CREDIT CARD FRAUD DETECTION

Shuddhendu Mishra, Vaibhavi Deshpande, Sayali Bhosale

February 26, 2023

OUTLINE

- Background
- Problem Statement
    - Why addressing this problem is important?
- Data Exploration
- Data Preprocessing
- Objectives
- Challenges
- Tasks
- ML Pipeline
- References

BACKGROUND

Fraud Detection

- As mentioned in the Kaggle competition by IEEE Computational Intelligence Society for Fraud Detection, there is an increased amount of persistent fraud transactions happening all around the globe and companies are taking measures to minimize the attempts

- While detecting, there are times when the customer wants to do a purchase but that's detected as fraud and results in unwanted situation for the customer

- IEEE-CIS wants to improve the accuracy of fraud detection which would eventually result in customer satisfaction

Why Addressing This Problem Is Important?

- Problem - Can you detect fraud from customer transactions?
- In 2022, PricewaterhouseCoopers reported that fraud has impacted 46% of all businesses in the world.
- The shift from working in person to working from home has brought increased access to data. According to a FTC (Federal Trade Commission) study from 2022, customers reported fraud of approximately $5.8 billion in 2021, an increase of 70% from the year before.
- The majority of these scams were imposter scams and online shopping frauds.
- Detecting these frauds accurately would result in reduction in losses incurred by the company and increase customer satisfaction.

OBJECTIVES

- To predict the probability of a transaction being fraudulent.
- To perform data cleaning and preprocessing on the dataset provided by Vesta's real-world e-commerce transactions after completing necessary data exploration and analysis.
- To choose among different machine learning models for their efficiency and compatibility for working on a large dataset.
- To analyze different machine learning solutions and approaches, compare them and conclude which effectively predicts a fraud.
- To test the performance of our machine learning models on unseen data.

CHALLENGES

- **Changing fraud patterns over time –** Fraudsters are always looking to find innovative ways to fool system and people. Hence, it becomes important for the models to be updated with upcoming patterns.

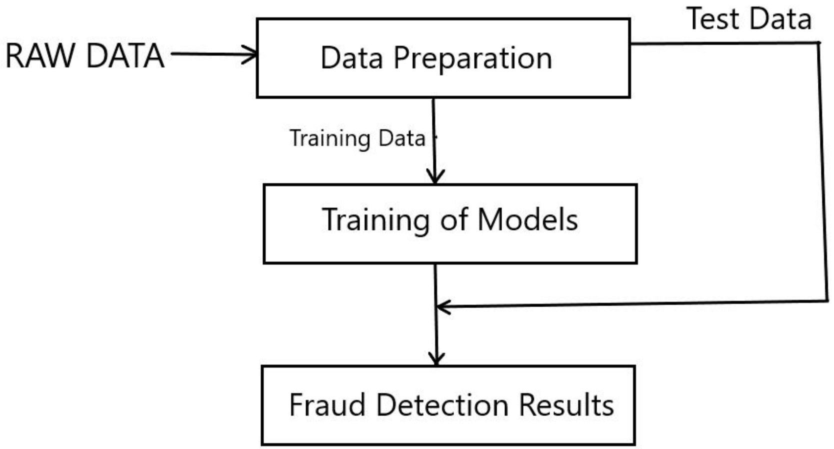
- **Class Imbalance –** As there are very few customers having fraudulent intentions, hence there is an imbalance in the prediction of fraud detection.

- **Feature generation can be time consuming –** Feature generation for this problem can be time-consuming which in-turn slows down the fraud detection process.

TASKS

- Data Collection – Data is collected and provided by **Vesta**.
- Data Preparation – Data will be cleaned and preprocessed for missing and null values along with feature creation, extraction and selection.
- ML Modeling – Appropriate ML models will be taken as a base and will be improved upon prediction accuracies.
- Evaluation – We will utilizing various evaluation metrics and checks to test our models.

