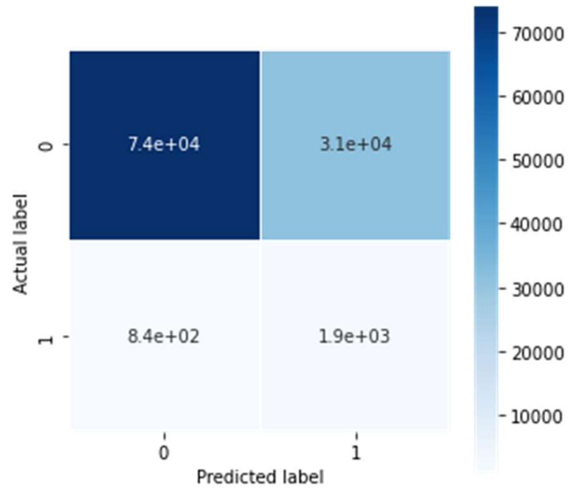

Figure 11

The accuracy of this model was 0.7053 and F1 score was 0.1078. The accuracy score improved by .0326 and F1 score improved by .0117. Unlike Naïve Bayes and Logistic, AdaBoost did benefit from Grid Search CV with a boost to both accuracy and F1 score. However, like the previous models, AdaBoost does not effectively minimize false negatives and true positives.

Model: Random Forest

Random Forest is an ensemble method that trains a lot of decision trees on a random subset of the data and features. Each tree then makes a classification prediction, and the algorithm averages the predictions into a final prediction. Random Forest predictions are better than individual decision trees because the variance of a random forest is lower and is resistant to overfitting.

The algorithm for Random Forest (RF): The model creates several decision trees during training and adds them together to vote on a final prediction, called ensemble:

$$RFfii = \frac{\sum_{j \in alltrees} [norm(fiij)]}{T}$$

- T = total numbers of trees,
- RFfii = importance of features i calculated from all trees in the random forest,
- $normfii = \frac{fii}{\sum_{j \in alltrees} fij}$ is the normalized feature importance for i in tree j,
- $fii = \frac{\sum_{j:node\ j\ splits\ on\ features\ i} *nij}{\sum_{k \in all\ nodes} nik}$ which is the importance of feature i, and nij is the importance of node j.
- Random Forest ensures each decision tree is uncorrelated by randomly sampling from the data with replacement. This is also done to the features. Since decision trees are sensitive to the data they are trained on, this results in different tree structures and predictions.

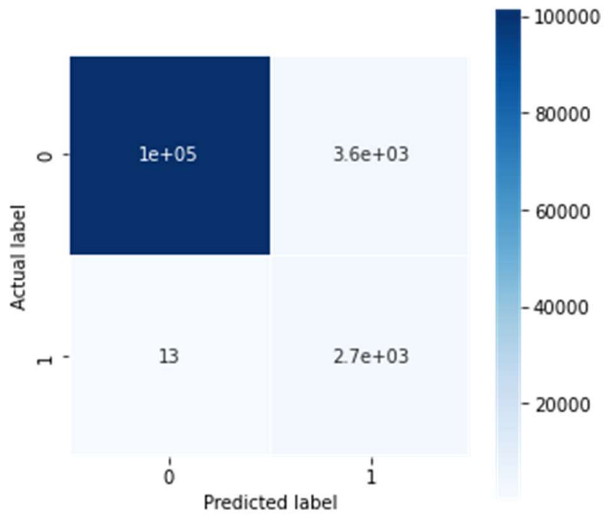

Figure 8

Figure 8 is the confusion matrix for the random forest model with default hyperparameters. The accuracy of this model was 0.9668 and F1 score was 0.6057. This is a significant improvement in both accuracy and F1 score compared to the previous models tested.

Random Forest After Hyper-Parameter Tuning

Grid Search CV hyperparameter:

1. **Criterion** - The function to measure t he quality of a split. Supported criteria

are “gini” for the Gini impurity and “entropy” for the information gain: ['gini', 'entropy']

2. **Min_samples_split** - The minimum number of samples required to split an internal node: [5, 10]

The best hyperparameters combination for Grid Search CV was Criterion= “gini” and min_sample_split = 5.

