Part 1 Core: Exploring and understanding the Kaggle process

1. Business Understanding

It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase. We will build a Machine Learning model to detect higher accuracy fraud. And then customers can easily get on with his shopping without any hassle.

2. Data Understanding

The datasets presents occurred credit cards transactions provided by company. We need to predict the probability that an online transaction is fraudulent (the binary target isFraud). Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

Features contains Transaction and identity which are nominal or numerical input variables, as denoted by the Table 1 and 2.

Table 1: Transaction

Feature	Description	Type of value
TransactionDT	time delta from a given reference datetime (not an actual timestamp)	Numeric
TransactionAMT	transaction payment amount in USD	Numeric
ProductCD	product code, the product for each transaction	Categorical
card1 - card6	payment card information, such as card type, card category, issue bank, country, etc.	Categorical
addr	address	Categorical
dist	distance	Numeric
P_ and (R__) emaildomain	purchaser and recipient email domain	Categorical
C1-C14	Counting, such as how many addresses are found to be associated with the payment card, etc. The actual meaning is masked.	Numeric
D1-D15	time delta, such as days between previous transaction, etc.	Numeric
M1-M9	match, such as names on card and address, etc.	Categorical
V1-V339	Vesta engineered rich features, including ranking, counting, and other entity relations.	Numeric

Table 2: Identity

Feature	Description	Type of value
id12 - id38	Identity information – network connection information (IP, ISP, Proxy, etc) and digital signature (UA/browser/os/version, etc) associated with transactions.	Categorical
DeviceType	timedelta from a given reference datetime (not an actual timestamp)	Categorical
DeviceInfo	transaction payment amount in USD	Categorical

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

Due to confidentiality issues, the dataset cannot provide the original features and more background information about the data. So we need to explore the dataset further to find interesting patterns.

3. Data Exploration

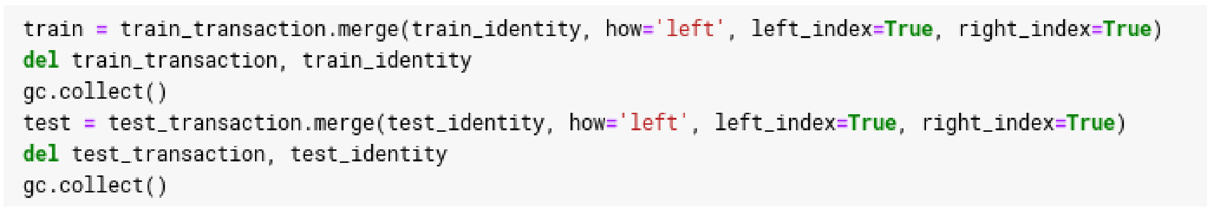
● Download the Dataset

First, we download the dataset. Then we will do some findings in each pattern and discuss the potential consequences.

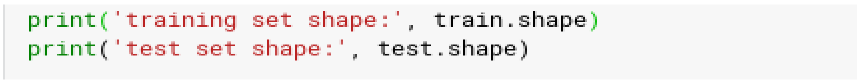
● Quality: Basic information of the datasets

From the csv file, we found that these four datasets include huge instances. So we need to merge the transaction and identity in train and test dataset respectively to explore features of instances together.

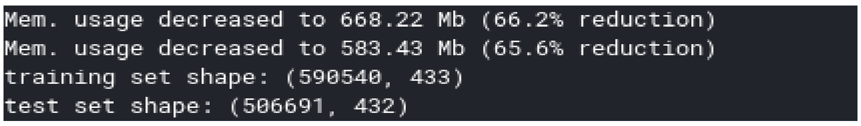
Also, it is necessary to reduce memory usage (the code based on the kernel: IEEE_MEMORY_REDUCE https://www.kaggle.com/sid00733/ieee-memory-reduce) to ensure the normal running, and then look at the shape of the dataset.



```python
train = train_transaction.merge(train_identity, how='left', left_index=True, right_index=True)
del train_transaction, train_identity
gc.collect()
test = test_transaction.merge(test_identity, how='left', left_index=True, right_index=True)
del test_transaction, test_identity
gc.collect()
```



```python
print('training set shape:', train.shape)
print('test set shape:', test.shape)
```

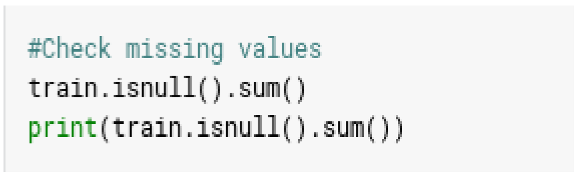


```
Mem. usage decreased to 668.22 Mb (66.2% reduction)
Mem. usage decreased to 583.43 Mb (65.6% reduction)
training set shape: (590540, 433)
test set shape: (506691, 432)
```

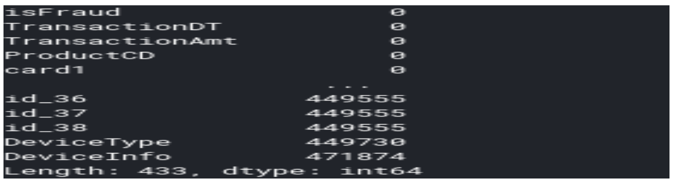
There are 20663 frauds out of 590540 transactions (Using find function from csv file.).The dataset is highly unbalanced, the positive class (frauds) account for 3.499% of all transactions.
Potential consequence: We will consider use some powerful machine learning to handle the dataset to accomplish the task.

● Completeness: Missing value & repeated value

1. Missing value



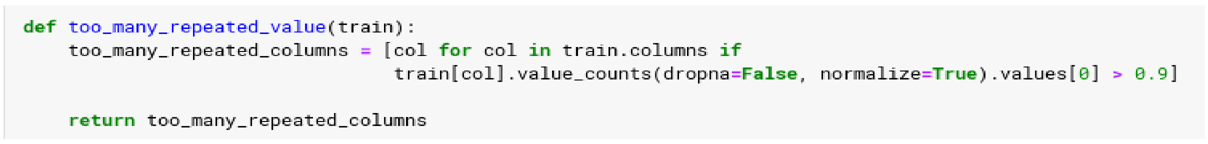
```python
#Check missing values
train.isnull().sum()
print(train.isnull().sum())
```



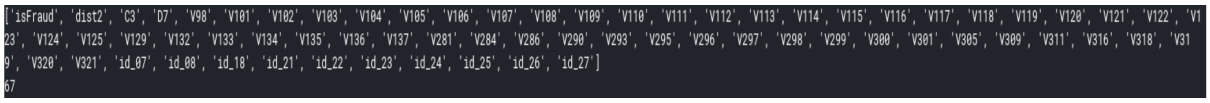
```
isFraud                  0
TransactionDT            0
TransactionAmt           0
ProductCD                0
card1                    0
                      ...
id_36               449555
id_37               449555
id_38               449555
DeviceType          449730
DeviceInfo          471874
Length: 433, dtype: int64
```

Potential consequence: In the original identity dataset, there are many missing value in id_01-id_38 features. Not all transactions have corresponding identity information.
Thus, there are more missing value in the train dataset after merged transaction and identity dataset.

