GROUP 3

CREDIT CARD ALLOCATION PREDICTION

N. Ashraff - s14308
D. S. Sonnadara - s14508
D. M. P. Disanayaka - s14472

Abstract

This report aims to outline the findings of the descriptive and advanced analysis carried out on the Credit Card Approval Prediction dataset sourced from Kaggle. Banks have seen a rapid increase in applicants looking to apply for the use of a credit card. With the high influx of potential customers banks are required to streamline their application processes so that they may filter out customers in terms of the risk of liability they pose to the business. Different graphs were plotted as part of the descriptive analysis which were used to create the foundation for the subsequent advanced analysis. Different machine learning models were fitted, their accuracies and precisions were compared to choose the final model to be used in the data product.

Table of Contents

List of Figures

List of Tables

1 Introduction

From its inception, credit cards have offered significant advantages over other forms of money as they are pocket size, easily portable, relatively secure and have no intrinsic value in themselves. Credit cards also provide users the option to pay their bills over a modest period of time. Banks have also pushed towards offering credit card services to more customers due to the significant long term returns it can offer. With the increase in demand for customers looking to apply for credit cards, banks are required to streamline their application processes to filter out customers that could be a potential liability. By doing so banks are able mitigate any potential losses to the business while also saving on the time and effort it might take process and maintain these customers.

2 Objectives

This project aims to create a machine learning model that can be used to predict if a customer is 'good' or 'bad' based on a given set of attributes. In this context 'good' refers to a customer who is predicted to make credit card payments on time, while 'bad' refers to a customer who is predicted to not make credit card payments on time. This model will then be used to build a data product that will be used by bank employees to filter out potentially risky customers.

3 About the dataset

The dataset is divided into 2 parts, one part consists of personal information of a customer while the other contains information collected on credit payments of the relevant customer at the bank, the description of the variables are as follows.

Variable	**Description**
ID	Client Number
CODE_GENDER	Gender
FLAG_OWN_CAR	Is there a car
FLAG_OWN_REALTY	Is there a property
CNT_CHILDREN	Number of children
AMT_INCOME_TOTAL	Annual income
NAME_INCOME_TYPE	Income category
NAME_EDUCATION_TYPE	Education level
NAME_FAMILY_STATUS	Marital status
NAME_HOUSING_TYPE	Way of living
DAYS_BIRTH	Days since born
DAYS_EMPOYED	Days since start of employment
FLAG_MOBIL	Is there a mobile phone
FLAG_WORK_PHONE	Is there a work phone
FLAG_PHONE	Is there a phone
FLAG_EMAIL	Is there an email
OCCUPATION_TYPE	Occupation
CNT_FAM_MEMBERS	Family size
MONTHS_BALANCE	The month of the extracted data is the starting point, backwards, 0 is the current month
STATUS	0: 1-29 days past due 1: 30-59 days past due 2: 60-89 days overdue 3: 90-119 days overdue 4: 120-149 days overdue 5: Overdue or bad debts, write-offs for more than 150 days C: paid off that month X: No loan for the month

TABLE 1: DESCRIPTION OF VARIABLE

4 Data Preparation

The response variable ‘STATUS’ consists of payment information from multiple months for a specific client ID, based on this information customers who have had at least one payment delayed by more than 2 months were considered as potentially risky , and were marked as ‘Yes-1’ in the new response variable ‘Target’, customers who have not had a payment delayed by more than 2 months were considered as not risky, and labeled as ‘No-0’ in the response variable ‘Target’. This new response variable will be used to classify potential customers accordingly.