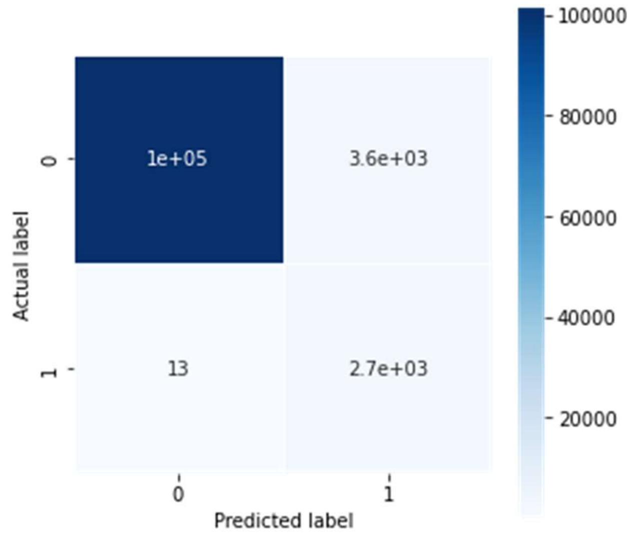

Figure 9

Figure 9 is the confusion matrix for the Grid Search CV Random Forest. The accuracy of this model was 0.9668 and F1 score was 0.6057, unchanged from the base random forest model. The accuracy and F1 score of the default and Grid Search CV random forest models are identical to each other. Out of the 8 models that were built, base models plus the hyper-tuned models, the best model can go to the base random forest model or the hyper-tuned random forest model. However, the team believes with more tuning parameters the hyper-tuned random forest model would be better. This model produced the least number of false negatives, false positives, a high accuracy score, and the best F1 score out of the set.

Results

The final and best model selected was the random forest model. This model produced the least number of false negatives, false positives, a high accuracy score, and the best F1 score out of the set. Random forest is an ensemble method that trains a lot of decision trees on a random subset of the data and features. Each tree then makes a classification prediction, and the algorithm averages the predictions into a final prediction. Random forest prediction is better than individual decision trees because the variance of the random forest is lower. Furthermore, random forests are popular because predictions are interpretable, they can handle large datasets efficiently, and the algorithm is resistant to overfitting.

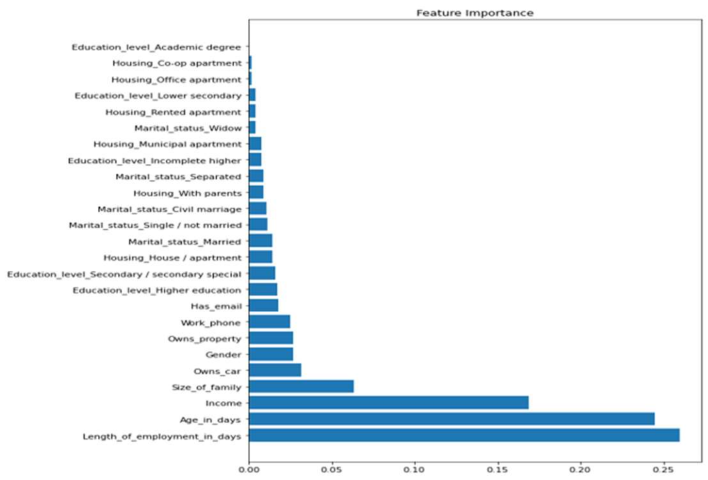

Figure 12

As illustrated in Figure 12, the key features in predicting default for the random forest are *length of employment in days, Age in days, Income, and size of family*. Additionally, *if an applicant owns a car, their gender, owns a phone*, and *owns property* are the next four best contributors. These four contribute about the same towards an accurate classification. The rest of the variables do not have a significant role in determining classification outcomes.

