STAT- 432

Basics of Statistical Learning Project Report

Fraudulent Customer Transaction Detection

Detecting if a customer transaction is fraudulent or not?

Team Member:

Swaraj Thakre (sthakre2)

Jayant Malhotra (jayantm2)

University of Illinois Urbana-Champaign

Department of Statistics

Table of Content

1. Project Description

As countries across the world move toward digital currency and cashless societies, it's important to be aware of the risks that technology can pose for one's finances. With the prevalence of eCommerce and online payments, anyone can be a target of fraud and identity theft. According to a report from PwC, fraud losses totaled US$42 billion in 2020, affecting 47% of all companies in the past 24 months.

In this project, we have used different machine-learning models on a challenging large-scale dataset. The data comes from Vesta's real-world e-commerce transactions and contains a wide range of features from device type to product features. Founded in 1995, Vesta pioneered the process of fully guaranteed card-not-present (CNP) payment transactions for the telecommunications industry. Since then, Vesta has firmly expanded data science and machine learning capabilities across the globe and solidified its position as the leader in guaranteed eCommerce payments.

On the other end, IEEE-CIS works across a variety of AI and machine learning areas, including deep neural networks, fuzzy systems, evolutionary computation, and swarm intelligence. They're partnering with the world's leading payment service company, Vesta Corporation, seeking the best solutions for the fraud prevention industry.

1 Goal

In this project, we are predicting if an online transaction is fraudulent or not, as denoted by the binary target *isFraud*. This will help improve the efficacy of fraudulent transaction alerts for millions of people around the world, thereby helping hundreds of thousands of businesses to reduce their fraud loss and increase their revenue. It will also save people from the hassle of false positives.

2 Approach

This work uses the dataset provided by Vesta Corporation through IEEE-CIS Fraud Detection Prediction Challenge on Kaggle. The complete dataset was provided in 2 separate tables named *Transaction* table and *Identity* table. These tables were joined to create a final training and testing dataset having a total of around 1 million records and 434 input features. Several data pre-processing steps like missing value imputation, dimensionality reduction, categorical variable encodings, additional feature creation, etc. were implemented to create a processed dataset that would help us learn a high-performing unbiased machine learning classifier. Several supervised machine learning models (Logistic Regression, KNN, Decision Tree, Random Forest, XGBoost) were chosen as a candidate to help predict whether transactions will be fraudulent or not.

3 Conclusion

Overall, among all the supervised machine learning classifiers XGBoost and Random Forest had the best ROC AUC on the test set. Featuring engineering helped in reducing overfitting and noise, thus improving the prediction and also reducing computation time.

2. Literature Review

In this section, we have discussed related works on IEEE-CIS Fraud Detection Prediction Challenge on Kaggle.

In (LACHAKE, 2022)[1], the authors used a tree-based supervised machine learning algorithm. The author used a Random Forest classifier to get a simple prediction model based on the training data (80%) and then fine-tuned it to suit the validation dataset (20%). The author dropped all the columns with more than 20% Nan values, resulting in 180 variables from 434. Then imputation of missing data was done using mean imputation for numerical data and mode imputation for categorical data. To deal with class imbalance, SMOTE (Synthetic Minority Over-sampling Technique) was used. Finally, the model was applied to test data given in the competition and a public score of 0.909376 was obtained.

In (thej ravichandran, 2020)[2], the authors instead of dropping columns with NaN values, filled them with an artificial number. After that, the author reduced the 339 Vesta-engineered features into 139 using a group by on the number of NaN values and then binned the correlated features together and took the one with the highest number of unique values. All the categorical variables were label encoded. The author fit the data with Random forest, XGBoost, Catboost, and LGBM. After evaluating each model with 5 Fold CV he chose XGBoost to further hyperparameter tune the model. Finally, the author tried to create 5 different UID (Unique Identifiers) to identify the fraudulent transactions and after testing each one of them he achieved a public score of 0.9543.

3. Data Description

The data comes from real-world e-commerce transactions from Vesta, a leading payment service company, and contains a wide range of features from device type to product features. The data is broken into two files *identity* and *transaction*, which are joined by TransactionID. In total, we have 434 features, out of which 48 are categorical features.

