IEEE-CIS Fraud Detection Project Report

Baiting Gai, Jiatian Xie, Ziyi Zhang

Table of Contents

Research Problem

Finance and banking are important sectors in our present-day generation, where almost everyone has to deal with banks either physically or online. The productivity and profitability of both the public and private sectors have tremendously increased because of the banking information system. Nowadays, financial services have provided many conveniences to a lot of people and created a huge economic benefit to society. However, financial frauds are also more likely to happen in our daily life. For example, in the US alone, the number of customers who experienced fraud hit a record of 15.4 million people, which is 16 percent higher than in 2015 (AltexSoft). Most E-commerce application system transactions are done through credit cards and

online net banking. These systems are vulnerable to new attacks and techniques at an alarming rate. Fraud detection in banking is one of the vital aspects nowadays as finance is a major sector in our life. The fraud prevention system is actually saving consumers millions of dollars per year (Kaggle). Researchers from the IEEE Computational Intelligence Society (IEEE-CIS) want to improve this figure, while also improving the customer experience. The project's data is provided by the world's leading payment service company, Vesta Corporation. We will use approximately 400 features to predict whether a transaction is a fraud with supervised learning techniques, and we are going to use AUC, which is the area under the ROC (receiver operating characteristic) curve, to evaluate our models.

Data Description

The IEEE-CIS Fraud Detection competition provides four datasets: two transaction datasets (train and test) and two identity datasets. The identity datasets include identity information about network connection and digital signatures associated with transactions. The train transaction dataset has 394 columns, with a total of 590,540 observations. The test transaction dataset has 393 columns and it does not have a target variable. The train identity dataset has 41 columns and 144,233 observations. The test identity dataset has the same size as that of the train dataset. The transaction dataset has the following features, and ProductCD, card 1 - card 6, addr, P/R_email domain, and M1-M9 are categorical features:

- transaction ID: the ID number of a transaction
- isFraud: the target variable, 1 = is fraud and 0 = is not fraud
- TransactionDT: timedelta from a given reference time
- TransactionAMT: the payment amount in USD
- ProductCD: product code for each transaction:
- card 1 - card 6: payment card information including card type, category, issue bank country, etc.
- Addr, addr1, addr2: address
- Dist: distance
- P_and R_emaildomain: purchaser and receipt's email domain
- C1-C14: counting, such as how many addresses are found to be associated with the payment card
- D1-D15: time delta, such as the number of days between the previous transaction
- M1-M9: matches, such as names on card and address
- Vxxx: Vesta engineered rich features such as ranking and other entity relations

The identity datasets have the following features:

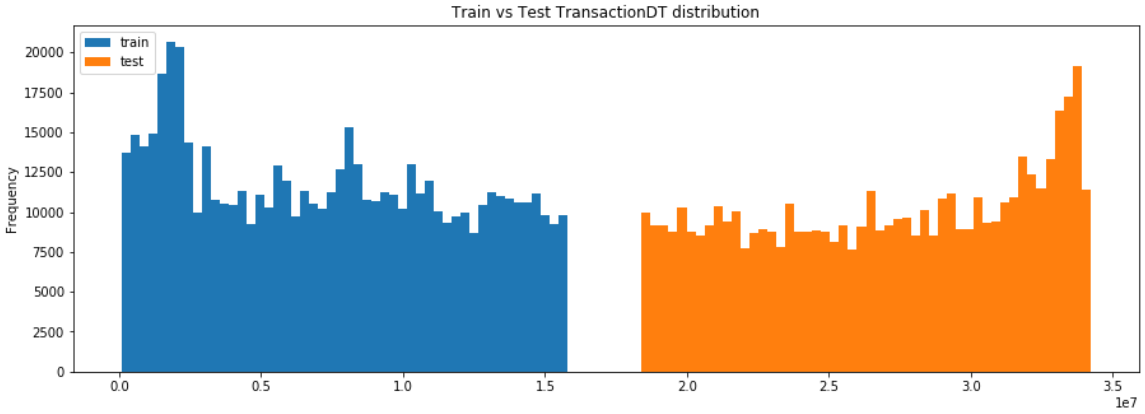
- TransactionID: the ID number of a transaction
- DeviceType: the type of device used during the transaction
- DeviceInfo: the information about the device used during the transaction
- Id_01 - id_38: the id number for each transaction

Exploratory data analysis

The task of the project is to predict fraud clients. It is a classification task and the most important thing that we need to notify is a class imbalance. If the data are mostly concentrated in one class, the established model will produce a high bias.