PIPELINE


RAW DATA
Data Preparation
Test Data
Training Data
Training of Models
Fraud Detection Results

DATA EXPLORATION

- Understanding the data : We need to have a strong understanding of the data beforehand.
- Data visualization using graphs
- Extracting important variables : Feature selection based on the importance of predictor variables
- Understanding relationships : We not only need to find out the relation between the variables but also the lack of relationship between them.
- Null Values Analysis : We will find out how many columns have a huge percentage of null or missing values.
- Outlier Detection : We need to find data which differs strongly from the rest of the dataset.
- Finding correlation among features and target variable.

DATA PREPROCESSING

- Data Reduction : The data is highly dimensional, so we will reduce the dimensions of the columns by first recognizing the important columns and eliminating the less useful columns.
  a. Remove features that have more missing data.
  b. Remove too-skewed features.
  c. Highly correlated features : If the features are strongly correlated, we can use one of the features instead of using multiple features saving computation resources.
- One Hot encoding : We will convert the categorical variables and perform binarization on them. We might even drop some of the categorical data used for one hot encoding.
- Data Imputation : We will fill the columns which have some missing data by using mean, median, most frequent or constant values.
- Feature Engineering : We will also be creating some new features by combining two or more columns.

CHECK POINT 1

OUTLINE

- Data Exploration and Visualization
- Data Preprocessing
- Preliminary ML Pipeline / system
- Experimental Results
- Future Plans
- References

Data Exploration and Visualization - 1

- Data is collected and provided by **Vesta**
- Considering the raw data set provided, we have two types of files, **identity** and **transaction** which are connected by TransactionID column
- Both of the files are further divided into **train** and **test** data
- In total we have 4 files, test_identity, test_transaction, train_identity and train_transaction
- Transaction data includes features like, TransactionID, Transaction Amount, card details, email domain used, etc
- Identity data includes TransactionID, device details, etc
- All transaction does not have corresponding identity information
- Possibly due to security reasons, we are not provided with the exact column information

Data Exploration and Visualization - 2 [Transaction Dataset]

- **TransactionID** — Id of the transaction and is the foreign key in the Identity Dataset
- **isFraud (Target variable)** — 0 or 1 signifying whether a transaction is fraudulent or not
- **TransactionDT** — timedelta from a given reference datetime (not an actual timestamp)
- **TransactionAMT** — Transaction Payment Amount in USD
- ProductCD — Product Code
- **card1 — card6** — Payment Card information, such as card type, card category, issue bank, country, etc
- **addr** — Address
- **C1-C14** — counting, such as how many addresses are found to be associated with the payment card, etc. The actual meaning is masked
- **D1-D15** — timedelta, such as days between previous transactions, etc
- **M1-M9** — match, such as names on card and address, etc
- **V1-V339** — Vesta engineered rich features, including ranking, counting, and other entity relations

Data Exploration and Visualization - 3 [Identity Dataset]

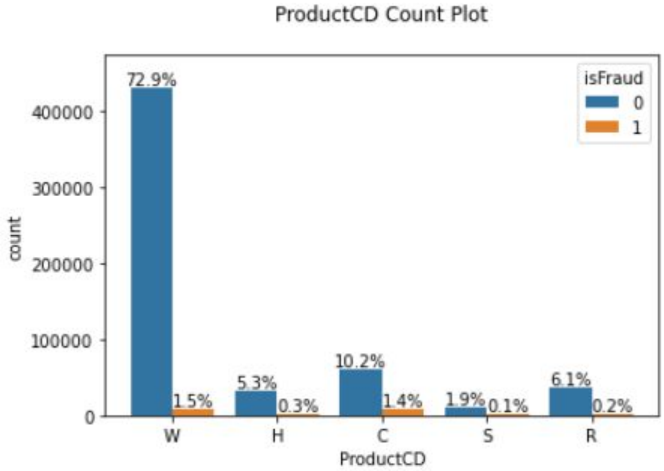
- **TransactionID —** Foreign key to the Transaction Dataset
- **id_01-id_38** — Masked features corresponding to the identity of the card holders
- **DeviceType —** Type of Device used to make the Transaction
- **DeviceInfo —** Information regarding the characteristics of the Device

Data Exploration and Visualization - 3 [datasets]