The different features present in the Transaction Table are:

- TransactionDT: timedelta from a given reference datetime (not an actual timestamp).
- TransactionAMT: transaction payment amount in USD.
- ProductCD: product code, the product for each transaction.
- card1 - card6: payment card information, such as card type, card category, issuing bank, country, etc.
- addr(1 and 2): address associated with card.
- dist: distance.
- P_ and (R__) email domain: purchaser and recipient email domain.
- 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.
- M1-M9: match not match, such as names on card and address.
- Vxxx: Vesta engineered rich features, including ranking, counting, and other entity relations.

The different Categorical Features present in the Transaction Table are:

- ProductCD
- card1 - card6
- addr1, addr2
- P_emaildomain
- R_emaildomain
- M1-M9

Variables in the Identity Table are:

- Identity information such as network connection information (IP, ISP, Proxy, etc) and digital signature (UA/browser/os/version, etc) associated with transactions.

The different Categorical Features present in the identity table are:

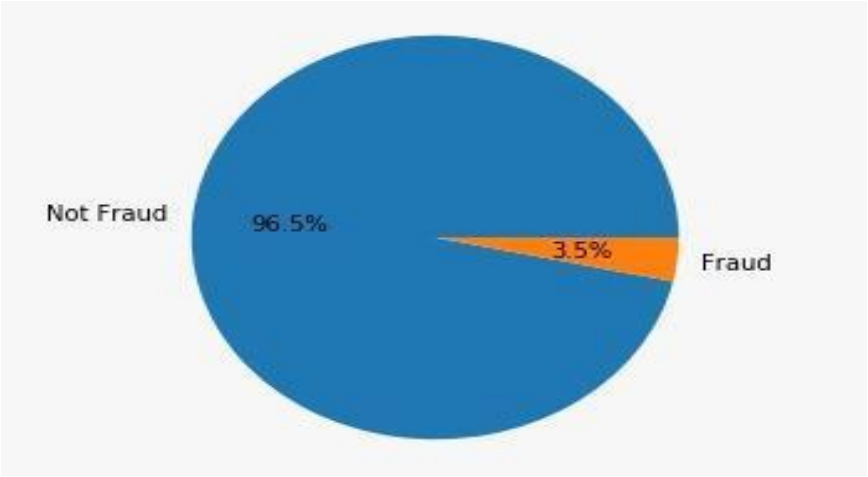
- DeviceType: type of device that was used for making transactions.
- DeviceInfo: specific information pertaining to each device.
- id_12 - id_38: numerical features for identity, which is collected by Vesta and security partners such as device rating, ip_domain rating, proxy rating, etc. Also, it recorded behavioral fingerprints like account login times/failed to login times, how long an account stayed on the page, etc.

Outcome Column: "isFraud":

The logic to define fraud transaction (isFraud=1) is whether transactions posterior to it with either user account, email address or billing address directly linked to these attributes were fraud too. If none of the above is reported as fraud beyond 120 days, then we define it as a legit transaction (isFraud=0).

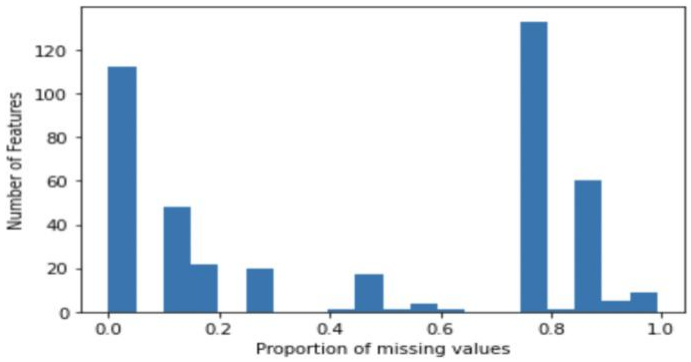
Imbalance in Data:

From the pie chart we can clearly see that the dataset is highly imbalanced as 96.5% of the response variable corresponds to 'NotFraud' and only 3.5% of the response variable corresponds to 'Fraud'.