2. Repeated value



```python
def too_many_repeated_value(train):
    too_many_repeated_columns = [col for col in train.columns if
                                 train[col].value_counts(dropna=False, normalize=True).values[0] > 0.9]

    return too_many_repeated_columns
```



```
['isFraud', 'dist2', 'C3', 'D7', 'V98', 'V101', 'V102', 'V103', 'V104', 'V105', 'V106', 'V107', 'V108', 'V109', 'V110', 'V111', 'V112', 'V113', 'V114', 'V115', 'V116', 'V117', 'V118', 'V119', 'V120', 'V121', 'V122', 'V1
23', 'V124', 'V125', 'V129', 'V132', 'V133', 'V134', 'V135', 'V136', 'V137', 'V281', 'V284', 'V286', 'V290', 'V293', 'V295', 'V296', 'V297', 'V298', 'V299', 'V300', 'V301', 'V305', 'V309', 'V311', 'V316', 'V318', 'V31
9', 'V320', 'V321', 'id_07', 'id_08', 'id_18', 'id_21', 'id_22', 'id_23', 'id_24', 'id_25', 'id_26', 'id_27']
67
```

● **<u>Representation of individual features</u>**

We will explore and analyse by choosing some data from categorical and numerical Variables since it contains so much attributes.

3.1 Categorical Variables

1. ProductCD

There are 5 Levels in ProductCD variable. There are no missing values in the variable.
Type W takes the maximum values and S takes the least. The proportion of Fraudulent Transactions are higher in Product Code of W and C and are least or negligible with other Product Codes.

2. Card1-Card6

Card1, card2, card3 and card5 have a very high numbers, especially column card1.

3. P_emaildomain and R_emaildomain

Gmail takes the maximum percentage in value variable and Proton-mail takes the highest in fraud percentage.

4. M1 to M9 Variables

Most of the M1-M9 columns have either 2 or 3 Unique Values. All of these columns have a lot of Missing Values. Except M4, rest of the columns have either True or False Values.

5. Id_01 to id_38 Variables

The percentage of values missing in id columns is high. As we talked before, the reason would be that not all identity data had details of TransactionID in the transaction dataset.
Some of them are Binary. In addition id_30, id_31 have more categories, but can be reclassified.

6. Device Type Variable

Although the Number of Mobile devices are less in number, the percentage of Fraudulent Transactions is highest.

3.2 Numerical Variables

1. TransactionDT

Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset.

2. TransactionAmt Variable

There is no missing Values, and some difference between the Mean Transaction Amounts of Fraudulent and Non Fraudulent Cases.

3. Dist1 and dist2 variables

There are a lot of Missing values in this variables. Maximum distance is 10286, considering the distribution of this variable, this definitely has to be an outlier. The Minimum value=0 also doesn't make sense for distance.

4. C1 to C14 Variables

None of the Columns C1-C14 have missing values. Some of these have high numbers.

5. D1 to D15 Variables

Apart from D1, rest of the columns have high percentage of Missing values. Except D9, rest of the columns have lot of Unique Values. This can be attributed to the fact that it is associate to the fix time, such as days between previous transaction, etc.

6. V1 - V339 Variables

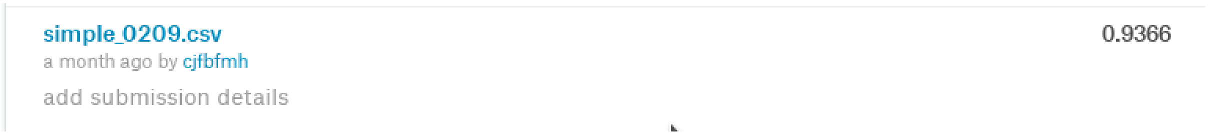
These values accounts are the main columns in the dataset, which will cause the overfitting and obtain more weight in that types value. We may consider drop some columns not useful.

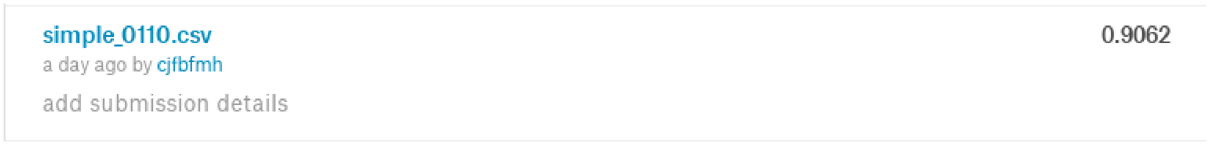
● Feature interactions

We also check the correlation between the variables and eliminate them that have high correlation. And we may also consider to remove some feature correlated less variables between each and isFraud. This will be explain more using visualization.

●Modelling & Score

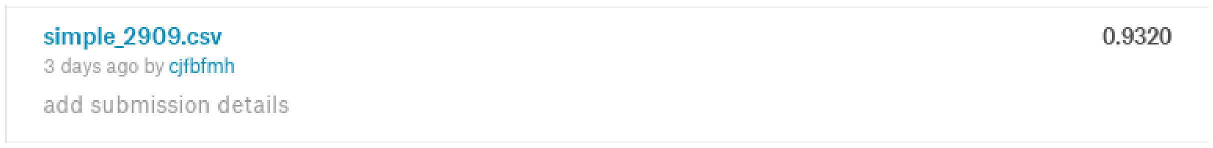
We choose the Xgboost as the classification method to handle the dataset which including massive instances.


simple_0209.csv 0.9366
a month ago by cjfbfmh
add submission details


simple_0110.csv 0.9062
a day ago by cjfbfmh
add submission details

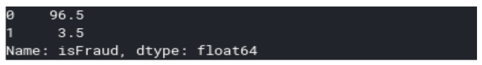
In the first round, we just fill in the missing value and en label the data to standardize them. The result is 0.9366. The second round, We consider to remove some not useful values and tuning some parameters, but the score is lower compared with the first time. The reason may be in this case, the dataset is highly unbalanced so that there is overfitting in the first time. The other reason is probably because we need to change another method to improve the result.