- train_transaction dataset has 590540 rows/data points and 394 columns/features
- train_identity dataset has 144233 rows/data points and 41 columns/features
- test_transaction has a total of 506691 rows/data points and 393 columns
- test_identity had a total of 141907 data points and 41 rows
- We have merged the datasets train_transaction and train_identity as train_dataset, and test_transaction and test_identity as test_dataset
- test_dataset and train_dataset had a mismatch in the name of id features. So, we changed the format of id features in the test_dataset from id-x to id_x

Data Exploration and Visualization - 4 [ProductCode]

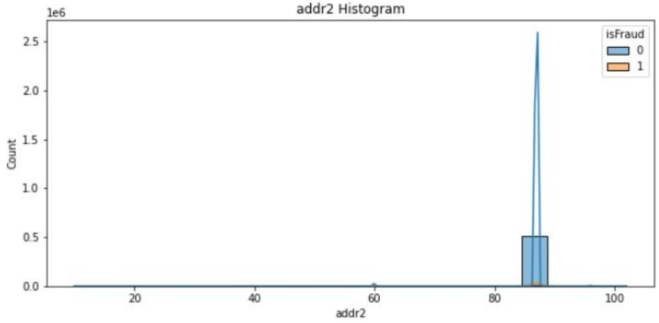
- Most of the transactions have ProductCD = ‘W’
- Second most frequent value for ProductCD is ‘C’ but the difference in frequency as compared to ‘W’ is huge
- Number of fraudulent transactions for ProductCD = ‘W’ is comparable to ProductCD = ‘C’
- This concludes that transactions done for ProductCD = ‘C’ had the highest chance for being fraudulent as compared to other ProductCD categories.



ProductCD Count Plot
isFraud
0
1
72.9%
1.5%
5.3%
0.3%
10.2%
1.4%
1.9%
0.1%
6.1%
0.2%
400000
300000
200000
100000
0
count
W
H
C
S
R
ProductCD

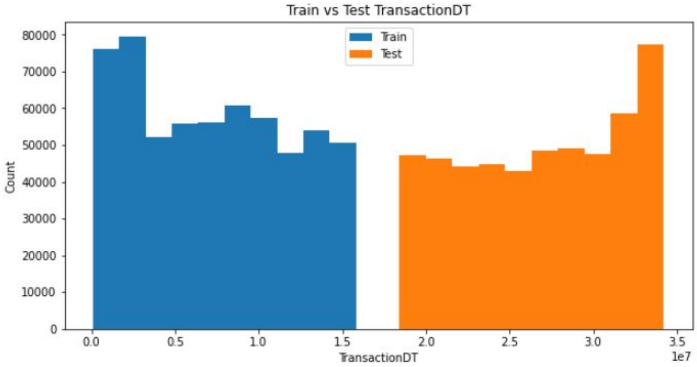
Data Exploration and Visualization - 5

- card4 and card6 features corresponded to the card company and the card type respectively
- Almost 99% of the transactions had the same value for addr2 feature which helped in concluding that this feature corresponds to the Country Code and most of the transactions belong to the same country



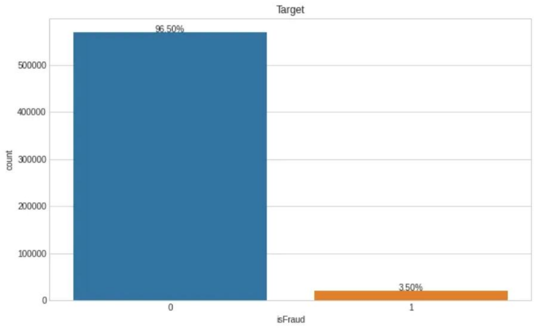
Data Exploration and Visualization - 6

- TransactionDT feature had the minimum value of 86400. It was concluded that this feature is actually the number of seconds elapsed, since, 86400 = 24*60*60.
- Train and Test Split was done based on time and there was some time gap between the train and test set.



