Predicting Credit Card Application Approval

Contents

1. Introduction

1.1 Context and Objective

Credit cards have been consistently replacing physical money ever since they first topped cash purchases in 2005. As our purchases become more digital, with most consumer businesses taking their businesses online, especially during the coronavirus pandemic, having a credit card has evolved from being a symbol of a good financial credit score to more of a banking necessity.

However, banks do not just issue credit cards to *everyone.* The Federal Reserve Bank of New York concluded in 2018 that rejection rates for credit cards rose every year [1]. And as of 2020, credit card rejections have accounted for the biggest percentage of credit rejections of the year, over 61 percent [2]. Being denied a credit card is a frustrating situation for a consumer, and the fact that banks provide no details about *why* the application was rejected makes it even more frustrating, and this ambiguity causes further rejections.

Therefore, in this project, we aim to predict whether a consumer's application for a credit card will be approved or rejected, with the help of unsupervised machine learning. The machine learning model takes various variables into account that will be discussed in the technical details of the project and predicts a binary result of whether the user's application will be *accepted* or *denied.*

1.2 Significance

The output of the model is expected to benefit both, the credit card issuing banks and the consumers. Using the model, the banks can automate the tedious process of sorting through credit card applications. On the other hand, consumers can evaluate their portfolios before applying for a credit card to make sure that they are eligible for one. As we have mentioned before, the COVID-19 pandemic has driven up the demand and applications for credit cards and having an automated process of application evaluation is crucial to efficiently handle every application.

2. Data Transformation

2.1 Data Description

This project utilizes the Credit Approval Data Sets from the Machine Learning Repository of the University College, London. The link to the data set: UCI Machine Learning Repository: Credit Approval Data Set.

The features of this dataset have been anonymized to protect the privacy. However, after observing the types of the features and their values, we believed that some features in the dataset are ‘Prior default’, ‘Years employed’, ‘Credit score’, ‘Income level’, ‘Age’, ‘Gender’, ‘Debt’ and finally ‘Approval Status’.

In this dataset, there is a good mix of attributes: continuous, nominal with small numbers of values, and nominal with larger numbers of values. There are also a few missing values. Therefore, the data should be preprocessed before inputting it into any machine learning model.

To process the data, we used the Python language in a Jupyter Notebook. We used the ‘sklearn’ package to handle the machine learning part.

2.2 Preprocessing

Firstly, we focus on the missing values. Much importance is given to missing values because they can affect the performance of a machine learning model heavily. While ignoring the missing values, a machine learning model may miss out on information about the dataset that may be useful for its training. Then, there are many models which cannot handle missing values implicitly. As observed, the missing values in the dataset are labeled with '?'. They are temporarily replaced by ‘NaN’, which represents the null value in the Python language.

Since there are missing values in both the numerical and non-numerical features, the missing values will be handled differently for each case. For the numerical features, the missing values were imputed using a strategy called mean imputation, in which the missing values were replaced by the mean value of that feature. For the non-numerical features, the missing values were replaced by the most frequent value of that feature.

Now that the missing values are successfully handled, we will focus on the following steps, which consist of converting all features to numeric, split the dataset for training and scale the feature values to a uniform range. The reason why all the non-numeric values are converted into numeric ones is because some machine learning models such as XGBoost require the data to be in a strictly numeric format. Additionally, when all data are numerical, this results in faster computation. A technique called label encoding is used. It encodes labels with a value between 0 and n-1, where n is the number of distinct labels. If a label repeats it assigns the same value to as assigned earlier, thus the categorical values are converted into numeric values.

Then the data is split into training data and testing data. Only one-third of the data is used as test data. The training data is explicitly used to train our models and the testing data is used to assess the performance of the models.

Then, the values of the training data and testing data will be scaled to a uniform range separately. This is because the dataset contains values from several ranges. Some features have a value range of 0 - 28, some have a range of 2 - 67, and some have a range of 1017 - 100000.

