Capstone Project Milestone Report: Prediction for value of transaction for Potential customers

Problem Definition:

According to Epsilon research, 80% of customers are more likely to do business with you if you provide personalized service. Banking is no exception. The digitalization of everyday lives means that customers expect services to be delivered in a personalized and timely manner and often before they have even realized they need the service. Santander Group aims to go a step beyond recognizing that there is a need to provide a customer a financial service and intends to determine the amount or value of the customer's transaction. This means anticipating customer needs in a more concrete, but also simple and personal way. With so many choices for financial services, this need is greater now than ever before.

In this project, I aim to help Santander Group identify the value of transactions for each potential customer. This is a first step that helps Santander to nail in order to personalize their services at scale.

Data Wrangling:

The dataset is from Bank Santander, a Spanish universal bank, and can be downloaded from this site. The data contains two part, training data and testing data. In training data, we have a data with 4993 columns and a sample of 4459 rows. In testing data, we have the same features with 49342 rows.

```
Training set:

Number of Records: 4459

Number of Features: 4993


Testing set:

Number of Records: 49342

Number of Features: 4992
```

At the first glance of data, people would notice that it is extreme high dimensional. The data has almost 5000 columns but only has under 4500 rows. So, we must find a way to reduce the dimension to avoid curse of dimensionality. However, the data is totally anonymous due to protection of clients privacy. Thus, we can't interpret any useful information from feature names. In addition, there are sparsity in some columns. These columns are populated with large portion of zeros, but target values are all above zero. On the other hand, one good thing is that the data doesn't contain any missing values.

To sum up, although there are no missing values, we need to implement a dimension reduction algorithm because of the high dimensional anonymous data. The detailed data explore analysis on training set will be present in the next session.

Data Explore Analysis:

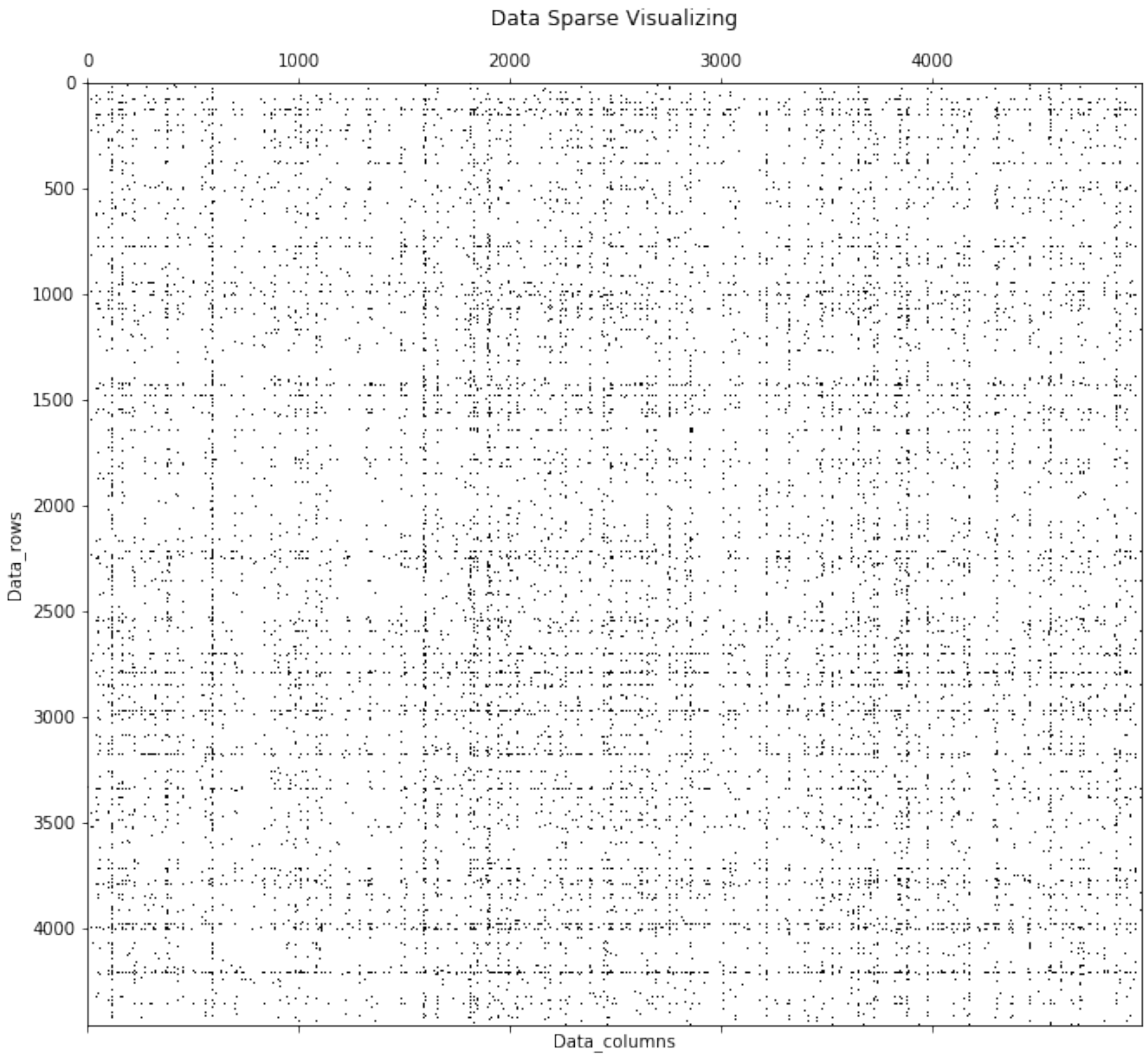
Basic Data information:

The basic understanding of the data, which is showed as follows. The one object type column is column 'ID' and all other columns are numeric.

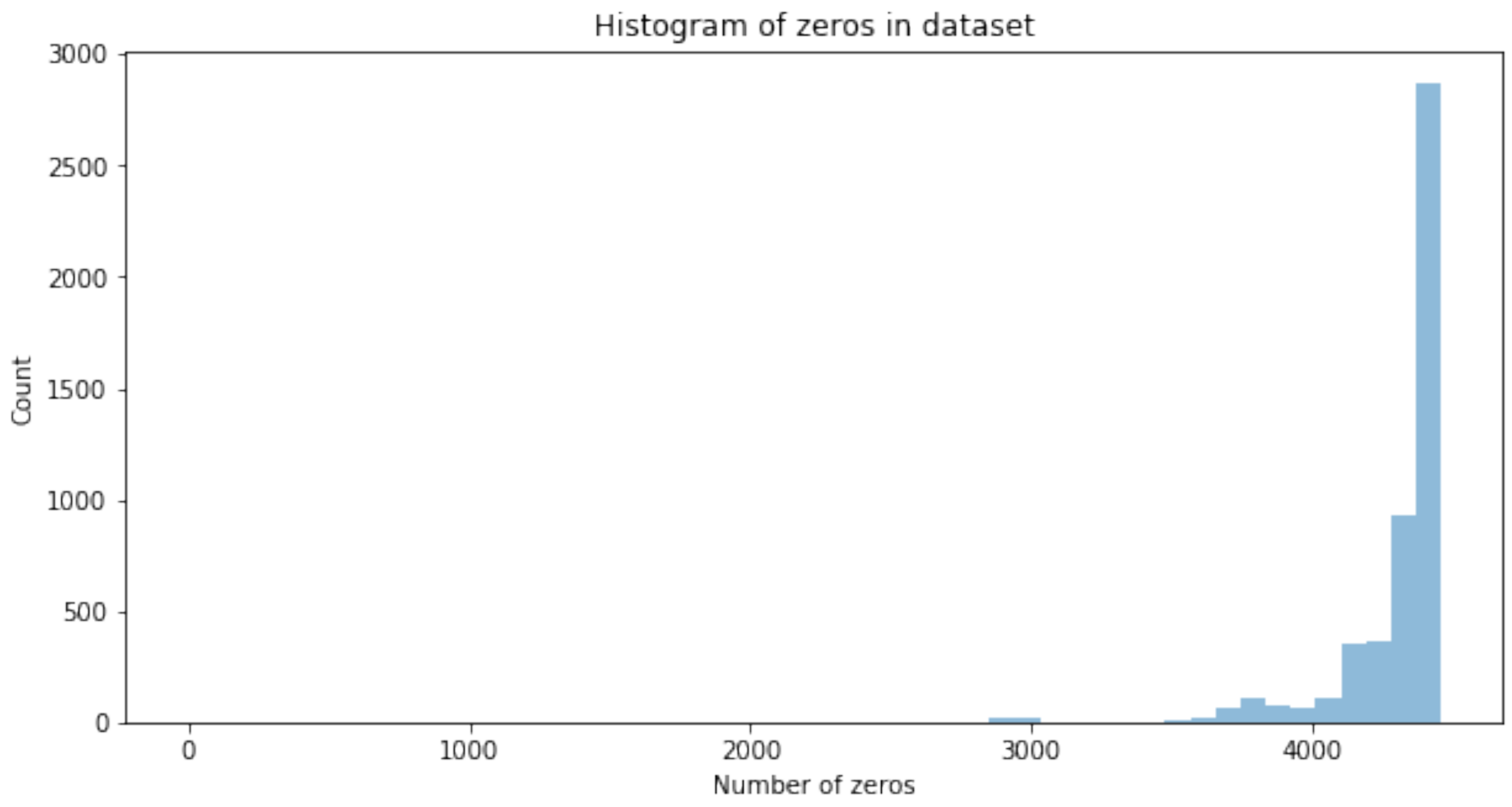
	Column Type	Count
0	int64	3147
1	float64	1845
2	object	1