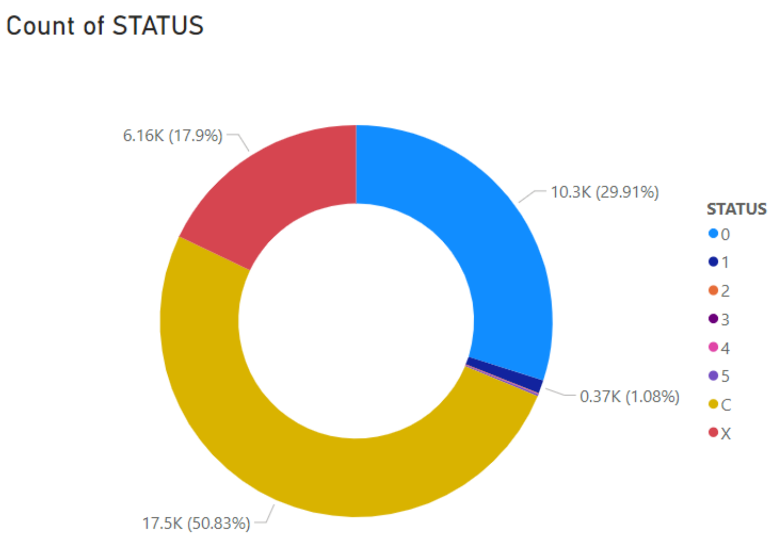


FIGURE 1: COUNTS OF STATUS

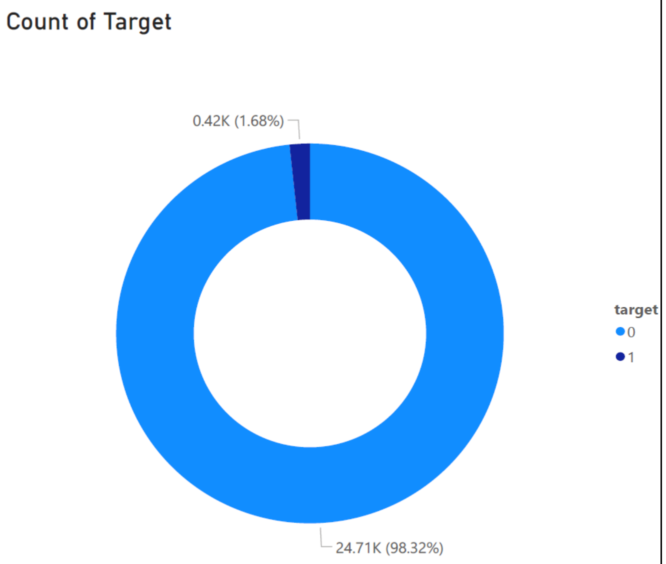


FIGURE 2: COUNTS OF TARGET

Records containing missing values were removed, variables containing two values were identified and encoded as binary variables. These include Gender, having a car, having a house, having a phone, having a work phone, and having an email. Continuous variables of Income, Number of children, Age, Working Years and Family Size were encoded as categorical variables by considering the trend in each variable individually. The updated levels of the variables are as follows

- Number of children – 0,1, 2 or more
- Annual Income – Low, Medium, High
- Age – Lowest, Low, Medium, High, Highest
- Working Years – Lowest, Low, Medium, High, Highest
- Family Size – 1,2, 3 or more

The number of values under the Occupation Type, Income Type and Education level variables were reduced by the means of combining them into similar groups so that it will be easier when fitting the machine learning models, the updated levels of the variables are as follows

- Income Type – State Servant, Commercial Associate, Working
- Occupation Type – Laborwk, hightecwk, officewk
- Education Level – Higher Education, Incomplete Higher, Lower Secondary, Secondary/Secondary special

5 Important Results of the Descriptive Analysis

5.1. Graph of Education Level with 'Yes' values from Target

Looking at the customers who hold a potential risk in terms of payment of credit card dues, we see that most customers are from a secondary/ secondary special background level. An interesting finding from this data is that customers with an academic degree are notably absent from this chart. This can be attributed to the fact that they may be financially stable and as a result have the capacity to not be considered as a potentially risky customer.