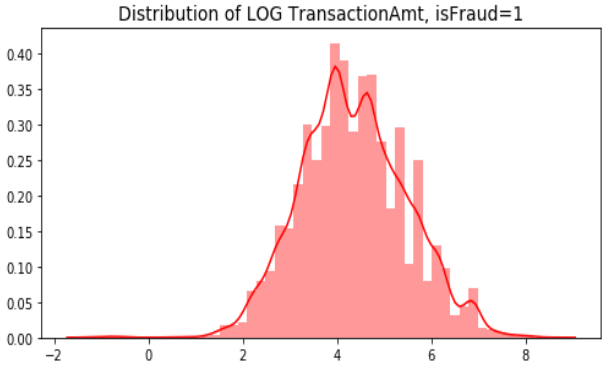
In the exploratory data analysis, we focused on data understanding and business understanding. Firstly, we analyzed the train and test data because we need to know how they are split and whether there exists a time series. Secondly, we examined the data imbalance problem. Thirdly, we understood some data meanings by exploring the data such as what card4 and card6 mean. Also, we got some basic information about meaningful features. However, there are some features we still can not understand because the distribution is very disorganized. We will try feature selection during modeling and if the features do not perform very well, we will remove some of them and adjust our model.

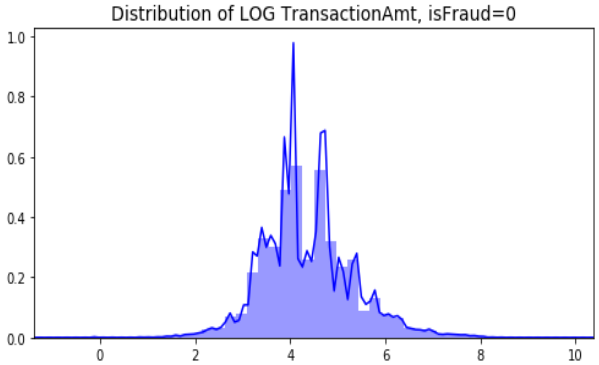
TransactionDT for both train and test



The TransactionDT feature is a timedelta from a given reference datetime (not an actual timestamp). The train and test transaction dates do not overlap, so it would be prudent to use a time-based split for validation.