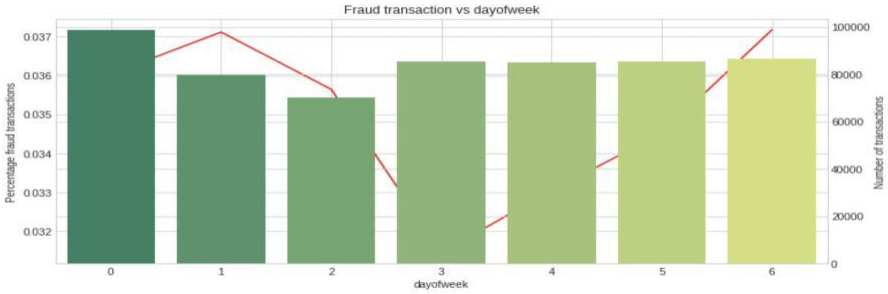


4. Key Insights

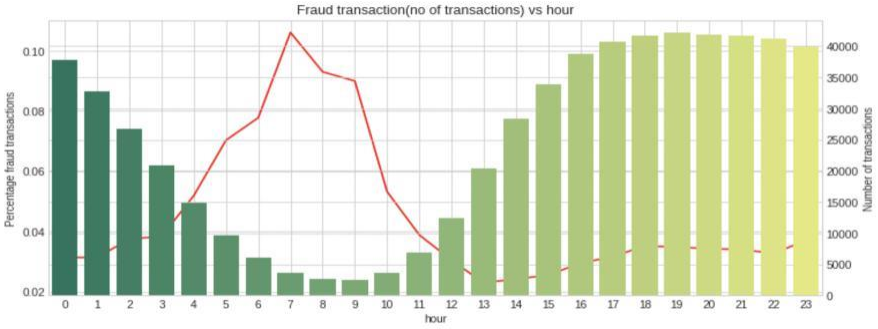
- High Sparsity in data. More than 99% of null values in 10 columns. Over 47% of features have more than 70% missing values.



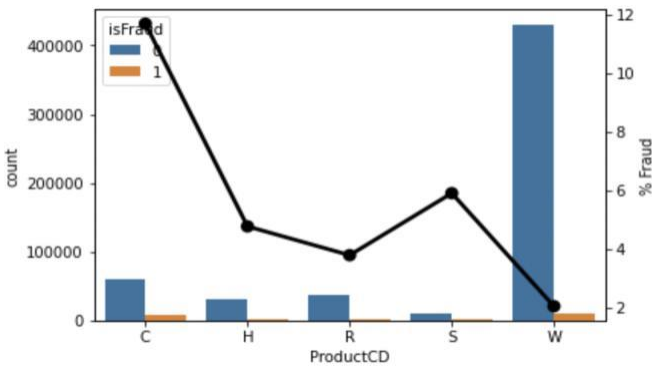
- Number of transactions decreases initially and then remains constant throughout the week, but one thing to focus on is that, the percentage of fraudulent transactions on the 3rd day of the week is very less.



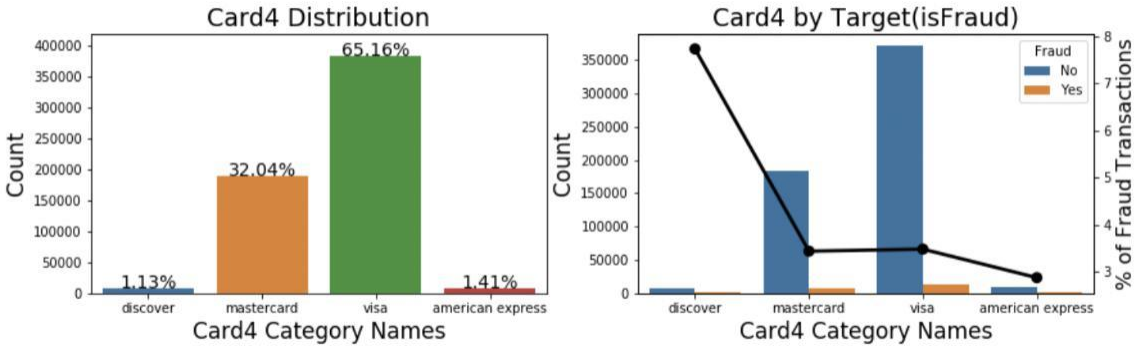
- Maximum transactions take place between 6 PM and 10 PM. However, It is from 4 AM to 10 AM when the Fraudulent Transaction Percentage is higher than usual, peaking at the 7th hour.



