Final Report

Loan Default Prediction

Name: Annas Imtiaz

Student Number: 501203680

Supervisor's Name: Ceni Babaoglu

Submission Date: July 17, 2023

Ryerson University

Table of Contents

Introduction

Credit is the central revenue-generating activity for the banking / credit industry and comes in two primary forms. In Unsecured credit, there is no security / collateral for the bank to fall back on in case a user defaults. Secured credit, on the other hand, has some form of collateral for the bank to fall back on. Therefore, in the case of unsecured credit, it becomes even more important for the issuer to improve its decision-making accuracy.

Over the past few decades, a lot of academic research and application has been focused towards improving prediction accuracy of whether a new loan applicant will default or not, and issuers are always striving to improve this accuracy. Many machine learning algorithms including neural networks have been used to classify new applicants as potential good or bad borrowers.

Usually, an issuer’s portfolio, or a publicly sourced dataset, would be imbalanced. Imbalance here means that the number of defaulted cases is less than the number of non-defaulted cases. Default is usually a binary classification, with 1 representing default and 0 representing non-default. As per Canadian Bankers Association (CBA), more than 70% of Canadians pay their credit card balance in full each month. This is itself reflective of the skewed nature of the problem, with a greater proportion of non-default instances. This makes training a model for default cases a bit difficult, since the model is unable to learn the characteristics of defaulted customers, due to low volume / high volatility.

A lot of academic research has tried to balance this imbalance, using either Undersampling, or Oversampling. Undersampling means that non-defaulted observations are deleted to make their count as close to the defaulted cases as possible. Oversampling, on the other hand, refers to adding sample defaulted observations, which have similar characteristics as actual defaulted observations, to have a sufficient volume for the algorithm to learn.

In some areas, academic research falls short of what is required by industry. One such instance is default prediction. Academic research is based primarily on predicting whether a new applicant is likely to default or not. However, a yes / no decision is not what banks / credit issuers are looking for. Credit issuers are more interested in a rank-order of borrowers based on their probability of default / credit score. They can then use this rank-order to define what level of risk they are willing to take, based on their risk appetite and overall economic forecasts. This allows them to set a benchmark through which they can make a risk – return trade-off decision, i.e., what level of risk are they willing to take if it offers a specific level of return. This benchmark / cut-off is obviously dynamic, and as economic conditions improve, issuers generally are more likely to take on risk, while risk aversion increases when economic uncertainty looms.

Literature Review

The first paper studied was ‘An Investigation of Credit Card Default Prediction in the Imbalanced Datasets’[1]. The hypothesis was whether models developed using different ML techniques are significantly different from each other, and whether sampling techniques would improve the model’s performance. The study used three different datasets and compared performance across each.

Three imbalanced datasets were used. Once data was preprocessed and resampled, Gradient Boosted Decision Tree (GBDT) model, an ensemble-based learning method, was used for modeling and its results were compared with traditional machine learning models (Random Forest, Bagging, KNN, Logistic Regression, Ada Boost and Stacking).

Next, a few resampling methods were used to undersample and oversample the data. A Gradient Boosted Decision automatically selects significant features during the modelling phase. It also gives the relative importance, also known as relative influence, of the independent features. To control overfitting of the Gradient Boosted decision Tree, a regularization shrinkage was also added.

To evaluate the models, Accuracy, Precision, Recall, F-Measure and AUROC. Additionally, Geometric Mean was also used to address imbalanced classes of the original datasets.

Results: For the initial datasets, GBDT outperformed all the other algorithms. In terms of Undersampling methods, Cluster Centroids has outperformed Random Undersampling as well as Near Miss methods. Amongst all the Oversampling techniques, K-means SMOTE outperformed Random oversampling, ADAYSN, SMOTE, Borderline-SMOTE and SMOTETomek. Across application of the algorithms on oversampled dataset, Random Forest had the best results when using Random Oversampling, while GBDT had the best results across all the other oversampling methods. The research concluded that balanced datasets had better accuracy as compared to imbalanced datasets, and that oversampling techniques produced better results compared to under sampling techniques.

The ANOVA test rejected the hypothesis and showed that the proposed methods using imbalanced datasets have significantly improved performance from the baseline model. Credit amount, marital status and education level were found to be significant features.

Authors in ‘A hybrid interpretable credit card users default prediction model based on RIPPER’[2] focus more on interpretability of default prediction models rather than their accuracy. Specifically, the study focused on operation time and stability. By models being difficult to interpret, researchers here refer to domain experts being unable to understand the mechanics behind a model, and thus not able to improve on it. Therefore, for this research, interpretability is defined as the model can be understood, the rules are completely understandable, and the number of rules is reasonable. Researchers are of the view that in some jurisdictions, it is a legal obligation to justify a refused credit decision, which is only possible if the model is interpretable.

After data normalization, RELIEF Algorithm was used for feature selection. This is a feature weighted algorithm, which assigns weights to features based on correlation with other features. Then, features with weight lower than a certain threshold are dropped.

Since the dataset being used is imbalanced, SMOTE is used to add some sample data points to create equality between both classes. RIPPER is customized 2-loop algorithm that the authors recommend.

In order to measure performance, F-Value and AUROC are used. The authors conclude that their proposed refined Relief algorithm boasts of better performance as compared to traditional algorithms.

Authors of ‘Credit Card Default Prediction using Machine Learning Techniques’[3] aim to find the correlation and predictive power of factors contributing to credit card default. For data pre-processing, some variables were converted from numeric to factor, while Correlation based Feature Selection was used to reduce dimensionality. For this paper, researchers used Logistic Regression, Decision Trees and Random Forest to predict default.

Based on a comparison of true positive vs false positive, Random Forest displayed the best accuracy and the highest area under the curve.

In their research paper ‘Comparison of Different Ensemble Methods in Credit Card Default Prediction’[4], the authors try to apply ensemble machine learning methods on both originally imbalanced as well as after applying some balancing techniques to the dataset. In both cases, min-max scalar has been used to scale the data. The study primarily aims to answer:

- How well ensemble methods work on credit card default prediction?
- Are ensemble methods better than other machine learning methods when used on skewed datasets?
- Does balancing the dataset have an impact on relative performance gain of ensemble methods?

The ensemble techniques used in this research are Bagging, Boosting (AdaBoosting and XGBoosting), Voting, and random forests (RF). Authors have pertinently identified that accuracy on positive decisions is relatively more critical in this case, as an incorrect credit decision of issuing a credit card to a customer who will default will result in greater loss for the issuer. Some of the metrics used in this paper are Accuracy, Precision, Recall, ROC and AUC – however, F1 score is used to assess ensemble methods.

For imbalanced dataset, F1 was used as the evaluation metric. Stacking had the best F1 score, followed by Neural Networks. In terms of AUROC, Stacking was followed by XGBoost. Dataset balancing was done by down sampling it. Once the data is balanced and reshuffled, XGBoost comes out to have the highest accuracy, as well as AUROC.

The paper 'Research on Default Prediction for Credit Card Users Based on XGBoost-LSTM Model'[5] uses XGBoost, which is widely used in financial classification models, and Long-Short Term Memory (LSTM), which is widely used in time-series information. In this research, default prediction is primarily based on account, credit bureau, and transaction flow data.

Machine learning algorithms and deep learning algorithms are used on the same data set to construct default prediction models, and the prediction accuracy and modeling workload are compared, ultimately revealing that the deep learning model has high prediction accuracy. One interesting approach used by researchers here was calculating 'desire of funds' which is based on the number of days the credit card was issued and it's first use.

Performance evaluation metrics include accuracy, precision, recall and AUC. KNN and SVM algorithm for default prediction appear to be less effective, and the decision tree, random forest, AdaBoost, and XGBoost algorithms are better; of these, the XGBoost algorithm is the best. LSTM is a form of Recurrent Neural Network (RNN). The results suggest that a fusion of XGBoost and LSTM yields significantly better results than a simple XGBoost method.

The last paper referenced here is ‘Research on Credit Card Default Prediction Based on k-Means SMOTE and BP Neural Network’[6]. This paper proposes a prediction model based on k-means SMOTE and BP neural network. K means SMOTE is used to change the data distribution. Next, random forest is used to calculate importance of the features and this importance is then substituted into the initial weights of BP neural networks. The paper uses KNN, logistic regression, SVM, random forest and decision trees and then compares the results of these 6 methods.

The 23 features are converted into 89 input variables. Next, PCA is used to reduce these 89 to 27 input variables. The results suggest that the proposed algorithm leads to an increased performance compared to all the other algorithms.

Research Questions:

A recap of the research questions is important here, and throughout the report, we will see how our modelling process helped answer these questions:

- Which is the classification algorithm that produces the best results in terms of F1-Score and AUROC?
- Are any of the independent variables correlated, leading to potential over-fitting if one of them is not excluded from analysis?
- Do the chosen algorithms have the same level of accuracy on both the train and test dataset?
- Do balanced datasets deliver better results as compared to imbalanced datasets?