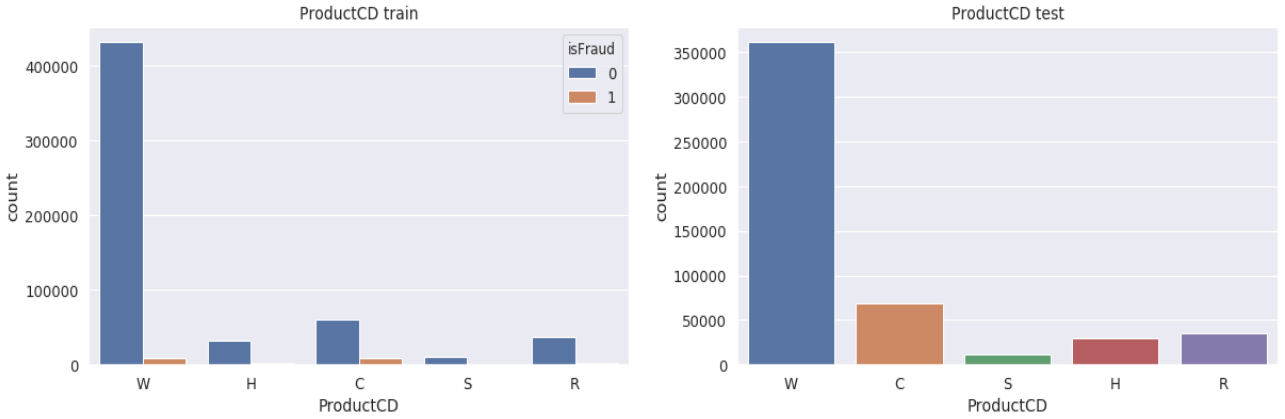
Transaction amount





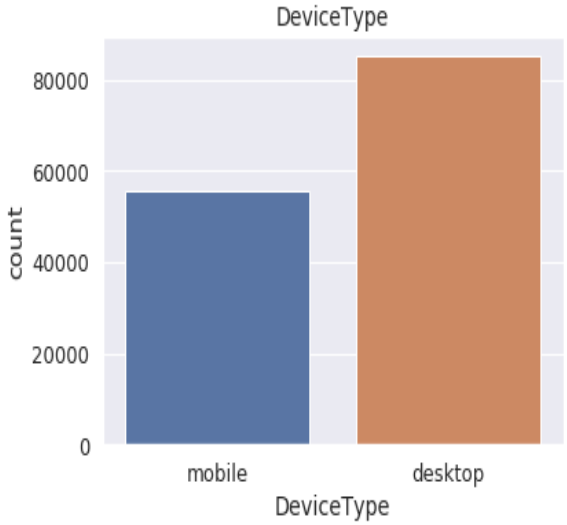
The above charts show the distribution of transactions amount. We have taken a log transform in some of these plots to better show the distribution, otherwise, the few, very large transactions skew the distribution. Because of the log transformation, any values between 0 and 1 will appear to be negative. From the distribution plot, fraudulent charges appear to have a higher average transaction amount.

Product CD



Even though for now we don’t know exactly what these values represent. But as you can see from the bar plots, we know that W has the highest number of observations, followed by C. The amount distribution in the train and test dataset is similar with a little difference which will not have an effect on the results. This information tells us that using the train data to predict the test data is reasonable.

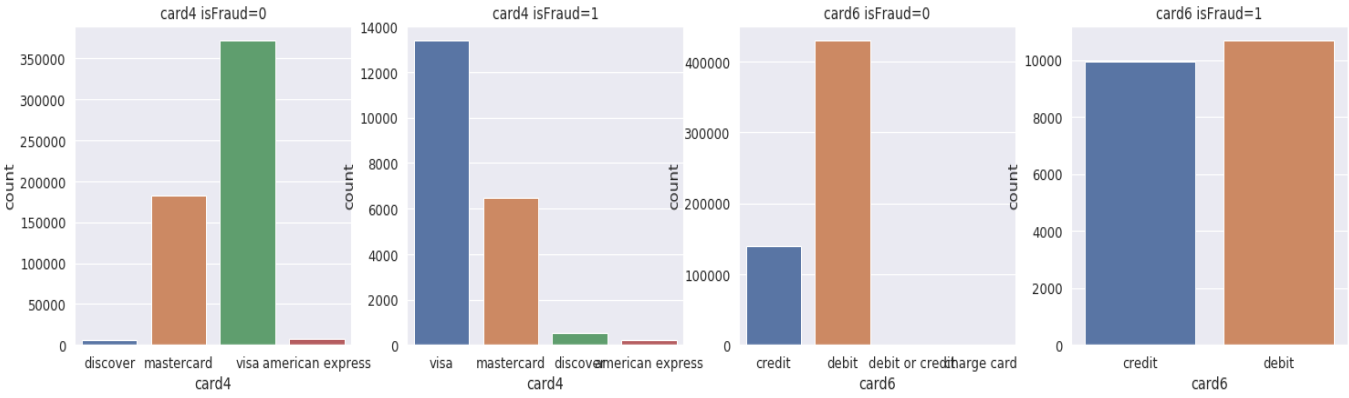
Device type and information




```
Unique Devices =  1786
Windows         47722
iOS Device      19782
MacOS           12573
Trident/7.0      7440
rv:11.0          1901
Name: DeviceInfo, dtype: int64
```

The above bar plot shows the count number of device types used during transactions. More transactions are completed on desktop than on mobile devices. Among all the transactions, approximately 53% are Windows. IOS and Mac are also major device types.

Card information (card4, card6)



From the above plots, we can guess what Card4 and Card6 mean since the feature meanings in the datasets are not very clear. Card4 represents the issuing bank of a card and Card6 represents the card type, which is debit card or credit card. Visa has the most frauds but also the most non-frauds. The fraud events in Discover is more than the non-fraud events in Discover. The last two plots show that credit cards are more likely to occur fraud events because the proportion of fraud to non-fraud of credit cards is higher than that of debit cards.