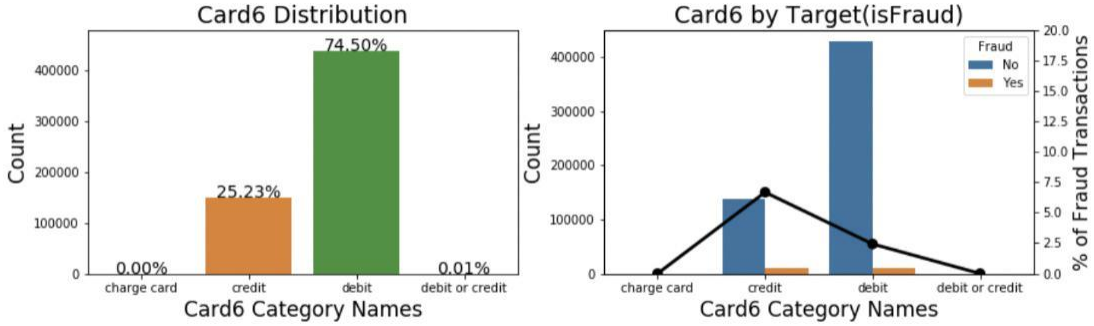
- Product code W despite being involved in the highest number of transactions had the lowest percentage of fraudulent transactions, whereas product code C had 12% of fraudulent transactions.



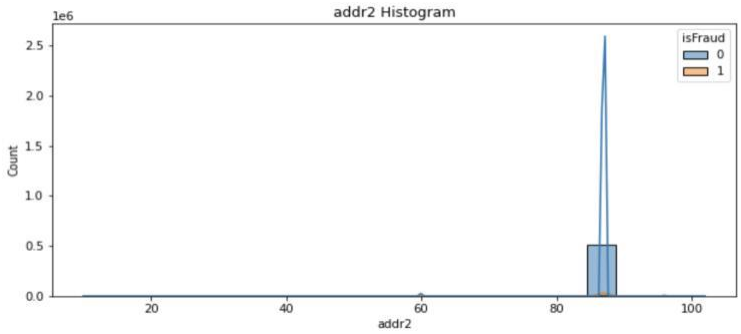
- 97% of our data have either Mastercard (32%) or Visa (65%). Also, we have the highest value of fraud transactions in Discover (~8%) against ~3.5% in Mastercard and Visa and 2.87% in American Express.



- All data is on Credit and Debit Cards. We can see a higher percentage of Fraud in Credit than in Debit transactions.



- Almost 99% of transactions had the same value for addr2 features, from this information we can conclude that this feature corresponds to Country code and most of the transactions belonged to the same country



- About 76% of values are missing in the R-email domain. In purchaser email domains 90% of fraudulent transactions come from the domain protonmail.com which is a serious issue.

5. Data Pre-Processing

1. Feature Creation:

As one can see in Key Insights the day of the week and hour of the day can help in determining whether a transaction is fraudulent or not, so we decided to create the following two features from TransactionDT.

i) Day of the week

ii) Hour of the day

2. Outlier Removal:

On doing Outlier Analysis, 2 abnormal observations from TransactionAMT were removed since they had a transaction amount 200 times more than the average transaction amount.

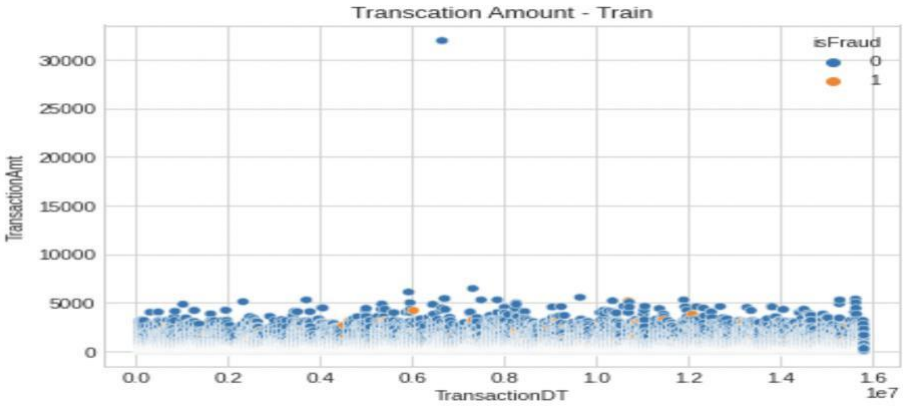


Fig 5.2 Plot of TransactionAmt vs TransactionDT

3. Feature Selection:

As discussed in the data analysis section above, we have high sparsity in our training data. Around 47% of the features have more than 70% missing values. Performing missing value imputations on such columns can thus underestimate the variance of those particular columns, which in turn could adversely affect the model performance. Thus, we decide to drop all columns that have greater than 30% missing values in them.

4. Transformations:

i) Imputation of missing values:

After performing the feature selection, to remove any missingness in the data we performed median value imputation for Numerical columns and performed mode/most frequent value imputation for Categorical columns.

ii) One Hot Encoding for Categorical Columns:

Most of the categorical columns were nominal in nature and they also had low cardinality. Thus we decided to use one hot encoding to convert feature labels to numeric values.