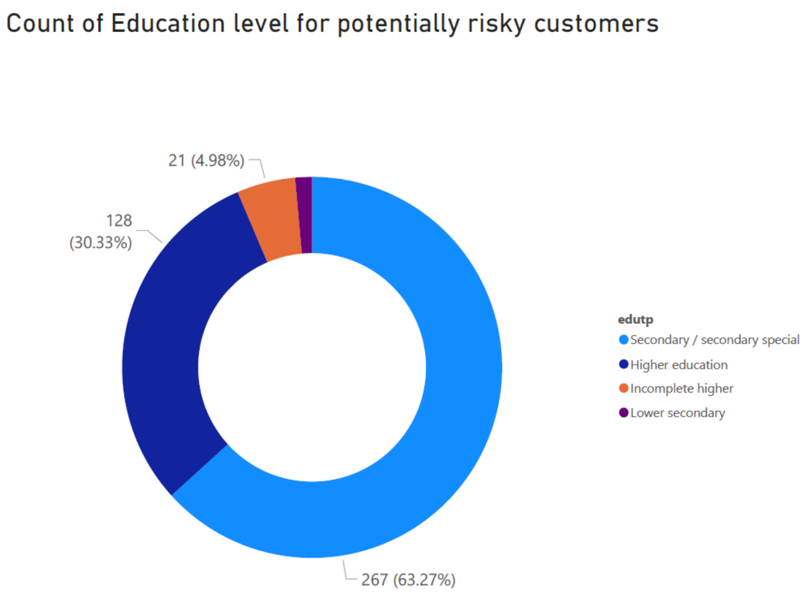


FIGURE 3: COUNTS OF EDUCATION LEVEL FOR POTENTIALLY RISKY CUSTOMERS

5.2. Graph of Education Level with 'No' values from Target

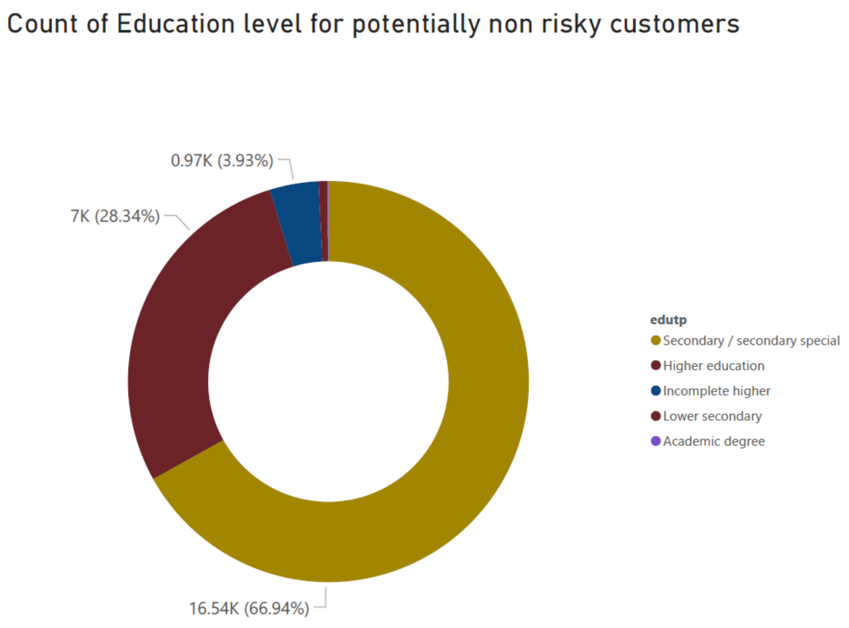


FIGURE 4: COUNTS OF EDUCATION LEVEL FOR POTENTIALLY NON RISKY CUSTOMERS

Looking at the customers who do not hold a potential risk in terms of payment of credit card dues, it was once again noted that most customers were from the secondary/ secondary special background level. It was also observed that all the individuals who hold a degree are not seen as risky customers. Thereby it can be assumed that the level of education could have some effect on the overall model.