2.3 Feature Selection

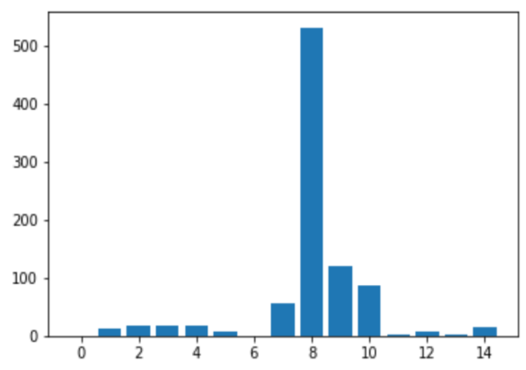
Irrelevant or partially relevant features can negatively impact the performance of many models, especially linear algorithms like linear and logistic regression. To remove these features, feature selection is performed. Feature selection is a process where you automatically select those features in your data that contribute most to the prediction variable or output in which you are interested. To make our credit card approval prediction model as accurate as possible, we aim to find carefully the best features to be inputted in our classifier and predictive models so that the output is the best possible with highest accuracy hence eliminating noise which in this case could be irrelevant attributes contributing negatively to the model and our focus can be set on the most important ones.

Three benefits of performing feature selection before modelling your data are:

- Reduces Overfitting: Less redundant data means less opportunity to make decisions based on noise.
- Improves Accuracy: Less misleading data means modelling accuracy improves.
- Reduces Training Time: Less data means that algorithms train faster.

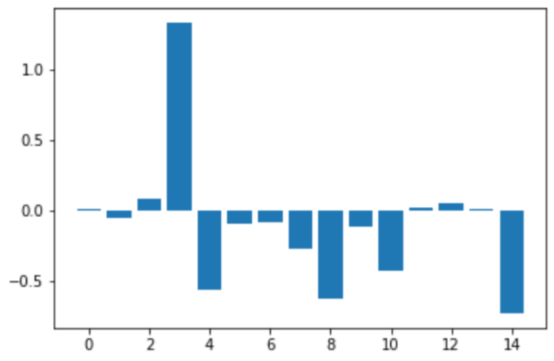
We will use a few algorithms to hence reduce errors and improve accuracy in running the further classifiers to extract which attributes are of more importance and should be used as inputs.

For classification predictive modelling, ANOVA f-test is a commonly used feature selection for numerical data when fitting and evaluating a classification model. Using this method, we get the following results for the relative importance of the input variables and based on ANOVA f-test, we can assume 7, 8, 9 and 10 are most relevant features.



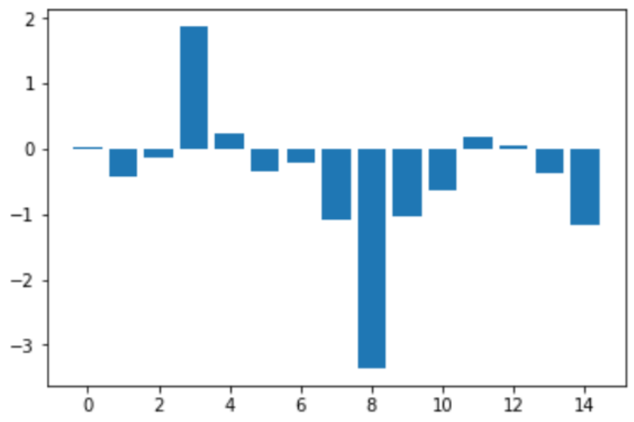
```
Feature: 0, Score: 0.4
Feature: 1, Score: 11.5
Feature: 2, Score: 16.4
Feature: 3, Score: 18.3
Feature: 4, Score: 18.4
Feature: 5, Score: 6.7
Feature: 6, Score: 0.0
Feature: 7, Score: 57.1
Feature: 8, Score: 530.5
Feature: 9, Score: 120.2
Feature: 10, Score: 86.8
Feature: 11, Score: 2.8
Feature: 12, Score: 6.2
Feature: 13, Score: 1.1
Feature: 14, Score: 14.1
```