Algorithms Used:

The algorithms used were:

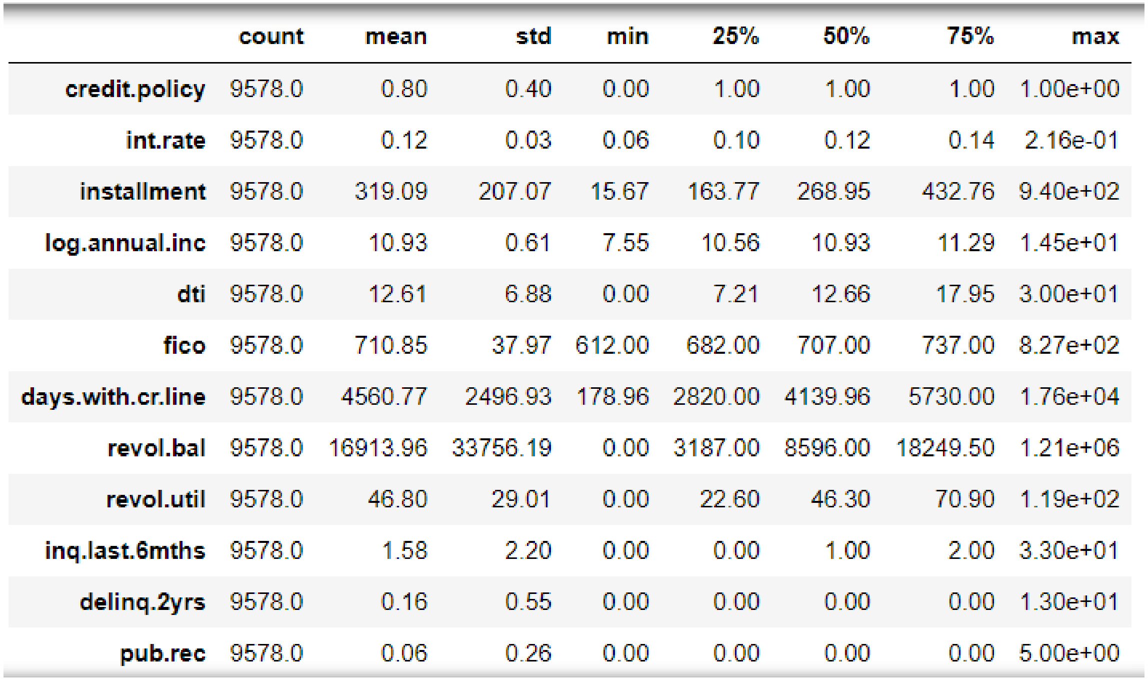
- Random Forecast Classifier (default and class_weights = balanced)
- Decision Tree Classifier (default and class_weights = balanced)
- KNeighbors Classifier (default and weights = distance)
- Logistic Regression Classifier (default and class_weights = balanced)
- ADA Boost
- XGBoost

GitHub Repository:

The final code has been uploaded to this public repository here: https://github.com/AnnasImtiaz/Loan-Default-Prediction

Dataset Description

The original dataset does seem to have a few outliers:

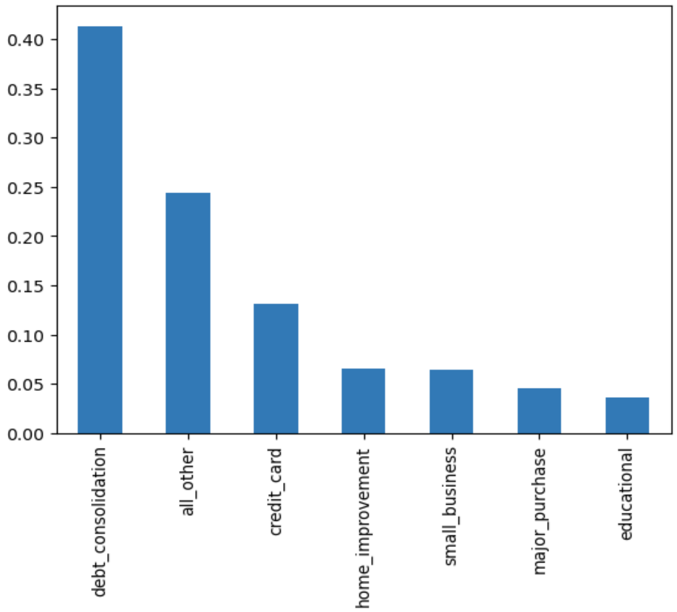


	count	mean	std	min	25%	50%	75%	max
credit.policy	9578.0	0.80	0.40	0.00	1.00	1.00	1.00	1.00e+00
int.rate	9578.0	0.12	0.03	0.06	0.10	0.12	0.14	2.16e-01
installment	9578.0	319.09	207.07	15.67	163.77	268.95	432.76	9.40e+02
log.annual.inc	9578.0	10.93	0.61	7.55	10.56	10.93	11.29	1.45e+01
dti	9578.0	12.61	6.88	0.00	7.21	12.66	17.95	3.00e+01
fico	9578.0	710.85	37.97	612.00	682.00	707.00	737.00	8.27e+02
days.with.cr.line	9578.0	4560.77	2496.93	178.96	2820.00	4139.96	5730.00	1.76e+04
revol.bal	9578.0	16913.96	33756.19	0.00	3187.00	8596.00	18249.50	1.21e+06
revol.util	9578.0	46.80	29.01	0.00	22.60	46.30	70.90	1.19e+02
inq.last.6mths	9578.0	1.58	2.20	0.00	0.00	1.00	2.00	3.30e+01
delinq.2yrs	9578.0	0.16	0.55	0.00	0.00	0.00	0.00	1.30e+01
pub.rec	9578.0	0.06	0.26	0.00	0.00	0.00	0.00	5.00e+00

However, because a bank can expect a diversity of customers approaching for credit card applications, the application of this default prediction model will also be on populations which will exhibit such outliers. Therefore, we will not treat these outliers and instead let the model learn to predict on outlier values too. From the above, we can also note that there are no missing or null values in the dataset.

Univariate analysis of Numerical Features showed some skewness in a few of the numeric features. Some of the outputs from the Univariate and Bivariate analysis performed is show below:

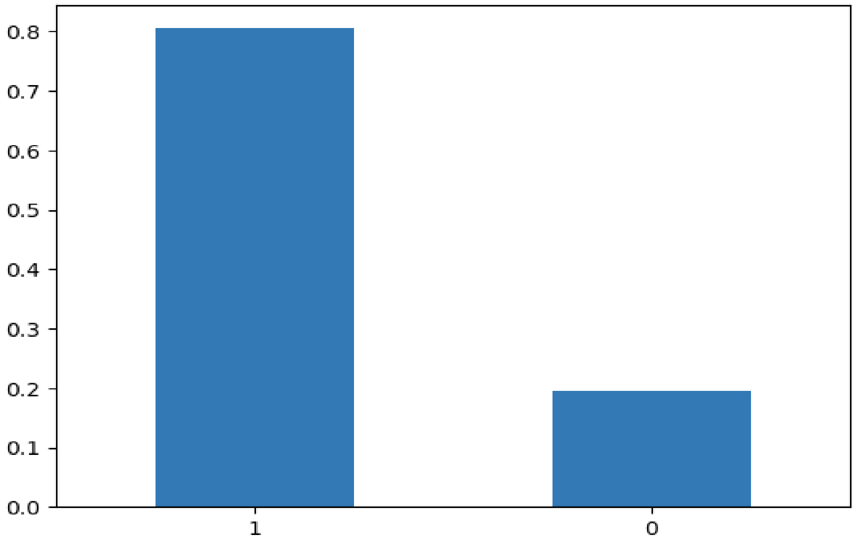
Purpose:



Observation:

- About 40% loans were taken for debt consolidation, followed by 15% credit cards, and smaller %s for small business, major purchase and education. All Others comprised of about 25%

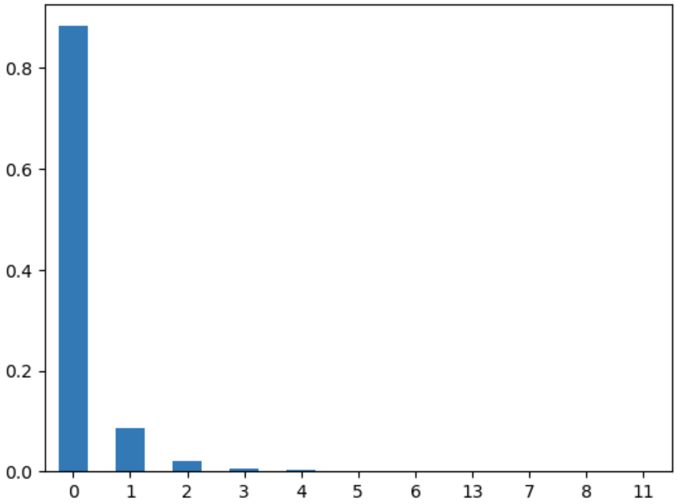
Credit Policy:



Observation:

- Almost 80% customers meet the lending criteria of LendingClub, while 20% do not meet this criteria

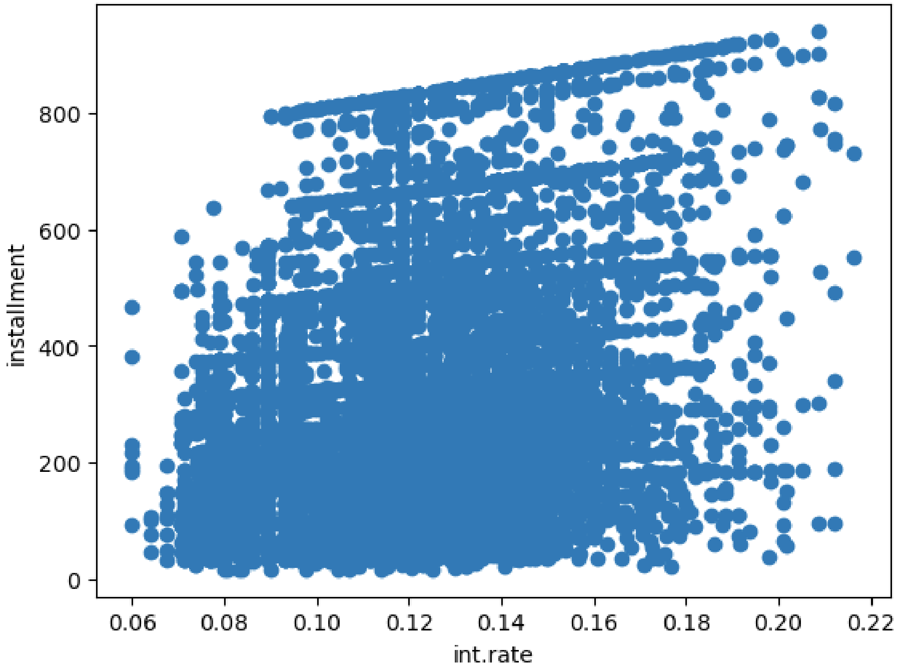
Delinq.2years:



Observation:

- Delinquency is defined by the data provider as being 30+ days past due. More than 80% customers have not had any delinquency in the past 2 years, while less than 2% have had 1. Obviously, as number of delinquencies increase, the customer moves from a good customer to a bad customer

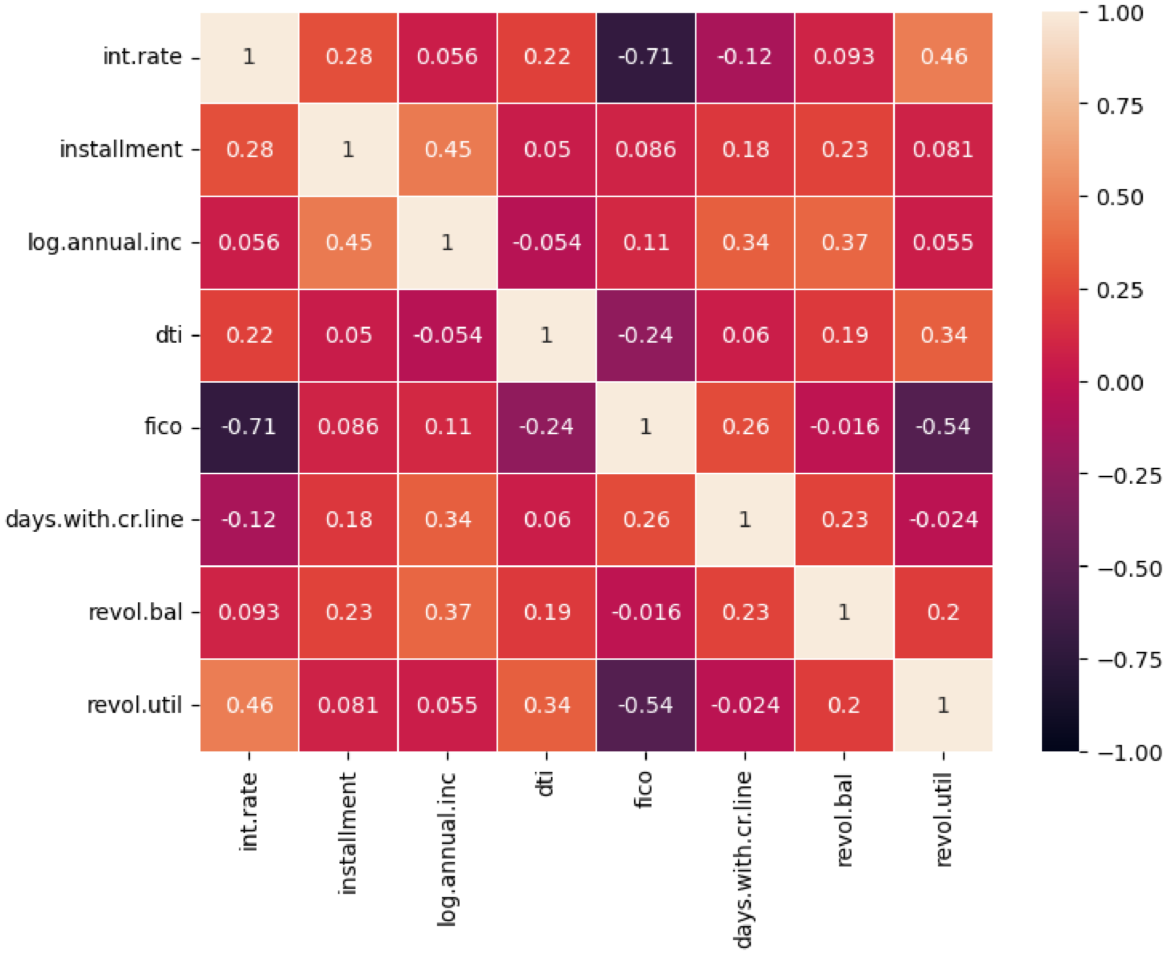
Installment & Int.Rate:



Reflects no correlation between int.rate and installment

The next step was identifying correlation. Pearson method was used for numeric features while Spearman method was used for categorical features.

Correlation of numeric variables:



	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util
int.rate	1	0.28	0.056	0.22	-0.71	-0.12	0.093	0.46
installment	0.28	1	0.45	0.05	0.086	0.18	0.23	0.081
log.annual.inc	0.056	0.45	1	-0.054	0.11	0.34	0.37	0.055
dti	0.22	0.05	-0.054	1	-0.24	0.06	0.19	0.34
fico	-0.71	0.086	0.11	-0.24	1	0.26	-0.016	-0.54
days.with.cr.line	-0.12	0.18	0.34	0.06	0.26	1	0.23	-0.024
revol.bal	0.093	0.23	0.37	0.19	-0.016	0.23	1	0.2
revol.util	0.46	0.081	0.055	0.34	-0.54	-0.024	0.2	1

A cutoff of 0.75 is set for this research, i.e., variables with correlation > 0.75 will be classified as highly correlated. The following variables are correlated - however, not as high as the set cut-off:

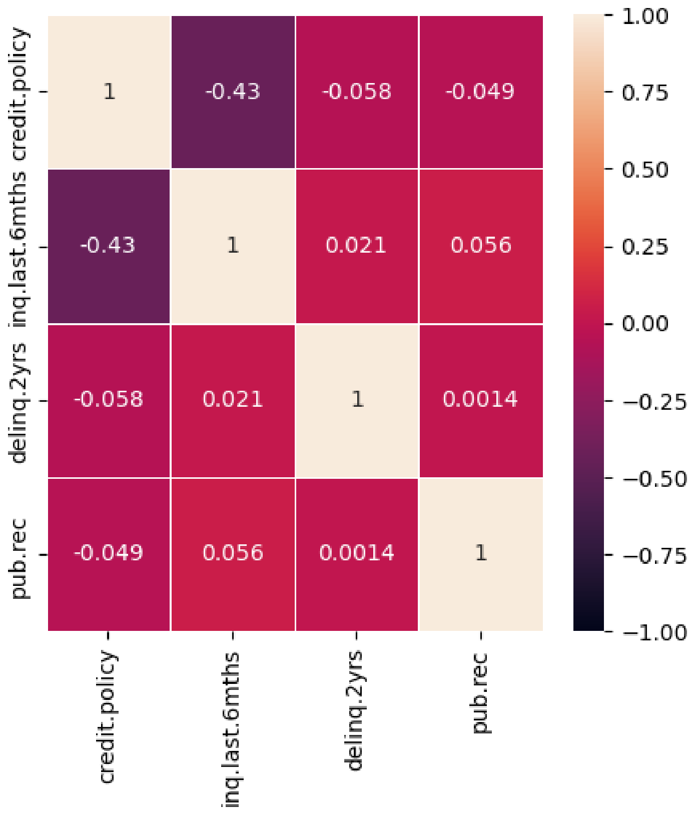
- int.rate and fico

- fico and revol.util

- int.rate and revol.util

These correlations are intuitively correct. Since the correlation is negative, it suggests that borrowers with a better FICO score get a lower rate of interest, and their revolving utilization is also lower.

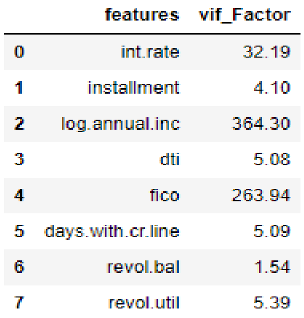
Correlation of categorical variables:



Based on the cutoff of 0.75, none of the categorical variables are correlated.

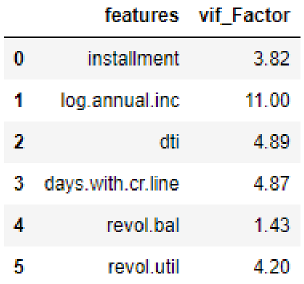
Another approach used to detect multicollinearity was through the use of Variance Inflation Factors (VIF).

VIF factors of the numeric variables from original dataset is presented below:



	features	vif_Factor
0	int.rate	32.19
1	installment	4.10
2	log.annual.inc	364.30
3	dti	5.08
4	fico	263.94
5	days.with.cr.line	5.09
6	revol.bal	1.54
7	revol.util	5.39

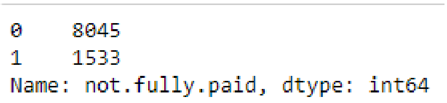
VIF factors with fico and int.rate feature removed is show below:



	features	vif_Factor
0	installment	3.82
1	log.annual.inc	11.00
2	dti	4.89
3	days.with.cr.line	4.87
4	revol.bal	1.43
5	revol.util	4.20

We can see that with these two factors removed, vif factor scores have significantly reduced. However, since the number of features is already limited, with fico scores being a critical determinant while assessing new credit applications, we have decided to not remove these two features from our dataset.