Sparsity check:

The dataset itself looks sparse. Below is the sparse visualizing of training set('ID' and 'target' columns are removed)



The histogram below shows that there are over 2500 columns are populated with more than 4000 of zeros.



Variance Check:

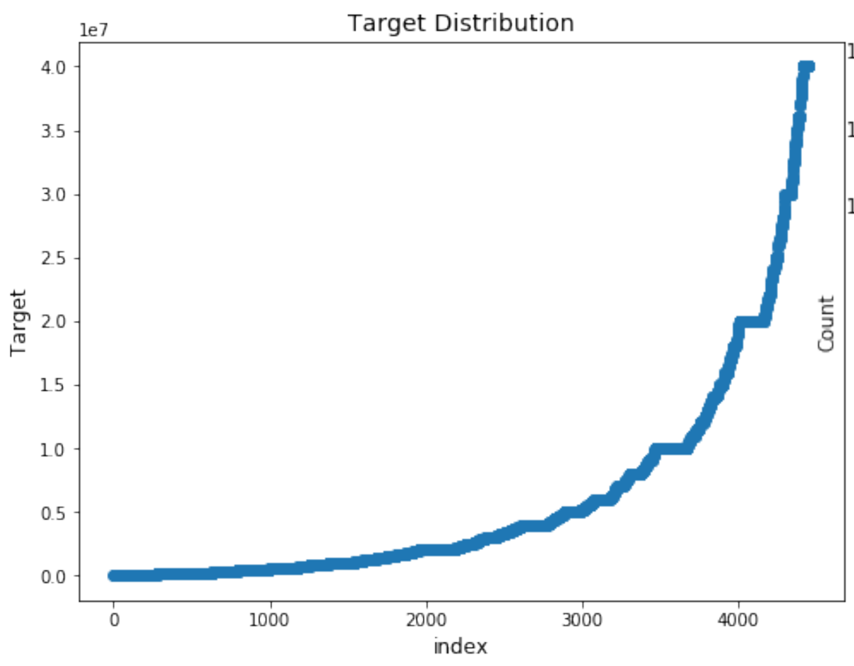
As showed in the table below, There are 256 columns are populated only with zero. I don't think I can get useful information from them, so I will just remove them.

