Credit Card Approval Classification

Author: Jesse Gempel

University of Missouri-St. Louis
College of Arts and Sciences

Python for Statistical Computing

December 6, 2024

1 Problem Definition

The premise of this machine learning project is to predict whether a theoretical client should be approved or rejected from obtaining a credit card. Each prediction centers around information that was collected from a credit card application, such as income type, education, and housing type. This project attempts to solve a binary classification problem with a supervised learning model.

2 Background

2.1 Motivation

Determining potential factors that correlate to a person's credit card approval serves as a motivation for undertaking this machine learning project. It is true that different credit card companies carry unique financial requirements. Some of these requirements, such as income, credit history, and even car ownership are reviewed with large variations of stringency. The same scenario can be observed with applications for mortgages, student loans, and car notes—among other lending methods. Oftentimes, clients face rejection on these lending methods despite carrying a substantial credit history or credit score. These clients become unsure—or even frustrated—when trying to determine why a bank would reject those individuals.

Perhaps these frustrations stem from the lack of detailed knowledge about a bank's decision making process. It is evident that banks observe a multitude of different factors when deciding a person's creditworthiness for a credit card. However, not many people understand which factors are most important (or least important) for this decision making process. That is why some individuals are rejected for reasons that appear to be trivial, nonexistent, or even "made up." This project aims to dissect many of these factors to discover the importance of each one. A model will then utilize these same factors to ultimately predict one's creditworthiness.

2.2 Machine Learning Source

The credit card classification problem was previously analyzed in a Jupyter notebook called *Credit_Card_Approval_Prediction*, by Іван Ніколайченко[1]. The author employed the *Credit Card Approval Prediction* dataset from Kaggle[2] to follow the source's project definition: to perform binary classification on a client's creditworthiness.

To act upon the project definition, Ніколайченко first divided his data into training and test sets. 70% of the data was relocated into the training set, while the remaining 30% was placed into the test set[1]. The author also utilized two models: one with LogisticRegression and one with RandomForestClassifier[1]. The ROC-AUC score served as the primary performance metric, although Ніколайченко employed precision, recall, F1, and support scores as well[1].

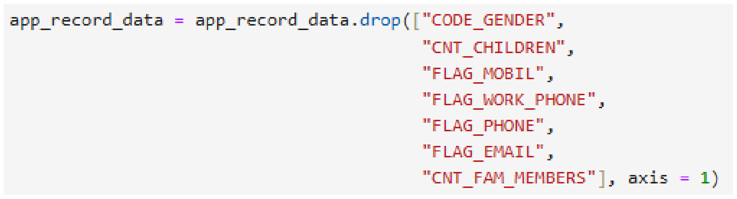
For the **LogisticRegression** model, the ROC_AUC score turned out to be **0.502**[1]. The best implementation of the author resulted in a *precision* score of **0.78**, *recall* score of **0.81**, *F1-score* of **0.79**, and *support* score of **5317**[1]. Meanwhile, the **RandomForestClassifier** model obtained an *AUC_ROC* score of **0.785**, a *precision* score of **0.82**, a *recall* score of **0.82**, an *F1_score* of **0.82**, and a *support* score of **5317**[1].

3 Project Implementation

3.1 Data Source

The *Credit Card Approval Prediction* dataset [2] was gathered from Kaggle, a website that allows users to access machine learning datasets and Python notebooks. This source contains two different csv files: application_record.csv and credit_record.csv.

The **application_record.csv** file stores 18 attributes from 438,557 clients—each attribute containing information shared by a client through a credit card application. The attributes are as follows: ID, CODE_ GENDER, FLAG_OWN_CAR, FLAG_OWN_REALTY, CNT_CHILDREN, AMT_INCOME_ TOTAL, NAME_INCOME_TYPE, NAME_EDUCATION_TYPE, NAME_FAMILY_STATUS, NAME_ HOUSING_TYPE, DAYS_BIRTH, DAYS_ EMPLOYED, FLAG_MOBIL, FLAG_WORK_PHONE, FLAG_PHONE, FLAG_EMAIL, OCCUPATION_TYPE, CNT_FAM_MEMBERS, and MONTHS_ BALANCE. Every attribute, with the exception of ID, serves as a possible *predictor*; some of these predictors were not used in the classification problem as shown below:



```
app_record_data = app_record_data.drop(["CODE_GENDER",
                                        "CNT_CHILDREN",
                                        "FLAG_MOBIL",
                                        "FLAG_WORK_PHONE",
                                        "FLAG_PHONE",
                                        "FLAG_EMAIL",
                                        "CNT_FAM_MEMBERS"], axis = 1)
```

Figure 1: A list of attributes from the application_record.csv dataset that were dropped. These attributes store gender, family member count, and contact information—none of which influence the creditworthiness of a client.