Data Exploration and Visualization - 7

- Target Feature ‘Is_Fraud’ marks 3.5% of total transactions as fraud. Graph -



Data Preprocessing - 1 [Missing Values and UID]

- As the data is huge, we declared the data types for each feature to reduce the memory usage
- **Missing values** - In training dataset, 414 columns had missing values where 12 columns have missing values > 90%
- Similarly, for test dataset, 385 columns are having missing values where 10 columns have missing values > 90%
- Ultimately, we decided to drop the features where missing values are > 99%
- Features removed - id_21, id_22, id_23, id_24, id_25, id_26, id_27, id_01,id_07,id_08
- **UID** - Finding the UID is key to improving our score. After analysing, we found that card1, D1, and addr1 will help us to identify the client. So, these were combined to be used as UID

Data Preprocessing - 2 [Removing unnecessary features]

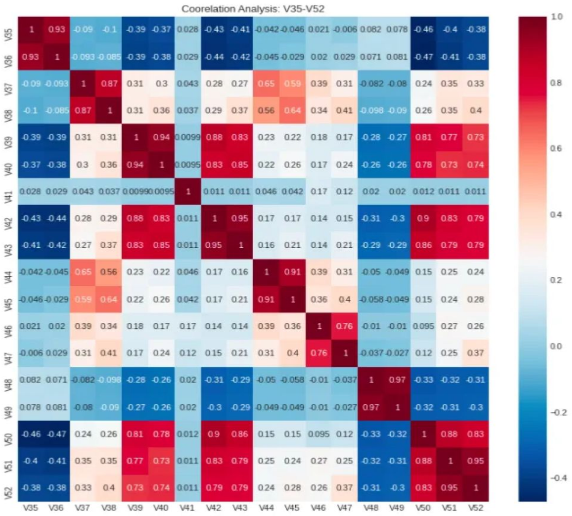
- In the training data, there are cases where transaction amount is greater than $30,000.
- This can be considered as outlier and hence we will be removing such transactions from training set
- TransactionDT and TransactionID columns are having unique values and having these in our model won’t make much sense. So, we will be removing these features.

Data Preprocessing - 3 [Feature selection based on correlation matrix]

- **V-columns -** There are 339 V-columns in the transaction dataset having strong correlation. So, we reduced the number of columns based on NaN values by doing correlation analysis.
- For the cases where different V-columns had similar number of missing values, we considered them to be in the same group.
- For each column in a group, we found the correlation with other columns and took only columns with a correlation coefficient > 0.75.
- In the end, we are left with 128 V-columns, removing 211 columns

Data Preprocessing - 4

- Correlation Matrix -



	V35	V36	V37	V38	V39	V40	V41	V42	V43	V44	V45	V46	V47	V48	V49	V50	V51	V52
V35	1	0.93	-0.09	-0.1	-0.39	-0.37	0.028	-0.43	-0.41	-0.042	-0.046	0.021	-0.006	0.082	0.078	-0.46	-0.4	-0.38
V36	0.93	1	-0.093	-0.085	-0.39	-0.38	0.029	-0.44	-0.42	-0.045	-0.029	0.02	0.029	0.071	0.081	-0.47	-0.41	-0.38
V37	-0.09	-0.093	1	0.87	0.31	0.3	0.043	0.28	0.27	0.65	0.59	0.39	0.31	-0.082	-0.08	0.24	0.35	0.33
V38	-0.1	-0.085	0.87	1	0.31	0.36	0.037	0.29	0.37	0.56	0.64	0.34	0.41	-0.098	-0.09	0.26	0.35	0.4
V39	-0.39	-0.39	0.31	0.31	1	0.94	0.0099	0.88	0.83	0.23	0.22	0.18	0.17	-0.28	-0.27	0.81	0.77	0.73
V40	-0.37	-0.38	0.3	0.36	0.94	1	0.0095	0.83	0.85	0.22	0.26	0.17	0.24	-0.26	-0.26	0.78	0.73	0.74
V41	0.028	0.029	0.043	0.037	0.0099	0.0095	1	0.011	0.011	0.046	0.042	0.17	0.12	0.02	0.02	0.012	0.011	0.011
V42	-0.43	-0.44	0.28	0.29	0.88	0.83	0.011	1	0.95	0.17	0.17	0.14	0.15	-0.31	-0.3	0.9	0.83	0.79
V43	-0.41	-0.42	0.27	0.37	0.83	0.85	0.011	0.95	1	0.16	0.21	0.14	0.21	-0.29	-0.29	0.86	0.79	0.79
V44	-0.042	-0.045	0.65	0.56	0.23	0.22	0.046	0.17	0.16	1	0.91	0.39	0.31	-0.05	-0.049	0.15	0.25	0.24
V45	-0.046	-0.029	0.59	0.64	0.22	0.26	0.042	0.17	0.21	0.91	1	0.36	0.4	-0.058	-0.049	0.15	0.24	0.28
V46	0.021	0.02	0.39	0.34	0.18	0.17	0.17	0.14	0.14	0.39	0.36	1	0.76	-0.01	-0.01	0.095	0.27	0.26
V47	-0.006	0.029	0.31	0.41	0.17	0.24	0.12	0.15	0.21	0.31	0.4	0.76	1	-0.037	-0.027	0.12	0.25	0.37
V48	0.082	0.071	-0.082	-0.098	-0.28	-0.26	0.02	-0.31	-0.29	-0.05	-0.058	-0.01	-0.037	1	0.97	-0.33	-0.32	-0.31
V49	0.078	0.081	-0.08	-0.09	-0.27	-0.26	0.02	-0.3	-0.29	-0.049	-0.049	-0.01	-0.027	0.97	1	-0.32	-0.31	-0.3
V50	-0.46	-0.47	0.24	0.26	0.81	0.78	0.012	0.9	0.86	0.15	0.15	0.095	0.12	-0.33	-0.32	1	0.88	0.83
V51	-0.4	-0.41	0.35	0.35	0.77	0.73	0.011	0.83	0.79	0.25	0.24	0.27	0.25	-0.32	-0.31	0.88	1	0.95
V52	-0.38	-0.38	0.33	0.4	0.73	0.74	0.011	0.79	0.79	0.24	0.28	0.26	0.37	-0.31	-0.3	0.83	0.95	1