5. Feature Extraction:

Non-parametric models suffer from **the curse of dimensionality** when the number of input covariates is large. Few of the parametric models suffer from **the high variance** problem when the number of input covariates is large. We thus reduced the number of input covariates by performing PCA separately on two sets of features which are C1-C14 (14 features) and Vxxx (339 features) respectively. As evident from the graphs below,

i) For C1-C14 :

4 principal components explain around 99% variance. This helped us reduce the number of features from 14 to 4.

ii) For Vxxx :

70 principal components explain around 98% variance. This helped us reduce the number of features from 339 to 70.

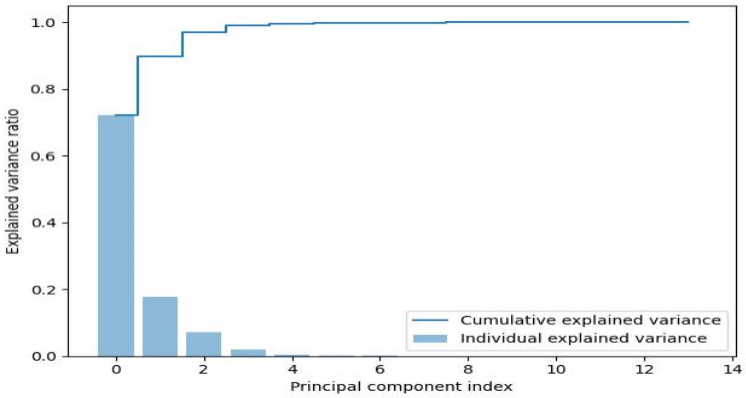


Fig 5.3.1 Cumulative and Individual variance explained by principal components of C1-C14 columns.

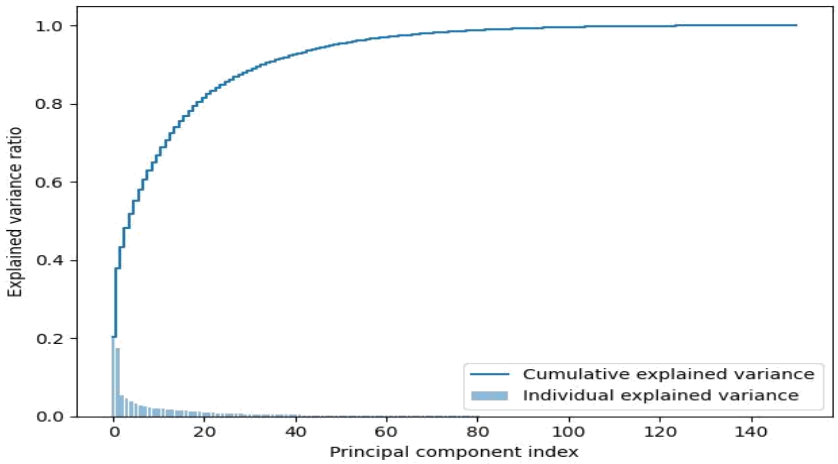


Fig 5.3.2 Cumulative and Individual variance explained by principal components of Vxxxx columns.

6. Model Training and Evaluation

We used 6 machine learning models - Logistic regression with L1 penalty, Logistic regression with L2 penalty, KNN, Decision Tree, Random Forest, and XGBoost for the classification of fraud. The training dataset is used for learning model parameters and hyper-parameter tuning while the test dataset is used for evaluating the model on unseen data. The right combination of parameters and hyperparameters results in a model with high performance.

1. Evaluation metrics and Hyper-parameter tuning

As the training data is highly imbalanced we decided to use the **ROC AUC** (Area under the curve) metric to tune our model using K-fold (with K as 5) cross-validation approach. The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

Further, we also evaluated the **Balanced Accuracy** [3] metric for our best model to get an unbiased estimate of the classification accuracy of our model without having to create a balanced training dataset by oversampling the minority classes or undersampling the majority ones.

