IEEE Fraud Detection

Karthik R Udupa
IMT2018510
karthik.r@iiitb.org
IIIT Bangalore

C P Vikram Adithya
IMT2018085
Vikram.Adithya@iiitb.org
IIIT Bangalore

Sairama Sashank Kadiyala
IMT2018064
sairama.sashank@iiitb.org
IIIT Bangalore

ABSTRACT

Given data on transactions and other identity details, we have to determine whether the result pertaining to the target must be Fraud or Not Fraud. (The reason why we are using the word 'target', here, will be mentioned in Introduction section). Train and test dataset, each, contains over 400 columns and has to be reduced first. This has been achieved by us, by performing EDA over the merged train datsets (train_transaction and train_identity) and merged test datasets (test_transaction and test_identity). The next challenge we faced was choosing the best model and method to tackle the high imbalance of the given data. Following this, feature engineering and hyperparameter tuning was performed to further improve our results. We notice that an ensemble of certain GBDTs obtained us the most significant results. Post processing actually reduced our score and we shall address that issue as to why it happened, further in this report, under the experiments section.

I. INTRODUCTION

'Imagine standing at the check-out counter at the grocery store with a long line behind you and the cashier not-so-quietly announces that your card has been declined. In this moment, you probably aren't thinking about the data science that determined your fate.
Embarrassed, and certain you have the funds to cover everything needed for an epic nacho party for 50 of your closest friends, you try your card again. Same result. As you step aside and allow the cashier to tend to the next customer, you receive a text message from your bank. "Press 1 if you really tried to spend $500 on cheddar cheese."
While perhaps cumbersome (and often embarrassing) in the moment, this fraud prevention system is actually saving consumers millions of dollars per year.
So, In this competition you are predicting the probability that an online transaction is fraudulent, as denoted by the binary target isFraud.'

While this overview/explanation given to us seems to be clean and pristine, there was a major clue hidden inside of it. The target we are predicting as Fraud/not Fraud is not the transaction itself, but the card used for the transaction. We ourselves realised it only when we were done with the EDA and halfway through with the project, however, this single discovery gave way for further methods to improve our score. The last line of the overview made us assume the wrong direction, however, we were able to steer ourselves onto the right path by properly understanding what the target represents.

II. DATASET

The data is broken into two files identity and transaction, which are joined by TransactionID. Not all transactions have corresponding identity information.

i. File descriptions

train_transaction, identity.csv - the training set
test_transaction, identity.csv - the test set (you must predict the isFraud value for these observations)
sampleSubmission.csv - a sample submission file in the correct format

ii. Transaction Table

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, issue bank, country, etc.
addr: address.
dist: distance.
P_ and (R_) emaildomain: 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 transaction, etc.
M1-M9: match, such as names on card and address, etc.
Vxxx: Vesta engineered rich features, including ranking, counting, and other entity relations.

Categorical Features:

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

iii. Identity Table

Variables in this table are identity information – network connection information (IP, ISP, Proxy, etc) and digital signature (UA/browser/os/version, etc) associated with transactions.
They're collected by Vesta's fraud protection system and digital security partners.
(The field names are masked and pairwise dictionary will not be provided for privacy protection and contract agreement)

Categorical Features:

- DeviceType
- DeviceInfo
- id_12 - id_38

III. EXPERIMENTS: DATA

The following section contains different techniques used to analyse, modify and process the data.

i. Category Cardinality Reduction

Concept: For a categorical feature, if many of the categories have identical/(very similar) target distribution, then we can group them into a single category, thus reducing unnecessary categories.
For example, in P_emaildomain, there are 19 email domains that have 0 fraud rate such as yahoo.de, netzero.com etc. We have put all these seperate categories into a single category called "trusted domains".

We have applied this technique to 3 features namely R_emaildomain, P_emaildomain and DeviceInfo.

TABLE I
UNIQUE FEATURES BEFORE AND AFTER REDUCTION

Feature	Before Reduction	After Reduction
P_emaildomain	60	42
R_emaildomain	61	31
DeviceInfo	1516	237

ii. Minority Class Oversampling

Given dataset is highly imbalanced. The minority class (isFraud = 1) accounts for 3.6 % of the data. Given this condition it was an intuitive decision to use the imblearn package (imblearn link).
There are 3 main techniques when using imblearn.

