Will I get Approved? Credit Card Approval Prediction with Machine Learning

Fang Shu
Department of MS&E
Stanford University
fangshu@stanford.edu

Name: Wai Yam Gordon Tsai
Department of MS&E
Stanford University
wtsai869@stanford.edu

Abstract

American's total credit card balances have reached $925 million by the third quarter of 2022, the second highest record in history [9]. This increased debt and default incentivize banks and financial institutions to make more careful decisions when approving credit card applications. In this study, we use different machine learning algorithms to predict whether a person would get approved for a credit card application based on personal features. We observe a Neural Network resulted in the best performance, with an AUC score of 0.95 and an F1-score of 0.88.

1 Introduction

Credit cards are a commonly used payment method. Applicants file credit card applications at banks, but banks will evaluate the application by checking the applicant's eligibility relying on personal information. A bank may be at risk if too many existing applicants fail to meet their obligations. However, the process of checking an application can be time-consuming and result in manual errors. As a result, we hope to build models that may make financial institutions easier to handle credit card applications when checking applicant's eligibility. The input to our models are personal features and it predicts whether or not a person should be approved for a credit card application.

The primary motivation behind our problem is that we believe credit card usage pertains to our daily lives. When we apply for a credit card, we hope for approval, but banks will need to conduct a thorough background check to make the decision. And by employing machine learning algorithms, the bank or financial institutions can reduce the risk of granting credit cards to someone with potential financial default risk, speed up the decision process, perhaps reduce manual error, and even allow the application decision process to not over-rely on one or a few personal features.

2 Related Work

Some similar research has been conducted in the field. For example, one study [7] inspected credit card approval data taken from the UCI machine learning repository. The study concluded that Random Forest was the best learning method among a few other supervised learning methods. However, the study did not employ any methods to address the imbalance problem, nor did the study use any deep learning methods, whereas we plan to incorporate both in our research.

One study reached a 94 % accuracy by using a Neural Network and 95 % accuracy using Random Forest [4]. However, since they relied on relief-based feature selection, they could not detect highly correlated features. For our work, we will detect multicollinearity among features and perform feature selection differently.

Another study introduced higher dimensional prediction algorithms such as Classification and Regression Tree (CART) and Linear/Quadratic Discriminate Analysis to solve imbalanced data problems and obtained a 96% accuracy with the CART model [11]. However, they did not provide feature interpretations and did not provide any suggestions to the financial institutions.

One paper copes with the same credit card approval process [3]. They efficiently clean missing and wrongly labeled data and accurately perform data pre-processing by using different pipelines. However, the models have low accuracy, and since the main goal of handling such approvals is to detect "bad" applicants, we should aspire for better model performance.

Another paper introduced new methods of Tensor Factorization for model prediction and used the Prediction and Parallel Factors Factorization to reduce the feature dimension [13]. However, although they obtained a high accuracy of 98.5%, their prediction is unstable. In addition, we also believe more feature interpretation is necessary.

,

3 Dataset and Features

We work with a real bank dataset that is obtained from Kaggle [12]. The dataset comes in two files. The first file contains information on the socioeconomic status of the applicants. It contains 19 features, such as whether or not the applicant owns a car, a realty; the annual income, education level, marital status, etc. Due to page constraints, we will not list out all the features. The second file contains the payment or default records of the applicants. Each row in the second dataset corresponds to the monthly record of a applicant. Because an applicant may have no default record for the current month (when the dataset was extracted) but may have a default record for the previous month, there are duplicated applicant IDs in the second dataset.