The **credit_record.csv** file stores 1,048,575 samples of month-by-month credit history for some of the clients in application_record.csv. This dataset stores 2 attributes—ID and MONTHS_BALANCE—plus the target variable STATUS. The *MONTHS_BALANCE* predictor stores the number of months prior to the month of the credit application. The *STATUS* target variable classifies the *worst* payment status that a customer obtained on any previous loan. The 8 classifications are as follows:

1. C (bill paid for the month)
2. X (no bill for the month)
3. Numbers 0, 1, 2, 3, 4, and 5 (the number of months that a bill was late)

It is best that the two datasets are used for binary classification—not multi-class classification. As a result, I decided to modify the dataset's payment status classifications to reflect whether a customer should be approved or rejected for a credit card. Binary classification was arranged in this manner:

1. 1 for *approved* (includes categories C, X, and 0)
2. 0 for *rejected* (includes categories 1, 2, 3, 4, and 5)

This classification setup dictates that a client whom was less than 30 days late on a bill should be *approved* for a credit card. Any client whom does not meet the criteria should be *rejected*. After merging the two datasets together and mapping predictor information by client ID, the clients were divided into the two classes as shown below:

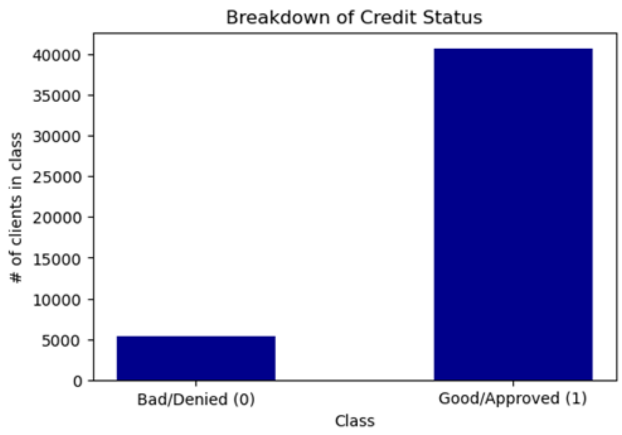


Figure 2: The **Bad/Denied (0)** class contains 5,350 samples while the **Good/Approved (1)** class contains 40,635 samples. The credit_record.csv dataset includes multiple entries with the same client ID, and some IDs were not included in both datasets. For these reasons, the number of samples significantly decreased to a total of **45,985 clients**.

The application_record.csv and credit_record.csv datasets' features were merged together for simplicity's sake. Once the merging process was complete, the features that were not deleted from the two datasets were compared to the Status target variable, as shown below:

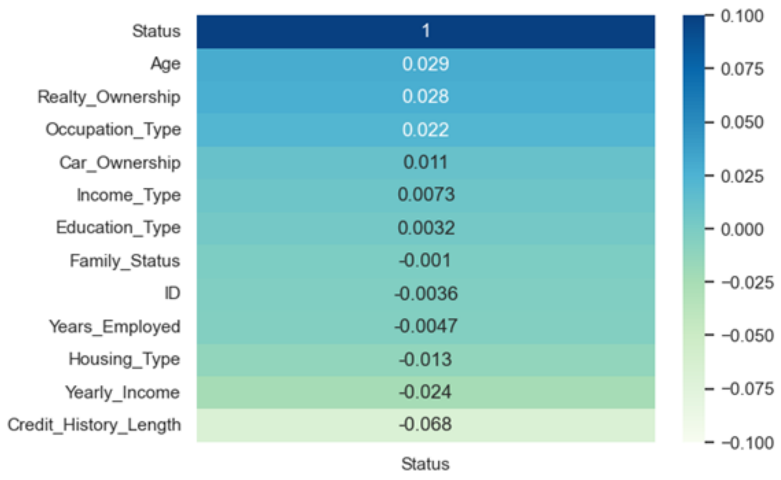
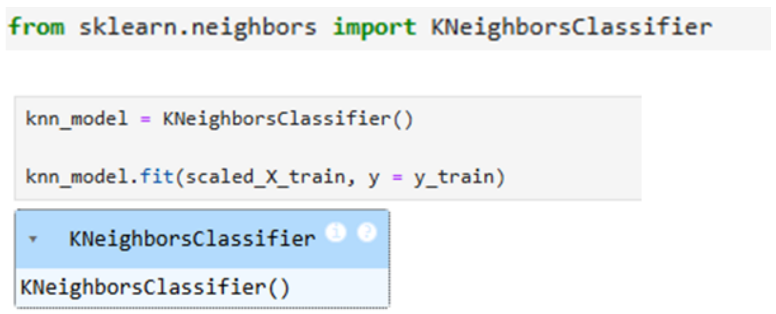