We also use another method (Lightgbm) and obtain the score:

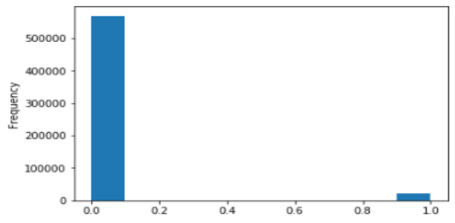

simple_2909.csv 0.9320
3 days ago by cjfbfmh
add submission details

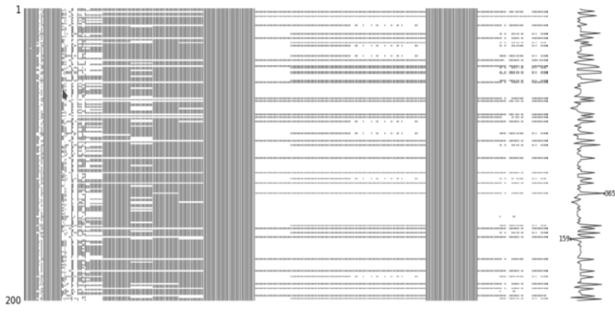
4. Visualisation

1. IsFraud & Missing value



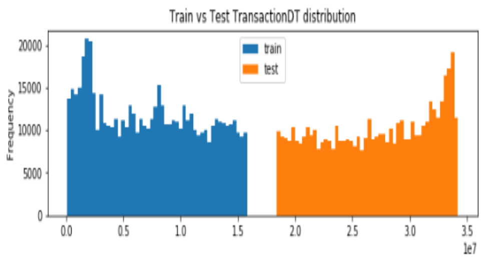
```
0    96.5
1     3.5
Name: isFraud, dtype: float64
```

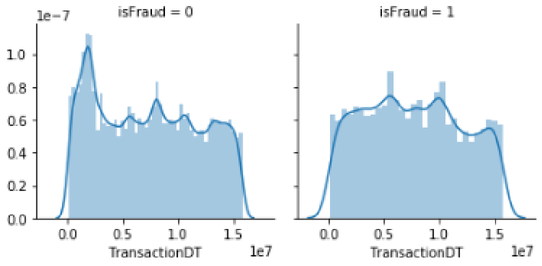




This is unbalanced dataset (Fraud:3.5% NoFraud:96.5%) which has missing value in many attributes.

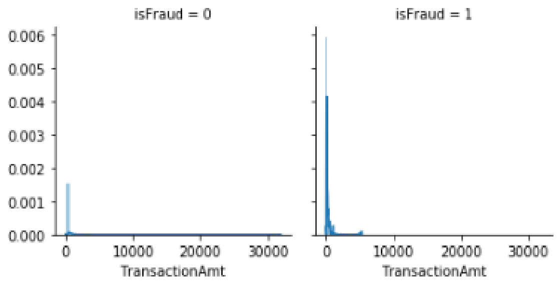
2. TransactionDT

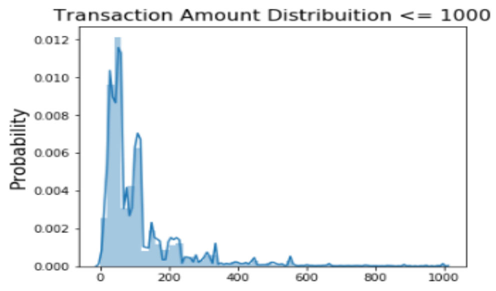




As expected, we can see that the transaction time in train and test set are not overlap. The transaction time in test dataset time is after train dataset, which is reasonable. As can be seen from the above pictures, the normal trading volume fluctuates with time, but the fraudulent trading is obviously relatively flat.

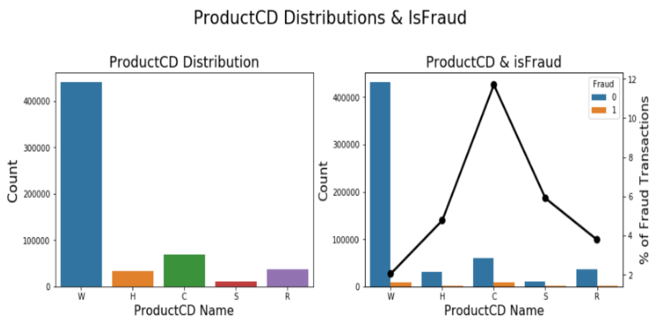
3. TransactionAmt



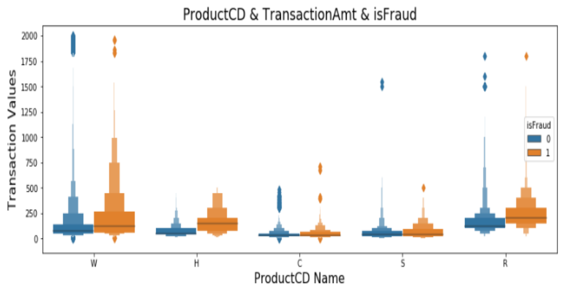


It is clearly that Transaction money of isFraud prefer the small amount, most of them less than 200$.