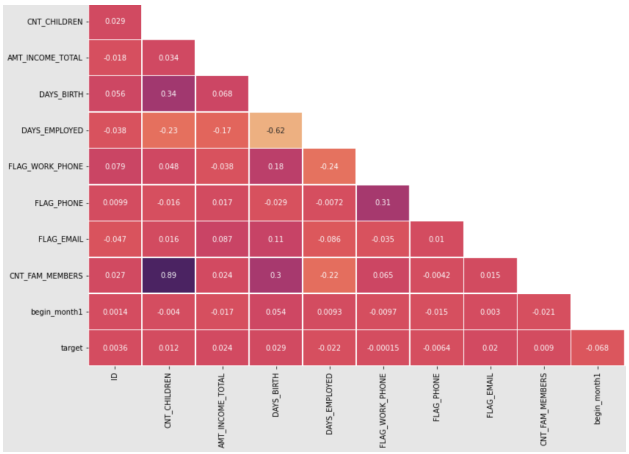
4 Methodology

4.1 Exploratory Data Analysis

We first transform the payment status (which exists in the second dataset) into our dependent variable. Currently, the feature *Status* lists the past-due period, such as 1-29 days past due, 30-59 days past due, etc. We will classify the applicants based on their payment status. We note that our data is highly imbalanced as about 98 % of all applicants have less than one past due. We decided to look at the records for the applicants for the past 12 months, as banks usually rely on personal information from the past 12 months to make a decision. If an applicant had at least one past-due status for the past 12 months, we will classify that applicant as a "bad" applicant, and if an applicant did not have any past-due records in the past 12 months, we say the applicant is a "good" applicant. Hence, we want to build algorithms that may predict whether or not an applicant is "good" or "bad" based on their personal features and therefore determine whether or not they should be approved for a credit card.

After defining the target variable, we combine the two datasets based on customer ID and only include the customers that occur in both datasets. This leaves us with 36,457 unique customers and their personal information. We check for missing values and drop a few features that may not be useful for our prediction purposes (such as whether or not a person has a mobile phone or email). By observing the feature correlation matrix in figure 1 (a). we see that the count of family members is highly correlated with the count of children and the *Days Birth* feature is highly correlated with *Days Employed*. Therefore, we drop the count of family members and *Days Birth*. We then normalize the numerical features (such as annual income) and transform all the numerical features into categorical ones. For example, we grouped all families with more than two children together and transformed the Children Count feature into a categorical variable with three categories: no children, one child, and more than two children. Finally, we perform one-hot encoding on the categorical variables. The final features we include in building our models are presented in figure 1 (b).

Finally, we see that about 94 percent of applicant can be classified as "good", whereas only about 6 percent can be classified as "bad". To address this imbalanced problem, we oversample the minority class. We employ the *Synthetic Minority Oversampling Technique* (SMOTE), which was first proposed in this paper [1]. This method creates new synthetic examples from the minority class, which are close in feature space to existing examples from the minority class. We only oversample our training data, as we do not want the testing data to be affected. Finally, we fit our models.



(a)

Feature Name	Feature Type
Gender	Binary
Car Ownership	Binary
Realty Ownership	Binary
Count of Children	Categorical
Work Time	Categorical
Occupation Type	Categorical
Housing Type	Categorical
Education Type	Categorical
Family Status	Categorical

(b)

Figure 1: (a) Feature Correlation Matrix (b) Feature type after Pre-Processing

4.2 Baseline Model

We fit a simple logistic regression model with L2-regularization as our baseline model. In general, logistic regression is one of the simplest binary classification models. For our task, it takes as input the different characteristics of the customers and outputs the probability of whether or not the customer is classified as bad or good. Logistic regression minimizes the loss function

$J(\theta) = \sum_{i=1}^{n}[-y^{(i)}log(h_\theta(x^{(i)})) - (1 - y^{(i)}log(1 - h_\theta(x^{(i)})] + \frac{\lambda}{2m}\sum_{i=1}^{n}\theta_i^2,$

where $x^{(i)}$ denotes the feature vector of the i^{th} customer, n is the number of customers, λ is the regularization hyperparameter, and $h_\theta(x^{(i)}) = \frac{1}{1+e^{-\theta^T x^{(i)}}}$. We let $y^{(i)} = 1$ if the i^{th} customer is a bad customer and $y^{(i)} = 0$ if the i^{th} customer is a good customer.

4.3 Decision Tree

Decision Tree is a nonparametric supervised learning method that may be used for both classification and regression tasks. The base of the decision tree is called the root node, and from the root node flows a series of decision nodes that depict decisions to be made. Each decision node resembles a question or split point, and the leaf nodes that stem from the decision node represent the possible answers. We use a decision tree as it is easy to implement and has high interpretability.

4.4 Random Forest

Random Forest is an ensemble of Decision Trees, with the advantage of reducing the error from an individual tree. It creates multiple Decision Trees using bootstrapped datasets of the original dataset, and randomly selects a subset of variables at each step of the Decision Tree. To classify a new sample based on the features, each tree gives a classification as if "voting" for that class. The Random Forest model then selects the classification that won the most votes among all trees.