Feature Engineering and Model Preparation

The fraud detection datasets have many null values and some features, such as those start with “v”, contain mostly null values. Therefore, we need to preprocess the data. After merging the identity and the transaction datasets into the train and the test dataset, we first ensured the consistency of column names. The id columns in the train datasets use underlines (e.g. id_01) but those in the test dataset use hyphens (e,g, id-01). We changed the hyphens to underlines for the id columns in the test dataset. Then, we started creating new features by calculating the means and standard deviations of features with few null values (card 1, card 4, id_02, D15, and email domains). We have also grouped the card columns by the means and standard deviations of the transaction amount and time delta (D columns). We have also split the email domains by period (“.”). By adding the standard deviations and means, we can provide more data for the models so that they can have a higher AUC score.

After feature engineering, we are going to prepare the datasets for modeling. If more than 90% of a column’s value is null, we are going to drop the column. We also dropped columns which only have one value, because they may not have a great effect on the outputs of the models. Then, we dropped columns with extreme values to avoid influence from outliers. By applying these rules, we dropped a total of 84 columns. After that, we concatenated the categorical columns, which are:

'id_12', 'id_13', 'id_14', 'id_15', 'id_16', 'id_17', 'id_18', 'id_19', 'id_20', 'id_21', 'id_22', 'id_23', 'id_24', 'id_25', 'id_26', 'id_27', 'id_28', 'id_29', 'id_30', 'id_31', 'id_32', 'id_33', 'id_34', 'id_35', 'id_36', 'id_37', 'id_38', 'DeviceType', 'DeviceInfo', 'ProductCD', 'card4', 'card6', 'M4','P_emaildomain','R_emaildomain', 'card1', 'card2', 'card3', 'card5', 'addr1', 'addr2', 'M1', 'M2', 'M3', 'M5', 'M6', 'M7', 'M8', 'M9','P_emaildomain_1', 'P_emaildomain_2', 'P_emaildomain_3', 'R_emaildomain_1', 'R_emaildomain_2', 'R_emaildomain_3'

We applied label encoder, which can Encode target labels with a value between 0 and the number of classes minus 1, on our concatenated columns to convert all the categorical values to numbers. Finally, we sorted the train and test datasets by transaction ID and divided them into X (the train dataset), X_test (the test dataset) and y (the target variable) and changed infinite values to null values. At this stage, our datasets are ready for modeling.

The challenges of the dataset

The dataset we dealt with has many challenges. Firstly, the dataset is very huge in terms of observations (about 1,200,000 rows) and features (393 columns). Secondly, the meaning of many columns is vague. The dataset is a real and raw dataset. The features are mostly some information about credit cards. The problem is, we only know that the columns are related to the credit card information but we do not know what the specific information is. For example, we know Vxxx columns represent Vesta engineered rich features such as ranking and other entity relations but we do not know how they are ranked and the specific meanings so that we could not know how to deal with the features. However, there exist some relationships among many columns because they are all related to one credit card. It means that in the transaction dataset, serval transactions are created by one credit card. So, the magic of predicting fraud detection in this project is to not just focus on using transactions to predict fraud transactions, but using the unique credit-card identification to predict fraud transactions. Therefore, we had better group features in modeling to improve accuracy. Thirdly, the dataset is imbalanced. This is a general problem in the fraud detection problem. Most transactions are not fraud while a little part of transactions is a fraud. There are several methods to deal with the imbalance problem. The first one is to set a threshold of judging fraud detections. But we did not know how much the threshold is better in the problem. The second one is resampling including undersampling and oversampling methods. This approach will be more effective than the first one and the histogram-based method of the LightGBM model will help us address the problem.

Models

eXtreme Gradient Boosting (XGBoost)

The algorithm of XGBoost

XGBoost is a widely used model in Kaggle competitions. It is an ensemble machine learning algorithm based on a decision tree that uses a gradient boosting framework. While decision can be used for small/medium dataset to represent of possible solutions to a decision based on certain conditions, XGboost is more an advanced technique which optimized gradient boosting algorithm through parallel processing, tree-pruning and dealing with missing values to avoid overfitting problems (Morde,2019). What’s good about XGBoost is that it has a wide range of applications which can be used not only for regression but also classification problems. It runs smoothly, faster and more accurately than the gradient boosting model. XGboost uses fewer resources to get superior results only in a short period of time.