- Random Undersampling
- Random Oversampling
- SMOTE

Random Oversampler duplicates the minority class to get the required ratio. This leads to overfitting in cases where the actual percentage of minority class is very less.
SMOTE stands for Synthetic Minority Oversampling Technique.
SMOTE uses interpolation techniques like K-Nearest Neighbors to upsample the minority class. The effect of SMOTE on the data is mentioned in the Model section.

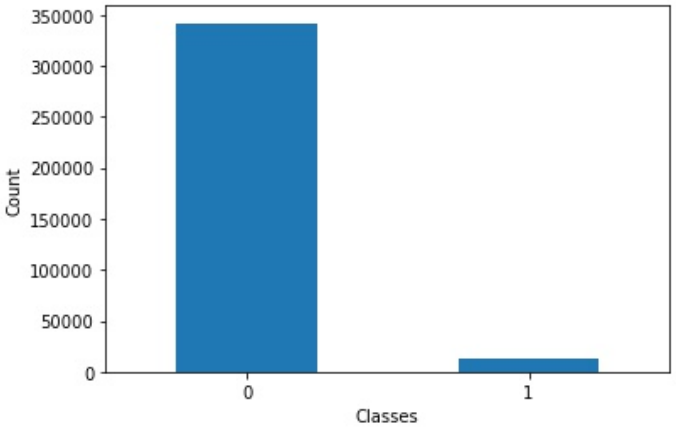

Fig. 1. Class imbalance in the dataset

iii. User Identification

The problem statement can be viewed from a different angle altogether. In general, if a card has been flagged as Fraud it is likely to remain so in the future. So instead of predicting transactions as fraudulent, we can identify the users(credit cards) and flag them.
With this mechanism, when we get a new transaction, we can check the history of that particular card and predict with improved accuracy.
The following mechanism was used to unique user identification.

$$\mathbf{TransactionDay} = \frac{\mathbf{TransactionDT}}{\mathbf{60*60*24}}$$

$$\mathbf{userid} = (\mathbf{TransactionDay} - \mathbf{D1})\mathbf{\_card1\_addr1}$$

More robust userid can be built by considering additional features such as addr2.

TABLE II
DISTRIBUTION OF UNIQUE CARDS IN THE DATA

Total no. of transactions in train	354324
No. of unique cards in train	160677
Total no. of transactions in test	236216
No. of unique cards in test	187602
Common cards in train and test	29643
Percentage of cards in test that are present in train(have a history)	25.81 %

The effect of user identity creation on performance is explained in the Post Processing and Evaluation sections.
Please note : The User Identity creation is solely for the purpose post processing and not for training as it will overfit(Explained in the Feature Engineering section).

iv. Principle Component Analysis

A dimensionality-reduction method that is often used to represent large dimensional datasets, by transforming the dataset into a smaller one that still contains most of the information and variance seen in the large dataset.
We plotted the variance captured vs the number of principal

components required by them, to decide on how many principal components we need to reduce the V and id columns to, such that it still retains the important information.