5.3 Boxplot of Target with Days Employed

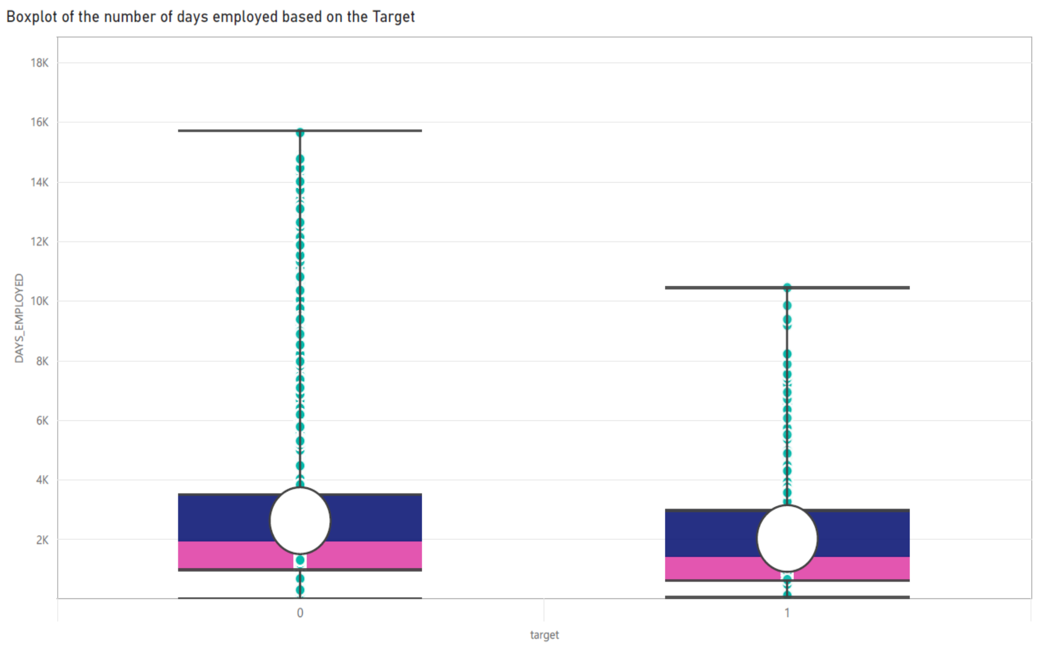


FIGURE 5: BOXPLOT OF TARGET WITH DAYS EMPLOYED

	Min	1st Quartile	Median	Mean	3rd Quartile	Max
Days Employed where Target = 0	17	985	1953	2635	3503	15713
Days Employed where Target = 1	65	622	1430	2037	2978	10454