In industry, usually defaulting / bad customers comprise of a much smaller part of a lender's overall portfolio as compared to non-defaulting / good customers. Therefore, it is expected that an industry database would also be imbalanced with more non-defaulting customers. Our data is exactly the same:



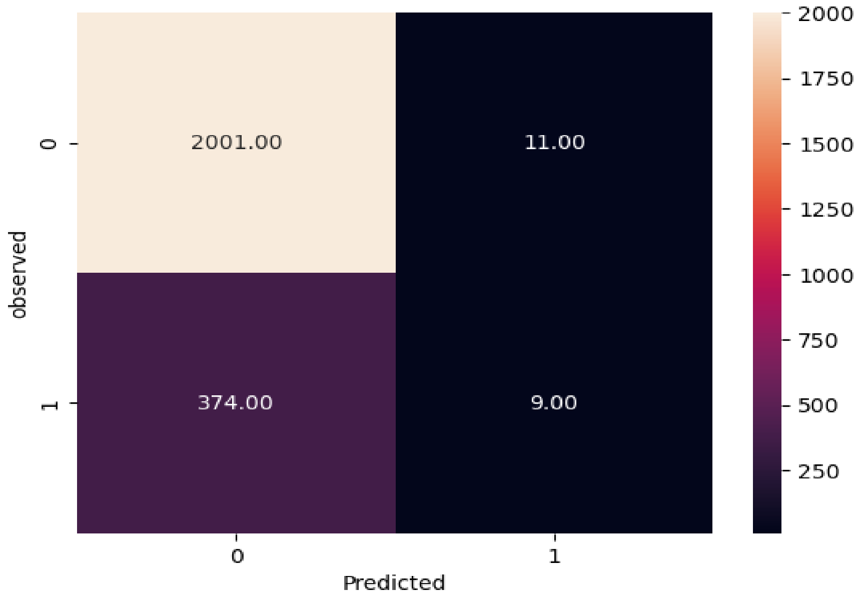
```
0    8045
1    1533
Name: not.fully.paid, dtype: int64
```

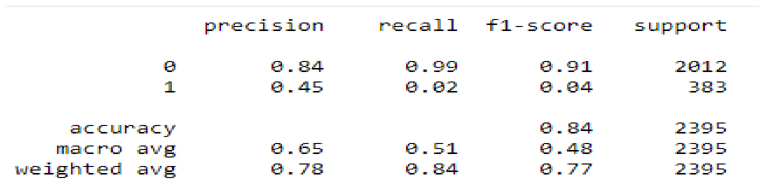
Therefore, with a 70% train data, our model would have a very small data to train on defaulting customers. To overcome this issue and let the model train better on all possible cases, for all non-ensemble algorithms we will use the default class weight as well as balanced class weight, and eventually compare if our model performance is better when balanced class weights are used.

Algorithms used and results

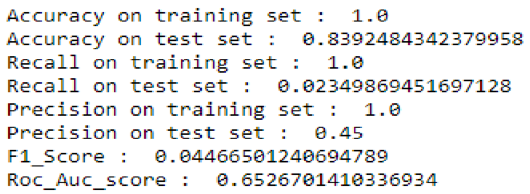
In this specific business case, it is extremely critical for a default prediction model to be good at predicting default cases correctly. If a non-defaulting customer is predicted as a defaulting customer, it will only be a case of missed business opportunity for the lender. However, if a defaulting customer is predicted as a non-defaulting customer, it would be detrimental for the business and result in losses.

1. Random Forest Classifier



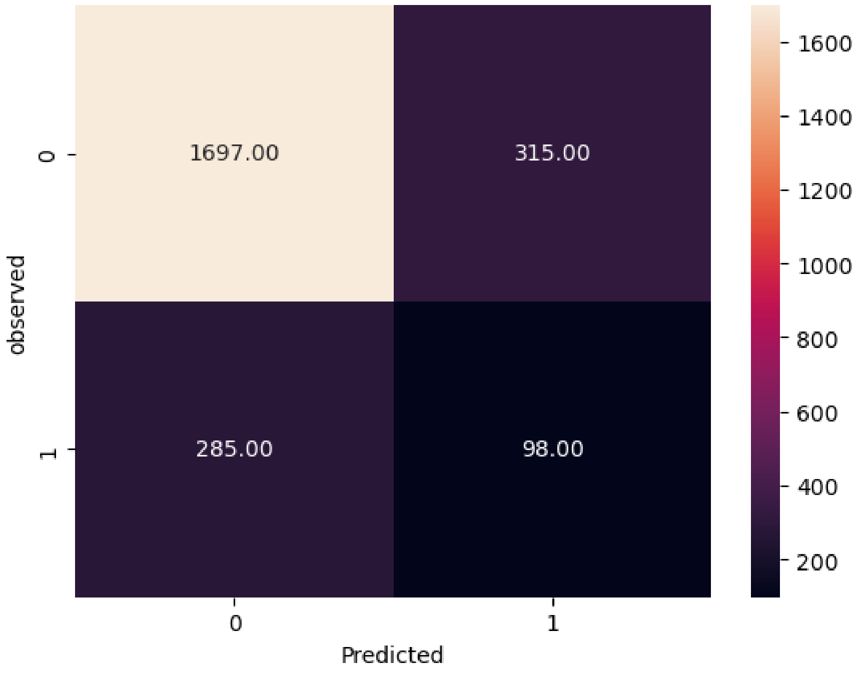


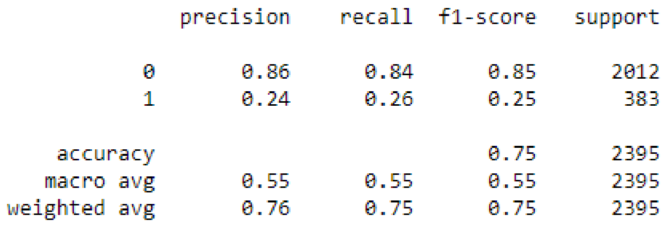
```
              precision    recall  f1-score   support

           0       0.84      0.99      0.91      2012
           1       0.45      0.02      0.04       383

    accuracy                           0.84      2395
   macro avg       0.65      0.51      0.48      2395
weighted avg       0.78      0.84      0.77      2395
```



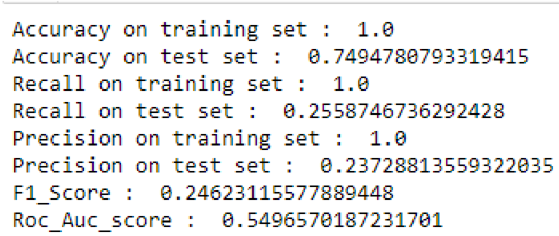
```
Accuracy on training set :  1.0
Accuracy on test set :  0.8392484342379958
Recall on training set :  1.0
Recall on test set :  0.02349869451697128
Precision on training set :  1.0
Precision on test set :  0.45
F1_Score :  0.04466501240694789
Roc_Auc_score :  0.6526701410336934
```

2. Decision Tree Classifier



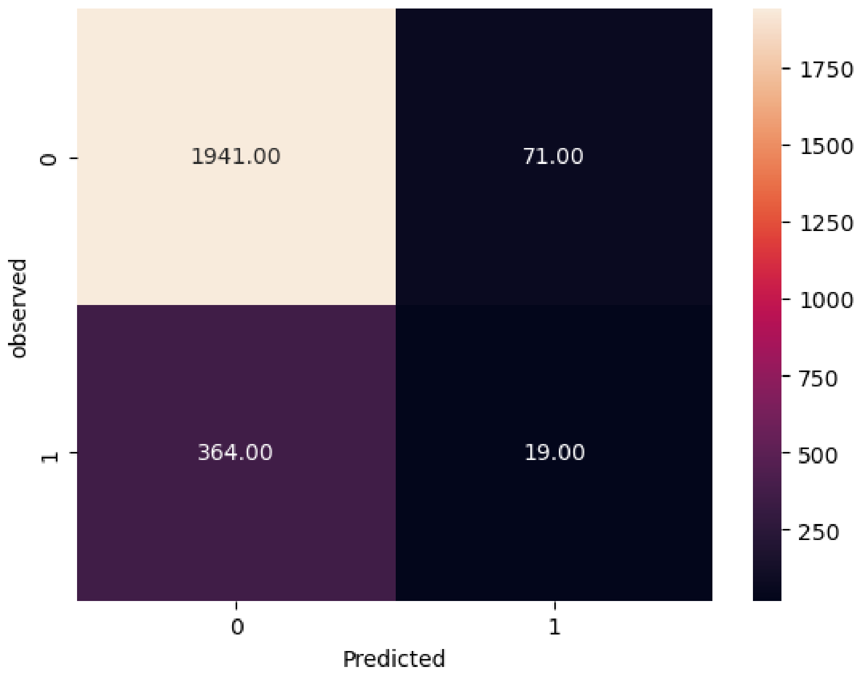


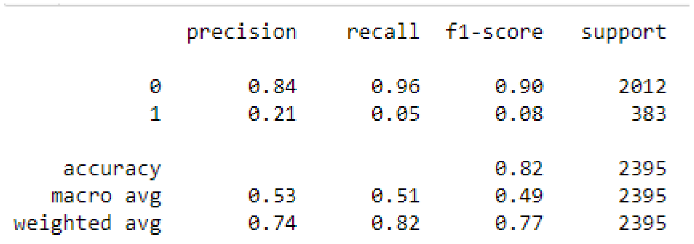
```
              precision    recall  f1-score   support

           0       0.86      0.84      0.85      2012
           1       0.24      0.26      0.25       383

    accuracy                           0.75      2395
   macro avg       0.55      0.55      0.55      2395
weighted avg       0.76      0.75      0.75      2395
```



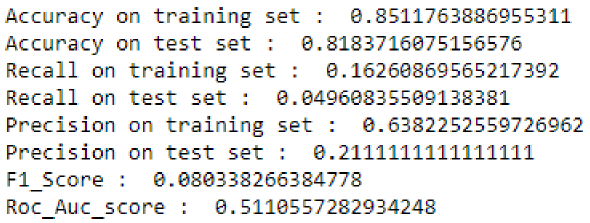
```
Accuracy on training set :  1.0
Accuracy on test set :  0.7494780793319415
Recall on training set :  1.0
Recall on test set :  0.2558746736292428
Precision on training set :  1.0
Precision on test set :  0.23728813559322035
F1_Score :  0.24623115577889448
Roc_Auc_score :  0.5496570187231701
```

