Wells Fargo Campus Analytics Challenge 2021

Project Report
10/13/2021

Ghanashyam Khanal, Rutgers University
Khanal@physics.rutgers.edu
Khanalg44@gmail.com

1. Introduction

This project is part of the Wells Fargo Campus Analytic Challenge 2021. The details can be found here https://www.mindsumo.com/contests/campus-analytics-challenge-2021.

2. Data Processing

Data processing was one of the main challenges of this project. We processed the given data with different data processing methods before deploying to any machine learning models.

Note: In order to reproduce the figures in this section, please refer to the following jupyter notebook.

`00_EDA.ipynb`

a) The Dataset

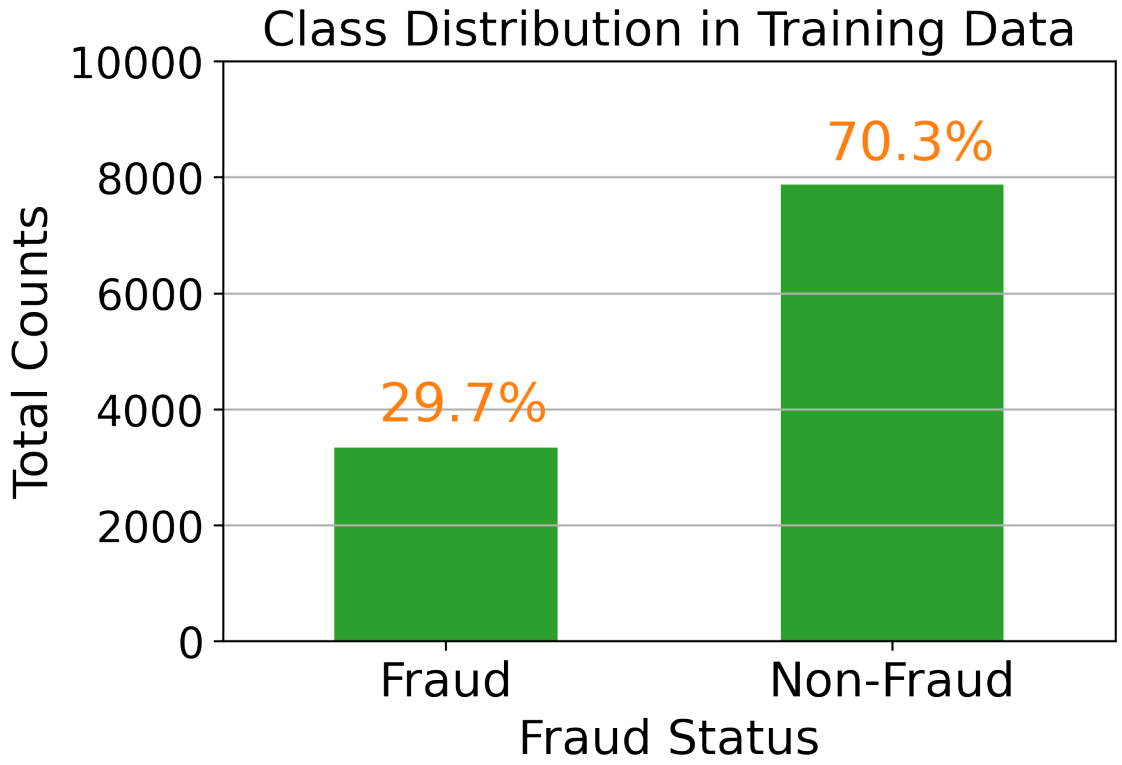


Figure 1: Class imbalance in the training data

b) Feature Engineering

Based on the features available on the original data we found a need to engineer more features which might have an influential role in the modeling. Hence, we came up with the following features.

PWD_UPDT_DAYS: Number of days since the last password update. Filled in with the account days if password update information is unavailable.

PH_NUM_UPDT_DAYS: Number of days since the last phone number update. Filled in with the account days if password update information is unavailable.

TRAN_DAYS: Number of days between the account opening date and transaction date.

PH_NUM_PWD_DAYS: Number of days between the phone number and password updates

c) Data transformation

After the exploratory data analysis (more on this in a later section), we found a need to transform some of the features. For example, the feature "CUST_STATE" has data from almost 40 different states from US. However, only a handful of them have a large number of data entry so we merged the rest of the features to a single categorical value for this feature.