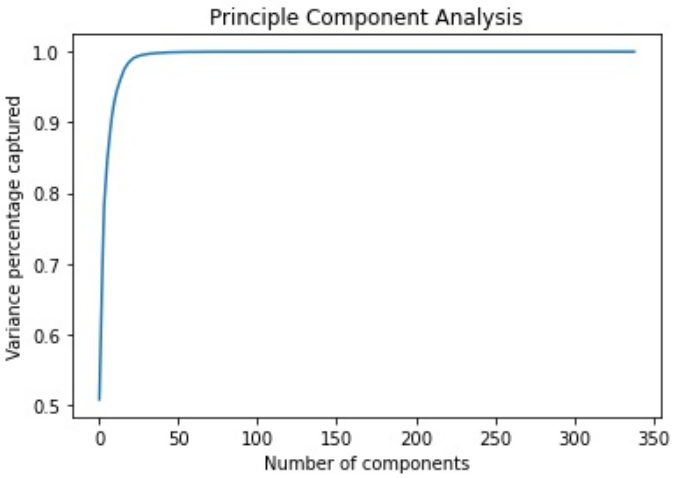

Fig. 2. Principle Component Analysis of V columns

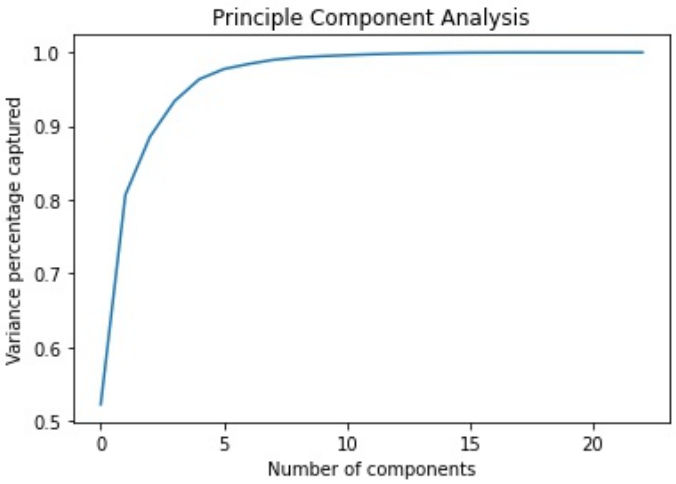

Fig. 3. Principle Component Analysis of ID columns

v. *Feature Engineering*

When model takes userid as a feature while training, it will definitely lead to overfitting, as it might have been explained before that once a card is detected as fraud, it is likely to remain so for a certain extent of time.
Since userid is basically a combination of given features, there is a high chance that a model might overfit on the given data. To prevent this overfitting, we create new features by aggregating features over userid and taking their mean, std and count as needed. We then remove the userid feature.
This ensures that the model is able to learn what makes the userid false rather than just whether or not that userid was false in train.

vi. *Recursive Feature Elimination*

Another method we came up with to further reduce the dimensionality of the data without modifying it by methods such as pca was to apply Recursive Feature Elimination method.
In this method, we recursively fit and eliminate features, in the model, based on their importance and choose the most optimal solution.
However the drawbacks were:

1) Takes too much time for the model to recursively try out each feature.
2) Certain features which act as outliers or important in test data but not in train, are removed, thus might lead to overfitting

In the Cross-validation score vs Number of features selected plot, we can see that training (ROC AUC) score peaked at 299 columns, almost three-fourths of the initial 432.
However the testing score decreased, thus a clear case of overfitting.
(since we passed step=7, the actual number of features is 7 times the one shown in the x-axis of the plot)

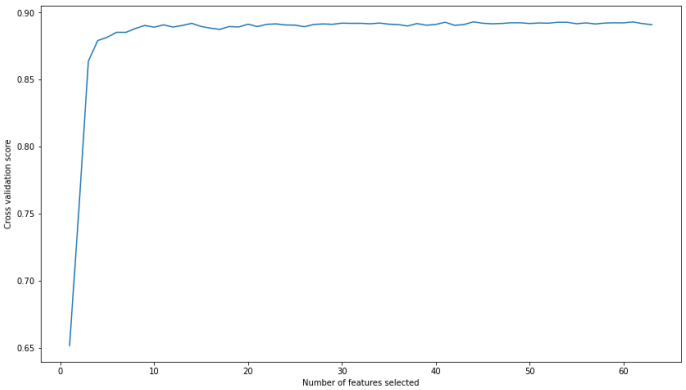

Fig. 4. Recursive Feature Elimination

vii. *Correlation*

Correlation is a statistical measure that describes the association between random variables. In this, Pearson correlation measures the linear association between continuous variables. Hence by analysing the correlation matrix, we can remove the redundant features and reduce the dimensionality of the dataset. Thus we can use this as a dimensionality reduction method.
For example, by removing the highly correlated features among the V columns, the initial hundreds of V columns can be reduced to just 58 in number.

IV. Experiments: Models

The following models were used in this competition in chronological order.