4.5 Support Vector Machine

Support Vector Machine is a supervised classification method that will find a hyperplane that separates the two classes of the data and will also maximize the margin between the two classes. For our dataset, the Support Vector Machine model tries to find a linear decision boundary that classifies a customer as good or bad, while maximizing the number of correct classifications and the distance from the decision boundary for the predictions. We can transform this as an optimization problem as follows:

$max_{\gamma,w,b}\gamma$ such that $y^{(i)}(w^T x^{(i)} + b) \geq \gamma, i = 1, ..., n$ and $||w|| = 1$,

where $y^{(i)}(w^T x^{(i)} + b)$ is the geometric margin for each customer, and the $||w|| = 1$ constraint guarantees that all geometric margin are at least γ.

4.6 Neural Network

Neural Network is a broad type of nonlinear parameterization that involve combinations of matrix multiplications and entrywise nonlinear operations. For our project, we use a multilayer perceptron, each with several hidden units using the ReLU activation function. Neural Networks has the advantage of automatically performing feature selection but comes with the disadvantage of having low interpretability.

4.7 Gradient Boosting

Gradient Boosting is an ensemble method that builds a stronger model from a class of weak learners. We use gradient boosting as it can prevent overfitting and is particularly suited for imbalanced data [8]. Gradient Boosting comes in the form of $\hat{F}_{a-1} = \sum_{i=1}^{n} \gamma_i g_i(x) + c$, where $g_i \in \mathbf{G}$, the class of weak learners. We start with the constant function $F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$ where L is the loss function and y_i are the labels. The algorithm will combine with previous models (weak learner) at each stage $F_a(x) = F_{a-1}(x) + \arg\min_{\gamma_a g_a : g_a \in \mathbf{G}} \sum_{i=1}^{n} L(y_i, F_{a-1}(x_i) + \gamma_a g_a(x_i))$. Gradient boosting also puts more weight on the incorrectly predicted samples at each step [5].

4.8 Stacked Model

The stacked model is an ensemble method that attempts to minimize the weakness of our weak learner but maximizes the strength of each individual model to become a meta-classifier [6]. In our case, we stack several of our algorithms together, including logistic regression, decision tree, support vector machine, random forest, and gradient boosting.

5 Experiments

5.1 Evaluation Metrics

In general, regardless of how we classify our customers as "good" and "bad", above 90 percent of customers will be defined as "good", and less than 10 percent as "bad". Therefore, to evaluate model performance, we will use the accuracy, Area under Receiver Operating Characteristic (AUC), and F1-score. It is critical for the model to both identify "bad" customers but also accurately classify "good" customers as well. Since we have an imbalanced dataset, a small number of correct or incorrect predictions can result in a large change in the AUC score. As a result, we choose the F1-score, the harmonic mean of Precision and Recall, as the major evaluation metric to look upon. Additionally, we will also look at the accuracy and the AUC to judge the models' performance.

5.2 Hyperparameter Tuning

We tuned the hyperparameters by using the Grid Search function in the *sklearn* library [10] for logistic regression, decision tree, support vector machine, and random forest. We also used the *Keras Tuner* [2] to tune the hyperparameters for the Neural Network. We found that a learning rate of 0.001, an epoch number of 12, along with the ReLU activation function in the hidden layers gave the best prediction accuracy. We also used a dropout rate of 0.25 and included Batch Normalization and early stopping to regularize the network and prevent overfitting.

5.3 Model Performance after EDA

As mentioned above, we will use the F1-score, AUC, and accuracy to evaluate our models' performance. The evaluation metrics are defined as follows, and the model performance after data pre-processing is presented in Table 1.

$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Customers}$

$F1score = \frac{2 * Precision * Recall}{Precision + Recall}$, where $Precision = \frac{True\ Bad\ Customers}{True\ Bad\ Customers + False\ Bad\ Customers}$ and $Recall = \frac{True\ Bad\ Customers}{True\ Bad\ Customers + False\ Good\ Customers}$

AUC = The area under the Receiver Operating Curve (ROC). The ROC curve plots the False Positive Rate against the True Positive Rate at different classification thresholds, and we record the area under the ROC.

Table 1: Model Performance after EDA

Model Name	Accuracy	Precision	Recall	AUC	F1-score
Logistic Regression	0.64	0.45	0.72	0.70	0.55
Decision Tree	0.79	0.79	0.80	0.88	0.79
Support Vector Machine	0.75	0.75	0.77	0.80	0.76
Random Forest	0.72	0.73	0.72	0.81	0.72
Gradient Boosting	0.63	0.64	0.63	0.72	0.63
Neural Network	0.80	0.81	0.79	0.89	0.80
Stacked model	0.81	0.81	0.81	0.81	0.81

From the table, we observe our models have room for improvement. For example, the baseline logistic regression model has a low F1-score of 0.55, an accuracy of 0.64, and an AUC of 0.70. The stacked model generated the highest F1-score of 0.81 among all models, but it only had an accuracy and AUC score of 0.81. We also observed in the models' confusion matrix that they all have relatively high false positives and false negatives. Therefore, we proceed with detecting what features might be causing these incorrect predictions.