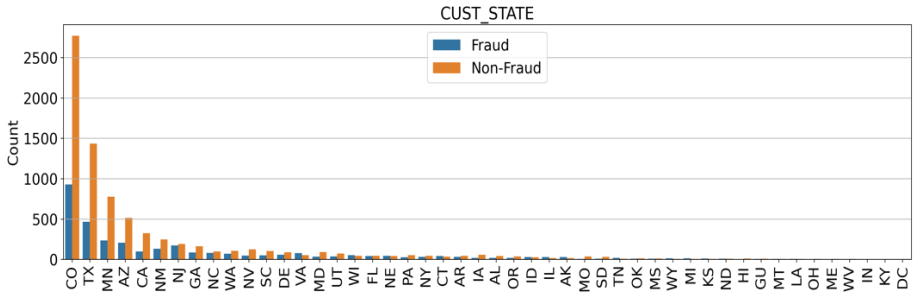


Figure 2: Distribution of the Customer States before feature transformation

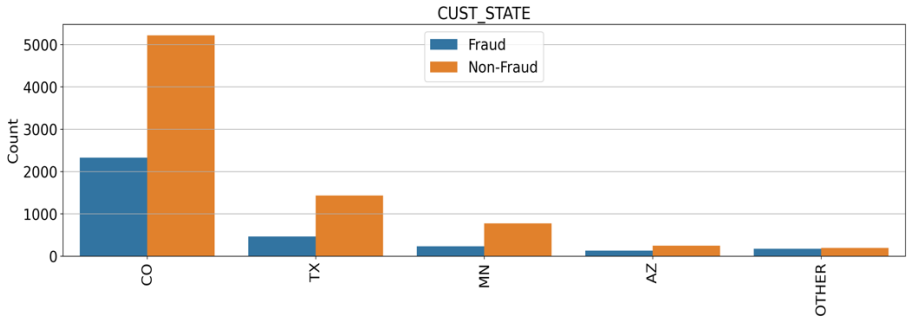


Figure 3: Distribution of the Customer States after feature transformation

d) Missing data and data imputations

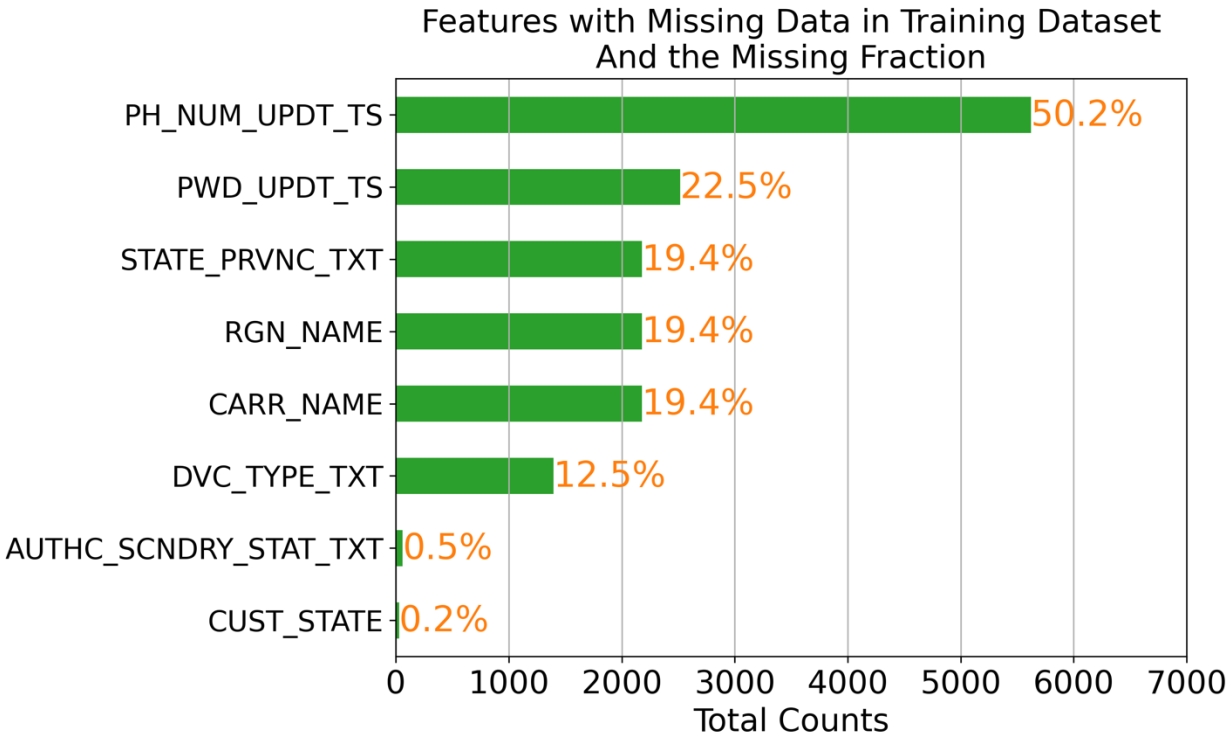


Figure 4: Missing Counts for different features in the training data

As shown in the fig. 4, there were several features in the data which had some number of missing data.

PH_NUM_UPDT_TS: As the feature with the greatest number of missing data, the first instinct was to completely ignore this, however upon deeper investigation we realized this feature

should be kept and so we filled in the missing values by the start date of the customer *CUST_SINCE_DT.*

PWD_UPDT_TS: Similar to *PH_NUM_UPDT_TS,* this feature was also filled in by the start date of the given customer for the missing values.

For other features, we imputed by the mode of the features if categorical and median of the data if numerical. The reason behind choosing median compared to more traditional mean of the data was to avoid having the outlier in the dataset any impact in the data.

We also saved the imputation data for all the features as a variable in anticipation of the missing data in new features in the test data.

e) Encoding the Categorical Variables

We use one-hot-encoding method to encode the categorical features.

3. Exploratory Data Analysis (EDA)

We present the distribution of some of the key features, both given and engineered in the following figures. From the distributions it's clear that Transaction amount is a strong predictor and so is the pre-transaction available balance. Please see supplementary information for more detailed distribution analysis of all the features.