TABLE 2: DESCRIPTIVE STATISTICS OF DAYS EMPLOYED FOR THE DIFFERENT TARGET VALUES

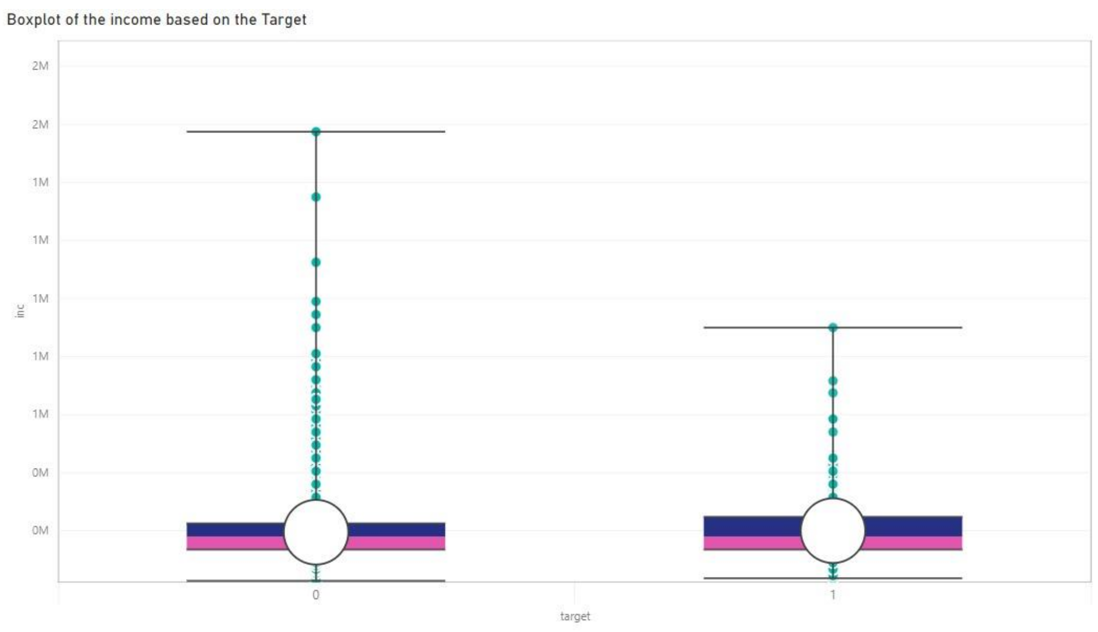
The individuals that do not have a risk associated with them seem to have been employed for a larger number of days than those who do have a risk associated with them. This is evident by comparing the descriptive statistics given in table.

5.4 Boxplot of Target with Income

	Min	1st Quartile	Median	Mean	3rd Quartile	Max
Income where Target = 0	27,000	135,000	180,000	194,745	225,000	1,575,000
Income where Target = 1	36,000	135,000	180,000	200,056	247,500	900,000

TABLE 3: DESCRIPTIVE STATISTICS OF INCOME FOR THE DIFFERENT TARGET VALUES

The individuals that do not have a risk associated with them seem to have a higher annual income compared to those who do have a risk associated with them. This is evident by comparing the descriptive statistics given in table, this supports the trend that a customer is considerably less risky when he/she has a higher annual income.



\

FIGURE 6: COUNTS OF INCOME BASED ON THE TARGET

6 Important Results of the Advanced Analysis

According to the given figure we notice an imbalance in the two options for the response variable Target, this could lead to a biased model being fit, to avoid any complication that could arise from this, the following procedure was followed. After removing the null values and outliers the dataset consisted of 20,313 values with the target variable labeled as "0" and 349 values with the target variable labeled as "1". A simple random sample consisting of 30% of the data

values with the "0" label (This amounted to 6094 datapoints) was selected. We then oversampled the 349 "Yes" datapoints to 6094 datapoints using SMOTE technique.

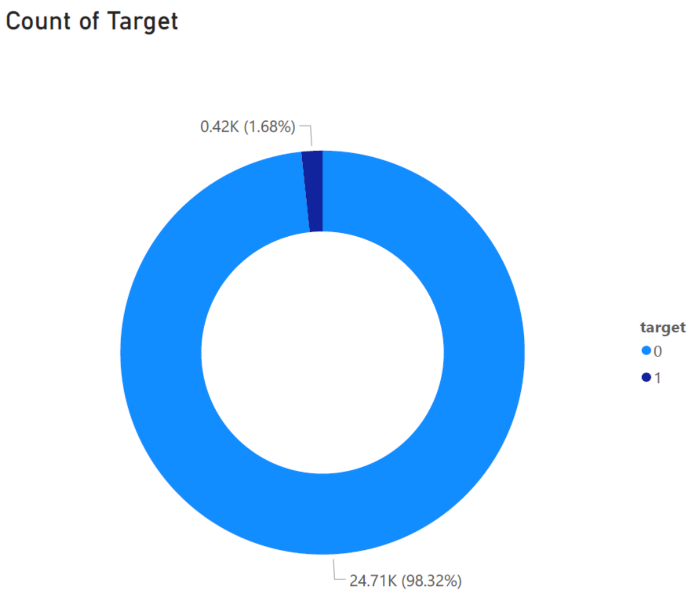


FIGURE 7: COUNTS OF TARGET

Different classification models were fitted to gain a rough idea on what path the analysis should proceed on, summaries of the different models fitted with default parameters are listed below. Note that 5-fold cross validation has been performed when deriving the results for each model given below