Although XGBoost and gradient boosting machine (GBM) are both ensemble tree methods, what is better about the XGBoost is that it improves the basic GBM through systems

optimization and algorithmic enhancements. XGBoost uses parallelized implementation to process the sequential tree building. The outer loop can compute the leaf nodes of a tree, the inner loop that enumerates the features. These two loops are interchangeable and limit the parallelization. Because the inner loops must be done before the out loop starts. By this way, it improves algorithmic performance. In addition to the parallelization, the stop criterion for tree splitting of the GBM model replies to the negative loss criterion of splitting. While Max_depth parameter can be used for XGBoost to start pruning trees backward (Morde,2019). Moreover, the algorithm can handle the hardware resources efficiently, which means it could be used for huge data frames that do not fit into the memory.

Why XGBoost fits our dataset

We chose XGBoost for the following three reasons.

Firstly, the model is helpful to handle missing values: Due to the fact that we have lots of missing values of many features. This is one of the most important things in dealing with the dataset to build models. XGBoost can naturally admit sparse features for input by automatically 'learning' missing value according to the training loss and handles different types of sparsity patterns in the data efficiently.

Secondly, we can utilize cross-validation using K-fold: XGBoost algorithm has a built-in cross-validation method so that we can use K- fold cross-validation to avoid overfitting. Also, we can get the AUC score for each fold of both training and validation datasets.

Lastly, XGBoot can deal with huge dataset in a short time: Since it is a huge dataset with millions of rows which may take a long time to build the models. XGboost is user-friendly for large datasets because it has the best combination of prediction performance and processing time compared to the traditional models like logistic regression, random forest, gradient boosting, etc. As a result, it only takes us about several minutes to wait.

Interpretation of XGBoost

The parameters that we used in the XGBoost model are n estimators, max depth, learning rate, subsmaple, colsample bytree, gamma, alpha and tree method. The learning rate is used for step size shrinkage to prevent overfitting. We can get the weights of new features after each boosting step, so the learning rate shrinks the feature weights to make the boosting process conservative. So we chose a relatively low number as our learning rate to be not that conservative and more accurate. Max depth means the maximum depth of a tree. If we increase the number, the model will be more complex and more likely to overfit. And due to the aggressively consumes memory, we chose a small number which is lower than the default (XGBoost doc). Subsample is related to the size of selecting the ratio of the training sets. We used 0.9 which means that XGBoost will randomly select 90% of the training data to grow trees. Since we had already used the built-in cross-validation function to prevent overfitting, we don't

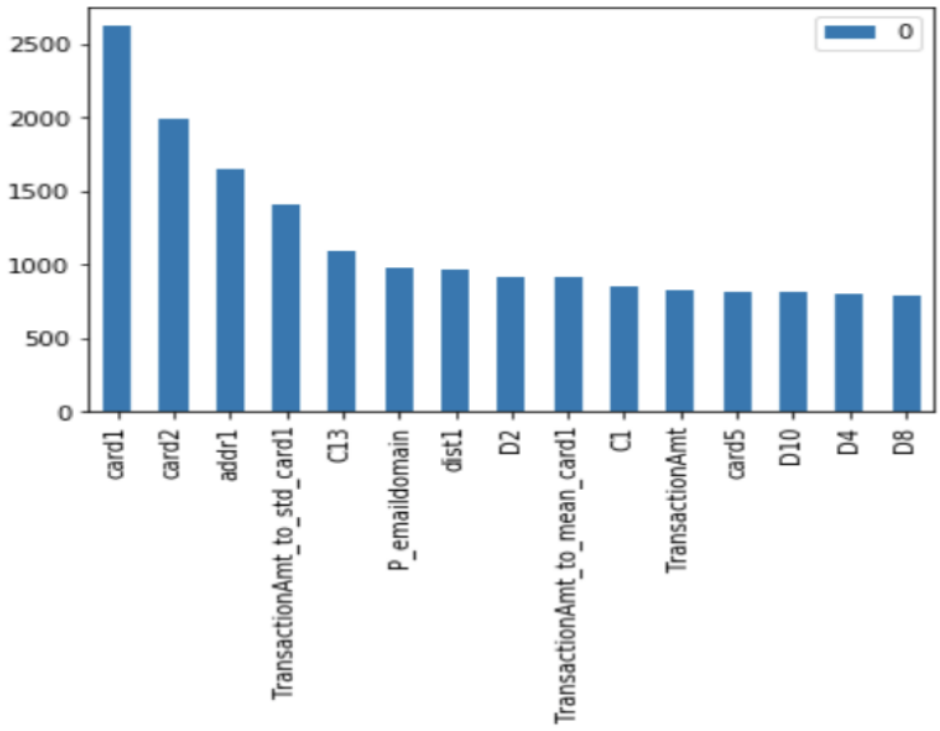
need to be too worried about this, that's the reason that we would like the model to select most of the data to train and grow trees.