3. KNeighbors Classifier



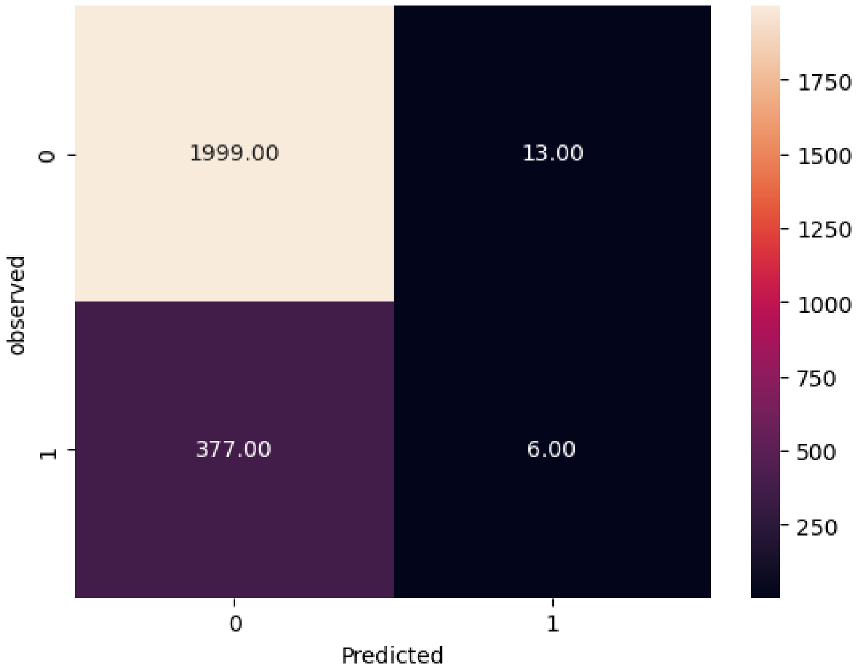


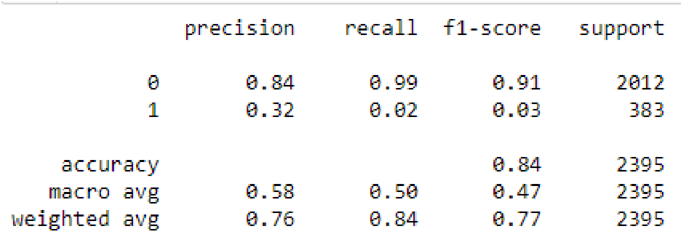
```
              precision    recall  f1-score   support

           0       0.84      0.96      0.90      2012
           1       0.21      0.05      0.08       383

    accuracy                           0.82      2395
   macro avg       0.53      0.51      0.49      2395
weighted avg       0.74      0.82      0.77      2395
```



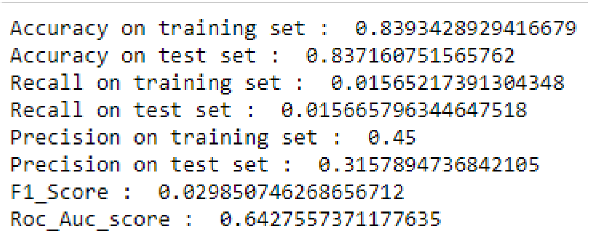
```
Accuracy on training set :  0.8511763886955311
Accuracy on test set :  0.8183716075156576
Recall on training set :  0.16260869565217392
Recall on test set :  0.04960835509138381
Precision on training set :  0.6382252559726962
Precision on test set :  0.2111111111111111
F1_Score :  0.080338266384778
Roc_Auc_score :  0.5110557282934248
```

4. Logistic Regression Classifier



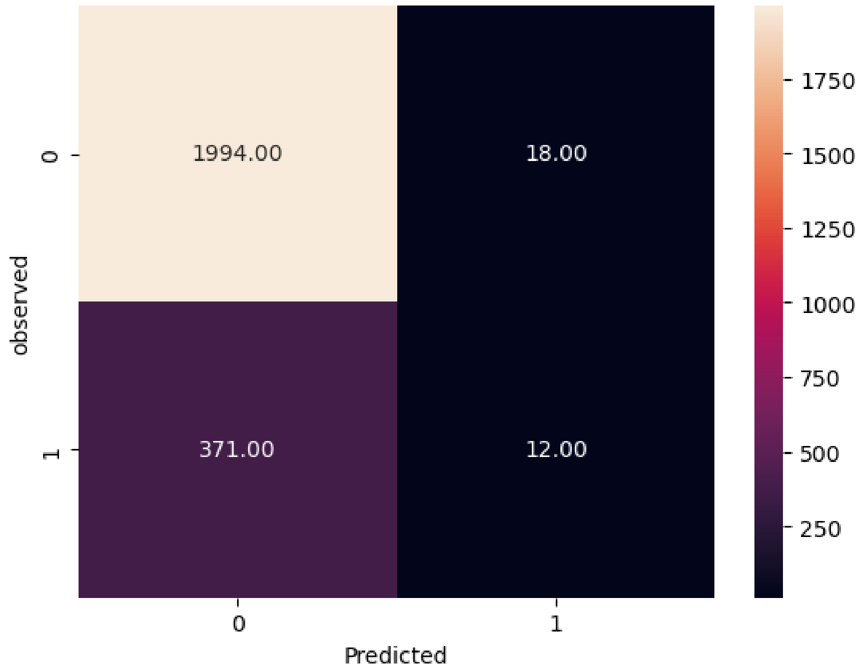


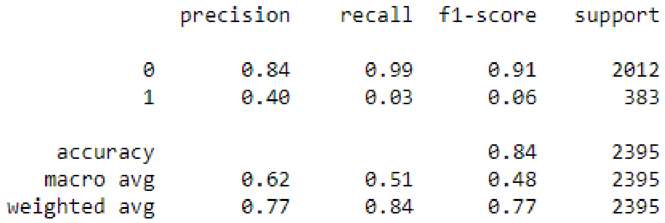
```
              precision    recall  f1-score   support

           0       0.84      0.99      0.91      2012
           1       0.32      0.02      0.03       383

    accuracy                           0.84      2395
   macro avg       0.58      0.50      0.47      2395
weighted avg       0.76      0.84      0.77      2395
```



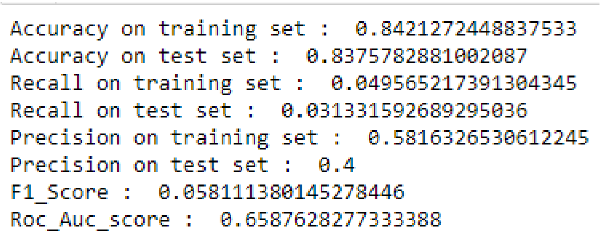
```
Accuracy on training set :  0.8393428929416679
Accuracy on test set :  0.837160751565762
Recall on training set :  0.01565217391304348
Recall on test set :  0.015665796344647518
Precision on training set :  0.45
Precision on test set :  0.3157894736842105
F1_Score :  0.029850746268656712
Roc_Auc_score :  0.6427557371177635
```

5. ADA Boost Classifier



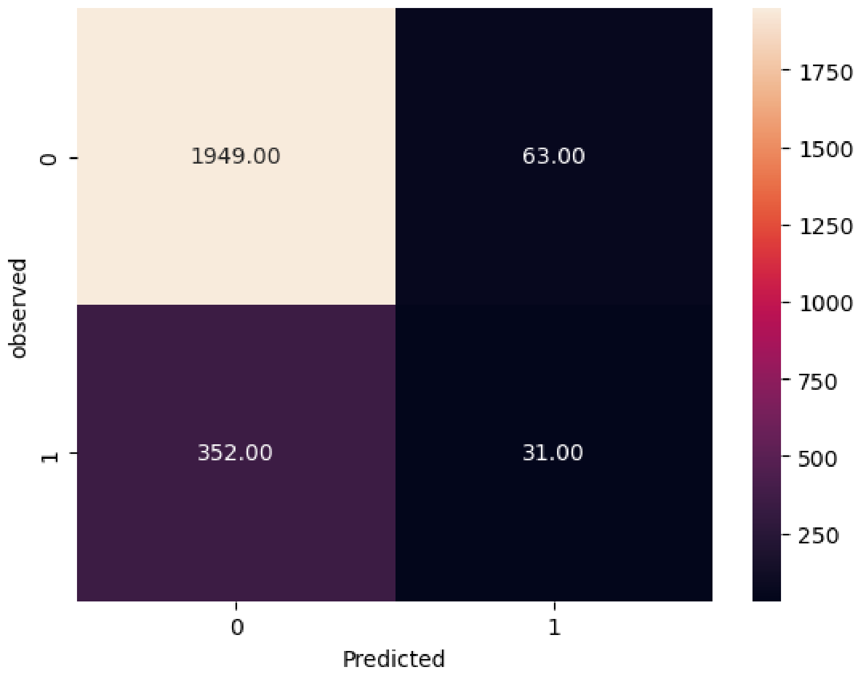


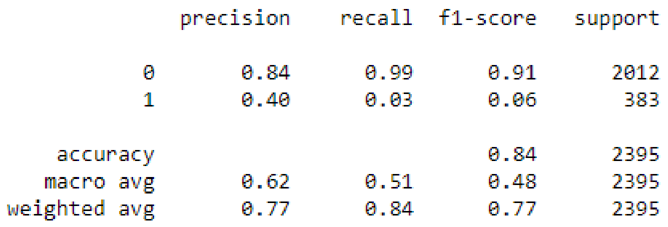
```
              precision    recall  f1-score   support

           0       0.84      0.99      0.91      2012
           1       0.40      0.03      0.06       383

    accuracy                           0.84      2395
   macro avg       0.62      0.51      0.48      2395
weighted avg       0.77      0.84      0.77      2395
```



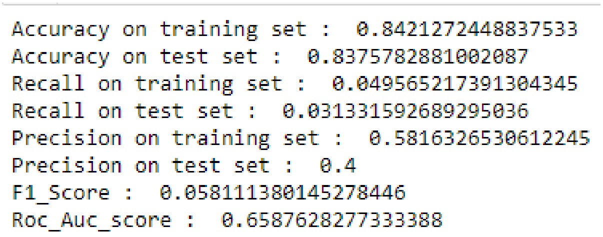
```
Accuracy on training set :  0.8421272448837533
Accuracy on test set :  0.8375782881002087
Recall on training set :  0.049565217391304345
Recall on test set :  0.031331592689295036
Precision on training set :  0.5816326530612245
Precision on test set :  0.4
F1_Score :  0.058111380145278446
Roc_Auc_score :  0.6587628277333388
```