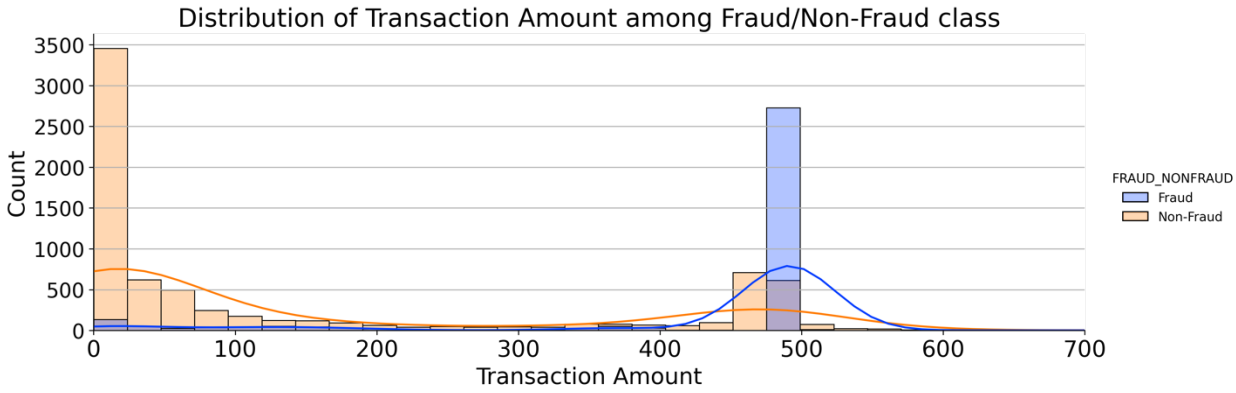


Figure 3: Distribution of Categorical variables

4. Method and Modeling

Various classical machine learning models were implemented for this project: from more classical models like Logistic Regression (LR), Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), XGBoost (XGB), LightGBM (LGBM) to deep learning models like Dense Neural Net (DNN) within Tensorflow (2.0) framework.

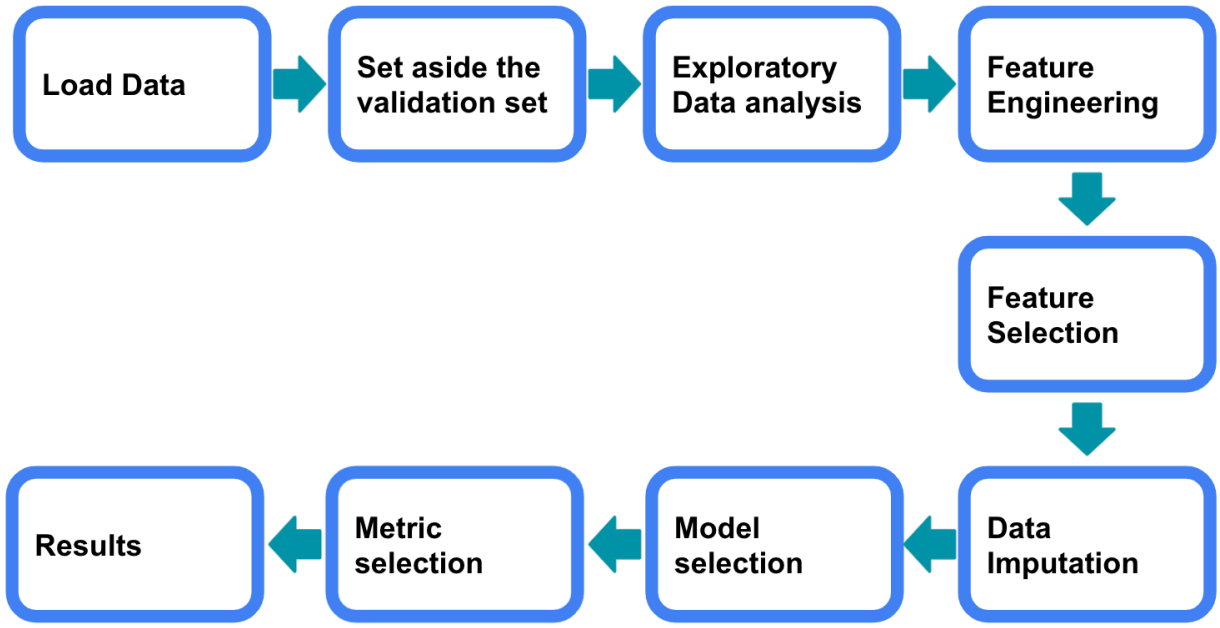
The hyperparameters were optimized using a Grid-Search Cross Validation method as implemented in scikit-learn package.

For all the models we train the models with a 80% of the train data and measure the performance with the test data (20% of the train data) which was put aside even before doing any exploratory data analysis.

For all the models we calculate accuracy score, f1 score, precision, recall, as well as plot the AUC curve, distribution of the predictions for each class and the feature importance score sorted by the score itself.

5. Workflow

Following is the tentative workflow we had for our work.



6. Results

As per the challenge criterion, we focused on evaluating our models based on the F1 score. In the table below we present the results of some of the best models that we tried. All the metrics are results on the validation set, which was set aside even before the exploratory data analysis. It turns out the ensemble methods perform much better than a simple logistic regression

model. Another thing we found was the model performed better when the categorical features were included in the training dataset. That means the features like Cust_State and Alert_Trgr_cd are important in making the correct decision. Among the many ensemble methods we tried, LightGBM method was a clear winner in terms of the ROC AUC score as well as the F1 score. And we use this model to make the prediction for the given test data.