Examining model diagnostics further, the precision of the model is 2700 ÷ (2700 + 3600) = 0.4285. Precision tells us how many of the correctly predicted cases actually turned out to be positive. The recall of the model is 2700 ÷ (2700 + 13) = 0.9952. Recall tells us how many of the actual positive cases the model predicted correctly. The random forest model excels at predicting true positives, true negatives, and false negatives. However, the model is average at handling false positive. Considering the objective is to determine principal factors & build a classification model to predict whether a candidate should receive a credit card. A great model minimizes both false negatives and false positives. Misclassifying false negatives is the most dangerous out of the two. The model is labeling these applicants as not risky when the applicants are not reliable customers. A high false-negative rate embeds a systemic risk to a company's financial health if its customer base is unpredictable and not trustworthy. On the other hand, a high false positive rate is not as perilous because the model is labeling these applicants as risky when they are trustworthy payers. If a company's model suffers from a high false positive rate, the company risks losing out on future clients as they are labeled risky and declined a credit card. This company would inevitably lose its competitive edge because of poor modeling and its market share over time. The final random forest model excels at minimizing false negatives 13 ÷ (13 + 100,000) = 0.0001 but does not do quite as good of a job at minimizing false positives 3600 ÷ (3600 + 2700) = 0.5714

Discussion

For this project, the team investigated the relationships between one's likelihood of default and various explanatory variables. According to our random forest model, the key features in predicting default are *length of employment in days, Age in days, Income, and size of family*. After these four models, the importance of the features significantly drops off. Since these features are significantly important in predicting default risk, creditors should make sure to include such questions in the application form for credit cards. Furthermore, these features can also help applicants understand how their credit application is scored which improves the transparency of the application process.

While the random forest model can appropriately classify the default risk of an applicant, there are many limitations and potential improvements. One such limitation is that the models built in this project were computationally expensive, especially when applying hyper-parameters to the base models. With higher compute power, it is feasible to do additional tuning which could improve the models. Furthermore, a collection of a new set of features guided by domain experts would meaningfully improve the quality of the dataset which directly improves the strength of the classification models. Another potential limitation of this model is the sampling technique (SMOTE) that was used to balance the imbalanced data from the training. It could be beneficial to try other balancing techniques to see if it yields stronger models.

This model could be enhanced in many ways. However, there is natural a tradeoff when you seek to improve a model. The easiest way to improve the final model is to run more tuning parameters to find the optimal parameters that classify observations accurately. As you test more tuning parameters, the model's complexity increases which would reduce bias but increase its variance. Additionally, as complexity increases the compute power required also increases. Another potential improvement is to run other models both supervised and unsupervised to see if there are other models which can classify the dataset better.

Work Cited

- Christopher, A. (2021, February 3). K-Nearest Neighbor. Medium. Retrieved March 16, 2022, from https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4
- GreatLearningTeam. "Hyperparameter Tuning with GRIDSEARCHCV." GreatLearning Blog: Free Resources What Matters to Shape Your Career!, 20 Sept. 2020, https://www.mygreatlearning.com/blog/gridsearchcv/.
- IBISWorld. (2021, July 31). Industry market research, reports, and Statistics. IBISWorld. Retrieved

March 15, 2022, from https://www.ibisworld.com/industry-statistics/market-size/credit-card-issuing-united-states/