6. XGBoost Classifier



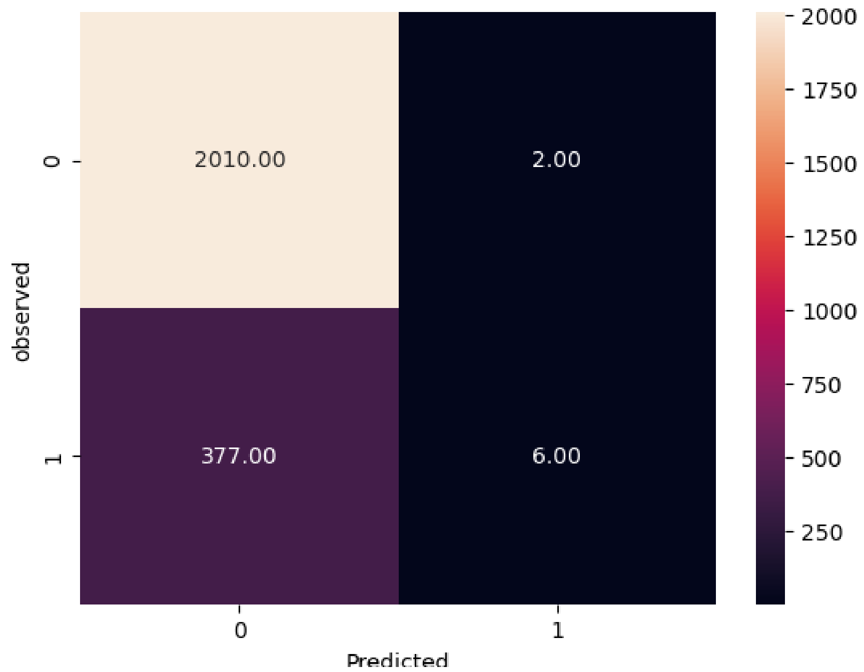


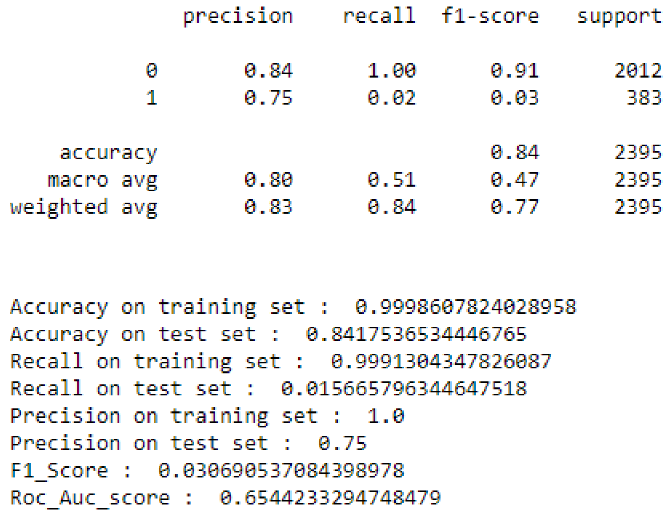
```
              precision    recall  f1-score   support

           0       0.84      0.99      0.91      2012
           1       0.40      0.03      0.06       383

    accuracy                           0.84      2395
   macro avg       0.62      0.51      0.48      2395
weighted avg       0.77      0.84      0.77      2395
```



```
Accuracy on training set :  0.8421272448837533
Accuracy on test set :  0.8375782881002087
Recall on training set :  0.049565217391304345
Recall on test set :  0.031331592689295036
Precision on training set :  0.5816326530612245
Precision on test set :  0.4
F1_Score :  0.05811138014527844б
Roc_Auc_score :  0.6587628277333388
```

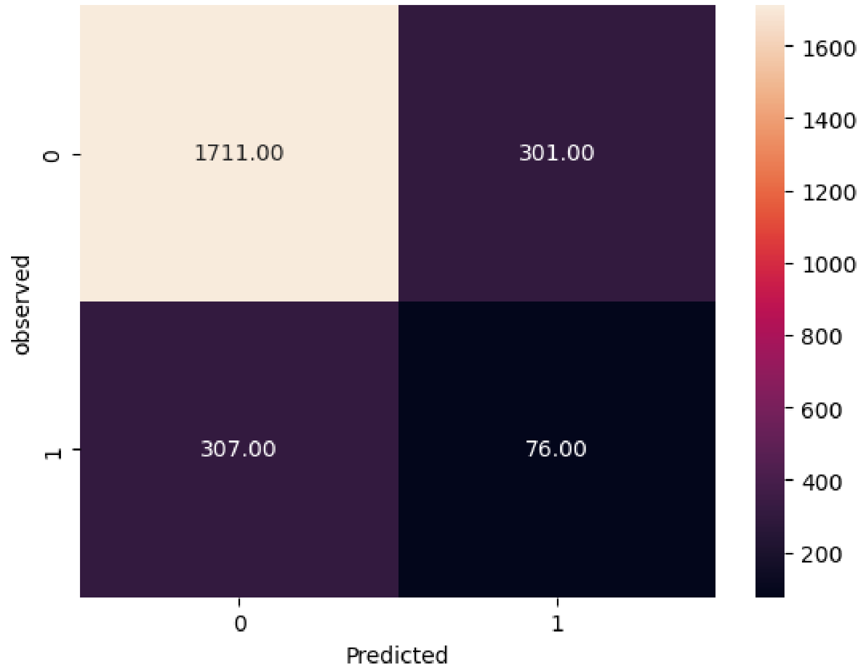
7. Random Forest Classifier (class_weight = balanced)

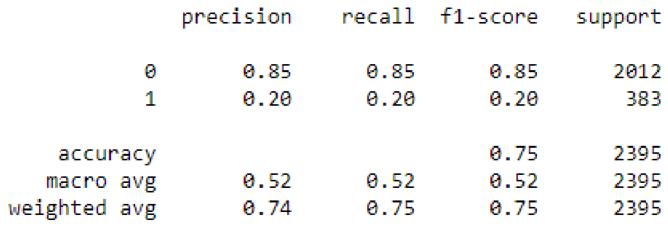




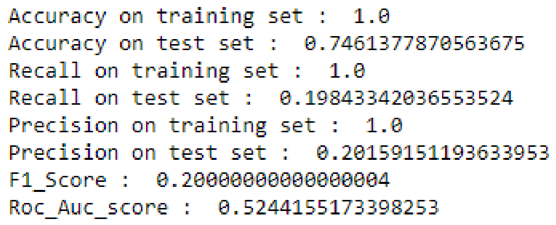
```
              precision    recall  f1-score   support

           0       0.84      1.00      0.91      2012
           1       0.75      0.02      0.03       383

    accuracy                           0.84      2395
   macro avg       0.80      0.51      0.47      2395
weighted avg       0.83      0.84      0.77      2395


Accuracy on training set :  0.9998607824028958
Accuracy on test set :  0.8417536534446765
Recall on training set :  0.9991304347826087
Recall on test set :  0.015665796344647518
Precision on training set :  1.0
Precision on test set :  0.75
F1_Score :  0.030690537084398978
Roc_Auc_score :  0.6544233294748479
```

8. Decision Tree Classifier (class_weight = balanced)



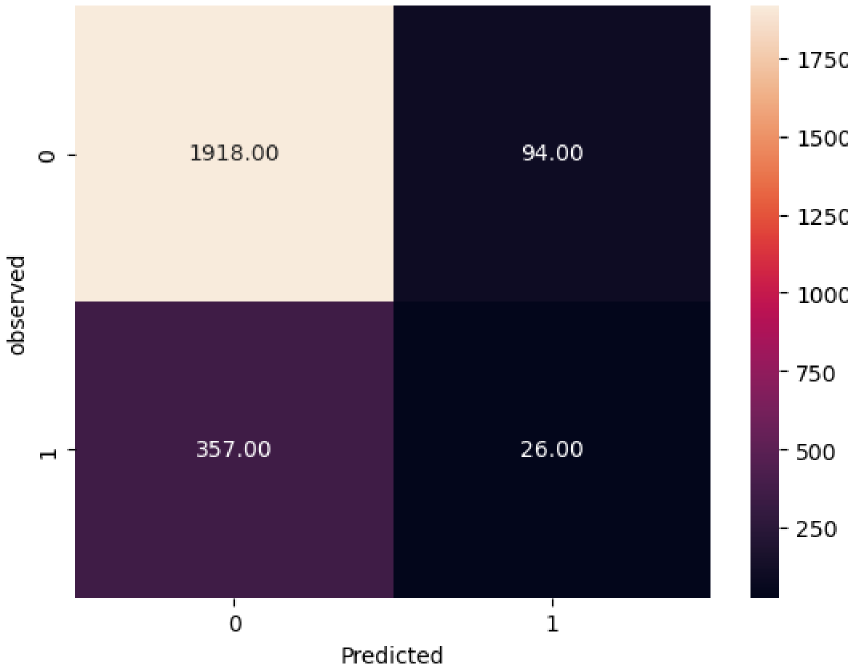


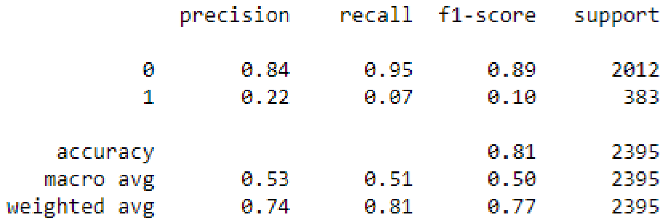
```
              precision    recall  f1-score   support

           0       0.85      0.85      0.85      2012
           1       0.20      0.20      0.20       383

    accuracy                           0.75      2395
   macro avg       0.52      0.52      0.52      2395
weighted avg       0.74      0.75      0.75      2395
```



