MS4610: Introduction to Data Analytics

Report

Group 22

PayBuddy Case Study Propensity Prediction Model

Group Members

Avinash Potnuru	ME19B006
B Sohana Preeth	ME19B008
Amritha K	ME19B078
Harshitha	ME19B111
Raghu Kattubadi	ME19B125
Saran Kumar	CS20B069

Contents

1. PROBLEM STATEMENT

Aim : To build a propensity model to predict the customer propensity to apply for the Evolution card using the following datasets:

1. fraud_update_features --> fraud based features
2. new_acct_memo_rcv --> received transaction based features
3. new_acct_fi --> financial instrument based features
4. new_acct_memo_sent --> emoji based features
5. new_acct_receiver --> Sender engagement features
6. new_acct_sender --> Receiver engagement features
7. payment_cashout_fail --> Failed withdrawal features
8. train_df --> Training Dataset
9. test_cust_id --> Testing Dataset

The Training Dataset contains 720000 rows and the Testing Dataset contains 180000 rows.

2.Initial Trial using Random Forest Model:

2.1 Data Preprocessing:

All the above mentioned 9 datasets were imported.

Feature Selection:

Correlation:

The correlations of features across the first 7 datasets were calculated and visualised using heatmaps. All the features which had a correlation greater than 0.9 were dropped from their respective datasets.