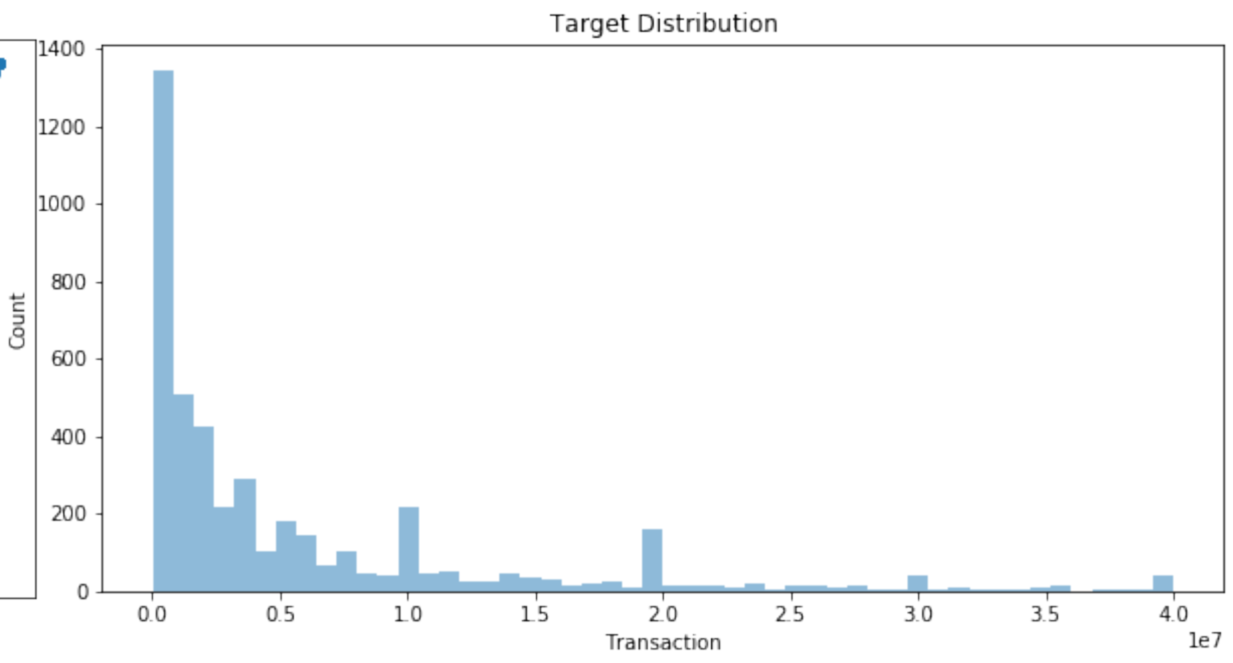
```
       col_name  unique_count value
28    d5308d8bc             1     0
35    c330f1a67             1     0
38    eeac16933             1     0
...         ...           ...   ...

[256 rows x 3 columns]
```