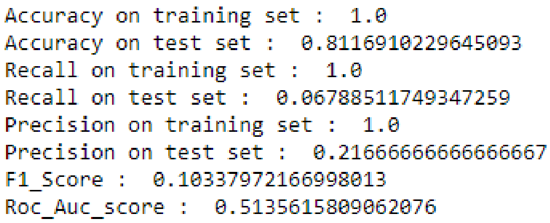
```
Accuracy on training set :  1.0
Accuracy on test set :  0.7461377870563675
Recall on training set :  1.0
Recall on test set :  0.19843342036553524
Precision on training set :  1.0
Precision on test set :  0.20159151193633953
F1_Score :  0.20000000000000004
Roc_Auc_score :  0.5244155173398253
```

9. KNeighbors Classifier (weight = distance)



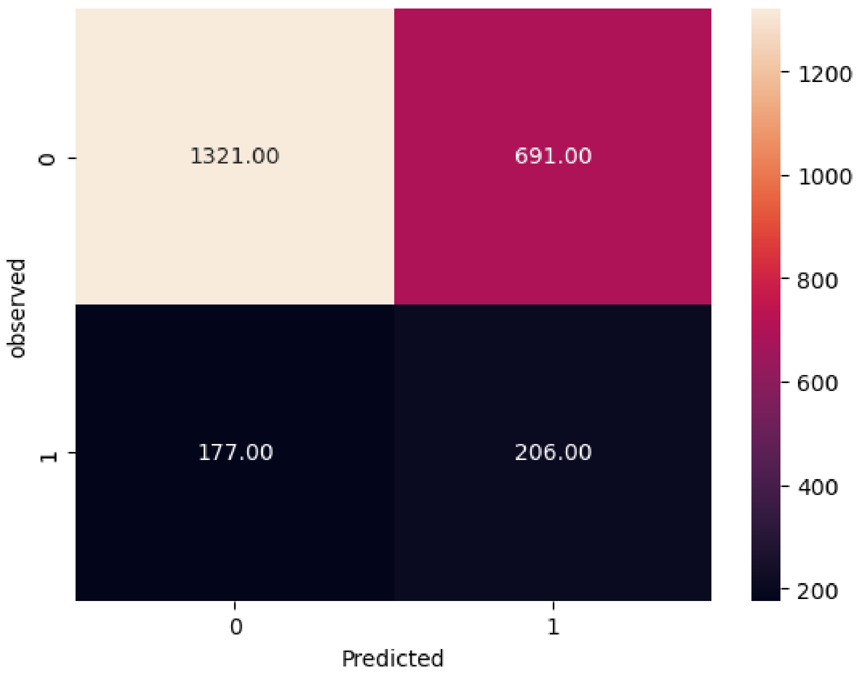


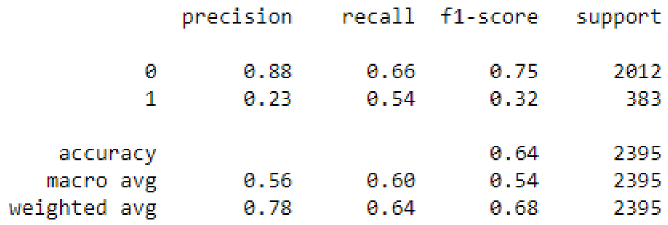
```
              precision    recall  f1-score   support

           0       0.84      0.95      0.89      2012
           1       0.22      0.07      0.10       383

    accuracy                           0.81      2395
   macro avg       0.53      0.51      0.50      2395
weighted avg       0.74      0.81      0.77      2395
```



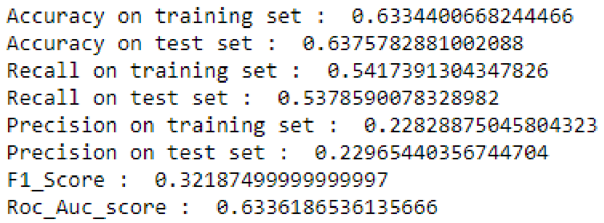
```
Accuracy on training set :  1.0
Accuracy on test set :  0.8116910229645093
Recall on training set :  1.0
Recall on test set :  0.06788511749347259
Precision on training set :  1.0
Precision on test set :  0.21666666666666667
F1_Score :  0.10337972166998013
Roc_Auc_score :  0.5135615809062076
```

10. Logistic Regression Classifier (class_weight = balanced)





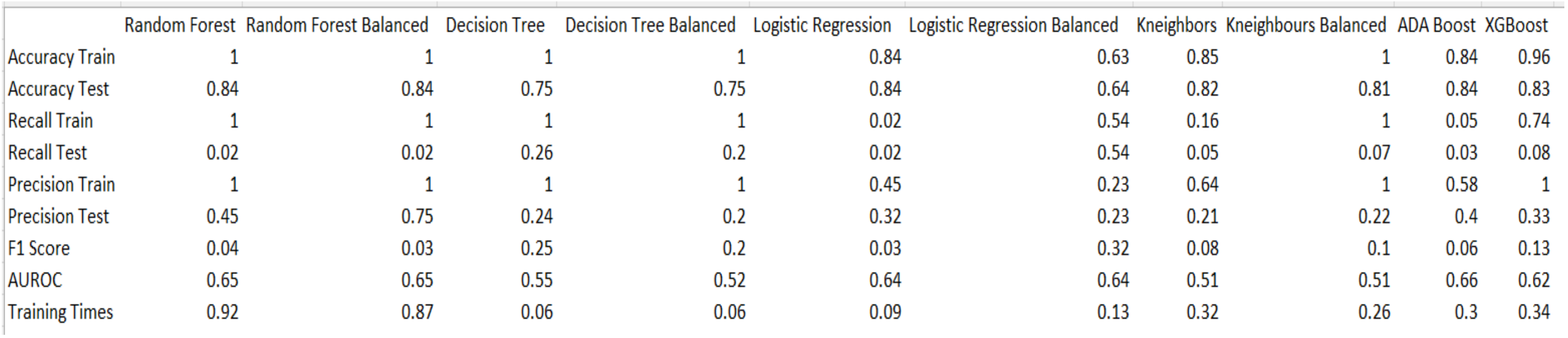
```
              precision    recall  f1-score   support

           0       0.88      0.66      0.75      2012
           1       0.23      0.54      0.32       383

    accuracy                           0.64      2395
   macro avg       0.56      0.60      0.54      2395
weighted avg       0.78      0.64      0.68      2395
```



```
Accuracy on training set :  0.6334400668244466
Accuracy on test set :  0.6375782881002088
Recall on training set :  0.5417391304347826
Recall on test set :  0.5378590078328982
Precision on training set :  0.22828875045804323
Precision on test set :  0.22965440356744704
F1_Score :  0.32187499999999997
Roc_Auc_score :  0.6336186536135666
```

Evaluation Metrics and Conclusion

To compare all the models together, I have created a separate data frame consisting of all the algorithms used in columns, and the evaluation metrics in rows. A snapshot of the results along-with training times for each model tabulated in an excel file is given below:



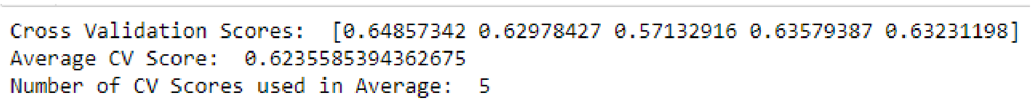
	Random Forest	Random Forest Balanced	Decision Tree	Decision Tree Balanced	Logistic Regression	Logistic Regression Balanced	Kneighbors	Kneighbours Balanced	ADA Boost	XGBoost
Accuracy Train	1	1	1	1	0.84	0.63	0.85	1	0.84	0.96
Accuracy Test	0.84	0.84	0.75	0.75	0.84	0.64	0.82	0.81	0.84	0.83
Recall Train	1	1	1	1	0.02	0.54	0.16	1	0.05	0.74
Recall Test	0.02	0.02	0.26	0.2	0.02	0.54	0.05	0.07	0.03	0.08
Precision Train	1	1	1	1	0.45	0.23	0.64	1	0.58	1
Precision Test	0.45	0.75	0.24	0.2	0.32	0.23	0.21	0.22	0.4	0.33
F1 Score	0.04	0.03	0.25	0.2	0.03	0.32	0.08	0.1	0.06	0.13
AUROC	0.65	0.65	0.55	0.52	0.64	0.64	0.51	0.51	0.66	0.62
Training Times	0.92	0.87	0.06	0.06	0.09	0.13	0.32	0.26	0.3	0.34

While we have calculated Precision, Recall, Accuracy, F1-Score and AUROC, we will be using both F1-Score and AUROC to determine the best model. Since both these metrics are critical, we need to choose models that are good in both metrics – being good in one while poor in the other would not suffice. We can see that ADA Boost has the highest AUROC, followed by Random Forest and Random Forest Balanced. However, for all of these, the F1-Score is poor. Therefore, based on a combination of these two metrics, Logistic Regression Balanced is the best model, because of its highest F1-Score and close to the highest AUROC. Next follow Decision Tree and XGBoost with one being better at F1-Score while the other at AUROC. We will now Cross-Validate these models to assess the stability of our top three models.

Cross Validation

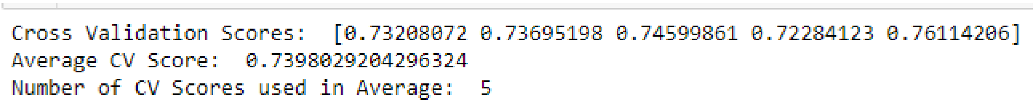
The purpose of cross validation is to see if the model is stable across multiple iterations of the data. If cross validation scores vary significantly across the multiple folds of data it is exposed to, it would reflect that the chosen model is not stable.