Model	Accuracy	Precision	Recall	Time Elapsed
Logistic Regression with parameters	0.681411066673444	0.6640920143345703	0.7328585033628738	5.414385795593262
Decision Tree Classifier	0.8003785635925235	0.7450022019258055	0.9138546629650135	0.9997541904449463
Support Vector Machine	0.6625393383317013	0.6556401205878338	0.6846211665056959	206.37223482131958
Random Forest	**0.9053995300102367**	**0.8733735789401503**	**0.9463438092274619**	**26.668754816055298**
Logistic Regression	0.6813290658810389	0.6639782565751251	0.7328585033628738	1.384354591369629
Linear Discriminant Analysis	0.6815746979755296	0.6630198720993479	0.7376166364257225	2.137072801589966
Gradient Boosting	0.7947995137851209	0.7693093023211783	0.8414904407634458	29.954367637634277
Light Gradient Boosting	0.8470652232821658	0.8204142334688992	0.8879281383566976	10.666882753372192
X Gradient Boosting	**0.874797413272212**	**0.8626283639836079**	**0.8912104594602968**	**69.60619020462036**

TABLE 4: DETAILS OF FITTED MODELS

From the **Table 4**, we notice that XGBoosting and Random Forest models seem to have a generally good accuracy score in terms of predicting the results from the test set. We chose to go forward with these models and looked to improve on them.

After performing Grid search to find the best parameters of these two models the results we got are as follows.

Model	Accuracy	Precision	Recall	Time Elapsed
Gradient Boosting	0.7865130240215868	0.7276244727209809	0.9153291278888858	3.2356746196746826
Random Forest	0.9060563442390572	0.8780321607687588	0.943060814606174	682.2166404724121

TABLE 5: DETAILS OF FITTED MODELS AFTER GRID SEARCH

```
Accuracy Score is 0.90129
      0     1
0  1583   246
1   115  1713
[[0.86550027 0.13449973]
 [0.06291028 0.93708972]]
```