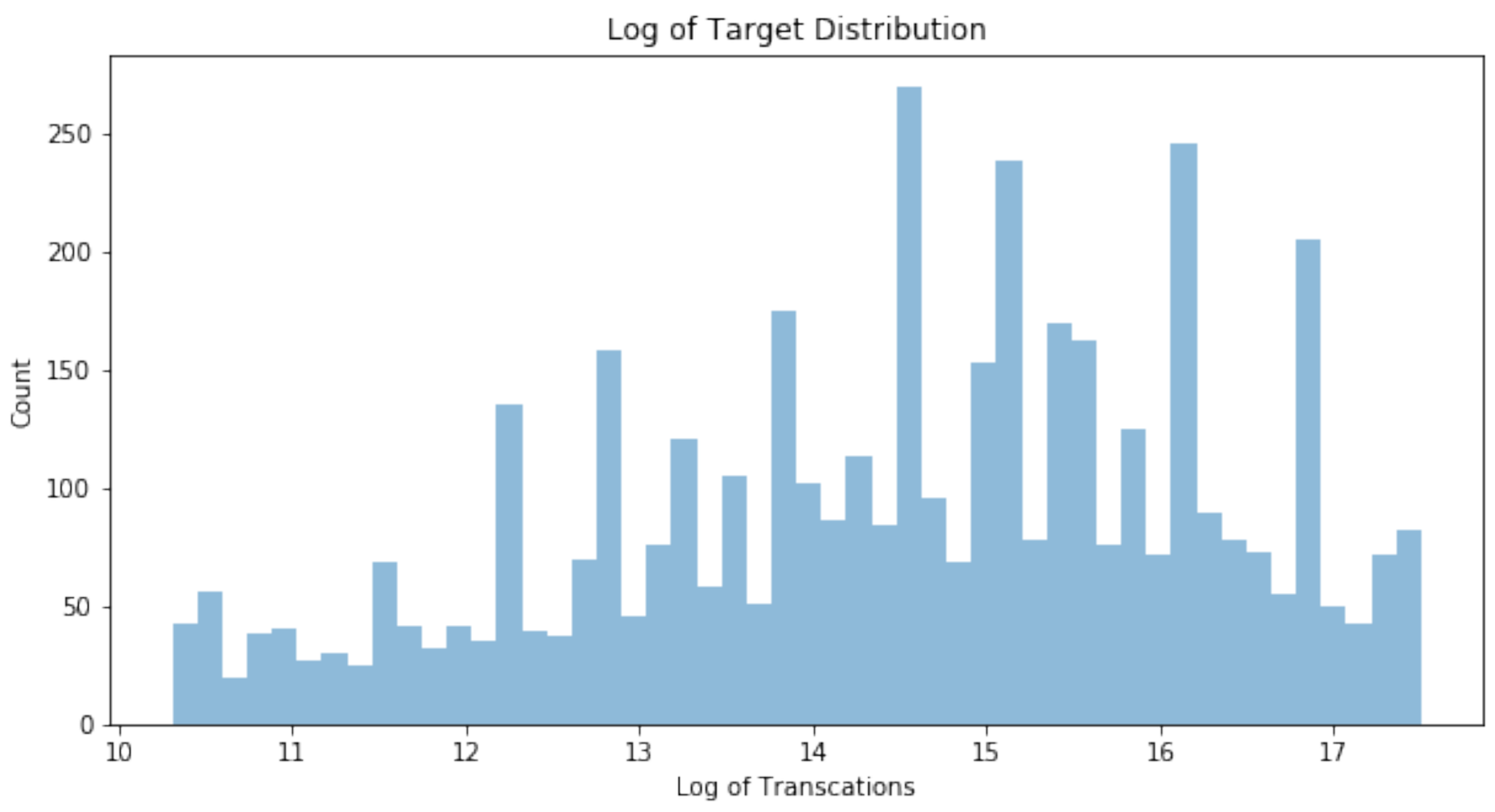
Target Value Outlier & Distribution Check:

So far so good for cleaning features data, target value needs to be taken care for the next. From the figures below, there are no obvious outlier in target value. However, the target values are obviously right skewed.