5.4 Detecting High False positive and False negative

We analyzed the features which gave rise to high false positives and high false negatives and therefore lead to a wrong customer type prediction. We observed that for all customers who were not married, over 70 % of their predictions were incorrect. Similarly, for the customers with secondary level education and the customers who had a housing type as an apartment, over 70 % of their predictions were incorrect as well. Therefore, we suspected that these three features were preventing our models from making correct predictions.

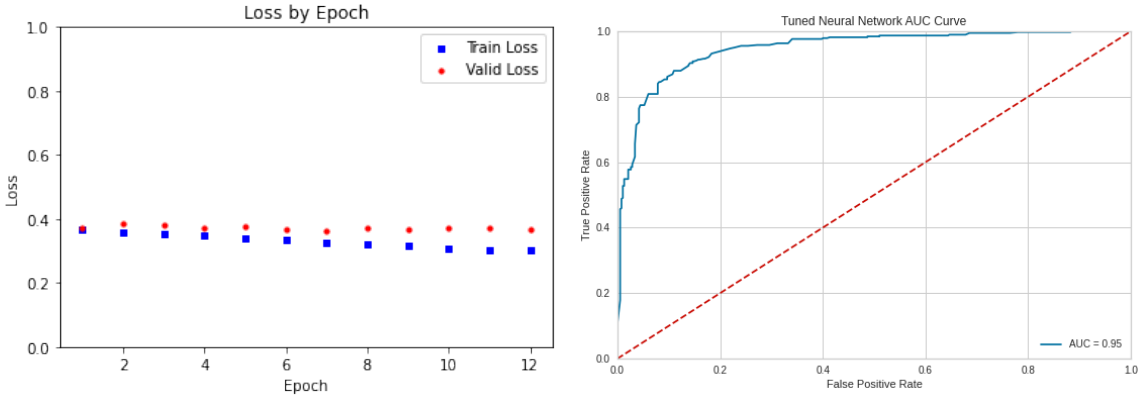
We first removed all customers that were not married. We see that all the models' performance increased significantly, as shown in the table. The F1-score, accuracy, and AUC values of all the models increased. We then attempted removing the customers that had a secondary level of education and those who lived in an apartment along with a combination of these three features, but the model's prediction improvement was minimal. Therefore, we retained the customers with a secondary level of education and those who lived in an apartment.

Table 2: Model Performance from filtered Married Condition

Model Name	Accuracy	Precision	Recall	AUC	F1-score
Logistic Regression	0.78	0.71	0.82	0.80	0.76
Decision Tree	0.87	0.88	0.85	0.93	0.86
Support Vector Machine	0.87	0.89	0.85	0.92	0.87
Random Forest	0.77	0.80	0.75	0.85	0.77
Gradient Boosting	0.84	0.87	0.82	0.87	0.84
Neural Network	0.88	0.89	0.87	0.95	0.87
Stacked model	0.89	0.91	0.88	0.88	0.89

5.5 Best Model Performance

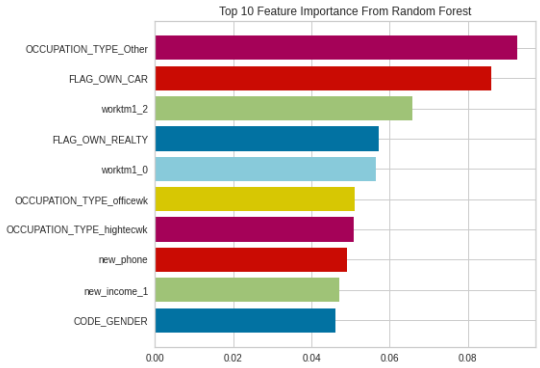
From the evaluation metrics in Table 2, we believe the Neural Network had the best performance. The Neural Network generated the highest AUC score of 0.95 (as shown in figure 2 (b), the second highest F1-score of 0.87, and the second highest accuracy of 0.88. The F1-score and the accuracy were only 0.02 away from the highest value of 0.89, which resulted from the stacked model. We can also see from figure 2 (a) that there is not a large difference between the validation loss and the training loss. Therefore, we may conclude that are Network is not overfitting the data.



