Credit Card Fraud Detection Report

Sherly Hartono

Overview

Credit card fraud by definition is the fraudulent use of a credit card through the theft of the cardholder’s personal detail. With the rise of e-commerce, the problem of fraudulent use of credit cards has become more acute.

Given previous transaction datas, can we predict whether or not a new transaction is a fraudulent one ?

Losses to fraud incurred by payment card issuers worldwide reached $27.85 billion in 2018 and are projected to rise to $40.63 bullion in 10 years. Answering the above problem can reduce these damages faced by merchants and credit card issuers. Our goal is to correctly classify the minority class of fraudulent transactions.

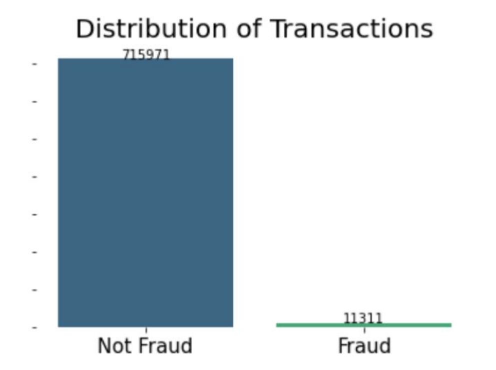
Data Wrangling

- Remove column with empty strings
- Impute Country and CountryCode with mode
- Convert datetime from string datetime object
- Remove useless columns: customerID, enteredCVV, cardCVV

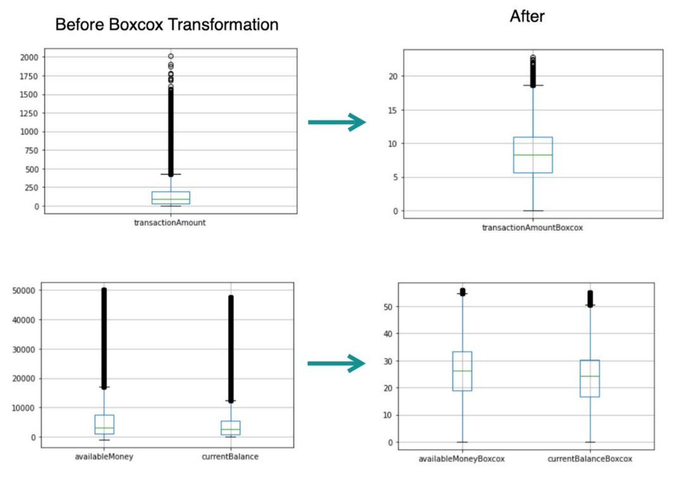
- Create new features:
- containsCom.
- lengthOfLast4Digits