4. ProductCD

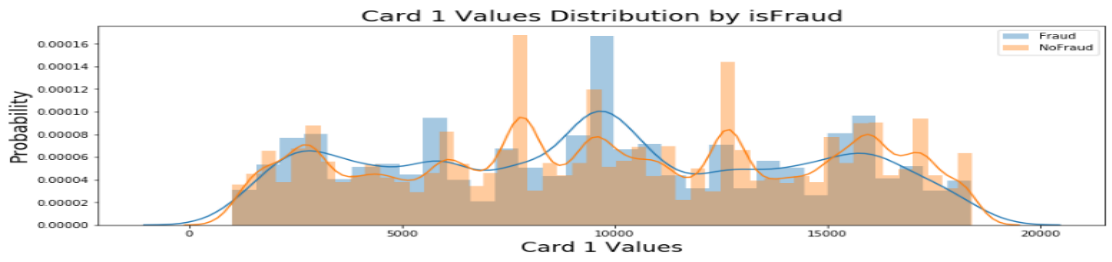


In

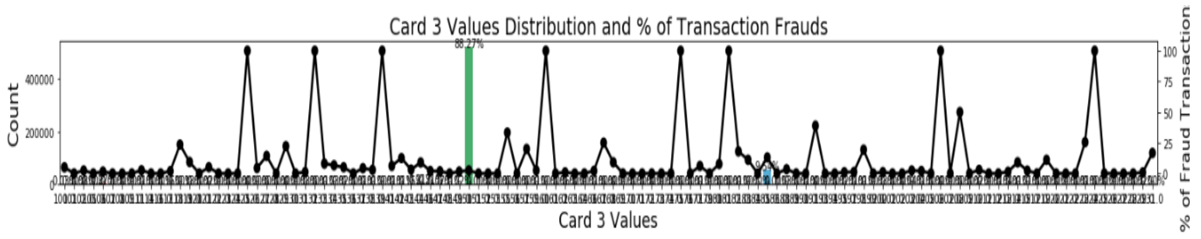


ProductCD, W has the highest count and lowest fraud proportion compared with C which has the less count but highest proportion. This conclusion has the same reason as TransactionAmt we observed because the TransactionAmt of fraud is usually small.

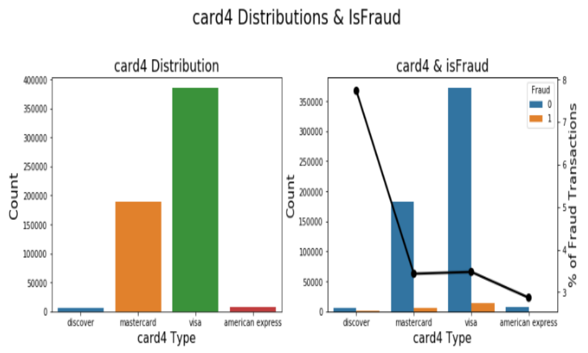
5. Card

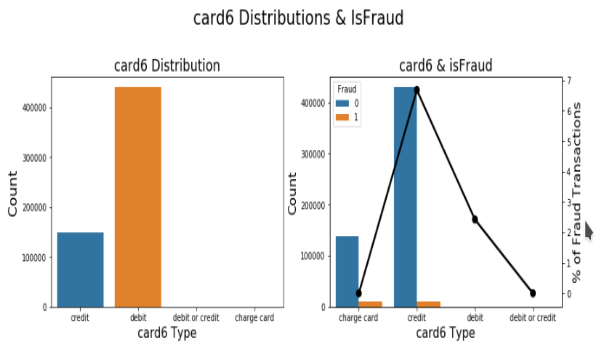


We can see there is no much correlated between card1 and isFraud and consider to delete it. We also can plot similarly for card2.



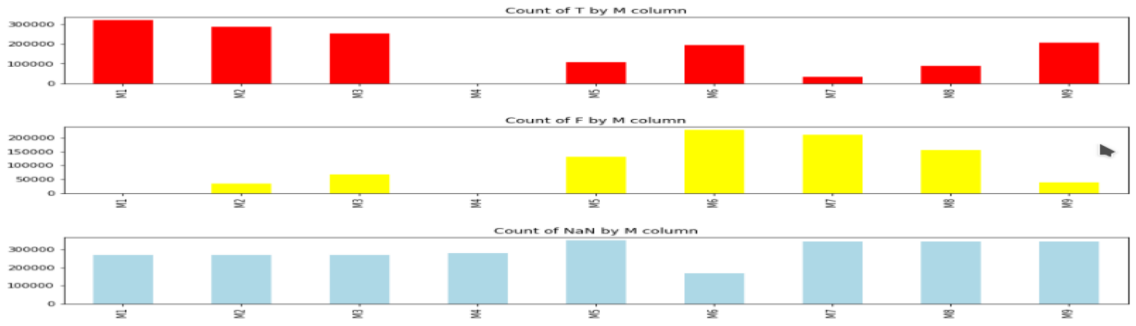
In card3, some value accounts for 88.27% has the lower percentage in isFraud while 9.54% has the higher percentage in isFraud.



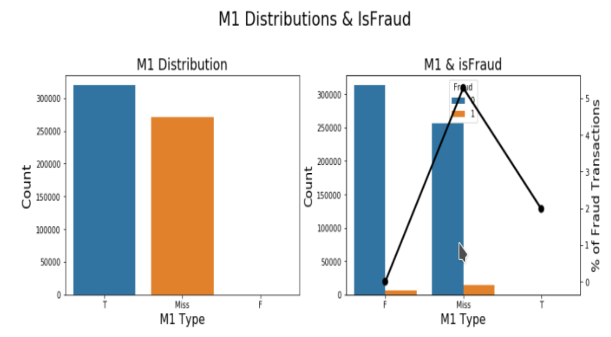


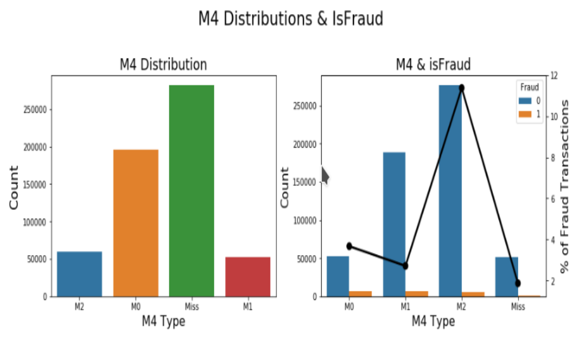
Discover card in card 4 and credit card in card6 has higher isFraud percentage respectively.

6. M1-M9

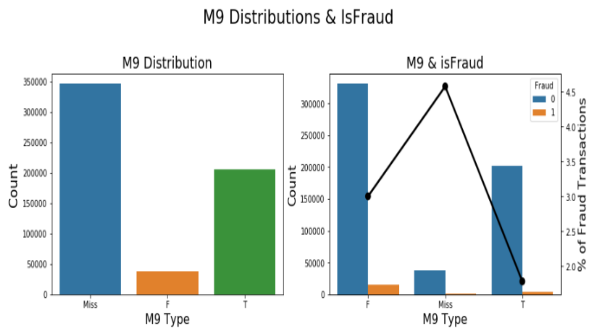


This is distribution of all the features in M. We can see that M4 is quite different from other attributes.