- Korstanje, Joos. "Smote." Medium, Towards Data Science, 30 Aug. 2021, https://towardsdatascience.com/smote-fdce2f605729.
- J. Kong, W. Kowalczyk, D. A. Nguyen, T. Bäck and S. Menzel, "Hyperparameter Optimisation for Improving Classification under Class Imbalance," 2019 IEEE Symposium Series on Computational Intelligence (SSCI), 2019, pp. 3072-3078, doi: 10.1109/SSCI44817.2019.9002679
- KNN(K-Nearest Neighbour) algorithm, maths behind it and how to find the best value for K. (2019, October 25). Medium. Retrieved March 16, 2022, from https://medium.com/@rdhawan201455/knn-k-nearest-neighbour-algorithm-maths-behind-it-and-how-to-find-the-best-value-for-k-6ff5b0955e3d
- LT, Zeya. "Essential Things You Need to Know about F1-Score." *Medium*, Towards Data Science, 25 Feb. 2022, https://towardsdatascience.com/essential-things-you-need-to-know-aboutf1scoredbd973bf1a3#:~:text=F1%2Dscore%20is%20one%20of,competing%20metrics%20%E2%80%94%20precision%20and%20recall
- Molnar, C. (2022, March 4). Interpretable machine learning. 5.2 Logistic Regression. Retrieved March 16, 2022, from https://christophm.github.io/interpretable-ml-book/logistic.html
- Rai, K. (2020, June 14). The math behind Logistic Regression | by Khushwant Rai | Analytics Vidhya. Medium. Retrieved March 16, 2022, from https://medium.com/analytics-vidhya/the-math-behind-logistic-regression-c2f04ca27bca
- Seanny. (2020). Credit card approval prediction. Kaggle. Retrieved March 15, 2022, from https://www.kaggle.com/rikdifos/credit-card-approval-prediction
- Yiu, T. (2019, June 12). Understanding Random Forest. How the Algorithm Works and Why it Is… | by Tony Yiu. Towards Data Science. Retrieved March 16, 2022, from https://towardsdatascience.com/understanding-random-forest-58381e0602d2

Appendix

Section 1: Continued EDA

Date: 04 – 29 – 2022

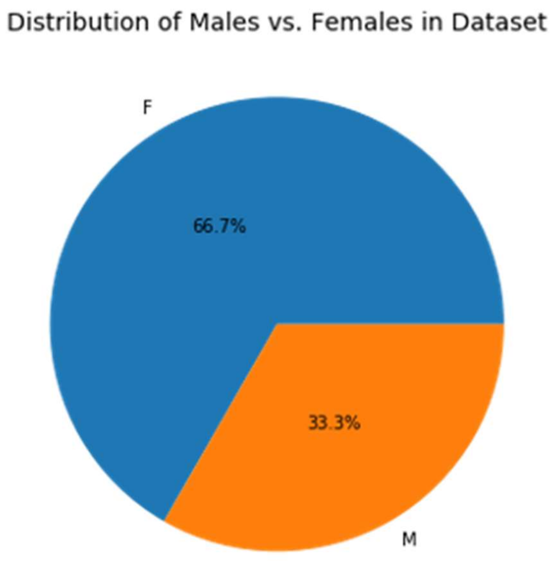

Figure A

With the processed and transformed dataset, a pie chart of the male and female ratio in the dataset was taken account. As illustrated in Figure A, majority of the dataset are women (66.7%).

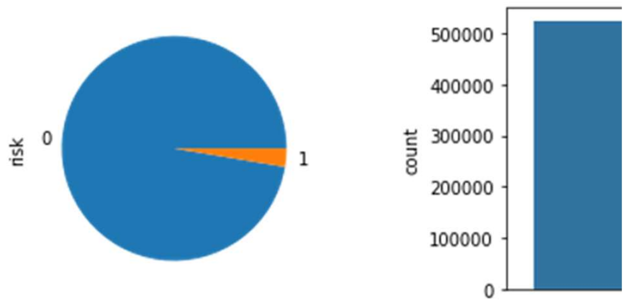

Figure B

Figure B further visualizes the imbalance issues of the dataset. Naturally, it is reasonable to expect much of the dataset to be represented as non-risky customers and a small minority class of customers to be labeled as risky. Those who have not defaulted, classified as non-risky, represent 98.5% of the dataset and the remaining 1.5% represent customers labeled as risky. This suggests that an oversampling or undersampling technique should be utilized to create a synthetic training set that is balanced and more robust for training.