- LogisticRegression
- RandomForest Classifier

- LGBM
- XGBoost
- Catboost

SMOTE technique drastically improves the performance of linear models like logistic regression. But from the table it is clear that it overfits for Tree based models like LGBM.

i. Model Selection and Ensemble Techniques

TABLE III
SCORE COMPARISON

Model	Public Leaderboard Score	Private Leaderboard Score
LGBM	0.90909	0.90812
XGBoost	0.967	0.971
CatBoost	0.78954	0.79694
LGBM*	0.72999	0.72618
Random Forest*	0.84779	0.84917
Logistic Regression*	0.77147	0.77427

* indicates the use of SMOTE for that particular prediction.

ii. Hyperparameter Optimization

Hyperparameter Optimization refers to the process of finding the optimal set of hyperparameters for the learning algorithm. It ensures that the model is tuned to handle the given data in a robust manner.
Bayesian Optimization technique has been used to tune model parameters. We have optimized XGBoost and CatBoost models.
Main concepts of Bayesian Optimization:

- Sequential Mode Based Optimization
  Unlike Random Search CV and Grid Search CV, Bayesian Optimization builds on the previous iterations to reach optimal set of values. This ensures faster convergence rate.
- Objective Function
  The objective function is the function that is maximised based on the learning algorithm that is provided.
  ROC AUC score has been provided as the objective function for tuning.
- The domain of hyperparameter space for the tuning is provided by evaluating performance on a trial and error basis.

*1) **XGBoost Optimization**:* BayesianOptimization Python Library was used for tuning (Github link) The most important parameter that improved the score is "scale_pos_weight" which handles imbalanced data and focuses on the minority class. The optimal parameters for XGBoost model are,

```
{'n_estimators': 1000,
 'colsample_bytree': 0.85,
 'learning_rate': 0.049,
 'max_depth': 9,
 'subsample': 0.84,
 'objective': 'binary',
 'metric': 'auc',
 'is_unbalance': True,
 'tree_method': 'gpu_hist',
 'reg_alpha': 0.15,
 'reg_lambda': 0.85
}
```

*2) **CatBoost Optimization**:* Optuna(put link here) is an automatic hyperparameter optimization software framework, particularly designed for machine learning. We have used it here as it makes it easy for defining search spaces, efficient optimization algorithms and some quick visualization.
The optimal parameters for CatBoost model are:

```
{'learning_rate': 0.25,
'depth': 8,
'bagging_temperature': 10,
'iterations': 478,
'random_strength': 3.6644e-06,
'border_count': 12,
'l2_leaf_reg': 18,
'scale_pos_weight': 0.97}
```

V. EXPERIMENTS: POST PROCESSING

The term indicates an operation performed on the predicted output of a model using techniques that are derived from the training data through feature engineering and analysis. In the context of our problem, Once a card is flagged as fraud, it is likely to remain in the same state.

How does this help?

If we have a mechanism to identify cards present in both test and train, then we can use the train target values along with the model prediction to strengthen the final prediction. This can be done through the unique user/card identification scheme.

Why does it fail in our contest?

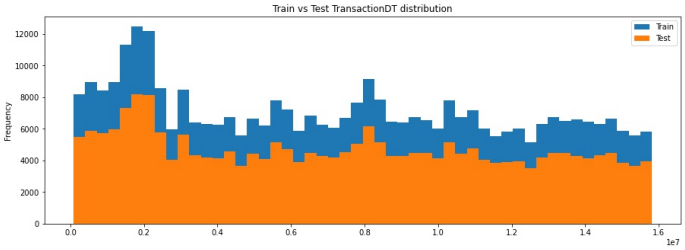

Fig. 5. Train Test Time line

If we look at the test and train distribution across time (in this fig), we observe that both span across the same time frame. Hence, we cannot go by the assumption that the train has occurred in the past and therefore cannot use its fraud values to determine the outcome of test data.
Nevertheless, we can use this concept to first combine the train and test, arrange them in a chronological manner, and then apply this concept.

VI. EVALUATION AND SCORES

The scoring in this competition is Area Under Receiver Operating Characteristic Curve (AUC ROC) curve.