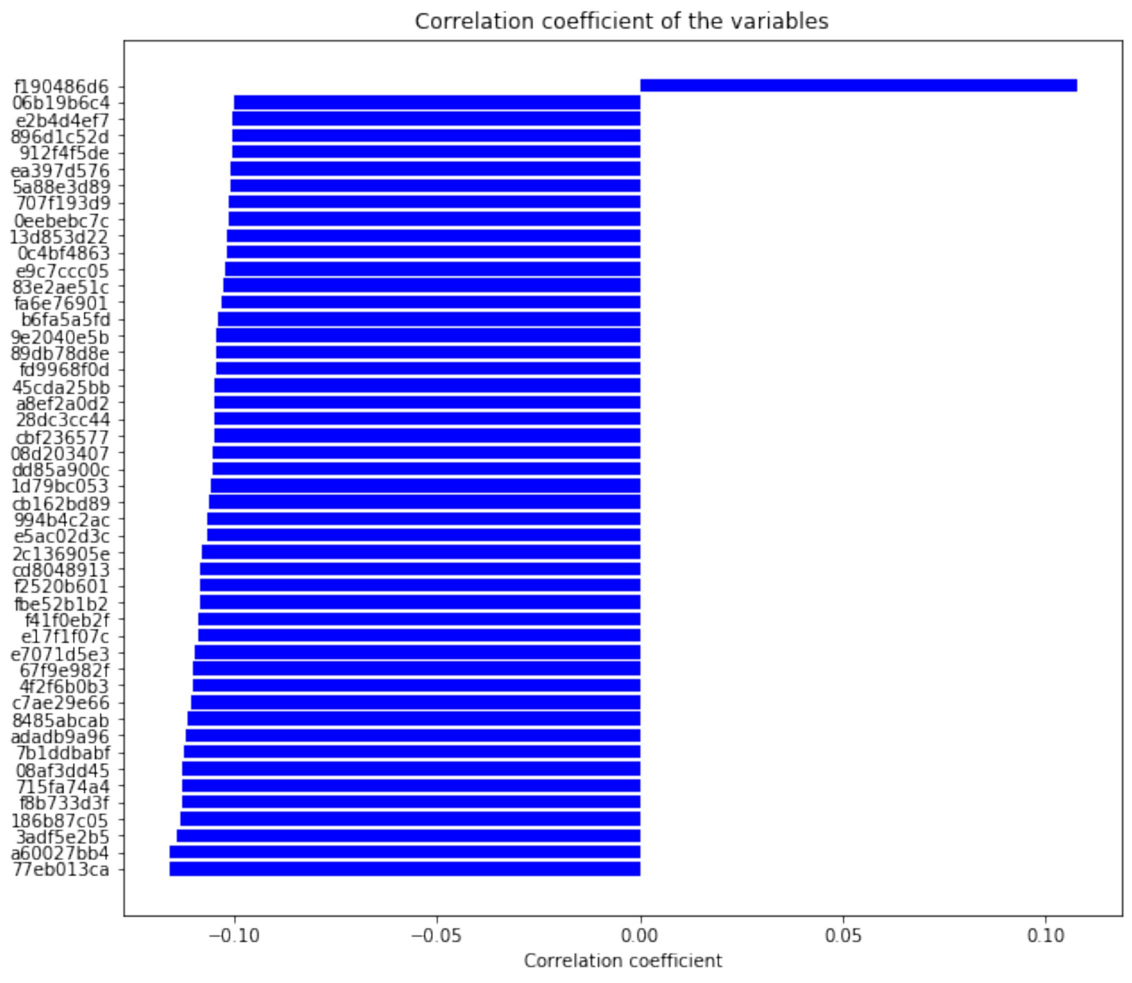


A Log transformation has been deployed to target values to fix the skewness, result shows as follow. It is not ideal normal shape but is much better than the original data.