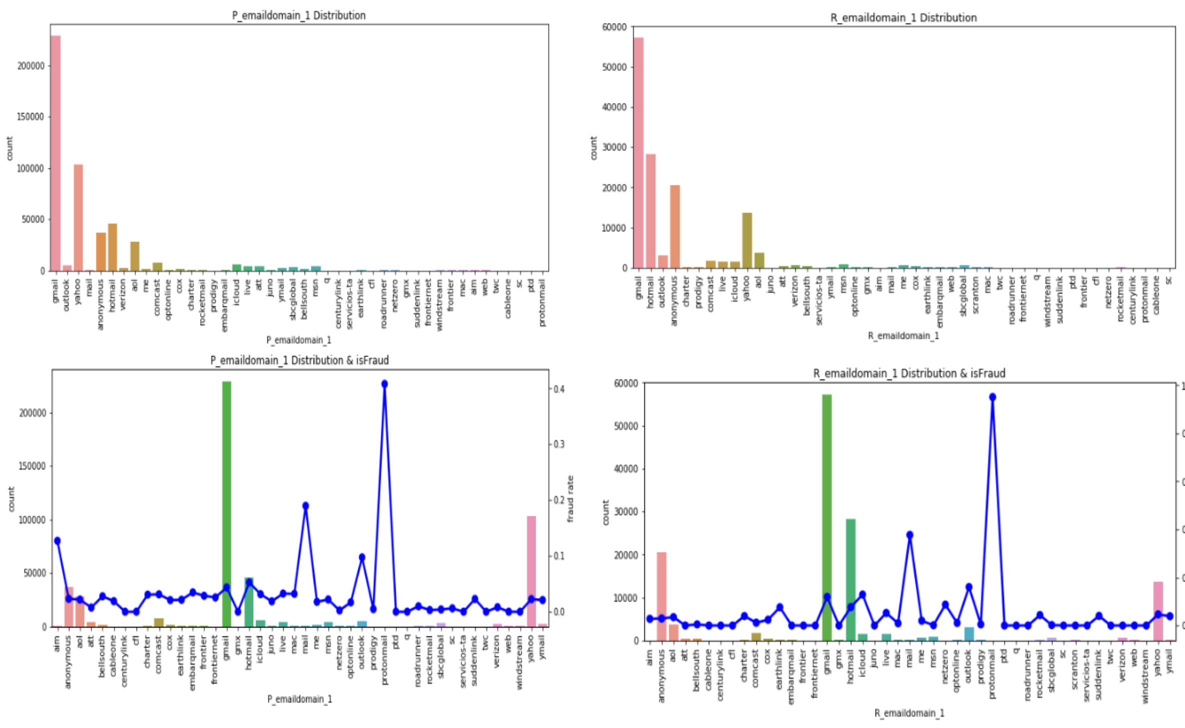


As is evident from the figure, T is the main value in M1 and has the low fraud percentage. M1 does not contribute to determining whether the transaction is fraudulent or legal.



In feature M4, M2's fraud is relatively high.
In feature M9, Missing value's fraud is relatively high.

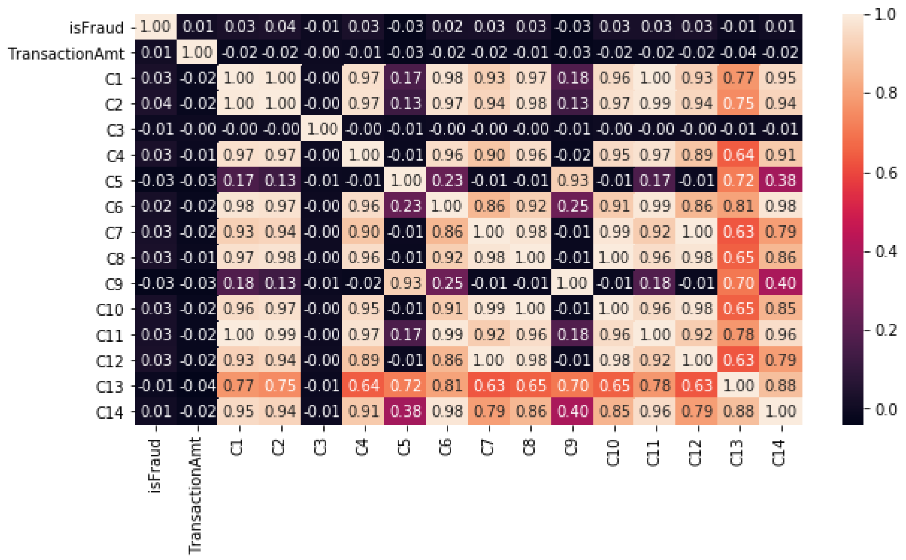
7. P_emaildomain and R_emaildomain



The proportion of fraud in two mailboxes is similar, and the fraud of Proton-mail is relatively high, and the proportion of fraud is nearly 100%.

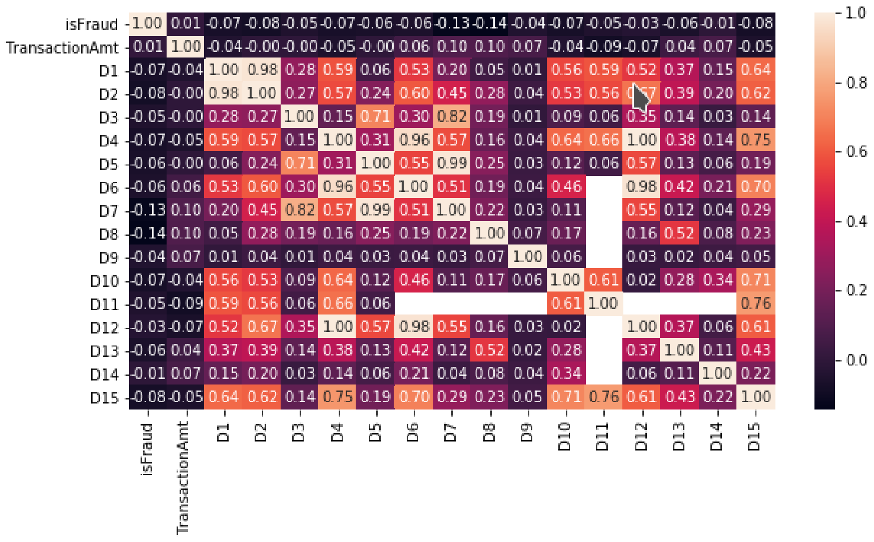
8. Feature interactions

1) C1-C14



We can see that the C3, C9 and C5 have less correlation with isFraud. Also, C1 and C2 has highly correlation in the internal, we can consider to choose one of the two.