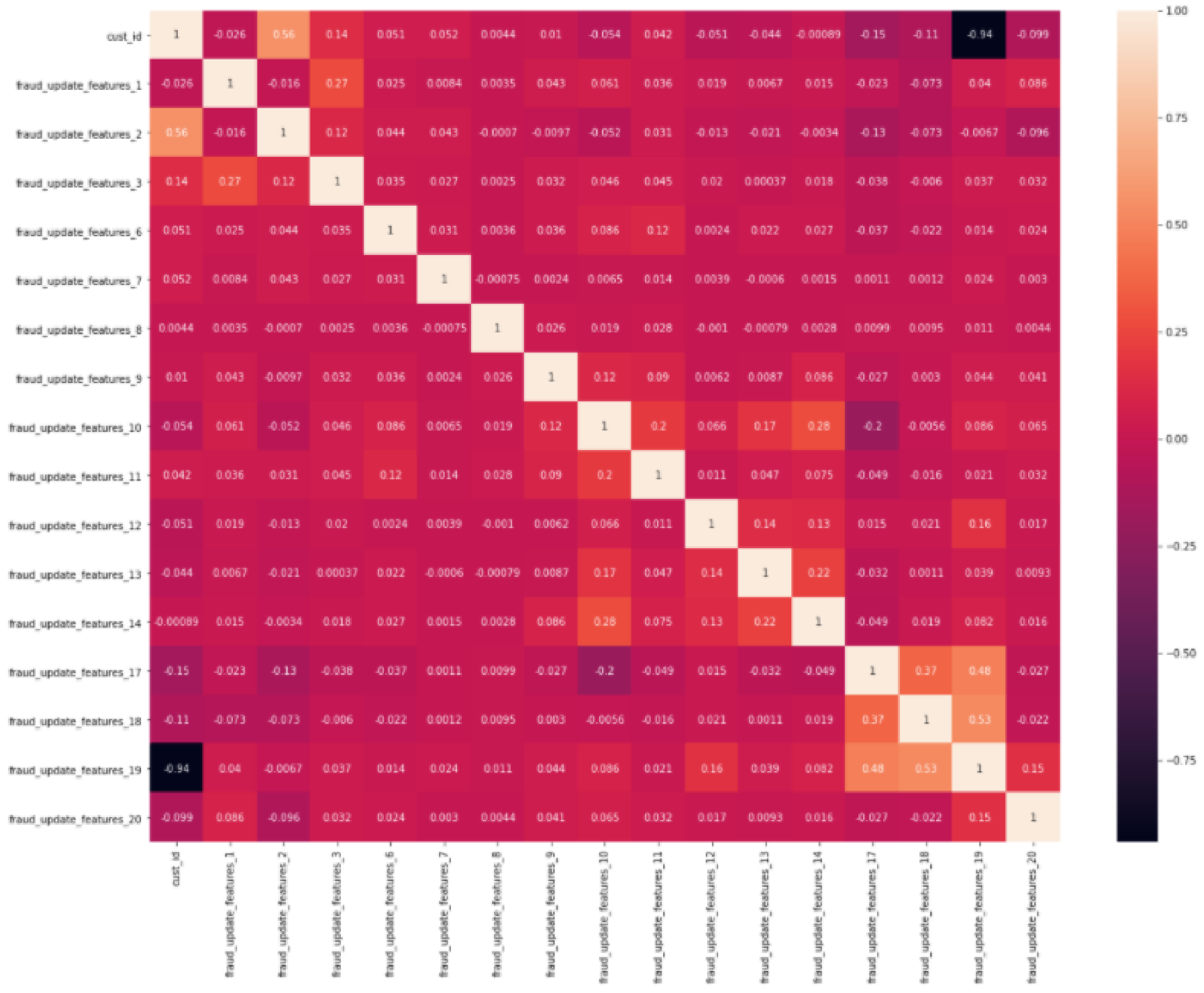


Figure: Correlation between features in 'fraud_update_features' Dataset

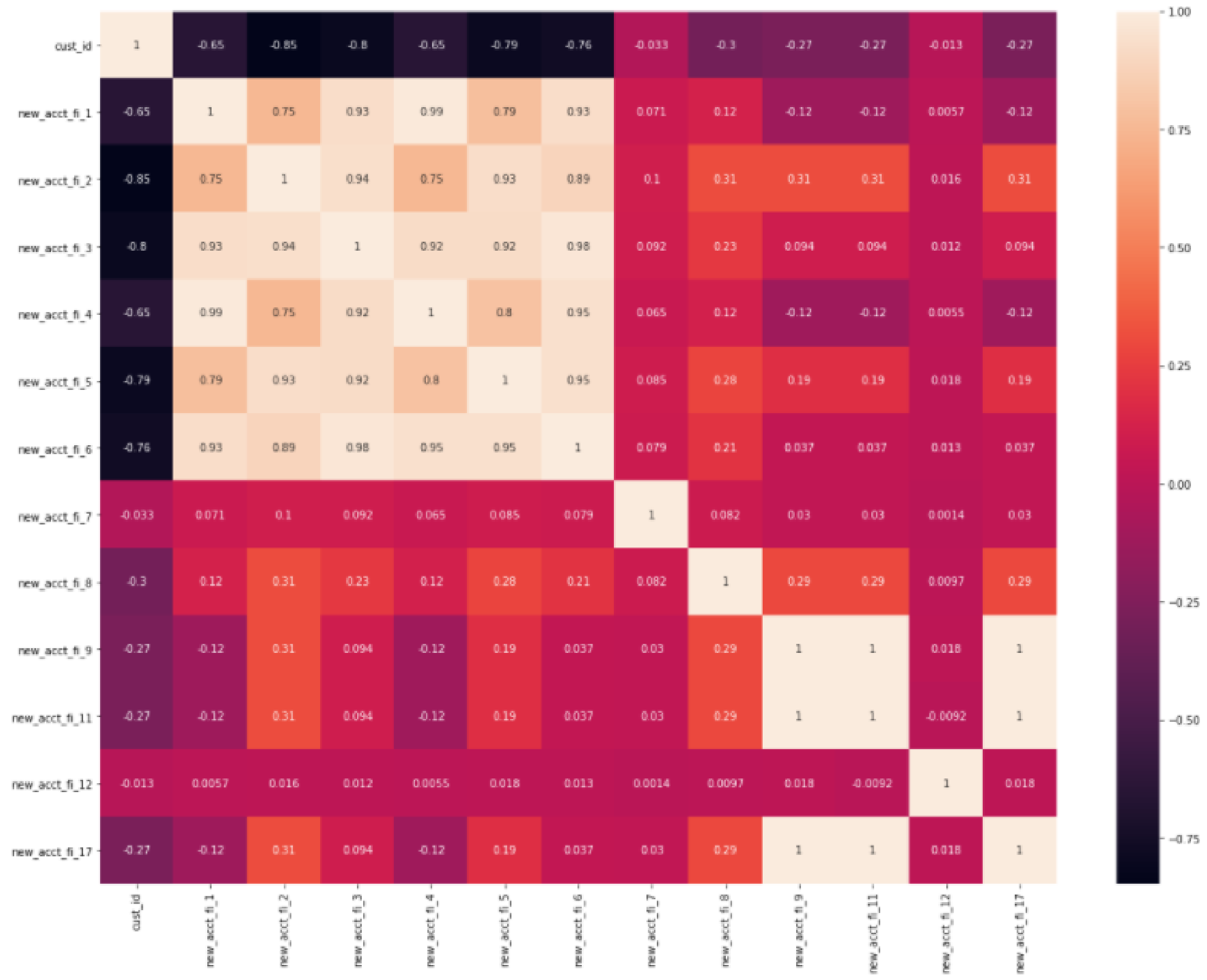
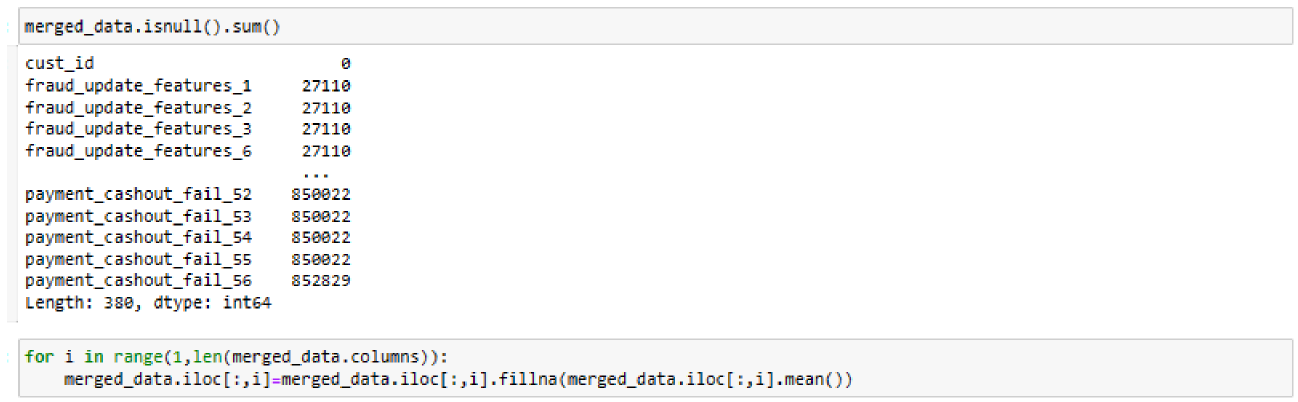


Figure: Correlation between features in ‘new_acct_fi’ Dataset

After removing all the correlated columns , the 7 datasets were merged using pandas.merge to form ‘merged_data’ dataset.

Missing Values:

All the columns of the merged dataset were checked for missing values. The missing values in each column were filled using the mean of that column.



```
merged_data.isnull().sum()
```

```
cust_id                        0
fraud_update_features_1    27110
fraud_update_features_2    27110
fraud_update_features_3    27110
fraud_update_features_6    27110
                            ...
payment_cashout_fail_52   850022
payment_cashout_fail_53   850022
payment_cashout_fail_54   850022
payment_cashout_fail_55   850022
payment_cashout_fail_56   852829
Length: 380, dtype: int64
```