Next, in linear regression, coefficients are the values that multiply the predictor values. Signs of coefficients indicate whether the relationship between predictor and response variable are positively or negatively related i.e., positive indicates predictor variable affects response variable positively and vice versa. Hence, a linear regression model is fitted on the dataset and the coefficients calculated for each input variable is retrieved. These coefficients can provide the basis for a crude feature importance score. The following are the results obtained from it. Here, for example, we see that features 0, 2, 3 and 12 are positively related.



```
Feature: 0, Score: 0.0
Feature: 1, Score: -0.1
Feature: 2, Score: 0.1
Feature: 3, Score: 1.3
Feature: 4, Score: -0.6
Feature: 5, Score: -0.1
Feature: 6, Score: -0.1
Feature: 7, Score: -0.3
Feature: 8, Score: -0.6
Feature: 9, Score: -0.1
Feature: 10, Score: -0.4
Feature: 11, Score: 0.0
Feature: 12, Score: 0.1
Feature: 13, Score: 0.0
Feature: 14, Score: -0.7
```

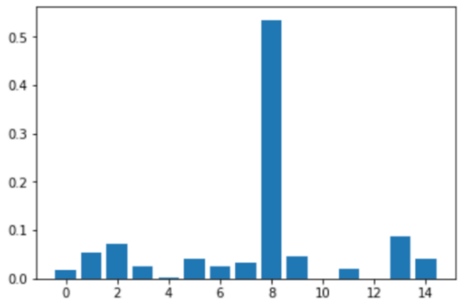
Logistic regression models the probabilities for classification problems with two possible outcomes. It's an extension of the linear regression model for classification problems. It treats classes as binary – 0 or 1. Hence, a Logistic Regression in the same way is also fitted on the dataset in order to evaluate coefficients for each input variable. Here are the following results:



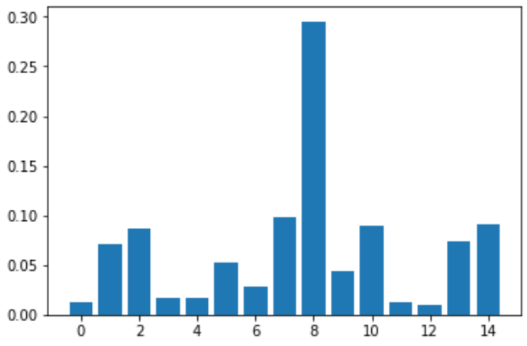
```
Feature: 0, Score: 0.0
Feature: 1, Score: -0.4
Feature: 2, Score: -0.1
Feature: 3, Score: 1.9
Feature: 4, Score: 0.2
Feature: 5, Score: -0.4
Feature: 6, Score: -0.2
Feature: 7, Score: -1.1
Feature: 8, Score: -3.4
Feature: 9, Score: -1.0
Feature: 10, Score: -0.6
Feature: 11, Score: 0.2
Feature: 12, Score: 0.0
Feature: 13, Score: -0.4
Feature: 14, Score: -1.2
```

For selecting best features using the Decision Tree Classifier Model, the importance is calculated as the decrease in node impurity weighted by the probability of reaching that node where the probability of the node can be calculated by number of samples in the node divided by total number of samples. Again, higher value implies more important feature. Here, most important features are 2, 8 and 13.

```
Feature: 0, Score: 0.02
Feature: 1, Score: 0.05
Feature: 2, Score: 0.07
Feature: 3, Score: 0.03
Feature: 4, Score: 0.00
Feature: 5, Score: 0.04
Feature: 6, Score: 0.03
Feature: 7, Score: 0.03
Feature: 8, Score: 0.54
Feature: 9, Score: 0.05
Feature: 10, Score: 0.00
Feature: 11, Score: 0.02
Feature: 12, Score: 0.00
Feature: 13, Score: 0.09
Feature: 14, Score: 0.04
```