Note: In order to reproduce the results presented on this section jupyter notebooks in python are attached. Among a few the one that produces the most important results is the one named

02_predictive_modelling_2.ipynb

And the file that generates the submission.csv file is named:

05_submission_file.ipynb

Model	Accuracy Score (%)	Precision Score (%)	Recall Score (%)	ROC-AUC (%)	F1 Score (%)
Numerical Features only [Given+ Engineered]					
Logistic Regression	85.2	73.01	65.72	77	35.2
Random Forest	94.21	93.07	87.03	98	89.95
XGBoost	94.07	91.74	88.0	98	89.83
CatBoost	-	-	-	-	-
LightGBM	-	-	-	-	-
Numerical +Categorical Features [Given+ Engineered]					
Logistic Regression	-	-	-	-	-
Random Forest	94.29	93.53	86.79	98	90.04
XGBoost	94.36	92.78	87.88	98	90.26
CatBoost	94.07	92.81	86.79	98	89.7
LightGBM	**94.5**	**93.25**	**87.88**	**99**	**90.48**

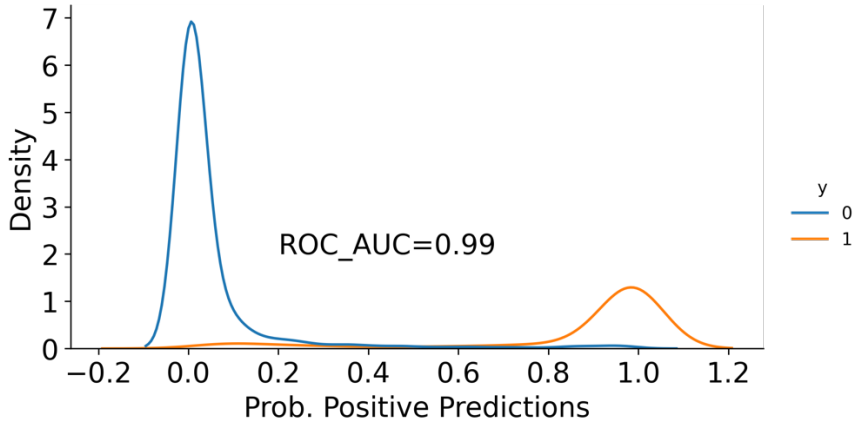


Figure 4: Distribution of the prediction for the Best model (LightGBM)

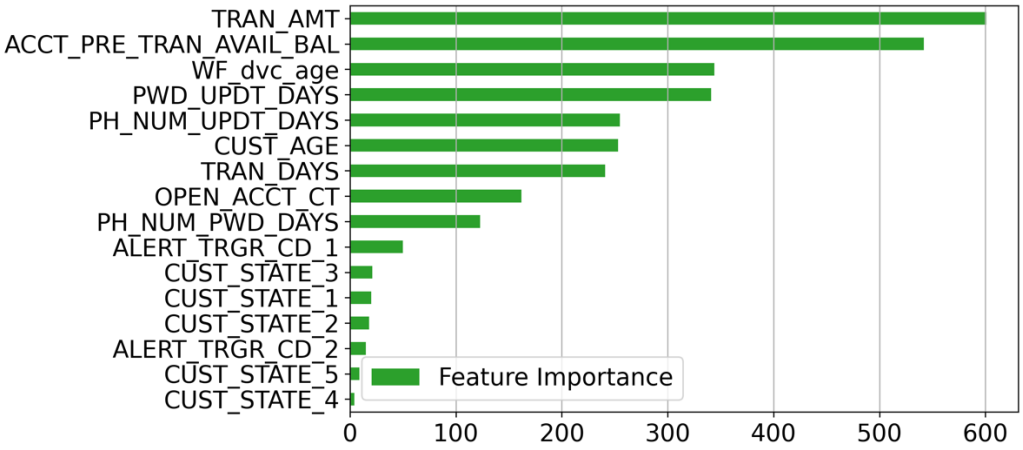


Figure 5: Feature importance obtained by the Best model (LightGBM)

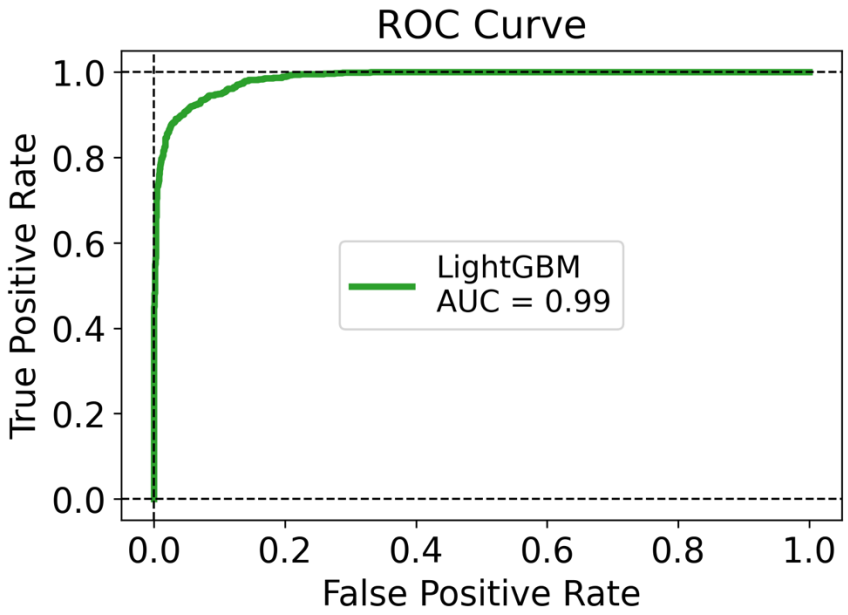


Figure 5: ROC curve and the area under the curve for the Best model (LightGBM)

7. Challenges

During the project, we faced some challenges. For example, for some data points the date on which the phone number was changed precedes the data on which the account was opened. We believe this is partly because of how the data was generated (Using GAN).