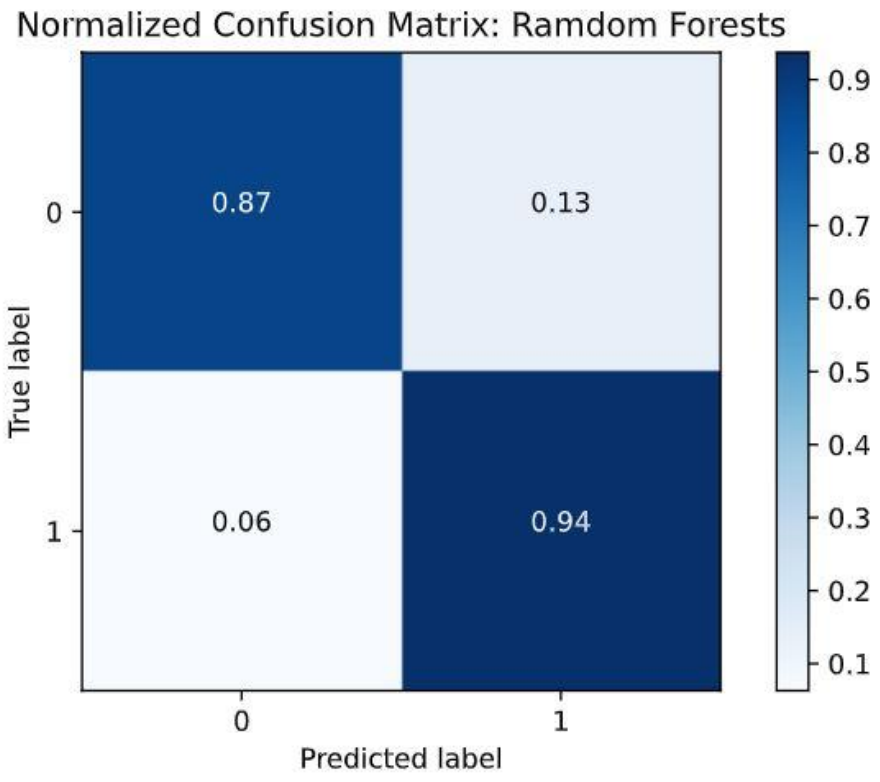


FIGURE 8: CONFUSION MATRIX OF THE FINAL MODEL

7 Issues encountered and Proposed Solutions

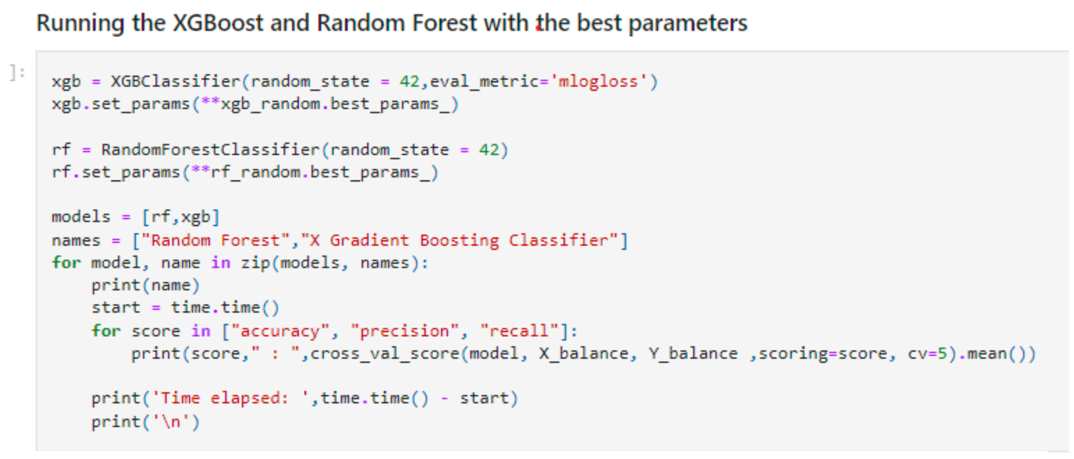
- Once the data was split into testing and train set, the overall count of observations which took the value “1” for the target variable was considerably low, more data could have led to more consistent and accurate results.
- Although the original dataset consisted of 438,558 observations after the removal of missing values there were 20,662 observations. If the dataset consisted of less missing values this could have led to the derivation of less biased results

8 Discussions and Conclusions

- Initial classification models yielded good accuracy scores, however, due to there being a risk of deriving biased results in each of these cases (due to the fact that only a small proportion of the original dataset was sampled after the removal of missing values) , further improvements were attempted using cross validation to yield more generalized results.

- While deeper exploration was done on Gradient Boosting and Random Forest, for further work we suggest to also explore other models deeply as well. This can be further tested by using a larger dataset and comparing how the models can improve.
- Gradient Boosting is computationally cheaper to run compared to Random Forest, thereby if the product is aimed to be used in devices with less processing power like smartphones, it would be ideal to go with a gradient boosting model.
- In conclusion, based on the advanced analysis carried out, it was decided to go with the Random Forest model based on its good accuracy and recall scores.