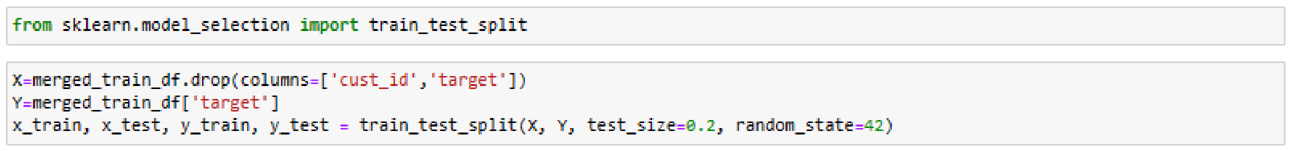
```
for i in range(1,len(merged_data.columns)):
    merged_data.iloc[:,i]=merged_data.iloc[:,i].fillna(merged_data.iloc[:,i].mean())
```

Figure: **Number of missing values in the data set and filling them using mean**

Merged train dataset was formed by merging merged_data and train_df datasets. Merged test dataset was formed by merging merged_data and testcust_id.

2.2 Train-Test Split Evaluation:

The training dataset was split into train and test data using train_test_split from sklearn.model_selection.



```
from sklearn.model_selection import train_test_split
```

```
X=merged_train_df.drop(columns=['cust_id','target'])
Y=merged_train_df['target']
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

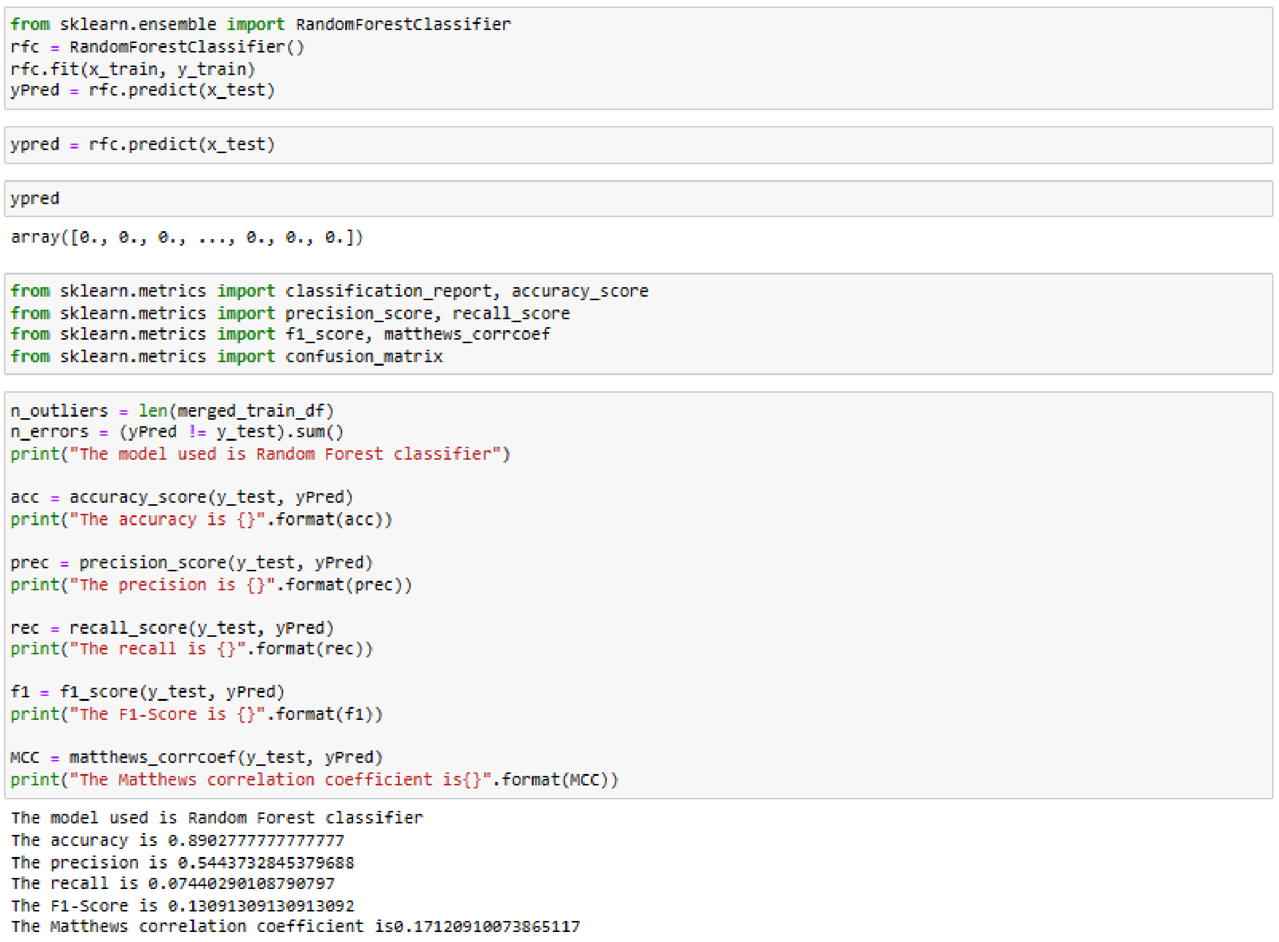
Figure: **train_test_split on training dataset**

2.3 Random Forest:

The model was trained using RandomForestClassifier from sklearn.ensemble library with x_train and y_train and tested on x_test and y_test which gave an accuracy of around 89%.

2.4 Predictions from the test dataset:

The model was applied on the test dataset to get the probabilities of customers applying for the credit card which gave a KS Score of 0.714 (approx).