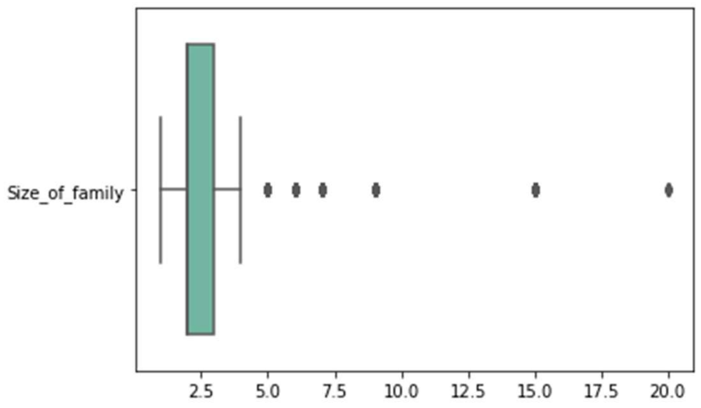

Figure C

Size of family is mostly distributed from 0 – 4 but there are a few observations that are considered outliers (Figure C).

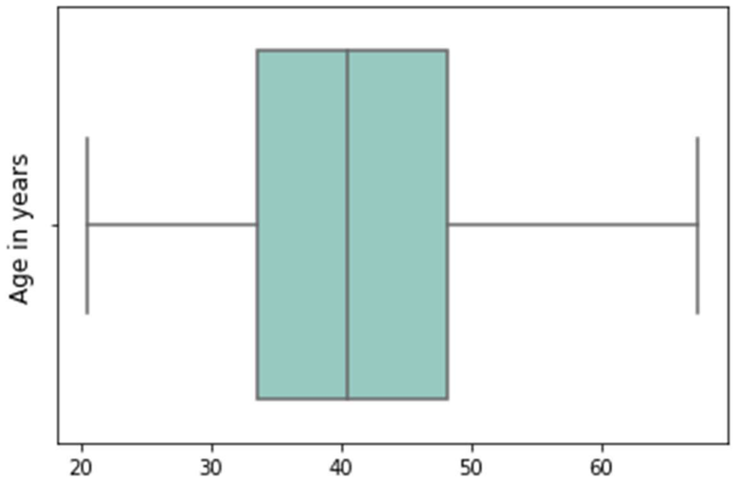

Figure E

Age in years distribution is normally distributed. A diverse set of ages are represented in the dataset (Figure E).

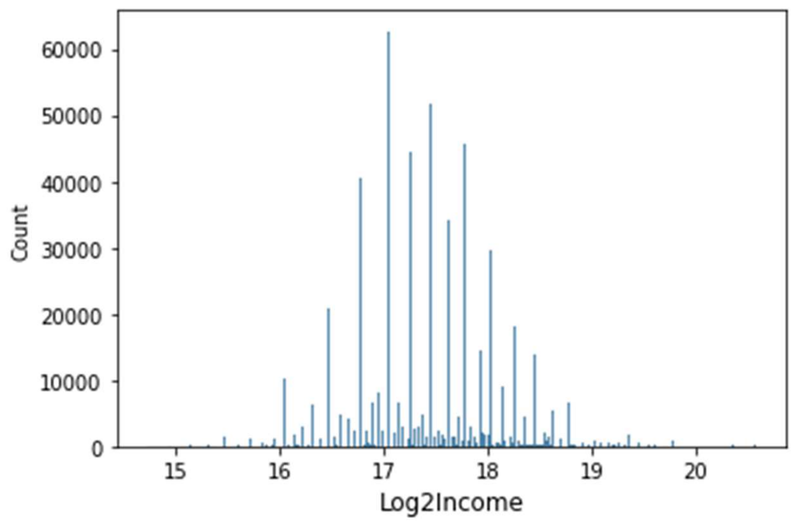

Figure F

Log2(Income) is normally distributed, it is beneficial to see a diverse set of income represented in the dataset. A normally distributed range of income creates a robust training set for predictive models (Figure F).

Section 2: Link to Codebook

GitHub:
https://github.com/KalideEndale/Supervised-Machine-Learning