Cross Validation: Logistic Regression Balanced



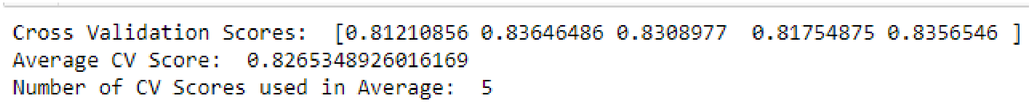
```
Cross Validation Scores:  [0.64857342 0.62978427 0.57132916 0.63579387 0.63231198]
Average CV Score:  0.6235585394362675
Number of CV Scores used in Average:  5
```

Cross Validation: Decision Tree



```
Cross Validation Scores:  [0.73208072 0.73695198 0.74599861 0.72284123 0.76114206]
Average CV Score:  0.7398029204296324
Number of CV Scores used in Average:  5
```

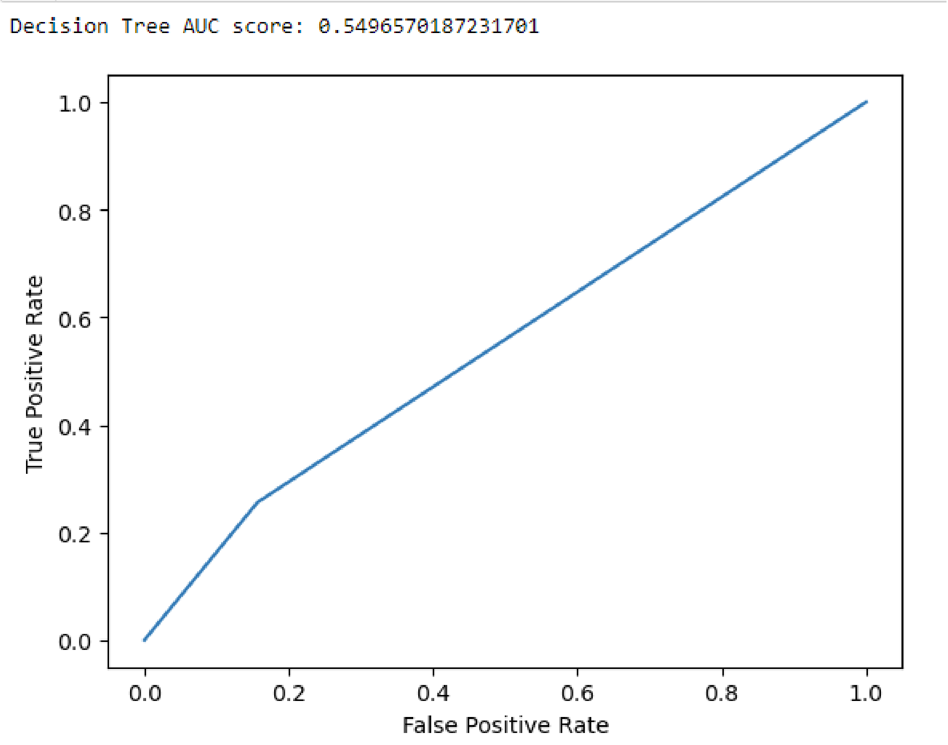
Cross Validation: XGBoost



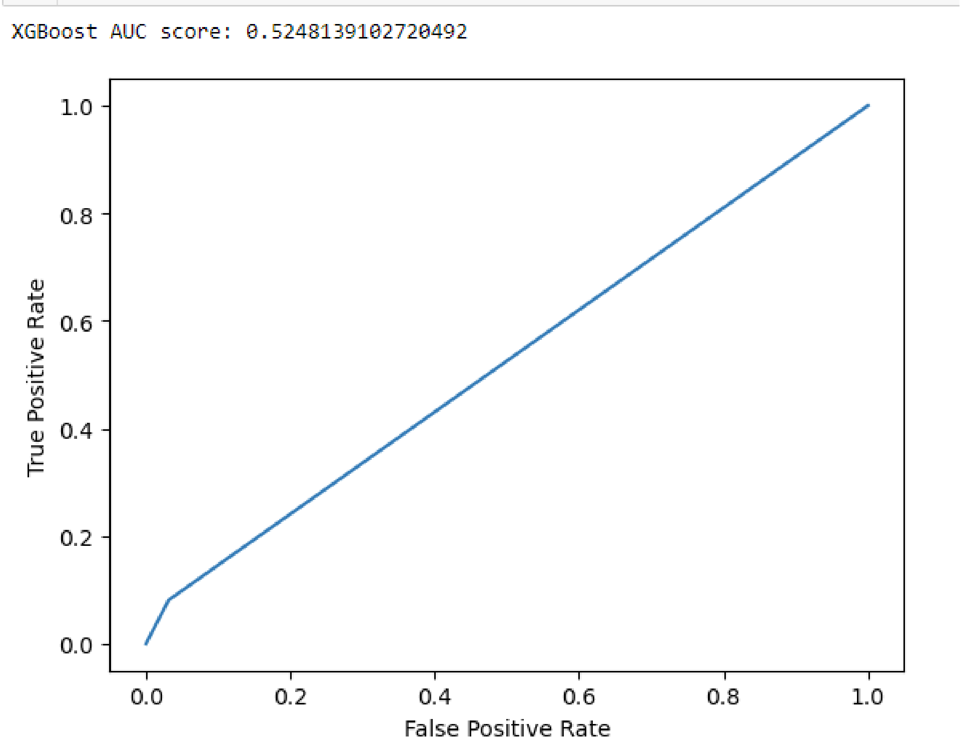
```
Cross Validation Scores:  [0.81210856 0.83646486 0.8308977  0.81754875 0.8356546 ]
Average CV Score:  0.8265348926016169
Number of CV Scores used in Average:  5
```

We see that for all the models, CrossVal scores are close enough across all folds. Not only that, but the average CrossVal score is almost the same as the Accuracy measure presented for the model based on one iteration only. This reflects that our models do not exhibit instability.

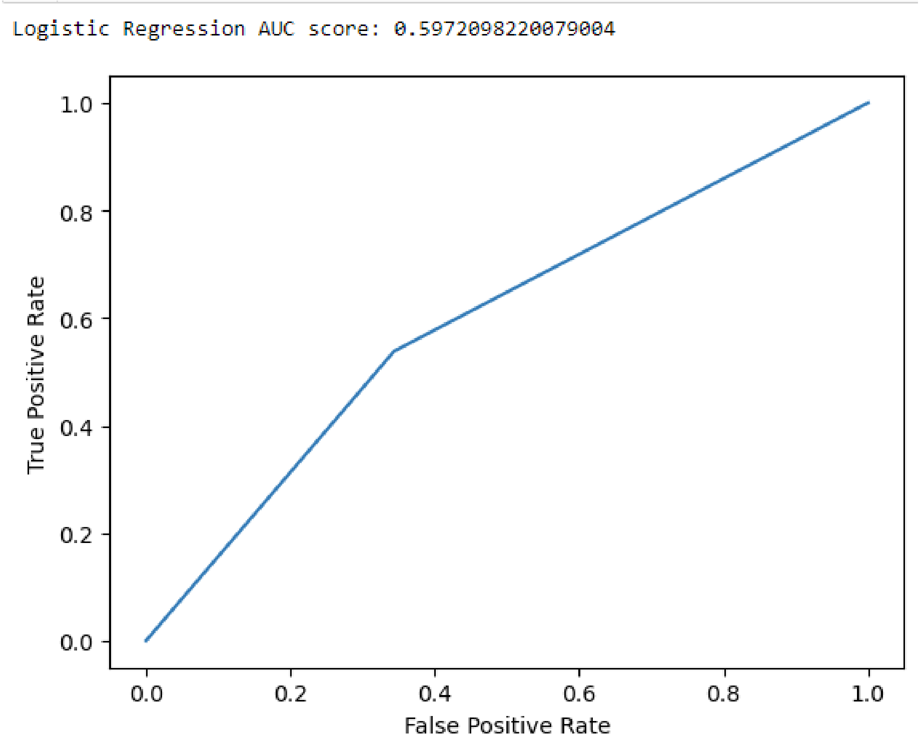
ROC Curve Plots



```
Decision Tree AUC score: 0.5496570187231701
```



```
XGBoost AUC score: 0.5248139102720492
```



```
Logistic Regression AUC score: 0.5972098220079004
```

Discussions and Limitations

At the start of our research, we posed a few research questions. In the first part of this section, we will review answers to those questions based on our findings and model results.

- Logistic Regression Balanced is the best classification algorithm while using F1-Score and AUROC as our evaluation criteria
- Setting a threshold of 0.75, none of the independent variables appeared to be correlated. Therefore, we did not remove any of the variables from our analysis
- Most models have the same (or very close) level of Accuracy on both train and test datasets. These include Logistic Regression, Logistic Regression Balanced, KNeighbors and ADA Boost. However, there were others which did not have the same level of accuracy across train and test datasets. These include Random Forest, Random Forest Balanced, Decision Tree, Decision Tree Balanced, KNeighbors Balanced and XGBoost
- There are some cases where balanced datasets perform significantly better than imbalanced datasets. Some examples where balanced datasets outperform imbalanced ones are:
    - KNeighbors: Accuracy Train, Recall Train and Precision Train are better
    - Logistic Regression: Recall Train, Recall Test and F1-Score are better
    - Random Forest: Precision Test is better
- However, there are a few instances where imbalanced datasets have performed not as well as imbalanced datasets. These include:
    - Logistic Regression: Accuracy Train and Precision Train are poorer
- Overall, the improvement in evaluation metrics from imbalanced dataset to balanced dataset is not as significant as expected

While this research has been completed, there are some limitations to this research, and some improvements that will be performed over the course of the next few months: Some of the limitations are hereunder:

- Hyperparameter tuning could not be performed due to time limitations - Since this project was subject to multi-phase delivery with a tight timeline, hyper parameter tuning exercise could not be performed. This would potentially have helped improve evaluation metric results across our chosen models. This will be done in the second phase of the research
- Usually, lenders are not looking for a yes / no decision. Infact, they are looking for a ranking of borrowers based on their probabilities of default. This process requires some additional research, after which, coefficients of each of the features will be determined which will then be used to calculate PD (Probability of Default)
- Factors considered in this research relate to the profiling borrowers only. However, decision making by banks depends on a number of external economic factors as well. These include economic growth, unemployment rates, the central bank’s monetary policy direction to name a few. However, these factors have not been covered as part of this research.