Model result and feature selection

According to the result of the 5 folds cross-validation, we got the AUC for both training and validation of each fold. It makes sense that the AUC training dataset score is higher than the validation dataset. Four folds of the validation AUC score are around 0.91, except fold 4 is 0.94. Overall the mean of the AUC score of 5 folds is about 0.92.

Fold	**AUC on Train data**	**AUC on Valid data**
k=1	0.980102	0.912620
k=2	0.981621	0.923396
k=3	0.978894	0.918738
k=4	0.983248	0.940381
K =5	0.976954	0.91559
mean	**0.922145**	

After we got the AUC score which can evaluate the accuracy of the model performance, we are also curious about the importance of feature to see which variables play vital roles in predicting the result. By using the feature selection function built in the XGBoost model, we sorted and selected the top 15 important variables. Card 1 and card 2 are the most important features in predicting whether the transaction is a fraud. Other variables including the address information, c13, are also important in the XGBoost model. The 'card1' represents the id of the credit cards and 'Transaction_to_std_card1' and 'Transaction_to_mean_card1' respectively represent the transaction amount standard deviation group by card1 and the transaction amount standard mean group by card1. It means that the algorithm predicts whether a transaction is a fraud based on the situation of a credit card. If the credit card is not reliable or the personal information is reliable, the transaction of the credit card is considered as a fraud. The 'card2' may represent bank-branch in some places. So it can be explained that some unreliable banks will lead to fraud transactions.



Light Gradient Boosting Machine (LightGBM)

The algorithm of LightGBM

The LightGBM model is an advancement model from the XGBoost model. The motivation of the LightGBM model is to solve the problem that modern datasets tend to be both large in the number of samples and the number of features. It has a lot of advantages over XGBoost including faster training efficiency, lower memory usage, higher accuracy, support for parallel learning, the capability of processing large amounts of data and capability of dealing with categorical features without too much feature engineering of categorical features like label encoding and one not encoding.

The improvements of XGBoost and LightGBM over boosting algorithms are mainly in two aspects: the tree-growth strategy and the method of finding the best split for each split. XGBoost and LightGBM both use the leaf-wise growth strategy which aims to reduce the loss the most. However, the leaf-wise strategy is more likely to overfitting. So, in the LightGBM model, it uses not only the number of leaves but also the depth of the tree to prevent overfitting. As for how LightGBM improves the accuracy by finding the best split for each split, We will explain it combining with our dataset and it is also why we choose the LightGBM model.

Why LightGBM fits our datasets

We choose LightGBM for the following three reasons.

Firstly, the histogram-based method which LightGBM used improved our computation time by reducing computation complexity, and the method can group similar features. In the process of finding the best split for each leaf in the GBDT model, it requires the algorithm to go through every feature of every data point. The amount of time it takes to build a tree is proportional to the number of splits that have to be evaluated. So, what XGBoost and LightBoost improved is to find the best split while avoiding going through all features of all data points. Often, small changes in the split don't make much of a difference in the performance of the tree. Histogram-based methods take advantage of this fact by grouping features into a set of bins and perform splitting on the bins instead of the features. This is equivalent to subsampling the number of splits that the model evaluates. Since the features can be binned before building each tree, this method can greatly speed up training, reducing the computational complexity. In conclusion, the histogram-based method can help us address the problems of big data volumes and group similar features. (Guolin Ke1, Qi Meng, et.al.)

Secondly, LightGBM works well in the spare data. Although the dataset has a lot of 0 in some columns, we need to find more information from these features and LightGBM can help us. Though lightGBM does not enable ignoring zero values by default, it has an option called zero_as_missing which, if set to True, will regard all zero values as missing. Besides, LightGBM both treat missing values in the same way as xgboost treats the zero values in sparse matrices; it ignores them during split finding, then allocates them to whichever side reduces the loss the most. (keitakurita, 2018)

Thirdly, the gradient-based one-side sampling method in LightGBM can solve the imbalance data on the fraud detection dataset. In the general boosting model, not all data points contribute equally to training; data points with small gradients tend to be more well trained (close to a local minima). This means that it is more efficient to concentrate on data points with larger gradients. The most straightforward way to use this observation is to simply ignore data points with small gradients when computing the best split. However, this has the risk of leading to biased sampling, changing the distribution of data. In order to mitigate this problem, lightGBM also randomly samples from data with small gradients. This results in a sample that is still biased towards data with large gradients, so lightGBM increases the weight of the samples with small gradients when computing their contribution to the change in loss (this is a form of importance sampling, a technique for efficient sampling from an arbitrary distribution).