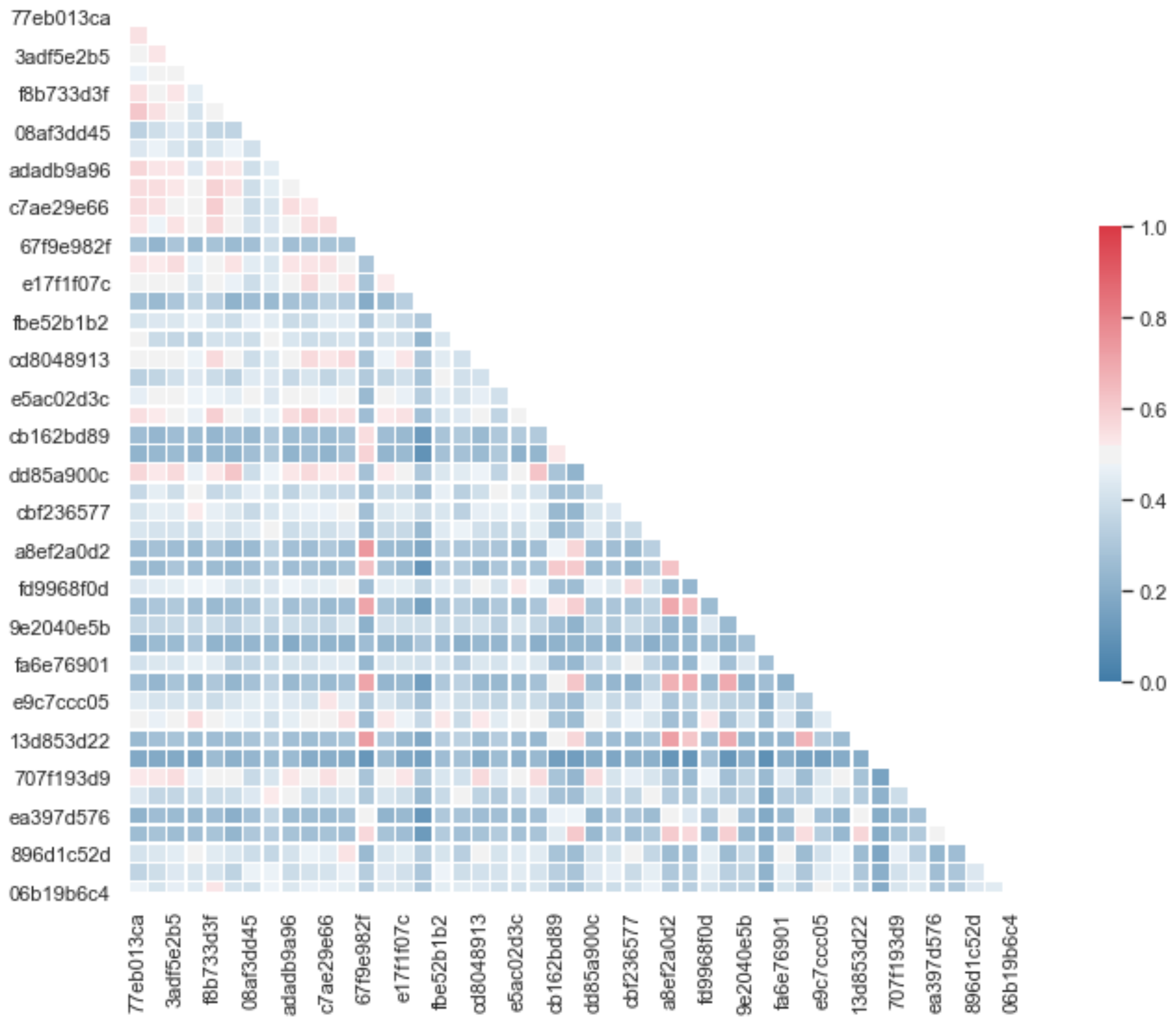
__Target Correlation Analysis:__

Since most features have high level of disparity, linear correlation is not suitable in this case. Instead of using Pearson linear correlation, Spearman correlation is a good idea. Because Spearman is computed on ranks and so depicts monotonic relationships while Pearson is on true values and depicts linear relationships. There are thousands of variables and so plotting all of them will give us a cluttered plot, thus only those variables whose absolute spearman correlation coefficient is more than 0.1 are plotted in the below figure. The minimal correlation coefficient is equal to -0.116095, maximal coefficient is equal to 0.1076785. Thus, there is no strong association between target variable and features.