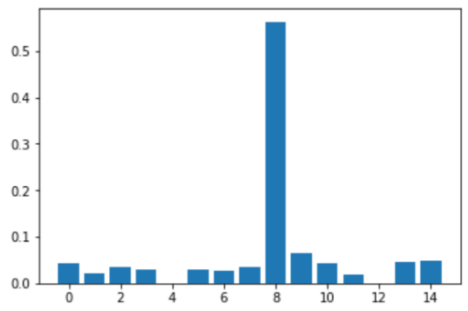


In the case where we select best features using Random Forest Model, the best features as shown below are 2, 7,14 and 8. It is basically in short, a set of Decision Trees. They use gini importance (The features for internal nodes are selected with some criterion, which for classification tasks can be gini impurity or information gain, and for regression is variance reduction. We can measure how each feature decrease the impurity of the split (the feature with highest decrease is selected for internal node). and mean decrease accuracy (not implemented in scikit learn package) mainly to evaluate the importance.



```
Feature: 0, Score: 0.013
Feature: 1, Score: 0.071
Feature: 2, Score: 0.087
Feature: 3, Score: 0.017
Feature: 4, Score: 0.017
Feature: 5, Score: 0.052
Feature: 6, Score: 0.028
Feature: 7, Score: 0.098
Feature: 8, Score: 0.295
Feature: 9, Score: 0.043
Feature: 10, Score: 0.090
Feature: 11, Score: 0.013
Feature: 12, Score: 0.010
Feature: 13, Score: 0.074
Feature: 14, Score: 0.091
```

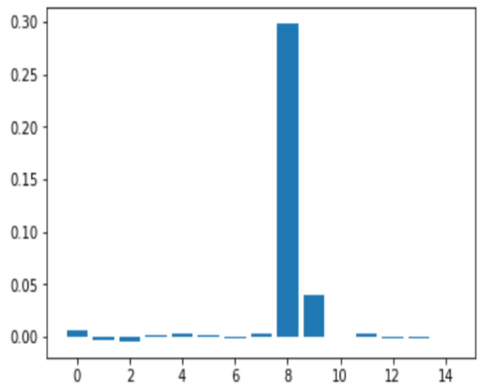
Similar implementation is again used by linear regression. The following results are shown below. We can hence see that the important features based on this classifier are 8,9,10,14 and 0. All scores other than 8 and 2 are similar so we can assume they are attributes that may have a chance to be correlated in other sense.



```
Feature: 0, Score: 0.042
Feature: 1, Score: 0.021
Feature: 2, Score: 0.035
Feature: 3, Score: 0.030
Feature: 4, Score: 0.000
Feature: 5, Score: 0.029
Feature: 6, Score: 0.026
Feature: 7, Score: 0.034
Feature: 8, Score: 0.563
Feature: 9, Score: 0.065
Feature: 10, Score: 0.044
Feature: 11, Score: 0.018
Feature: 12, Score: 0.000
Feature: 13, Score: 0.046
Feature: 14, Score: 0.048
```

The final one is with KNN classifier. Here we see the best features are given by 8, 0 and 9.

```
Feature: 0, Score: 0.00649
Feature: 1, Score: -0.00346
Feature: 2, Score: -0.00476
Feature: 3, Score: 0.00130
Feature: 4, Score: 0.00346
Feature: 5, Score: 0.00130
Feature: 6, Score: -0.00216
Feature: 7, Score: 0.00216
Feature: 8, Score: 0.29827
Feature: 9, Score: 0.04026
Feature: 10, Score: 0.00000
Feature: 11, Score: 0.00303
Feature: 12, Score: -0.00173
Feature: 13, Score: -0.00216
Feature: 14, Score: 0.00000
```



The above procedure involves running many types of classifiers to find input importance reason being different algorithms have different flaws and advantages. Running more can give a data scientist a glimpse into the actual scenario rather than the case where it can be biased. For example, the features return by XGBoost do not fall common in much with others except for 8. We can hence assume, the important features are 2,7,8,9,10,14 and we can base our next modelling step based on it to reduce noise and increase accuracy.

3. Prediction Models

3.1 Grid Search

The models used in this project are logistic regression, random forest classifier, XGBoost, k-nearest neighbors classifier and AdaBoost. The models are trained on the training data. Then the testing data is used to assess the performance of the models.

A model consists of different hyperparameters. Therefore, for each model, a set of parameters have to be tuned so as to obtain the optimal accuracy for that model. A grid search of each model parameters is performed to obtain the set of optimal parameters. The package ‘scikit’ has a function for this process called ‘GridSearchCV()’.