Interpretation of LightGBM

The parameters we adjusted in the model include num_leaves, min_child_samples, max_depth, learning_rate, subsample_freq, subsample, bagging_seed, reg_alpha, reg_lambda, colsample_bytree. Num_leaves, min_child_sampls, and max_depth are the parameters adjusting the shape of the tree. Since the leaf-wise growing tree strategy, we need to use both the number

of leaves and tree depth to control the complexity. For the learning_rate parameter, a lower learning rate will improve accuracy. The subsample, subsample_fre, and bagging_seed parameters are used to control the sub resampling situation in the process of creating recursive trees. The reg_alpha and reg_lambda are penalty terms on the objective function to prevent overfitting. We chose the best value of the parameters using Grid Search in Scikit-learn. The parameters we fit in the final LightGBM model include number of leaves of 256, minimum number of data needed in a leaf of 79, maximum tree depth for base learners of 13, learning rate of 0.03, 90% subsample of the training instances with frequency of 3 each time, L1 regularization of 0.3 and L2 regularization of 0.3.

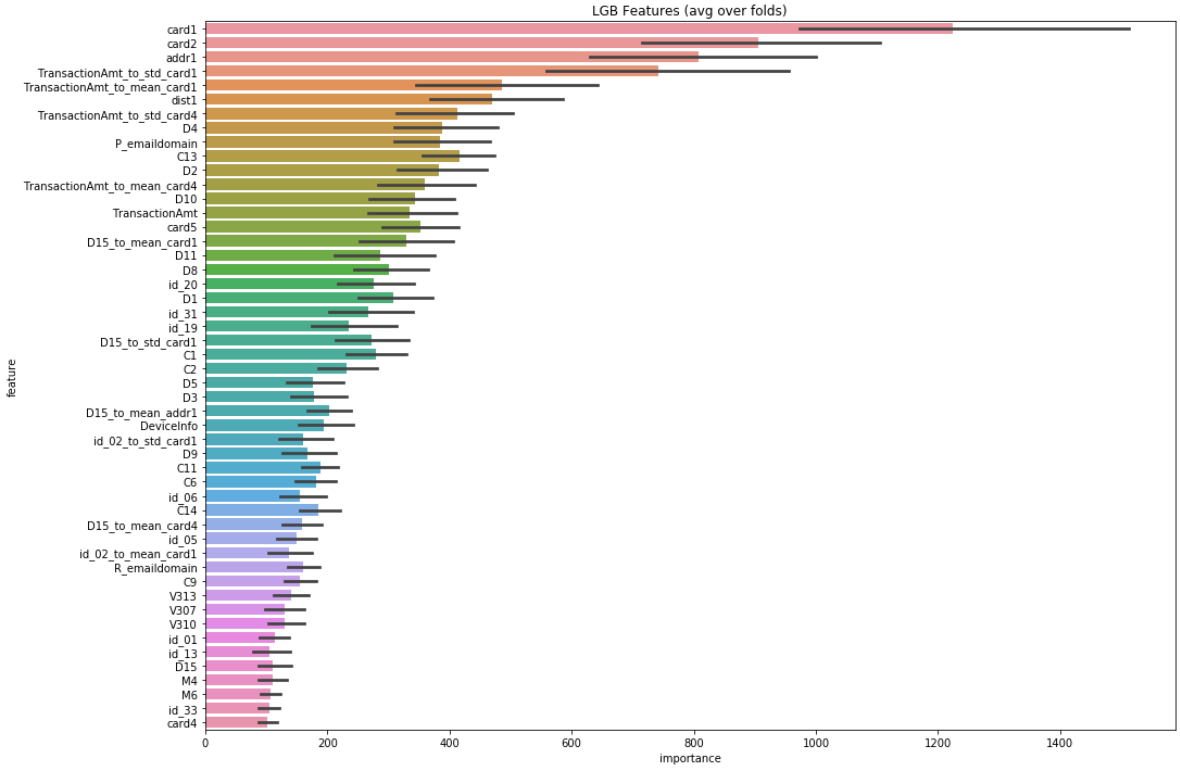
Model result and feature selection

The model results are below. We used a 6-fold cross-validation method to prevent overfitting. The mean AUC of the 5 folds is 0.9290 and the standard deviation is 0.0099. It means that the model can predict 92.9% fraud in the transactions.