```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train, y_train)
yPred = rfc.predict(x_test)
```

```
ypred = rfc.predict(x_test)
```

```
ypred
```

```
array([0., 0., 0., ..., 0., 0., 0.])
```

```
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score, matthews_corrcoef
from sklearn.metrics import confusion_matrix
```

```
n_outliers = len(merged_train_df)
n_errors = (yPred != y_test).sum()
print("The model used is Random Forest classifier")

acc = accuracy_score(y_test, yPred)
print("The accuracy is {}".format(acc))

prec = precision_score(y_test, yPred)
print("The precision is {}".format(prec))

rec = recall_score(y_test, yPred)
print("The recall is {}".format(rec))

f1 = f1_score(y_test, yPred)
print("The F1-Score is {}".format(f1))

MCC = matthews_corrcoef(y_test, yPred)
print("The Matthews correlation coefficient is{}".format(MCC))
```

```
The model used is Random Forest classifier
The accuracy is 0.8902777777777777
The precision is 0.5443732845379688
The recall is 0.07440290108790797
The F1-Score is 0.13091309130913092
The Matthews correlation coefficient is0.17120910073865117
```

Figure: RandomForestClassifier Model

3. FINAL MODEL:

3.1 Data cleaning and merging:

Relevant notebook- 'data_merging.ipynb'

- Removed features with unique value (0 variance).
- Removed irrelevant columns.
- Merged all 9 data sets to a single data frame 'df_f' which contains all information using 'cust_id' as a common column.

- Then we have divided the training data (for which the target variable is known) and test data (for which the target variable is unknown)
- The final 4 data frames X(training data features), y(training target), X_t (test data features) and test_cust (customer ids of test data) were uploaded in the files.