Figure 3: A heatmap containing the target variable **Status** and all features that were not deleted (Refer to Figure 1 to view deleted features). A higher numerical value reflects a stronger correlation between a specific feature and the **Status** variable.

3.2 Machine Learning Algorithm

To determine the proper machine learning algorithm to use, three requirements needed to be satisfied. First, the algorithm must be a *supervised learning technique*. The two datasets involve the use of labels, so the algorithm works with these labels to determine patterns between the inputs and outputs. The algorithm must also be simple and effective at its task. More complex tools can proficiently solve the classification problem, but such tools are unnecessary for a relatively simple problem. Lastly, the algorithm must be able to handle nonlinear relationships between the target variable (payment status) and the datasets' predictors.

The k-Neighbors Classifier satisfied the three requirements mentioned above. For this reason, we choose the **k-Neighbors Classifier** algorithm.



```
from sklearn.neighbors import KNeighborsClassifier
```

```
knn_model = KNeighborsClassifier()

knn_model.fit(scaled_X_train, y = y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier()
```

Figure 4: Implementation of the k-Neighbors Classifier algorithm in Python.

3.3 Performance Evaluation

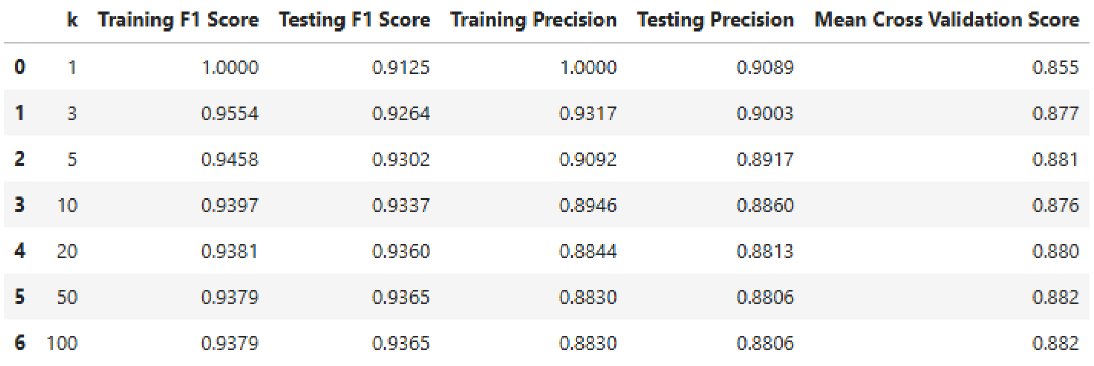
Three performance metrics were used to evaluate the performance of the k-Neighbors Classifier model: F1-score, precision, and mean stratified k-fold cross validation score.

As shown in Figure 2, the two classes were heavily imbalanced. Approximately 88.3% of the samples belonged to the Good/Approved (1) class, whereas only 11.7% of clients were in the Bad/Denied (0) class. Due to this imbalance of classes, the **F1-score** served as the primary metric for evaluating the model. The benchmark for the F1-score was **0.80** since the source of this study obtained scores up to 0.82 with similar models[1].