3.2 Logistic Regression

For the logistic regression, the parameters tuned and their values are:

- tol: [0.1, 0.01, 0.001 ,0.0001, 0.00001]
- max_iter: [multiples of 100 from 100 to 10000]

The best set of parameters are {max_iter: 100, tol: 0.1}. When using this set of parameters, the accuracy of the logistic regression model is 0.838. This is how it looks:

3.3 Random Forest Classifier

For the random forest classifier, the parameters tuned and their values are:

- max_features: [2, 3, 4, 5, 6, 7]
- min_samples_leaf: [2, 3, 4, 5, 6]
- max_depth: [8, 9, 10, 11, 12, 13, 14, 15, 16, 17]

The best set of parameters are {max_features: 7, min_samples_leaf: 2, max_depth: 13}. When using this set of parameters, the accuracy of the random forest classifier model is 0.860.

3.4 XGBoost

For the XGBoost model, the parameters tuned and their values are:

- max_features: [2, 3, 4, 5, 6, 7]
- eta: [0.05, 0.10, 0.15, 0.20, 0.25, 0.30]
- max_depth: [3, 4, 5, 6, 8, 10, 12, 15]
- min_child_weight: [1, 3, 5, 7]
- gamma: [0.0, 0.1, 0.2, 0.3, 0.4],
- colsample_bytree: [0.3, 0.4, 0.5, 0.7]

The best set of parameters are {colsample_bytree: 0.3, eta: 0.2, gamma: 0.4, max_depth: 3, min_child_weight: 5}. When using this set of parameters, the accuracy of the XGBoost model is 0.860.

3.5 K-Nearest Neighbors

For the random forest classifier, the parameters tuned and their values are:

- n_neighbors: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
- weights: ['uniform', 'distance']
- metric: ['euclidean', 'manhattan']

The best set of parameters are {metric: ‘euclidean’, ‘n_neighbors’: 15, ‘weights’: ‘distance’}. When using this set of parameters, the accuracy of the k-nearest neighbors model is 0.855.

3.6 AdaBoost

For the AdaBoost, the parameters tuned and their values are:

- n_estimators: [10, 50, 100, 500]
- learning_rate: [0.0001, 0.001, 0.01, 0.1, 1.0]

The best set of parameters are {learning_rate: 0.0001, n_estimators: 10}. When using this set of parameters, the accuracy of the random forest classifier model is 0.842.

4. Conclusion

After extensive application of machine learning techniques, there are 2 models with the best accuracy score, random forest classifier and XGBoost. These 2 models can be used to obtain an accuracy of 0.860. However, we believe the random forest classifier is better as it is easier to use and takes less computational time.

References

1. Jason Brownlee (June 5, 2020). *How to Perform Feature Selection With Numerical Input Data*
   Retrieved from:https://machinelearningmastery.com/feature-selection-with-numerical-input-data/#:~:text=ANOVA%20f%2Dtest%20Feature%20Selection,-ANOVA%20is%20an&text=The%20ANOVA%20method%20is%20a,as%20an%20ANOVA%20f%2Dtest.&text=The%20results%20of%20this%20test,be%20removed%20from%20the%20dataset.

2. Federal Reserve Bank of New York (December 03, 2018). *Credit Access Survey Shows Credit Tightening as Application Rejection Rates and Account Closings Increase in 2018.* Retrieved from: https://www.newyorkfed.org/newsevents/news/research/2018/an181203

3. Allie Johnson (November 23, 2020). *Denied: Many Americans have been turned down for credit in 2020.*
   Retrieved from: https://www.bankrate.com/finance/credit-cards/credit-denial-survey/#:~:texto%20can%27t%20get%20credit,rejected%20for%20a%20credit%20card

4. Link to the dataset: https://archive.ics.uci.edu/ml/datasets/Credit+Approval

5. Piotr Płoński (June 29, 2020). *Random Forest Feature Importance Computed in 3 Ways with Python*
   Retrieved from: https://mljar.com/blog/feature-importance-in-random-forest/#:~:text=Random%20Forest%20Built%2Din%20Feature%20Importance&text=It%20is%20a%20set%20of%20Decision%20Trees.&text=We%20can%20measure%20how%20each,average%20it%20decreases%20the%20impurity