We have seen that XGBoost model was the best model. Following the table containing various scoring metrics of the XGBoost model.
K Fold (K = 5) Validation has been used to understand training scores and to built a robust model.

TABLE IV
MEAN CV EVALUATION SCORES FOR THE BEST MODEL (XGBOOST)

Metrics	Score
AUC ROC	0.9628
F1 Score	0.7893
Precision	0.9000
Recall	0.7029

TABLE V
MEAN CV EVALUATION SCORES FOR THE CATBOOST MODEL

Metrics	Score
AUC ROC	0.8738
F1 Score	0.4563
Precision	0.8360
Recall	0.3140

As mentioned in Post Processing section, processing the prediction probabilities with userID feature reduced our accuracy due to inconsistent timeline between train and test.

The following is the comparision between XGBoost and Catboost confusion matrices.

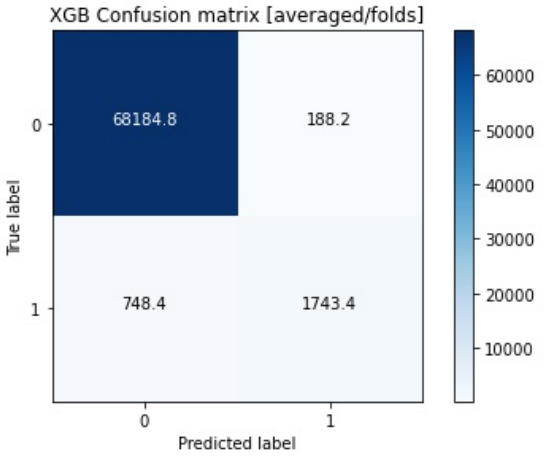

Fig. 6. XGBoost : Confusion Matrix

We observe that there are many false negatives in catboost prediction compared to XGBoost.

i. Tackling Overfitting

There was always the question whether our model was overfitting on the training data.
1. Though this point is trivial, it is important to mention

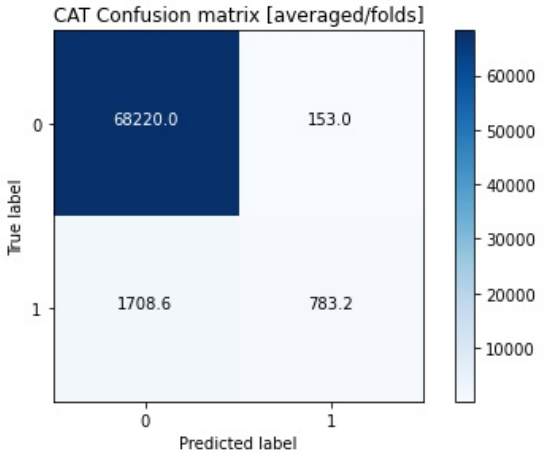

Fig. 7. CATBoost : Confusion Matrix

TABLE VI
SCORE BEFORE AND AFTER INCLUSION OF USERID

Before	0.96496
After	0.95109

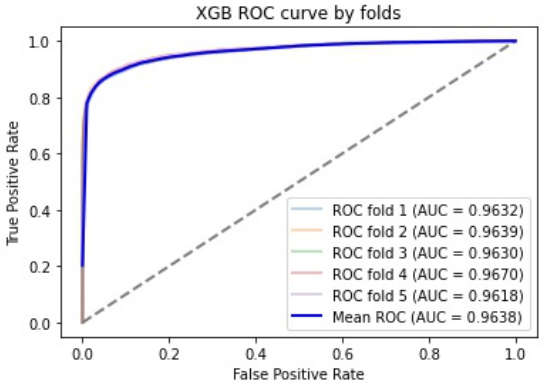

Fig. 8. XGBoost : ROC-AUC Curve

The above plot is the ROC Curve for each of the K folds The score is very similar to the private scores.