The precision score is a metric that is similar to the F1-score. It is better to have more than one metric that accommodates imbalanced classes, as in this study. For this reason, the **precision score** was also used with a benchmark of **0.85** (since the source obtained scores between 0.75 and 0.79 with similar algorithms)[1].

The **mean stratified k-fold cross validation** metric involves splitting a dataset into a k number of folds. However, it is different from standard mean k-fold cross validation because it takes the imbalance of classes into consideration[3]. Since the data involves an approximate 88%/12% split, it is best to use this metric as well. The benchmark for the mean stratified k-fold cross validation score was **0.80** since the value is close to the percentage of samples in the larger class.

4 Results



	k	Training F1 Score	Testing F1 Score	Training Precision	Testing Precision	Mean Cross Validation Score
0	1	1.0000	0.9125	1.0000	0.9089	0.855
1	3	0.9554	0.9264	0.9317	0.9003	0.877
2	5	0.9458	0.9302	0.9092	0.8917	0.881
3	10	0.9397	0.9337	0.8946	0.8860	0.876
4	20	0.9381	0.9360	0.8844	0.8813	0.880
5	50	0.9379	0.9365	0.8830	0.8806	0.882
6	100	0.9379	0.9365	0.8830	0.8806	0.882

Figure 5: Performance metric results after running the k-Neighbors Classifier model with seven different configurations of k. **NOTE**: The column "Mean Cross Validation Score" refers to the stratified—not the regular—cross validation scores.

As shown in Figure 5, three different metrics were used: F1-score, precision, and mean stratified mean k-fold cross validation score. Compared to Іван Ніколайченко's top F1-score values of 0.79 and 0.82[1], the k-Neighbors Classifier model obtained **testing F1-scores** between **0.91** and **0.93**. Not only did the model surpassed the benchmark of 0.80, it also outperformed the source's highest F1-scores.

A similar scenario occurred with the precision metric as well. According to Ніколайченко's Jupyter notebook, the LogisticRegression and RandomForestClassification models obtained precision values of 0.78 and 0.82 respectively[1]. My k-Neighbors Classifier model's **testing precision scores** ranged between **0.88** and **0.91**—a range that outperforms both source models as well as the benchmark score of 0.85.

The **mean stratified k-fold cross validation score** ranged between 0.85 and 0.88. Scores for all values of k appeared to surpass the benchmark of 0.80. However, Ніколайченко's Jupyter notebook does not mention this specific metric, so it is impossible to compare my model's performance to my source's performance in that regard.

5 Conclusion

The metrics acquired by the k-Neighbors Classifier model suggest that machine learning is suitable for classifying the creditworthiness of a client. For all seven configurations of k (as shown in Figure 5), the F1, precision, and mean stratified k-fold cross validation scores all managed to surpass the metrics' benchmarks. This metric comparison can be applied not only to benchmarks, but to the source Jupyter notebook's results as well. Every possible score with the k-Neighbors Classifier model proved to be better than those of Ніколайченко. It is surprising that the source's models—LogisticRegression and RandomForestClassifier—used relatively complex algorithms, yet a simple model yielded better results. These observations support the hypothesis that machine learning algorithms can accurately perform binary classification on a person's worthiness for a credit card.

Other discoveries include the ability to pinpoint correlations between a person's creditworthiness and specific features. It was discovered in Figure 3 that age, home ownership, and occupation type are relatively strong predictors for creditworthiness. Housing type, income, and credit history length appear to have strong (but somehow negative) correlations as well. Future clients should consider focusing on these factors to increase their credit card approval odds.

6 References

[title page] University of Missouri–St. Louis. (n.d.). (2024). *Horizontal university logo.* Retrieved from https://www.umsl.edu/branding/logos/index.html

[1] Ніколайченко, I. (2024, October 16). *Credit_ Card_ Approval_ Prediction* [Jupyter Notebook]. Kaggle. Retrieved October 24, 2024, from https://kaggle.com/code/frotfl1pp/credit-card-approval-prediction

[2] Moneyman. (2020, March 24). *Credit Card Approval Prediction* [Data set]. Kaggle. Retrieved October 24, 2024, from https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction

[3] *Stratified K Fold Cross Validation* (2023, January 10). GeeksforGeeks. https://www.geeksforgeeks.org/stratified-k-fold-cross-validation/#