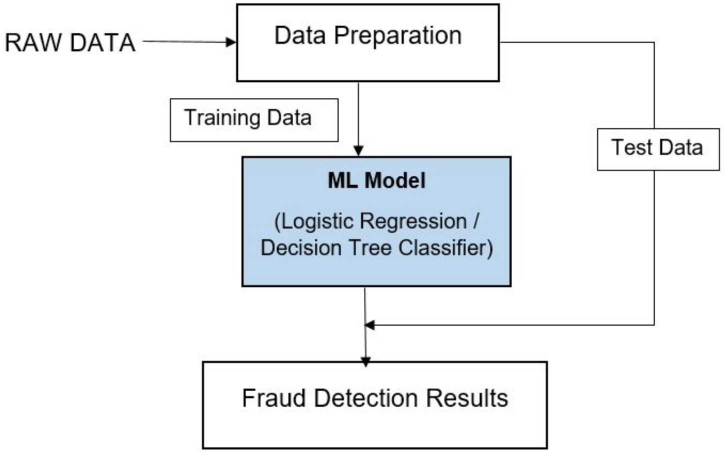
correlation plot

Data Preprocessing - 5 [Encoding features]

We created following features -

- **day of the week:** Day of the week in which transaction occurred
- **hour:** Hour in which transaction occurred
- **cent:** cents associated with the transaction amount
- **LogTransactionAmt:** Log of the transaction amount
- **P_email_company:** company to which P_email_domain belongs.
- **Device_corp:** This is basically parent company details obtained from DeviceInfo columns from identity dataset.
- **Hour fraud status:** We created 4 categories as very low, low, medium, and high so that each hour falls on any of these categories based on the fraudulent transactions in each hour.

Preliminary ML Model/Pipeline



Proposed Approach

Preliminary model consists of both tree and non-tree based models i.e. Decision Tree and Logistic Regression Classifier respectively.

Trained both the models on given dataset to see which one of the model is working best for the data and the one which performed best at this initial stage will be chosen to be used in later stages.