8. Summary and Outlook

During this month-long project, we implemented a few machine-Learning models to predict the fraudulent transaction. Our best model performs has a AUC score of 99% which is remarkably good. And so we believe the model is ready to deploy into production. However, given a little more time we would like to try few other techniques for example other encoding methods for the categorical features. And we can even try deep-learning based models.

9. Packages used

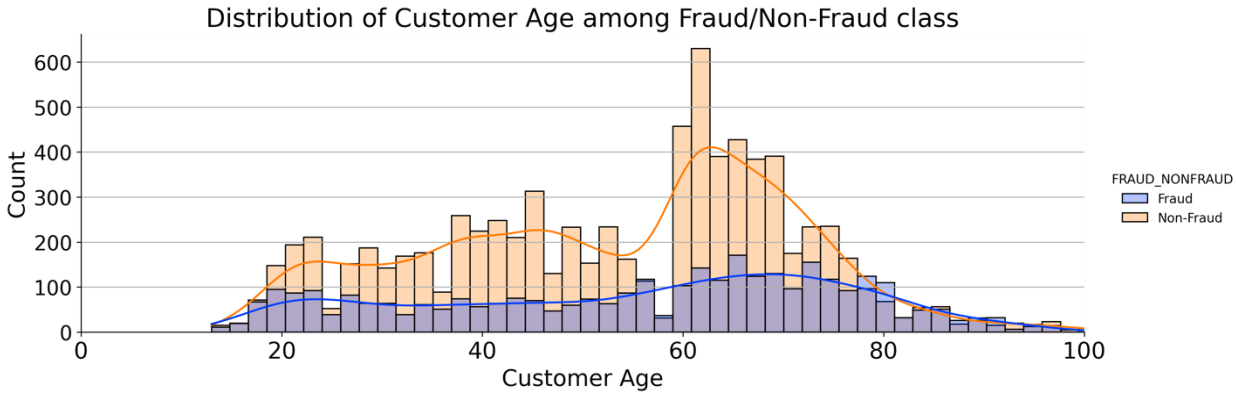
Following are the packages used in this project.

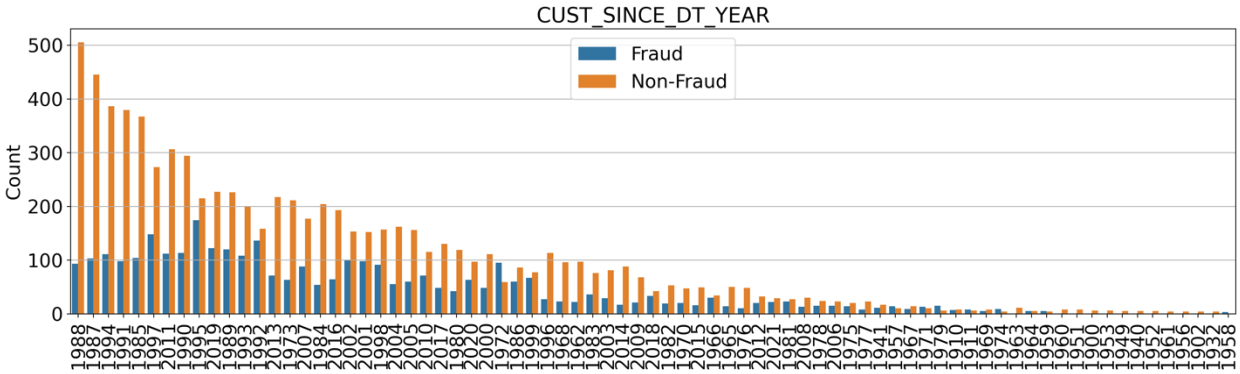
```
Python : 3.9.5
pandas : 1.2.4
```

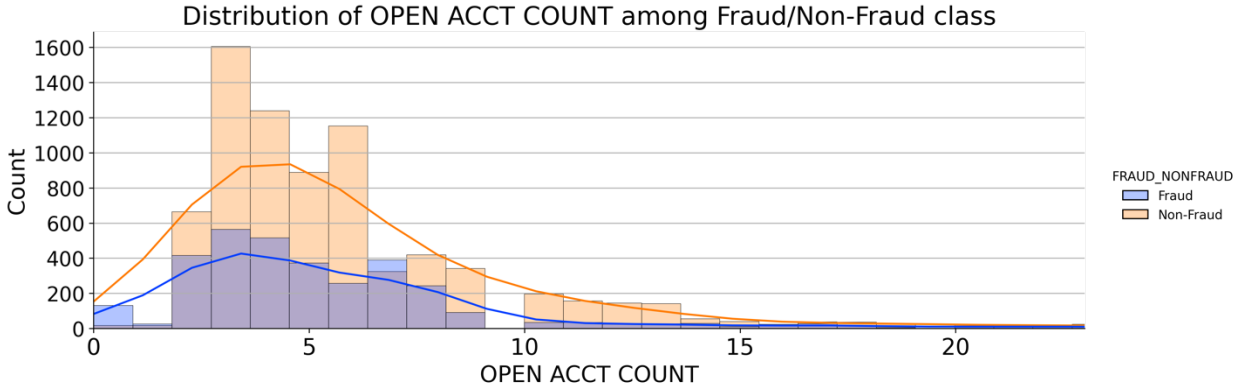
```
numpy : 1.19.5
matplotlib : 3.4.2
seaborn : 0.11.1
sklearn : 0.23.2
xgboost : 1.4.2
catboost : 1.0.0
lightgbm : 3.2.1
```