Fold	**AUC on Train data**	**AUC on Valid data**
k =1	0.994808	0.926728
k=2	0.990416	0.926965
k=3	0.994589	0.946387
k=4	0.993283	0.946645
k=5	0.994799	0.922091
mean	**0.9290**	

From the feature importance of LightXGB, the top five important features are ‘card1’, ‘card2’, ‘add1’, ‘Transaction_to_std_card1’ and ‘Transaction_to_mean_card1’. The results make sense and it has basically the same results with the XGBoost model. Important features include ‘TransactionAmt_to_std_card4’ ‘C13’’ and ‘TransactionAmt_to_mean_card4’. ‘Transaction_to_std_card4’ and ‘Transaction_to_mean_card4’ respectively represent the transaction amount standard deviation group by card4 and the transaction amount standard mean group by card4. ‘Card4’ represents the issuing company of the card. The results show that there is a relationship between issue bank and fraud detection. The importance of ‘C13’ and ‘addr1’ represents fraud detection is related to the address information including the number of addresses the credit card has used in the transactions and the first address. If the address in the transaction is different from the first address in the credit one, the transaction is a fraud. If there are too

many addresses linked with the credit card, it shows that the credit card is problematic and the transactions with the credit card are likely to be a fraud transaction.



Conclusion

For the IEEE-fraud detection datasets, our goal is to find out whether a transaction is a fraud. Because the data have many missing values, we handled those by adding standard deviations and means of variables with few missing values and then dropped the columns with the most missing values to prepare the data for modeling. After comparing the prediction results, we concluded that Light GBM has a higher mean AUC score, 0.9290, and therefore has a better performance. The Light GBM model also runs faster than the XGBoost model. Although Light GBM only performs slightly better than XGBoost, a slight accuracy improvement may have a great impact on fraud detection. According to the result of the feature importance, we found that information about the card type, the company of the card, transaction address, the address of the credit card are the relatively important features in predicting whether the transaction is a fraud. Since the rise of the digital economy has been matched by the rapid spread of fraud, using machine learning models like XGboost and Light GBM could be helpful for us to be better equipped to data silos and safeguard against present and future.

References

Can you detect fraud from customer transactions?. *Kaggle*. Retrieved from https://www.kaggle.com/c/ieee-fraud-detection/discussion/101203

Fraud Detection: How Machine Learning Systems Help Reveal Scams in Fintech, Healthcare, and ECommerce. *AltexSoft.* Retrieved from www.altexsoft.com/whitepapers/fraud-detection-how-machine-learning-systems-help-reveal-scams-in-fintech-healthcare-and-ecommerce/.

Guolin Ke, Qi Meng, et.al. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Nips*. Retrieved from https://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf

IEEE-CIS Fraud Detection. *Kaggle.* Retrieved from https://www.kaggle.com/c/ieee-fraud-detection/overview

Keitakurita. (2018). LightGBM and XGBoost Explained. *Kaggle*. Retrieved from https://www.kaggle.com/c/ieee-fraud-detection/discussion/107833

Morde, V. (2019, April 8). XGBoost Algorithm: Long May She Reign! Retrieved from https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

Payment card number. (2020, February 18). *Wikipedia.* Retrieved from https://en.wikipedia.org/wiki/Payment_card_number#Issuer_identification_number_(IIN)

Prusti, Debachudamani, and Santanu Kumar Rath. (2019). Web Service Based Credit Card Fraud Detection by Applying Machine Learning Techniques. *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*, doi:10.1109/tencon.2019.8929372.

Vidanelage, Harindu Mudunkotuwa Mudunkotuwe Hitiwadi, et al. (2019). "Study on Machine Learning Techniques with Conventional Tools for Payment Fraud Detection." *2019 11th International Conference on Information Technology and Electrical Engineering (ICITEE)*, doi:10.1109/iciteed.2019.8929952.