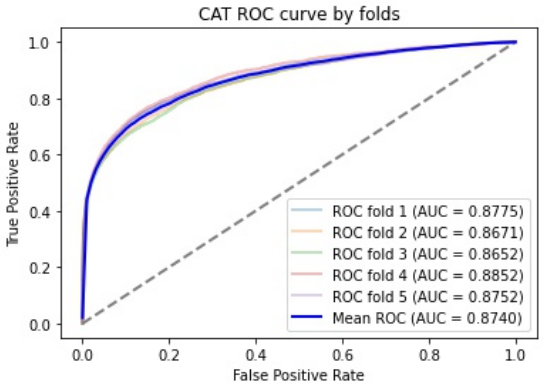

Fig. 9. CATBoost : ROC-AUC Curve

The following plot is the Precision Recall Curve.

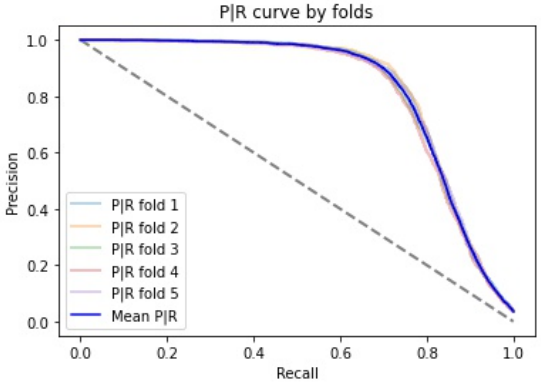

Fig. 10. XGBoost : Precision-Recall Curve

that as a best practise, ID's must not be used when training. We observed that using TransactionID column in training was overfitting in the training data and reducing our public leaderboard score.

Removing TransactionID column improved the score from 0.95876 to 0.96496 in the leader board.

2. In the process of improving our score in leaderboard, when do we stop and realize that overfit has taken place? To answer this question we did extensive feature engineering and understood the tipping point for the model. When the model overfit the data it gave 0.971 score in train and 0.94 in leaderboard, this indicated that overfit had started.

VII. Conclusion

In this competition, we came across different techniques. Out of these, techniques like feature engineering and encoding aided in the score improvement whereas PCA, Recursive indirectly helped prepare the data.

When we look at the feature importance given by the XGBoost model, we understand the importance of analysing the data carefully as features that seem to be unimportant at the first glance, in actuality could be a very important feature for the model to predict the result.

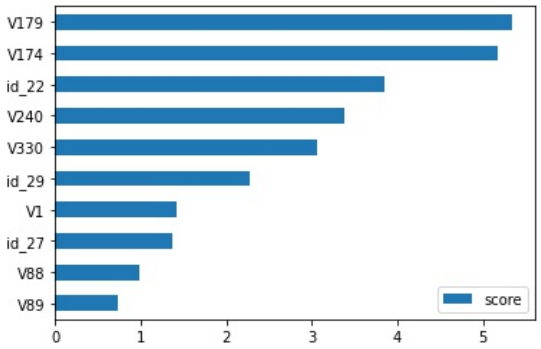

Fig. 11. Feature Importance

VIII. Future Scope

Fraud Detection remains one of the top priority requirement in the society and machine learning/deep learning concepts have proved to solve this in an entirely new data driven manner.

Acknowledgment

This project has taught us various aspects that goes into to preparing a possible solution for a machine learning challenge/problem. Our sincere thanks to Saurabh Jain for the smooth evaluations, to the professors for the detailed lectures and to all the TA's for the wonderful sessions.

References

[1] Jereme Jordan, "Hyperparameter tuning for machine learning models", 2 NOV 2017. link

[2] Will Koehrsen, "Bayesian Hyperparameter Optimization for Machine Learning", 2 JUN 2018. Medium link

[3] Will Koehrsen, "Bayesian Hyperparameter Introduction", 28 JUN 2018. Medium link

[4] Bayesian Optimization Code in Python. Kaggle link

[5] XGBoost Official Documentation (link)

[6] CatBoost Classifier Official Documentation (link)

[7] Optuna Hyperparameter Optimization (link)

[8] Understanding XGBoost Hyperparameters (link)

[9] Understanding D Columns (kaggle notebook link)

[10] User/Card Identification (kaggle discussion link)