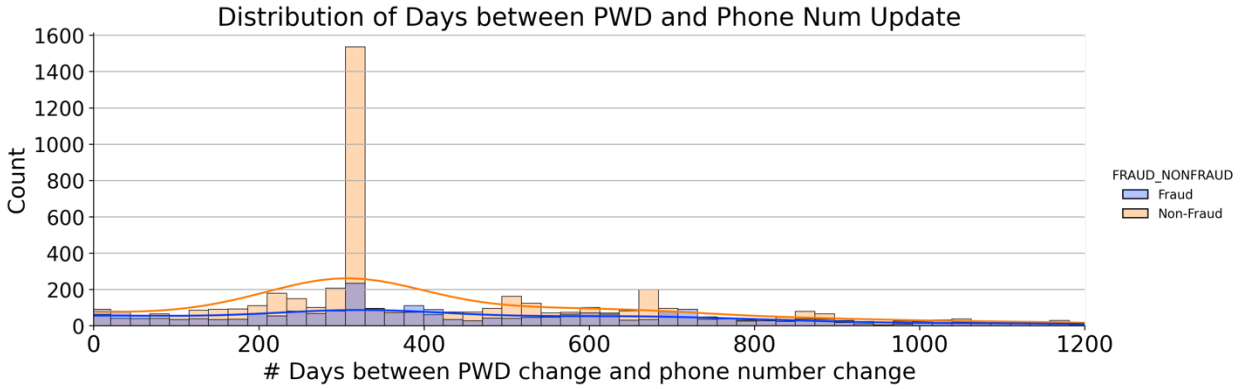
10. Supplementary figures for distribution plots

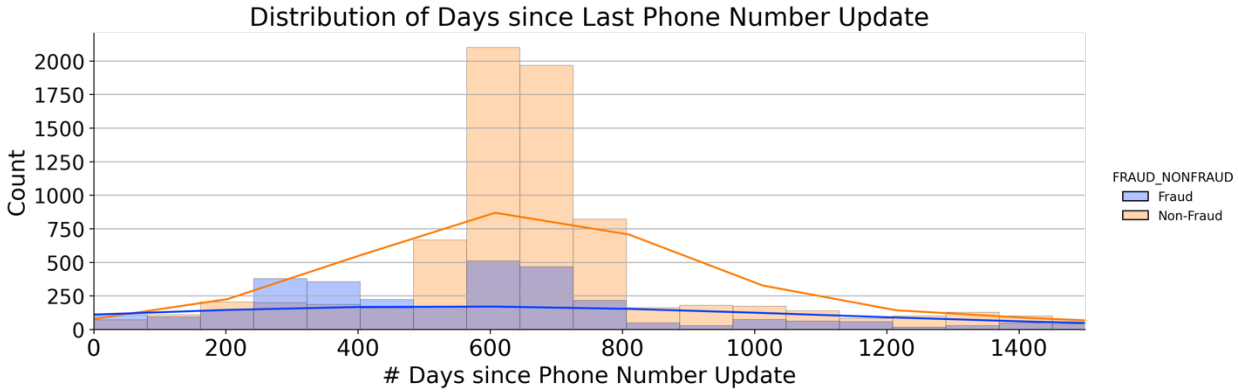
In the following we put some of the plots that we made for the exploratory analysis part.