To work with these classifiers, dataset has been encoded as per the needs of model

Data Preparation based on the model :

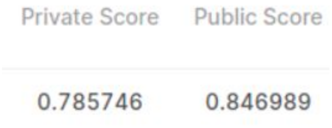
- Stored the “isFraud” column of train_dataset as y_train.
- Dropped “TransactionID” column from both the datasets since it was only a unique identifier for each of the transactions and was of no use in deciding the transaction Status.
- Dropped the “isFraud” column from the train_dataset.
- Dropped the “TransactionID” column of test_dataset.
- Stored the modified train_dataset and test_datasets as X_train and X_test respectively.
- Imputed all the non-numeric categorical missing values of the X_train and X_test with “missing”.
- Label Encoded the Non-Numeric Features of X_train.
- Label Encoded the Non-Numeric Features of the X_test using the X_train.

Experimental Results

- Each classifier is trained and evaluated using the pre-processed datasets and results obtained are as follows:

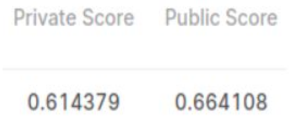
  For Logistic Regression model, 78.57% accuracy is achieved whereas the Decision Tree Classifier was able to achieve just 61.43%.

Logistic Regression



Private Score	Public Score
0.785746	0.846989

Decision Tree



Private Score	Public Score
0.614379	0.664108

Potential issues :

- The time and memory utilization can be reduced with further optimization.
- Lots of features in training data to be handled.
- Missing values have been handled by filling them with zeroes, they can be handled better using other methods.
- Many features that don’t contribute to predicting the fraud are identified and dropped.

Future plans:

- Improvements in data preprocessing by introducing more feature engineering and encoding techniques.
- Improvements in ML modeling techniques by using advanced ML algorithms to increase accuracy.
- Handling the data quality issues with improved data cleaning methods.
- Improving memory utilization and runtime.

CHECK POINT 2

Outline:

- Exploring improved methods
- Comparing results of ML models to propose final model

  Proposed ML model - adaBoosting and random forest
- Comparison of random forest model with other ML models
- Enhanced performance / results
- Future plans

Improved Methods (1):

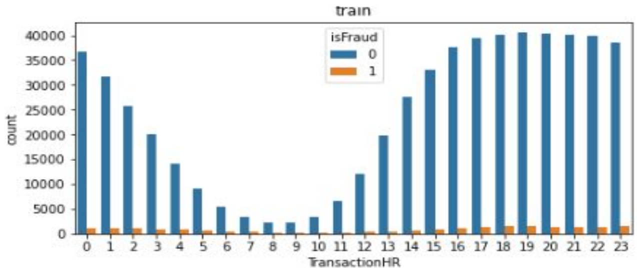
1. Feature Engineering
2. Hyperparameter Tuning

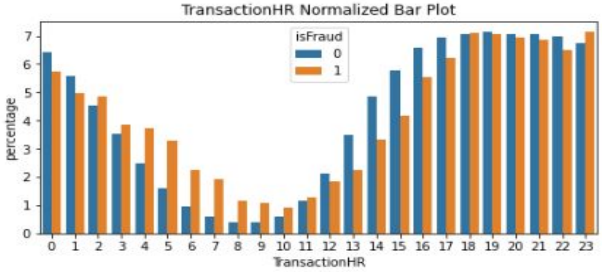
Improved Methods (2):

The following new features were created as part of the Feature Engineering —

- **Transaction Minute**
- **Transaction Hour —**
  - This feature was most interesting feature and we found that relationship between the hour with total number of transactions and the total number of fraudulent transactions is exactly inverse.
  - The larger the number of transactions in an hour the lesser is the fraud percentage of that hour and the lesser the total number of transactions the higher is the fraud percentage.

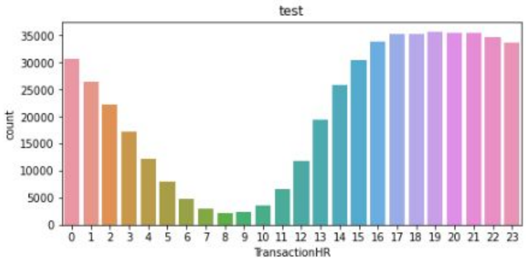
Improved Methods (3):





Improved Methods (4):

Cyclic Nature of Transaction Hour and Transaction Minute feature — The cyclic behavior of the time and hour features was incorporated as follows



Improved Methods (5) - Hyperparameter Tuning:

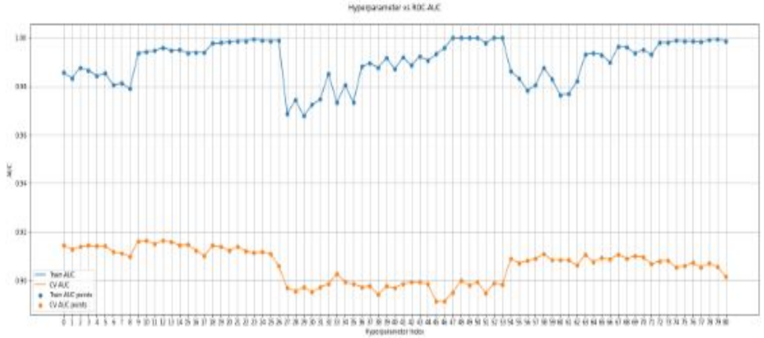
- We performed hyperparameter tuning for the baseline models to understand the improvement in the score.

```
# Hyperparameters

learning_rate = [2e-2, 3e-1, 1e-1]
max_depth = [8, 12, 16]
subsample = [0.6,0.8,1]
colsample_bytree = [0.6,0.8,1]
```

Improved Methods (6) - Hyperparameter Tuning:

- The results after performing Hyperparameter tuning -



- From the plot, the best Hyperparameter index is 12

ML Model: AdaBoost

- In Adaptive Boosting, the weights are re-assigned to each instance, with higher weights assigned to incorrectly classified instances.
- Adaboost is less prone to overfitting as the input parameters are not jointly optimized, so we decided to experiment through
- With the help of adaBoost, we were able to achieve 86% accuracy as compared to 62% of decision tree

ML Model: Random Forest

- Random Forest has strong abilities for handling the missing data, outliers, and categorical variables.
- Easily interpretable.
- It performs well with huge data.
- To reduce overfitting and to improve the accuracy of the model we implemented Random Forest classification algorithm.

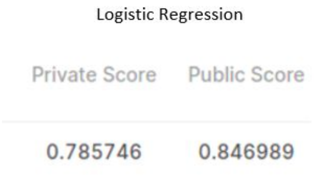
ML Model: Random Forest (2)

Hyperparameters used:

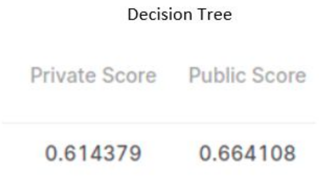
- class_weight='balanced'
- n_jobs=-1
- random_state=3

Experimental Results:

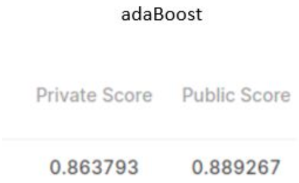
- We experimented with four ML models,


Logistic Regression

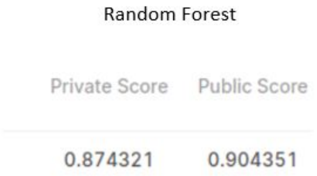
Private Score	Public Score
0.785746	0.846989


Decision Tree

Private Score	Public Score
0.614379	0.664108


adaBoost

Private Score	Public Score
0.863793	0.889267


Random Forest

Private Score	Public Score
0.874321	0.904351

Potential issues of Random forest:

- Overfitting - Random Forest may overfit when the number of trees in the forest is too high or when the individual trees are too deep
- Training Time - Random Forest can be computationally expensive, especially when dealing with large datasets
- Hyperparameters - Hyperparameter tuning of Random Forest (number of trees, the maximum depth of each tree, and the number of features to consider at each split.) is time-intensive
- Memory Utilization - Random Forest require more memory as they consist multiple decision trees

Future Plans (Considering Random forest):

- To overcome the issues of Random forest we can use XGboost which is a gradient boosting method.
- XGBoost has better accuracy and lower computational complexity compared to Random Forest.
- XGBoost is better at handling missing values, outliers, and imbalance data.
- XGBoost has an efficient regularization technique, which helps to prevent overfitting and improves generalization.

REFERENCES

- https://www.kaggle.com/competitions/ieee-fraud-detection/overview/timeline

Thank You!