The formula for the Balanced Accuracy metric for a binary classification case is given by,

Balanced Accuracy = (Sensitivity + Specificity)/2

2. Machine Learning Models

a. **Logistic Regression:** Logistic Regression is one of the powerful classifiers for linear separable datasets. Although it can be used for multinomial classification, it is extensively used for binary classification problems. We have used Logistic Regression with L1 and L2 penalties for our training purposes.

b. **KNN:** K nearest neighbor (KNN) is a simple non-parametric method. It can be used for both regression and classification problems. Being a non-parametric method, KNN suffers from the curse of the high dimensionality effect. To reduce this effect, we have pre-processed our data and have reduced our feature space from 434 to 159.

c. **Decision Tree:** Decision Tree belongs to the class of supervised learning algorithms. It can be used for both regression and classification problems. It enjoys the benefit of performing automatic feature selection and it can also handle categorical variables directly without us having to encode them in some numerical format. It is however prone to overfitting and it also cannot easily capture additive structures.

d. **Random Forest:** Random Forest is an ensemble technique that reduces model variance while retaining low bias through Boost Strapping Aggregation (Bagging) technique. In our project, Random Forest was able to handle large datasets with high dimensionality and slight correlation, all while not overfitting data with appropriate hyperparameter tuning.

e. **XGBoost:** XGBoost is another ensemble technique that improves a model bias through boosting approach. Here the errors made by previous models are tried to be corrected by succeeding models by adding some weights to the models. XGBoost helped us in reducing training time by providing parallel processing, along with regularizations which helped in reducing overfitting.

3. Model Comparison

Model	ROC AUC	Balanced Accuracy	Accuracy
Logistic Regression + L1 penalty	82.35	74.97	74.34
Logistic Regression + L2 penalty	73.56	67.10	67.81
KNN	84.24	82.94	92.53
Decision Tree	75.08	75.08	96.19
Random Forest	91.16	84.61	87.68
XGBoost	93.16	84.10	90.96

Fig 6.3.1 Table comparing ROC AUC, Balanced Accuracy, and Classification Accuracy on test set for various models.

Model	ROC AUC 159 features	ROC AUC 258 features
Logistic Regression + L1 penalty	82.35	83.05
Logistic Regression + L2 penalty	73.56	60.69
KNN	84.24	69.02
Decision Tree	75.08	79.03
Random Forest	91.16	92.46
XGBoost	93.16	93.52

Fig 6.3.2 Table comparing ROC AUC for various models on our pre-processed test set and partial pre-processed test set.

In fig 6.3.1 we can clearly see that Random Forest and XGBoost models outperform the rest of the models in terms of ROC AUC and Balanced accuracy metrics. We have also included the classification accuracy column in fig 6.3.1 so that we can contrast it with the balanced accuracy metric. Due to the unbalanced nature of the training data, we can clearly see how classification accuracy can be misleading. For e.g. AUC for the Random Forest model is greater than the AUC for the Decision Tree model but the classification accuracy for the Decision Tree comes out to be greater than the classification accuracy for the Random Forest model which is misleading.

In fig 6.3.2 we created a separate partially processed test dataset (having 258 features). This test dataset was obtained by removing columns having greater than 30% missingness, imputing the missing values, and one hot encoding the categorical variables (we didn't perform dimensionality reduction and feature creation). We can clearly see that Random Forest and XGBoost models on both test sets have similar performance owing to the factors that these models can perform automatic feature selection but we see drastic performance reduction in Logistic Regression and KNN models due to factors such as high variance and curse of dimensionality as a result of an increase in the number of input covariates.

7. Conclusion

Overall, among all the supervised machine learning classifiers XGBoost and Random Forest had the best ROC AUC on the test set. Featuring engineering helped in reducing overfitting and noise, thus improving the prediction and also reducing computation time.

Challenges:

We came across a lot of challenges during the project:

1. Large size of the training/testing data made the data loading, data pre-processing, and model fitting steps to be very computationally expensive. To address this issue we implemented a memory reduction trick that downscaled the data type of individual columns (wherever possible) based on the range of values that the column holds.
2. Data was highly sparse in nature. Handling such a huge amount of sparsity without letting the model performance drop was achieved by performing various data pre-processing steps.
3. Due to the lack of computational capabilities only essential hyperparameters were tuned using the user-defined function to find the best parameters one at a time.
4. Data was highly imbalanced. Thus to get an unbiased high-performing classifier we carefully selected metrics like ROC AUC and Balanced Accuracy.

Future Scope:

We want to try different ensembles of machine learning algorithms or neural networks for even better predictions.

8 References

[1]:https://www.kaggle.com/code/pradneshlachake/ieee-cis-fraud-detection-random-forest-classifier

[2]:https://www.kaggle.com/competitions/ieee-fraud-detection/discussion/220874

[3]: https://neptune.ai/blog/balanced-accuracy