2) D1-D15



We can see that the D1-D4, D7, D10 and D15 have less correlation with isFraud since the correlation approaches 0. Also, there are no much correlation in rest features.

Part 2 Competition: Developing and testing your machine learning system

1. Initial design

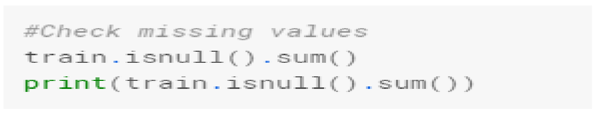
Before submitted to Kaggle, we design the initial system according to the following steps:

Step 1. Load Data

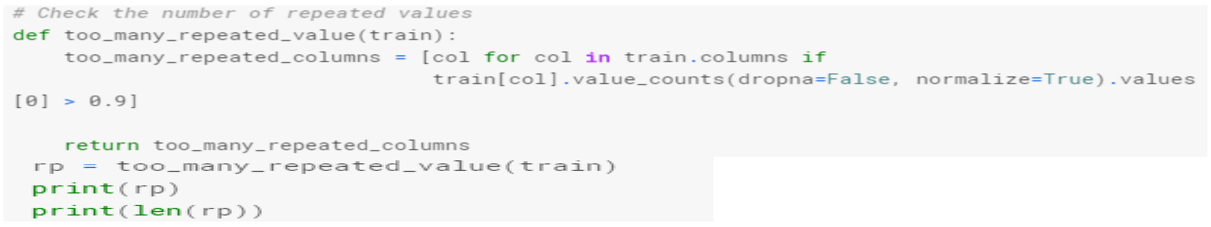
We download the five dataset from Kaggle and merge train_transaction.csv and train_identity.csv dataset to check and deal with them. Then do similarly merge test dataset.

Step 2. Initial Data Analysis
Carefully observe each attribute in the two datasets and the distribution of value. This enormous datasets contain 54k instances and 400+ attributes. We also consider properties of the dataset, such as a highly unbalanced dataset.

Step 3. Pre-process Data
After doing some feature engineering, we can filter data: drop columns, count encoding, fillna.
1. Check missing and repeated values
Remove attributes with missing values and repeated values greater than 90%.Then fill rest missing value using -999.



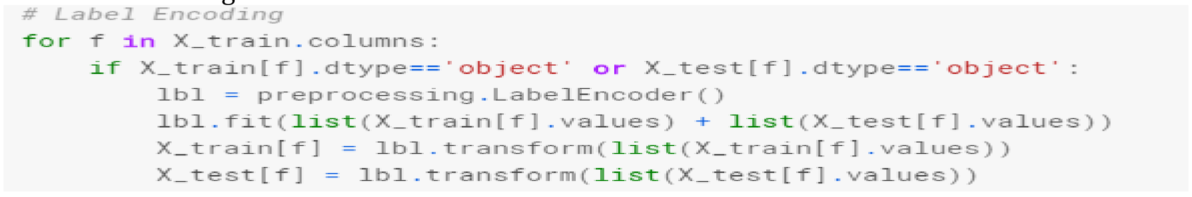
```
#Check missing values
train.isnull().sum()
print(train.isnull().sum())
```



```
# Check the number of repeated values
def too_many_repeated_value(train):
    too_many_repeated_columns = [col for col in train.columns if
                                  train[col].value_counts(dropna=False, normalize=True).values
[0] > 0.9]

    return too_many_repeated_columns
 rp = too_many_repeated_value(train)
 print(rp)
 print(len(rp))
```

2. Drop special values--generalisation/over-fitting
For independent variables, we mainly consider the relationship between each feature and isFraud (Target Class).
Delete M1, M5 and M7 due to less feature correlated with isFraud.
For group of data, we could put them together for analysed. Then observe their internal correlation (such as: synchronous changes) and external relations (vs isFraud).
Eliminate values that have high correlation, such as delete C3, C5 and C9 in C feature.

3. Encode the string value

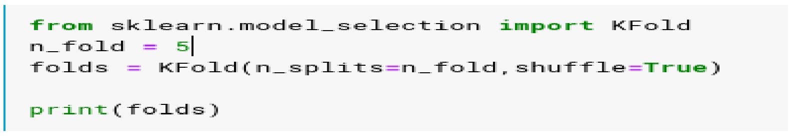


```
# Label Encoding
for f in X_train.columns:
    if X_train[f].dtype=='object' or X_test[f].dtype=='object':
        lbl = preprocessing.LabelEncoder()
        lbl.fit(list(X_train[f].values) + list(X_test[f].values))
        X_train[f] = lbl.transform(list(X_train[f].values))
        X_test[f] = lbl.transform(list(X_test[f].values))
```

Step 4. Exploratory Data Analysis
Visualize our data using matplotlib or seaborn to find other insights to make and improve our predictions, then observe the results (This step can be seen in Part1).

Step 5. Build regression models using the training data
The task is the measure of whether the transaction card is fraud or not (is Fraud: 1, Not Fraud: 0).Our classifier will use this to know what the output should be for each of the training instances. We Create and fit the suitable model, which is XGBoost as the algorithm since it could handle the huge dataset. The first step is definitely going to over fit our data since we just set simple parameter.

Step 6. Evaluate models by using cross validation
Split labelled data and use K-Fold (k=5)



```
from sklearn.model_selection import KFold
n_fold = 5
folds = KFold(n_splits=n_fold, shuffle=True)

print(folds)
```

Step 7. Assess model on the test data
Submit csv and file to upload to Kaggle, then we obtain final output of the system.
Go to the submission section, we will see our score on the leader board. And then we need to come back to make some change improve.

2. Intermediary systems design

When come to this stage, we can return here and tweak parameters in this cell. For example, we could reduce the maximum depth of the tree or change the learning rate with the following (The other parameters are listed below).

Change 1: Delete the unrelated features.
Change 2: Try to change a max depth to improve my score.
Change 3: Use K-Fold cross-validation (k=5)
Change 4: Import a different models such as the LightGBM Classifier to compare with initial score.

After submitted to Kaggle twice subsequent, the score has not improved. On the contrary, it become lower.
The scores are 0.9366, 0.9320 and 0.9062(the another one is 0.9212, but submit fail). Compare the results of XGBoost and LightGBM (0.9366 and 0.9320), there is not a significant difference. However, Light GBM is very fast when compared to XGBOOST and is a much better approach when dealing with large datasets.