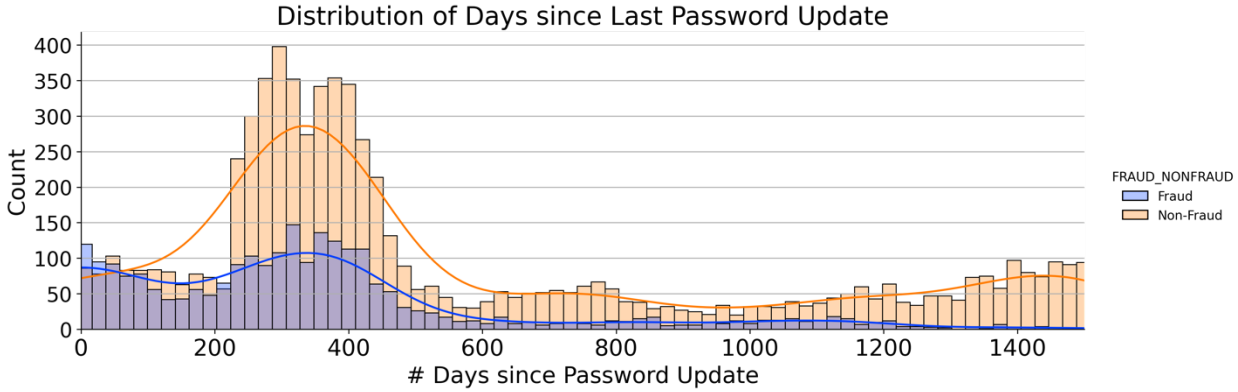


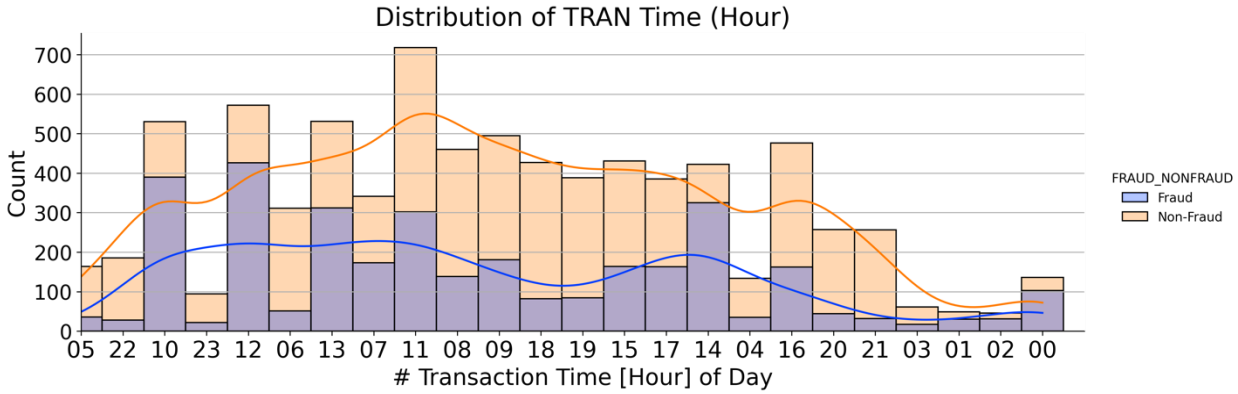

CUST_SINCE_DT_YEAR
Fraud
Non-Fraud
Count
500
400
300
200
100
0
1988 1987 1994 1991 1985 1997 2011 1990 1995 2019 1989 1993 1992 2013 1973 2007 1984 2016 2002 2001 1998 2004 2005 2010 2017 1980 2020 2000 1972 1986 1999 1996 1968 1962 1983 2003 2014 2009 2018 1982 1970 2015 1966 1965 1976 2012 2021 1981 2008 1978 2006 1975 1977 1941 1957 1967 1971 1979 1910 1911 1969 1974 1963 1964 1959 1960 1951 1909 1933 1949 1940 1952 1961 1956 1902 1932 1958

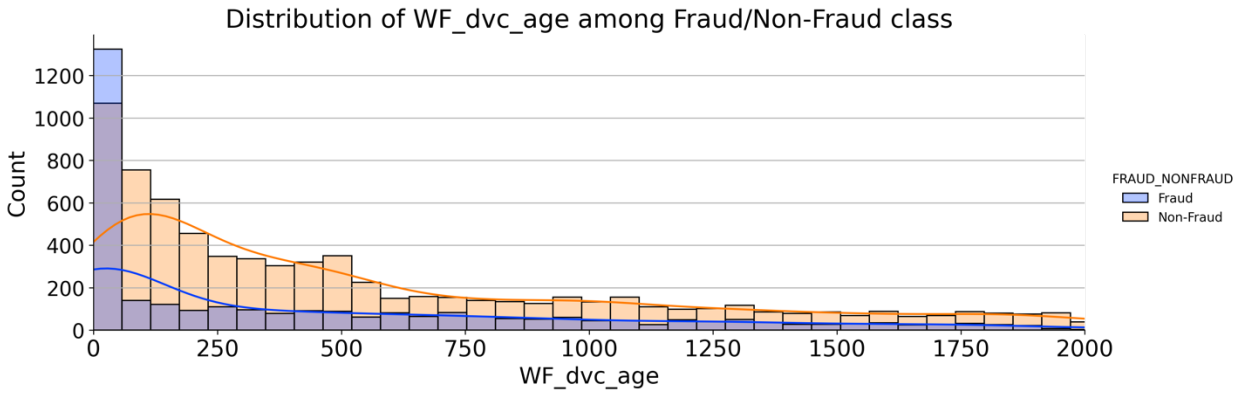

Distribution of OPEN ACCT COUNT among Fraud/Non-Fraud class
Count
1600
1400
1200
1000
800
600
400
200
0
0
5
10
15
20
OPEN ACCT COUNT
FRAUD_NONFRAUD
Fraud
Non-Fraud


Distribution of Days between PWD and Phone Num Update
Count
1600
1400
1200
1000
800
600
400
200
0
0
200
400
600
800
1000
1200
Days between PWD change and phone number change
FRAUD_NONFRAUD
Fraud
Non-Fraud


Distribution of Days since Last Phone Number Update
Count
Days since Phone Number Update
FRAUD_NONFRAUD
Fraud
Non-Fraud


Distribution of Days since Last Password Update
Count
Days since Password Update
FRAUD_NONFRAUD
Fraud
Non-Fraud


Distribution of TRAN Time (Hour)
Count
Transaction Time [Hour] of Day
FRAUD_NONFRAUD
Fraud
Non-Fraud