9 Appendix



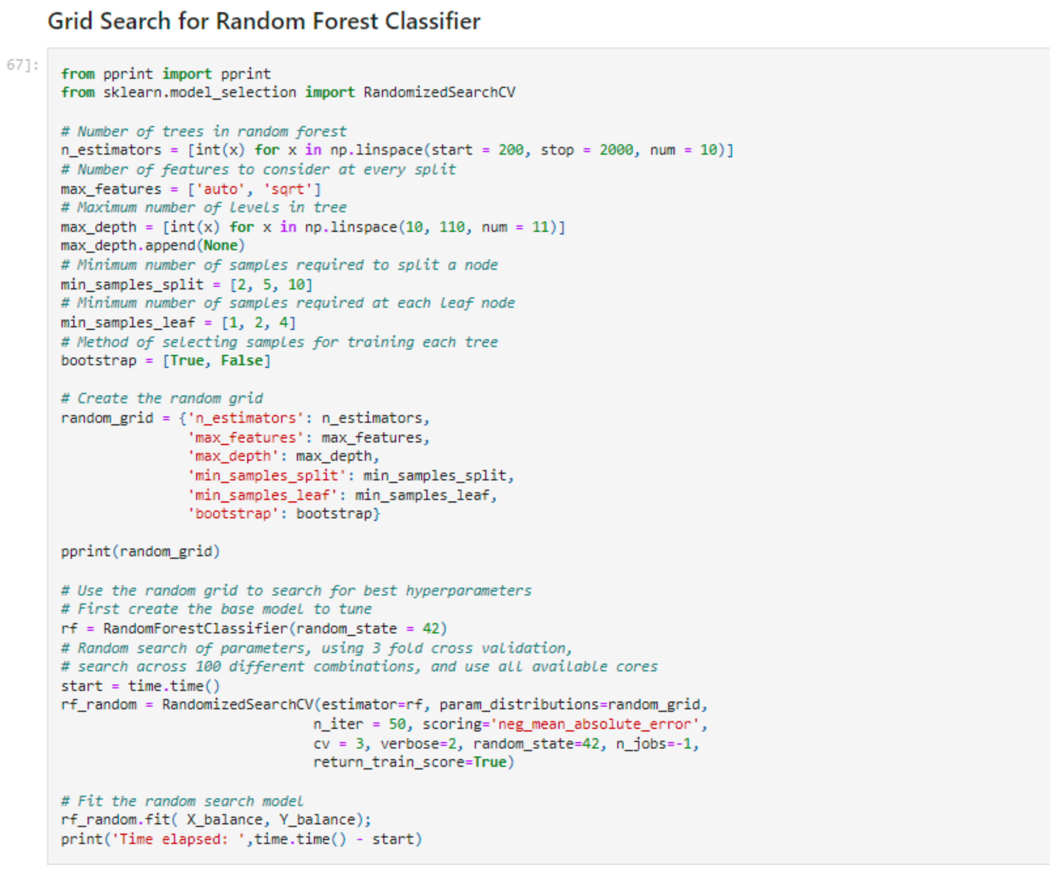
Running the XGBoost and Random Forest with the best parameters

```
xgb = XGBClassifier(random_state = 42,eval_metric='mlogloss')
xgb.set_params(**xgb_random.best_params_)

rf = RandomForestClassifier(random_state = 42)
rf.set_params(**rf_random.best_params_)

models = [rf,xgb]
names = ["Random Forest","X Gradient Boosting Classifier"]
for model, name in zip(models, names):
    print(name)
    start = time.time()
    for score in ["accuracy", "precision", "recall"]:
        print(score," : ",cross_val_score(model, X_balance, Y_balance ,scoring=score, cv=5).mean())

    print('Time elapsed: ',time.time() - start)
    print('\n')
```



Grid Search for Random Forest Classifier

```
from pprint import pprint
from sklearn.model_selection import RandomizedSearchCV

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]

# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}

pprint(random_grid)

# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestClassifier(random_state = 42)
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available cores
start = time.time()
rf_random = RandomizedSearchCV(estimator=rf, param_distributions=random_grid,
                              n_iter = 50, scoring='neg_mean_absolute_error',
                              cv = 3, verbose=2, random_state=42, n_jobs=-1,
                              return_train_score=True)

# Fit the random search model
rf_random.fit( X_balance, Y_balance);
print('Time elapsed: ',time.time() - start)
```

A more detailed breakdown of the codes related to data processing and model building can be found at our project repository here.