(a) (b)

Figure 2: (a) Training and Validation Loss by Epoch (b) Neural Network AUC curve

5.6 Feature importance



(a)

Figure 3: Random Forest Feature Importance Graph

In addition to trying out different algorithms and comparing their performance, we also want to detect what features are the most important in those algorithms. Therefore, we generated feature importance graphs for logistic regression, decision tree, random forest (as shown in figure 3), support vector machine, and gradient boosting. We observed that *Income*, *House Type*, and *occupation type* were the top three predictors that occurred among the models. Therefore, we believe banks or financial institutions should take a careful look at personal income, housing type, and occupation type when handling credit card approvals. These features can be classified under personal assets and revenue will and can significantly affect the approval rate.

6 Conclusion and Future Works

We fitted seven different machine learning algorithms in this project. By performing hyperparameter tuning and feature selection, we were able to increase the models' prediction performance and therefore reduce the number of false positives and false negatives. Before feature selection, we observed the decision tree model performed better than Random Forest, an Ensemble model. We speculate this is due to the bias-variance tradeoff in which the decision tree has a higher variance but a lower bias. After feature selection, we observed the Neural Network and the Stacked model performed the best. The Neural Network worked well as it is capable of learning nonlinear, complex, and hidden correlations in the data. We also speculate the stacked model worked well as it is an ensemble method that minimizes the weakness and maximizes the strength of each individual model. However, we also note that the stacked model still has room for improvement. We believe this is because we do not have a huge dataset, and we would like to test this out if we get a larger dataset for future work.

There are several things we would like to do for future work. First, our dataset does not contain important features such as credit scores and transaction historical data for the customers. If we are able to retrieve this information, we might be able to improve our model's performance even more. Additionally, although we were able to reduce the number of false positives and false negatives for all our models, we still see a number of false positives and false negatives that cannot be ignored. Therefore, if we had more time, we would attempt more ways to tackle the imbalanced data, such as under-sampling, combining our current dataset with a larger dataset, etc. Finally, We have tried using causal inference to observe the causality between personal income as a treatment and credit card approval rate as our outcome. However, as our data set does not contain critical confounding factors such as credit score and default history, it was difficult to take further action.

7 Contributions

Fang focused on performing exploratory data analysis and implementing machine learning models. Gordon focused on improving the prediction accuracy of the models and detecting feature importance. Both members contributed to this final report.

8 Acknowledgements

We thank our TA, Beri for his generous help.

References

[1] Nitesh V Chawla et al. "SMOTE: synthetic minority over-sampling technique". In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[2] Francois Chollet et al. *Keras*. 2015. URL: https://github.com/fchollet/keras.

[3] Naman Dalsania, Devang Punatar, and Deep Kothari. "Credit Card Approval Prediction using Classification Algorithms". In: ().

[4] Leonard Flores et al. "A Classification Approach in the Probability of Credit Card Approval using Relief-Based Feature Selection". In: *2022 2nd Asian Conference on Innovation in Technology (ASIANCON)*. IEEE. 2022, pp. 1–7.

[5] Jerome H. Friedman. "Greedy function approximation: A gradient boosting machine." In: *The Annals of Statistics* 29.5 (2001). DOI: 10.1214/aos/1013203451.

[6] Jen Wadkins. *Simple Model Stacking, Explained and Automated*. https://towardsdatascience.com/simple-model-stacking-explained-and-automated-1b54e4357916, 2021.

[7] Md Golam Kibria and Mehmet Sevkli. "Application of Deep Learning for Credit Card Approval: A Comparison with Two Machine Learning Techniques". In: *International Journal of Machine Learning and Computing* 11.4 (2021).

[8] Olga Lyashevska et al. "Class imbalance in gradient boosting classification algorithms: Application to experimental stroke data". In: *Statistical Methods in Medical Research* 30.3 (2021), pp. 916–925.

[9] Matt Schultz. *2022 Credit Card Debt Statistics*. https://www.lendingtree.com/credit-cards/credit-card-debt-statistics/, 2022.

[10] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[11] Jong-Woo Song. "A Comparison of Classification Methods for Credit Card Approval Using R". In: *Journal of the Korean Society for Quality Management* 36.1 (2008), pp. 72–79.

[12] Xiao Song. *Credit Card Approval Prediction*. https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction,

[13] Zaima Zarnaz, Dipannita Biswas, and KM Azharul Hasan. "Credit Card Approval Prediction by Non-negative Tensor Factorization". In: *2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*. IEEE. 2021, pp. 319–323.