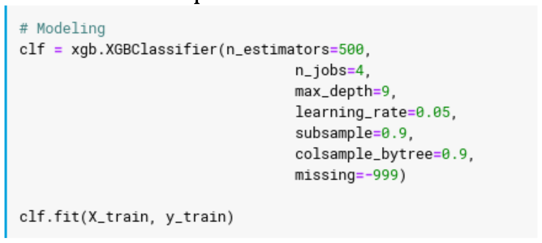
1. Delete the unrelated features

```
cols_to_drop = [col for col in train.columns if col not in useful_features]
```

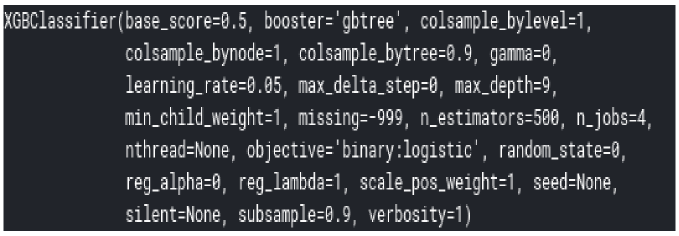
Create useful features based on the data explore. After tuning the parameter, the score on the leader board is lower (0.9366 to 0.9062).It is probably due to we remove some important features or there are some overfitting in the initial results.

2. Tweak parameters

The first round: parameter of XGBoost:

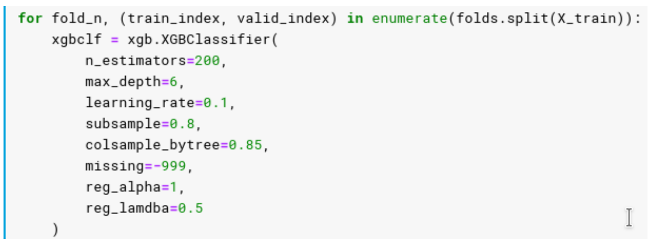


```
# Modeling
clf = xgb.XGBClassifier(n_estimators=500,
                        n_jobs=4,
                        max_depth=9,
                        learning_rate=0.05,
                        subsample=0.9,
                        colsample_bytree=0.9,
                        missing=-999)

clf.fit(X_train, y_train)
```

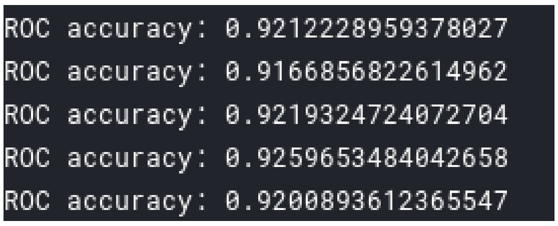


```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=0.9, gamma=0,
              learning_rate=0.05, max_delta_step=0, max_depth=9,
              min_child_weight=1, missing=-999, n_estimators=500, n_jobs=4,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=0.9, verbosity=1)
```

The second round: parameter of XGBoost:



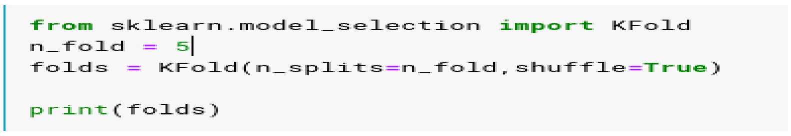
```
for fold_n, (train_index, valid_index) in enumerate(folds.split(X_train)):
    xgbclf = xgb.XGBClassifier(
        n_estimators=200,
        max_depth=6,
        learning_rate=0.1,
        subsample=0.8,
        colsample_bytree=0.85,
        missing=-999,
        reg_alpha=1,
        reg_lamdba=0.5
    )
```



```
ROC accuracy: 0.9212228959378027
ROC accuracy: 0.9166856822614962
ROC accuracy: 0.9219324724072704
ROC accuracy: 0.9259653484042658
ROC accuracy: 0.9200893612365547
```

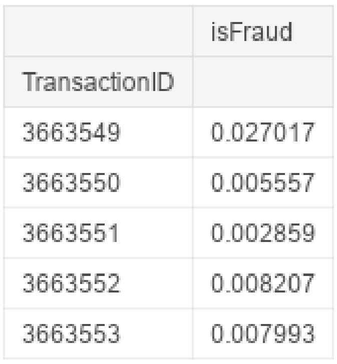
After tuning the parameter, the score on the leader board is lower (0.9366 to 0.9212). However, the execute time is less because we change the learning rate (from 0.05 to 0.1).

3. Use K-Fold cross validation



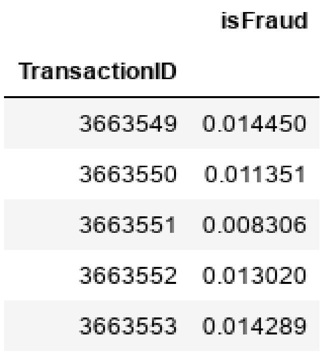
```
from sklearn.model_selection import KFold
n_fold = 5
folds = KFold(n_splits=n_fold,shuffle=True)

print(folds)
```

Before change:



	isFraud
TransactionID	
3663549	0.027017
3663550	0.005557
3663551	0.002859
3663552	0.008207
3663553	0.007993

After change:



	isFraud
TransactionID	
3663549	0.014450
3663550	0.011351
3663551	0.008306
3663552	0.013020
3663553	0.014289

We can see that most of the percentage of isFraud is higher, which could be the reason of the lower score. In the unbalanced dataset which contains mainly not fraud instances, the predict trend to not fraud. Thus, the improved percentage of probability predict will be regard as lower performance.