3.2 Model Validation:

Relevant notebooks: 'model_validation_xgb.ipynb', 'model_validation_lgbm.ipynb', 'model_validation_catboost.ipynb'

We tried out various ML models in which 'XGBClassifier' , 'LGBMClassifier' and 'CatBoostClassifier' gave the best scores.

The performance scores of these models on validation data are shown in the table.

ML Models	Accuracy	Precision	Recall	F1 score
XGBClassifier	0.891627	0.5798098	0.088476	0.153525
LGBMClassifier	0.892177	0.5873583	0.098529	0.168751
CatBoostClassifier	0.891505	0.5595695	0.109232	0.182784

NOTE:

1) We tried *SMOTE* for over sampling and *RandomUnderSampler* but they didn't give better results. Thus, we removed it.
2) We also tried a few feature selection methods like *SelectKBest* and *Random Forest feature selection* but the performance score decreased when we used the same.

3.3 Important Features:

Important Features according to the three models are shown below and the same can be found in the respective model validation code files.

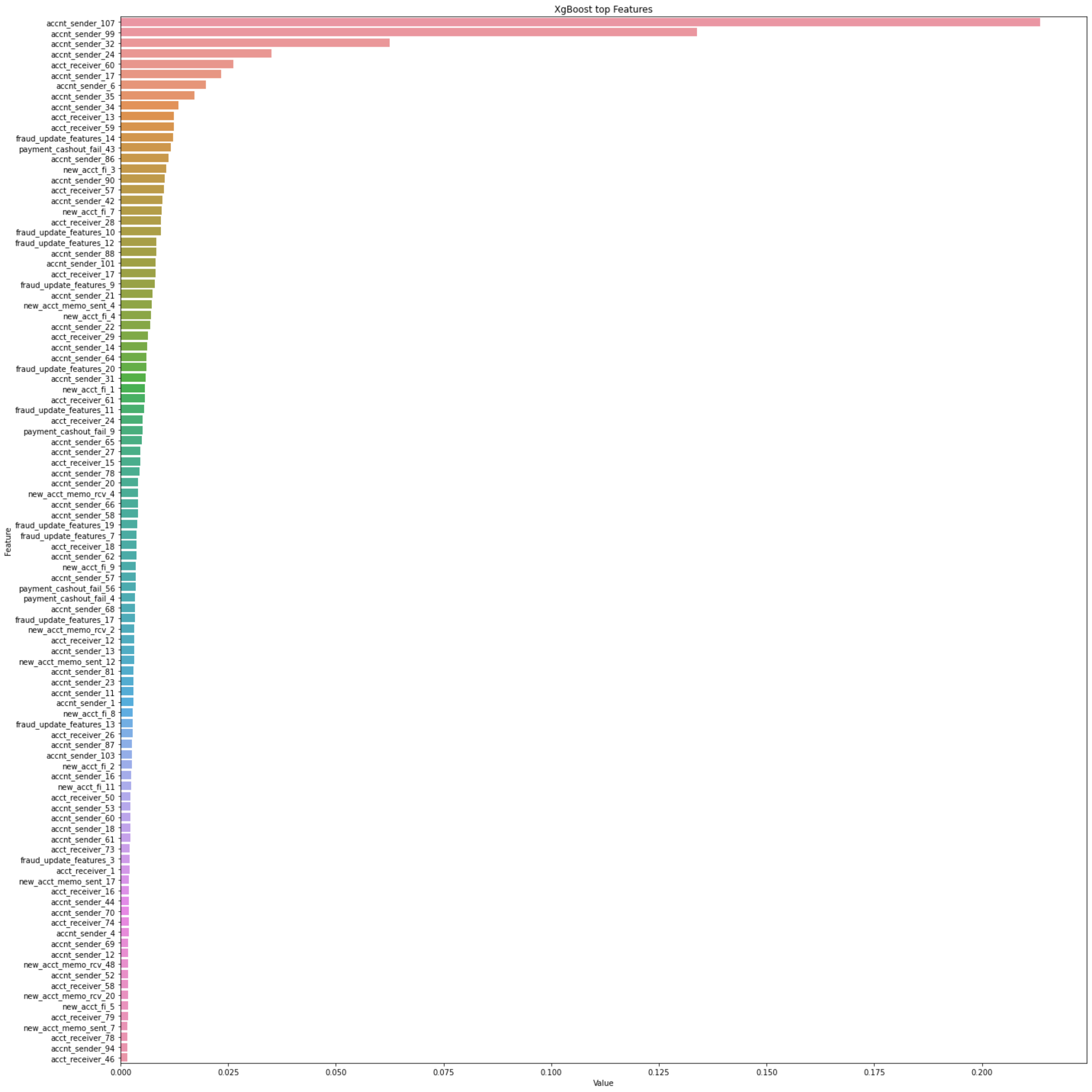


Figure: XGBoost Top Features

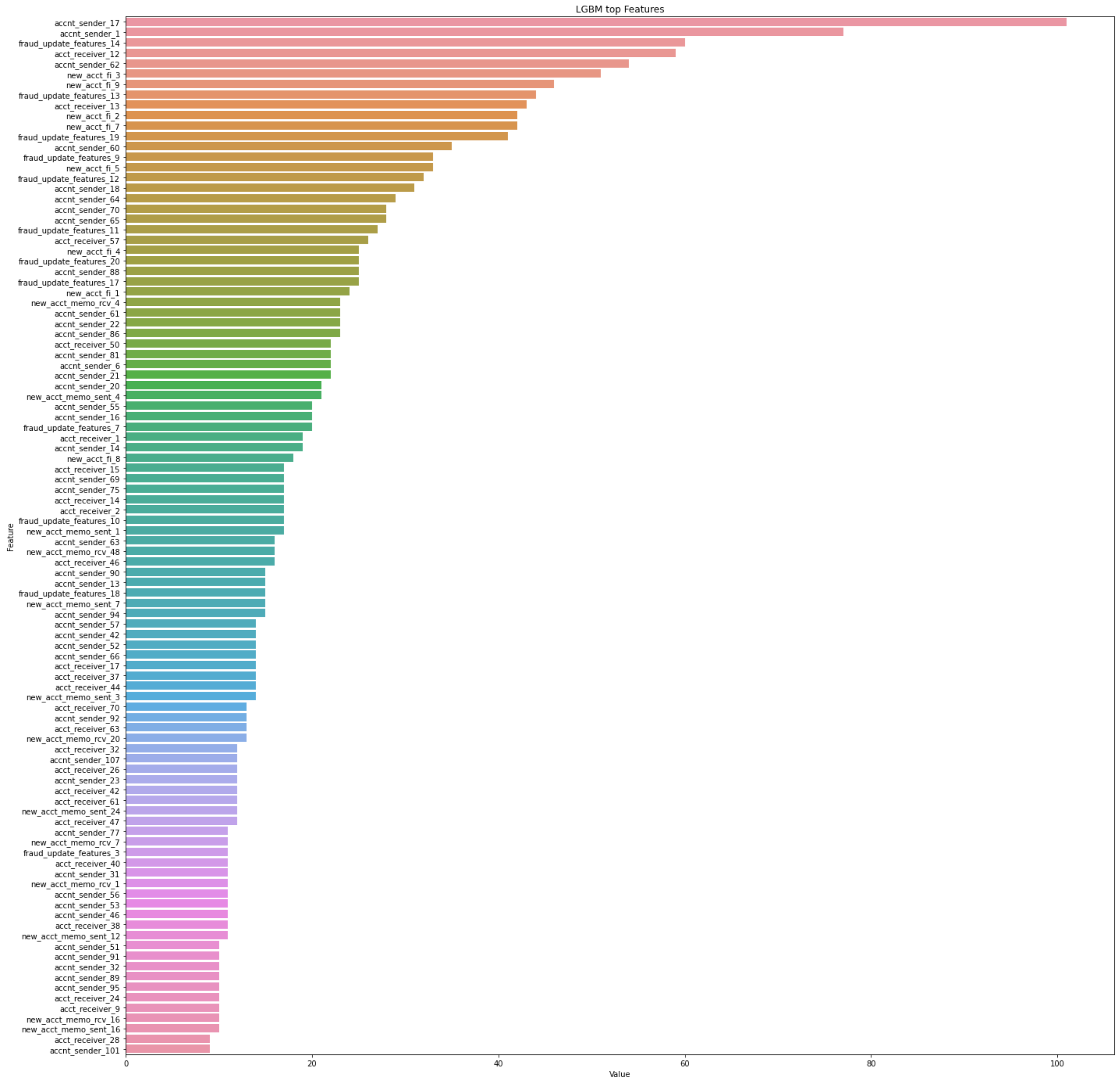


Figure: LightGBM Top Features

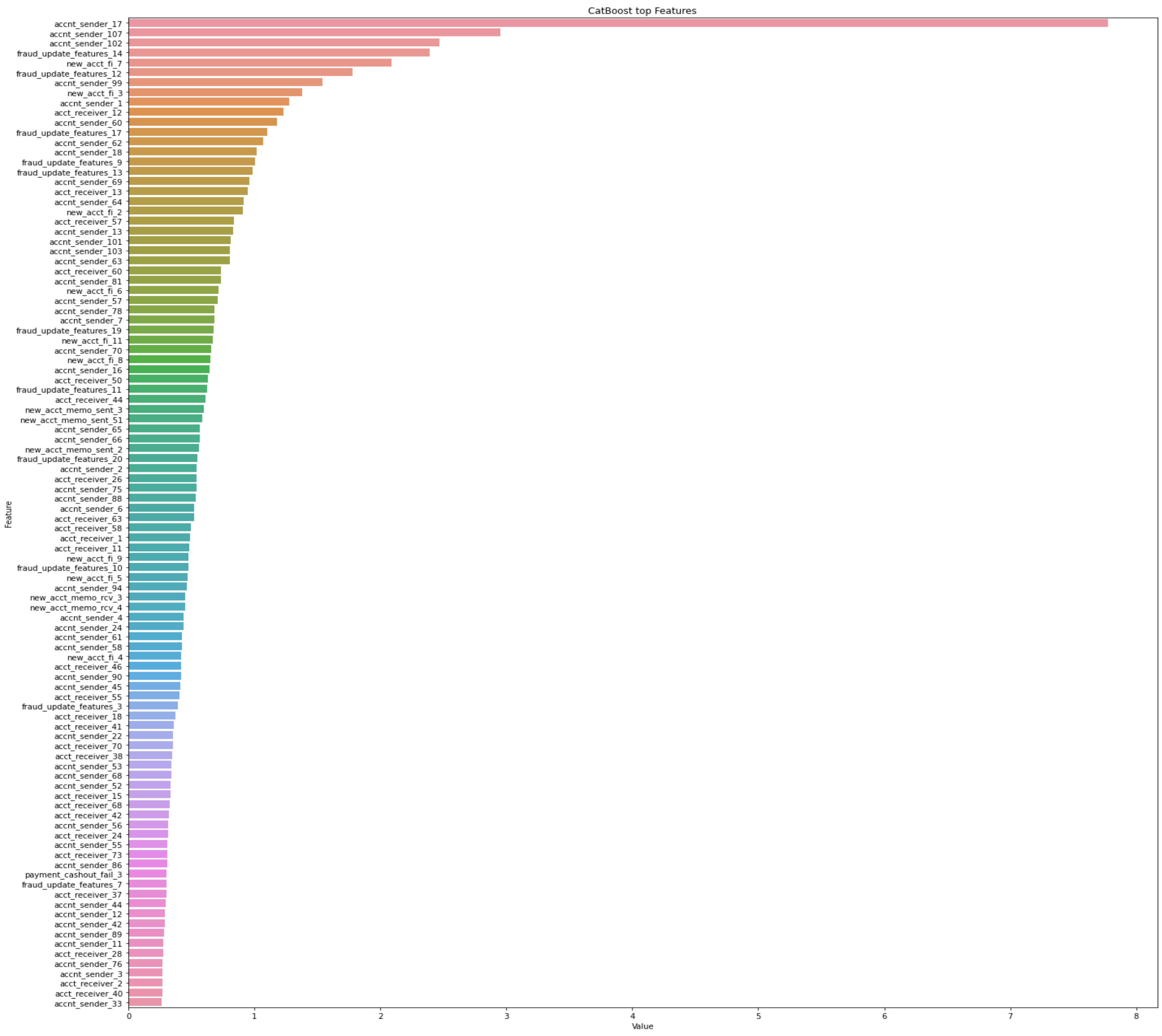


Figure: CatBoost Top Features

3.4 Final Results:

Relevant Notebook: 'final_result_catboost.ipynb', 'final_result_lgbm.ipynb', 'final_result_xgb.ipynb'.

Finally those three models were applied to the whole training data set and probabilities for the test data were also submitted in the competition portal.

Among the three models, CatBoostClassifier gave the best score of 0.75661 (as submitted on Kaggle leaderboard).