EDA

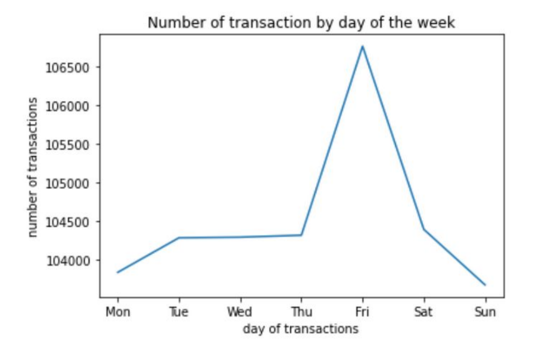
- Imbalance data



- Box Cox Transformation to treat collective outliers



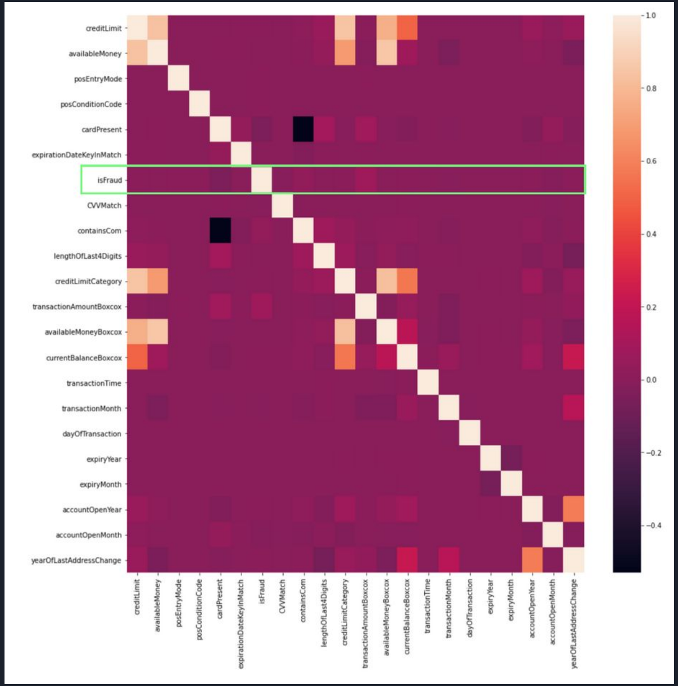
- Date time trends



EDA

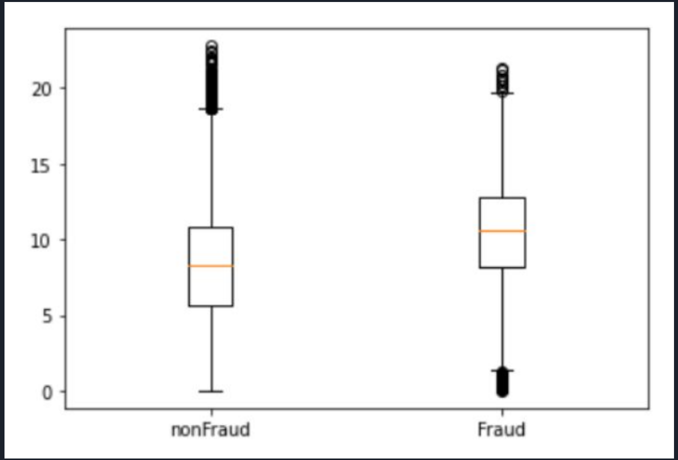
- ## Correlation Matrix

Positive correlation between fraudulent transaction and 'transactionAmountBoxCox' and 'containsCom',



- ## Hypothesis Testing using T-statistic

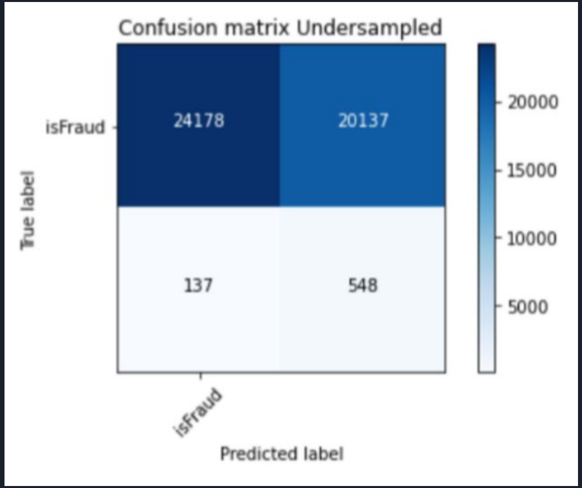
There is a significant difference between the transaction amount of fraud and non fraudulent transaction.



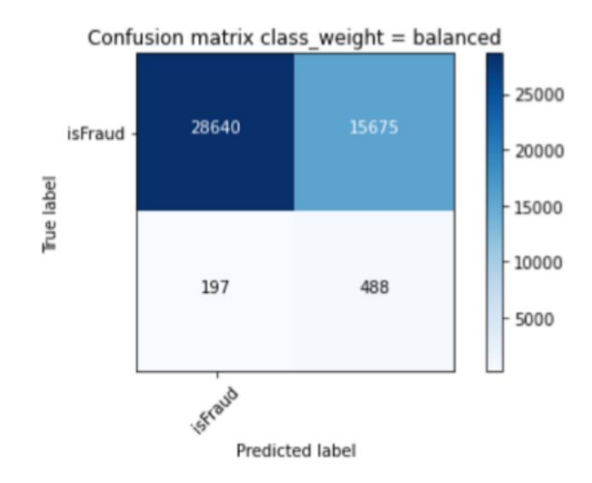
Model 1: Logistic Regression

Method 1: Undersampling

- Roc_auc score of 0.74



Method 2: set class_weight to balanced



Undersampling gives better result (Minimize False negative)

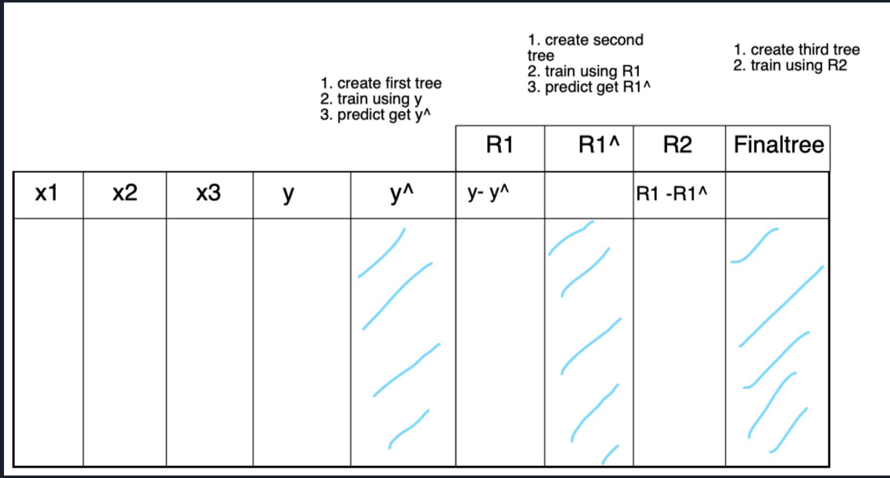
Model 2 : Random Forest

- n_estimators = number of trees in the forest
- max_features = max number of features considered for splitting a node
- max_depth = max number of levels in each decision tree
- min_samples_split = min number of data points placed in a node before the node is split
- min_samples_leaf = min number of data points allowed in a leaf node
- bootstrap = method for sampling data points (with or without replacement)
- Use RandomizedCV