4. XGBoost VS Lightgbm

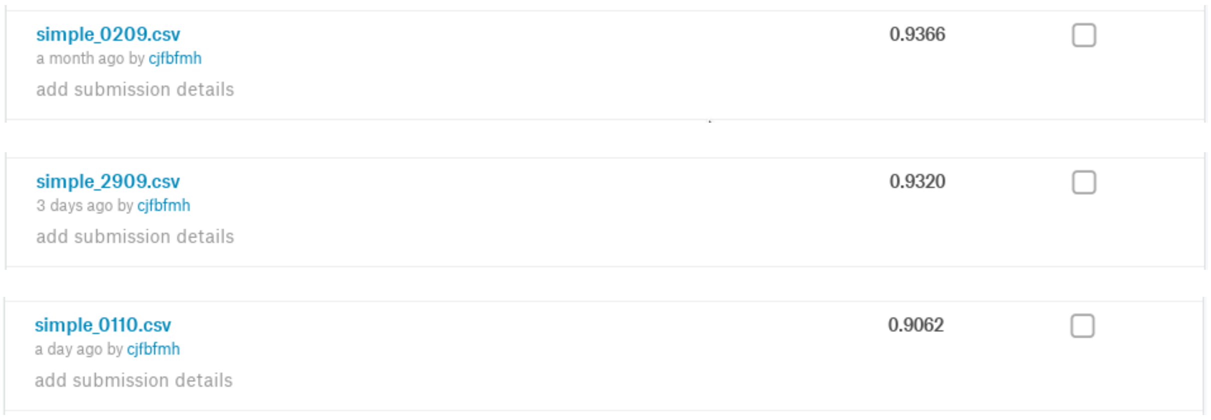
The dataset is a very sparse and huge data frame. Xgboost manages only numeric vectors. If there are categorical data, we will convert categorical variables to numeric one.
XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.
XGBoost improves upon the base Gradient Boosting Machines (GBMs) framework through systems optimization and algorithmic enhancements.
Regularization: It penalizes more complex models to prevent overfitting.
Sparsity Awareness: Automatically 'learning' best missing value depending on training loss and handles different types of sparsity patterns in the data more efficiently.
Cross-validation: built-in cross-validation method at each iteration.
The two reasons we use XGBoost are:
1) Execution Speed
XGBoost is really fast when compared to other implementations of gradient boosting.
2) Model Performance
XGBoost dominates structured or tabular datasets on classification and regression predictive modelling problems.
The evidence is that it is the go-to algorithm for competition winners on the Kaggle competitive data science platform (evidence from: https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions).

Light GBM is a fast, distributed, high-performance gradient boosting framework based on decision tree algorithm, used for ranking, classification and many other machine learning tasks.
Since it is based on decision tree algorithms, it splits the tree leaf wise with the best fit whereas other boosting algorithms split the tree depth wise or level wise rather than leaf-wise. The leaf-wise algorithm can reduce more loss than the level-wise algorithm and hence results in much better accuracy which can rarely be achieved by any of the existing boosting algorithms.

Faster training speed and higher efficiency
Lower memory usage: Replaces continuous values to discrete bins which result in lower memory usage (resource from: https://towardsdatascience.com/lightgbm-vs-xgboost-which-algorithm-win-the-race-1ff7dd4917d).

3. Choose the best system

The scores of the three submission are as follows:



simple_0209.csv
a month ago by cjfbfmh
add submission details
0.9366

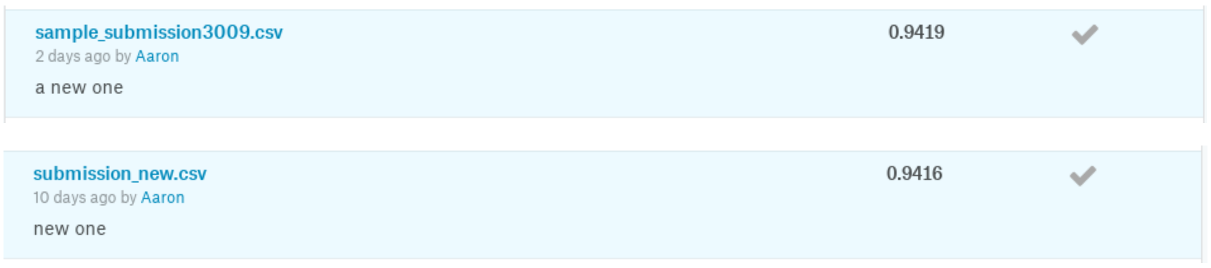
simple_2909.csv
3 days ago by cjfbfmh
add submission details
0.9320

simple_0110.csv
a day ago by cjfbfmh
add submission details
0.9062

The first is the initial XGBoost model, the second one is Lightgbm model. The last one is XGBoost after tweaking parameters.

In my models, it is better to choose the last model as the best one. Although it has lowest score than other model. The last one drops some not related feature, which is more reasonable.

Here are some shortcoming (have not done) in my model:
1. We may use other features that were dropped to improve our score since it is possible we miss some important data.
2 We may use GridSearchCV from sklearn.model_selection on the Classifier to tune the parameters and improve score.
Considering the competition, we choose Aaron's model to submit.



sample_submission3009.csv
2 days ago by Aaron
a new one
0.9419

submission_new.csv
10 days ago by Aaron
new one
0.9416

Part 3 Challenge: Reflecting on your findings

1. Team form

There are 3 people (Aaron, AMY61 and cjfbfmh) in our team (Team name: 228)
(My Kaggle username: compcjj   Display name: cjfbfmh)
Aaron and I study Comp309 (wrapped into Comp440), and AMY61 studied data science and want to do some research. The responsibility of us as follows:
1. AMY61 choose the common model: logistic model and Random Forest Regression.
2. Aaron and I (cjfbfmh) try to use XGBoost model to improve the prediction.

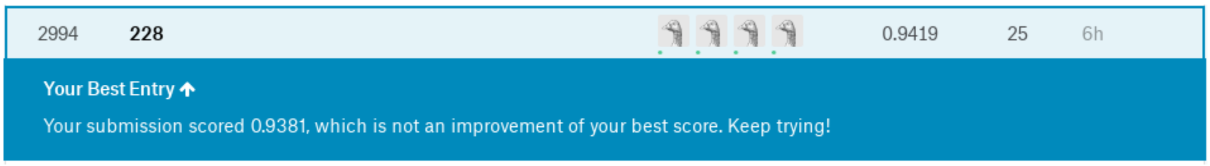
2. Aaron use Lightgbm model and make continuous optimization. While I choose to tweak the parameters on the XGBoost. Then we compare my results with Lightgbm used by Aaron's.
We discuss and continuously improve the result and refresh the score.

2. Benefit from team

Take the list from our team and then have the best models and compare the different prediction by other members.
In team cooperation, we help each other and make progress and contribution. This progress makes us trust teammates and share experiences so that we could obtain deeper understanding of different machine learning algorithm. After the team work, we improve the individual level and share team achievement.

3. Major difference

The major difference after the team formation is that we improve our ranking in the Leader board. Compared with XGB, Lightgbm has better performance, not only in higher score but also in less execute time.
Before team formed, the individual score around 2400 out of 4000.While after that, our team score ranks 2994 out of 6321.


2994 **228** 0.9419 25 6h

Your Best Entry ↑

Your submission scored 0.9381, which is not an improvement of your best score. Keep trying!