Supplementary Results

XGboost model

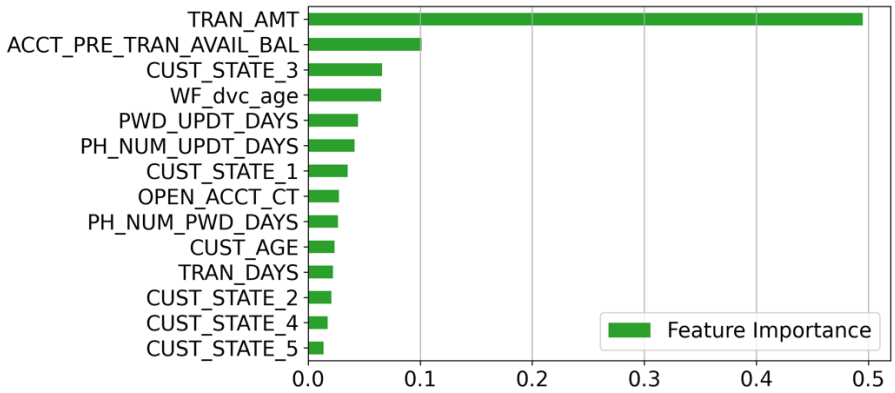
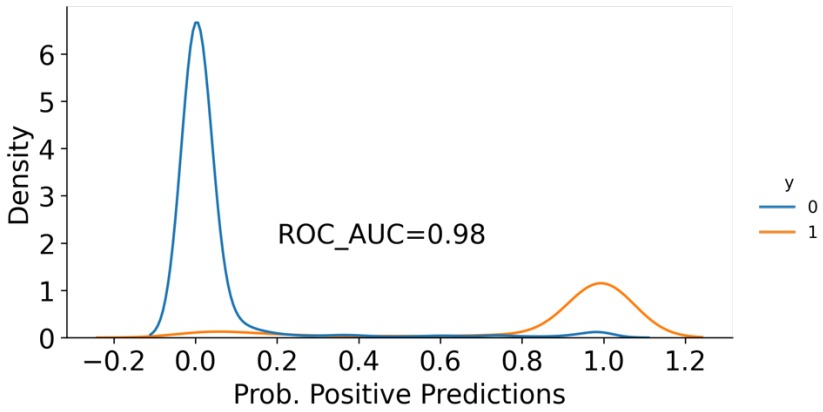


Figure: Feature importance for XGBoost model

Note: Class 0 and 1 represent non-fraud and fraud class respectively



Random Forest

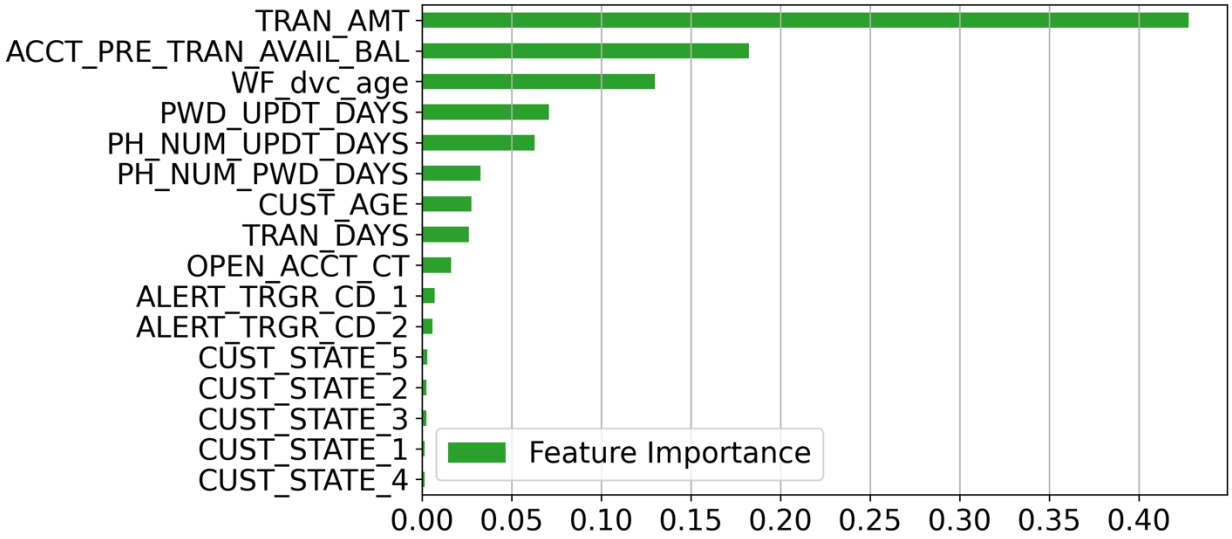
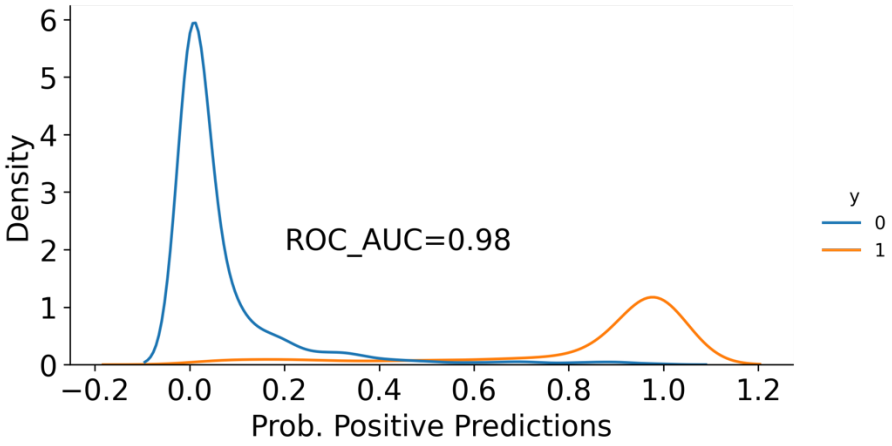


Figure: Feature importance for Random Forest model

Note: Class 0 and 1 represent non-fraud and fraud class respectively


Density
6
5
4
3
2
1
0
ROC_AUC=0.98
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Prob. Positive Predictions
y
0
1