- roc_auc score of 0.8

Model 3: XG Boost

- XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable.
- While regular gradient boosting uses the loss function of our base model (e.g. decision tree) as a proxy for minimizing the error of the overall model, XGBoost uses the 2nd order derivative as an approximation
- Each time we run a decision tree, we extract the residuals. Then we run a new decision tree, using those residuals as the outcome to be predicted. After reaching a stopping point, we add together the predicted values from all of the decision trees to create the final gradient boosted prediction.
- roc_auc at 0.79



1. create first tree
2. train using y
3. predict get y^

1. create second tree
2. train using R1
3. predict get R1^

1. create third tree
2. train using R2

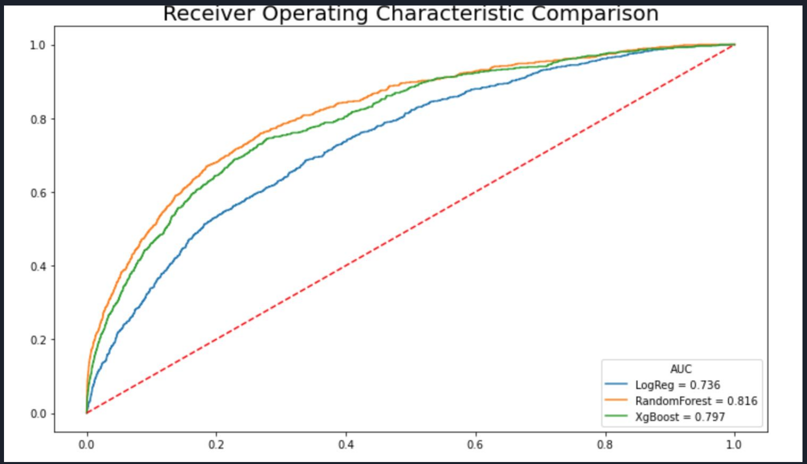
					R1	R1^	R2	Finaltree
x1	x2	x3	y	y^	y- y^		R1 -R1^	

Evaluation

RandomForest model performs the best.

Possible reason:

- XGB model is more sensitive to overfitting if the data is noisy.
- RF are harder to overfit than XGB.



Evaluation: Feature Importance

The top important features agree with our correlation matrix we did in EDA. Higher amounts of transactions did give us a clue on whether or not a transaction is fraudulent. Its importance is more than double than other features.

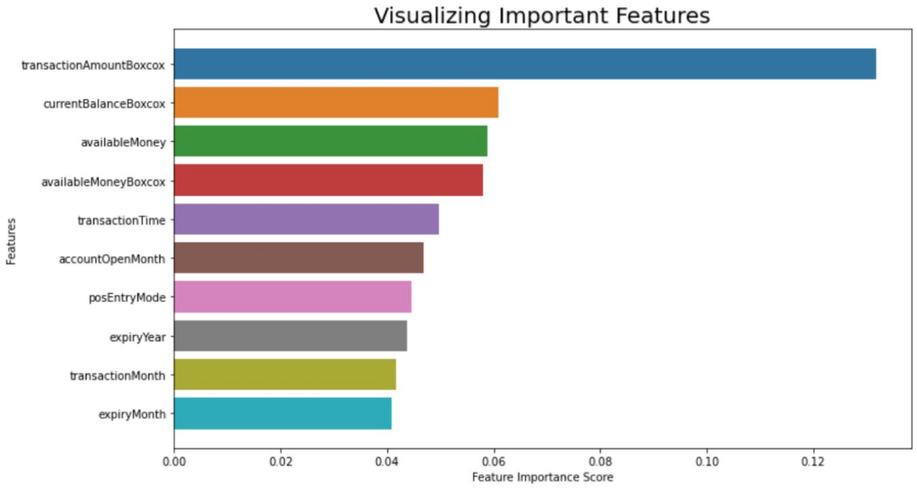


Fig 4.4 top ten important features in RandomForest model

Future Work

- Train on more data

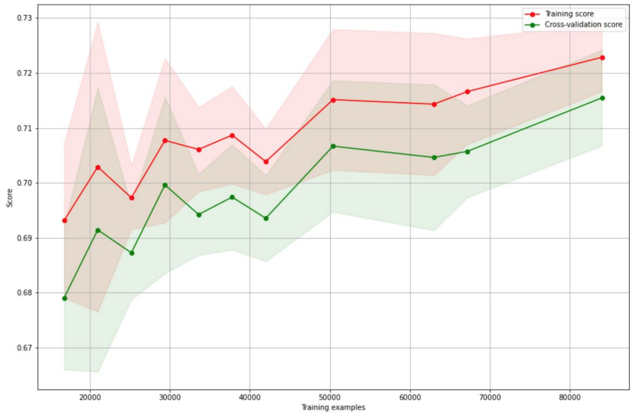


Fig 4.5 Learning curve for RandomForest model

- Use Bayesian Optimization instead of stopping at RandomSearch in our hyperparameter tuning.
- Use classification models to predict the missing values instead of only using mode
- Use target encoding instead of one-hot encoding to avoid curse of high dimensionality