Collinearity Analysis:

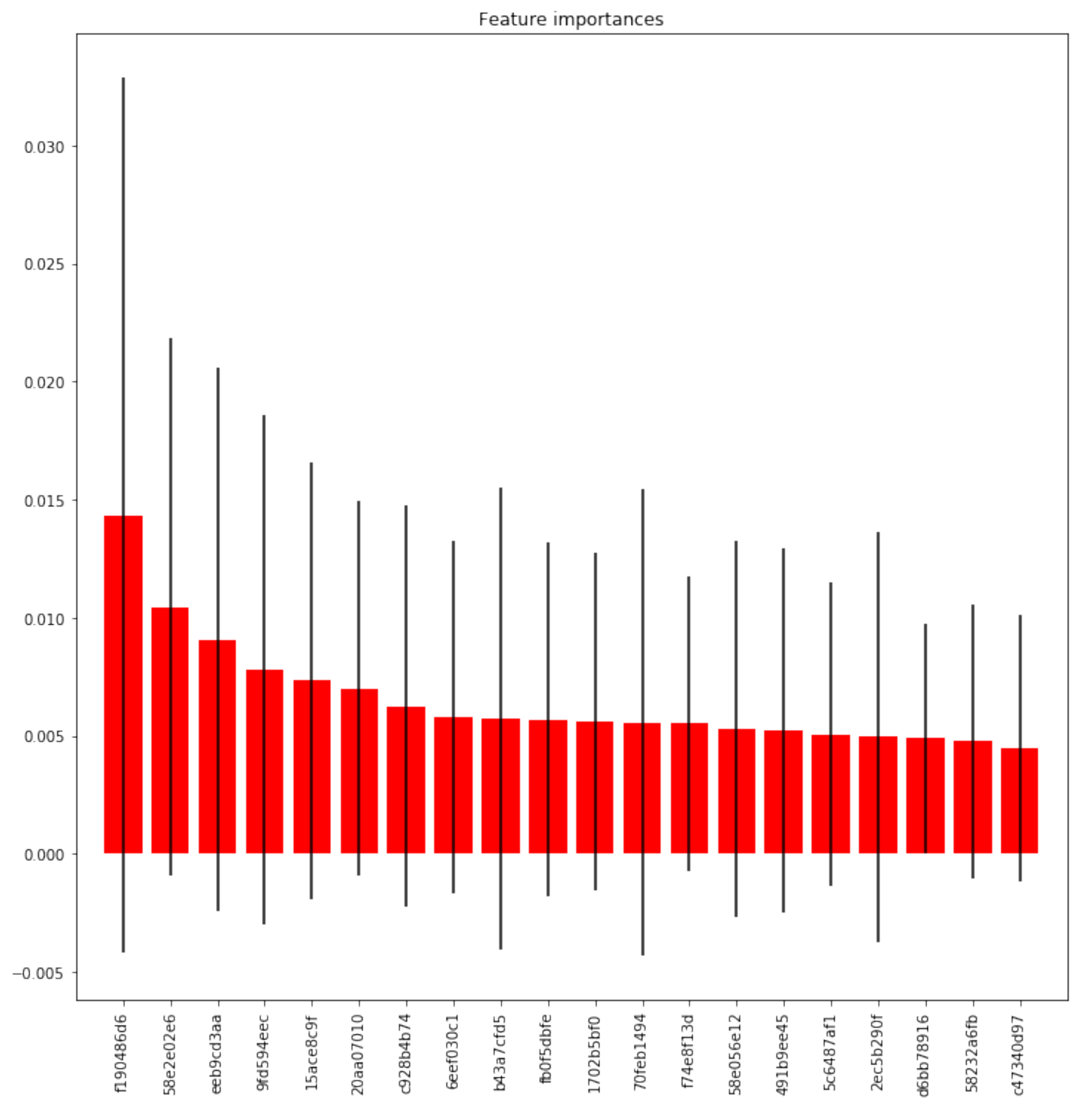
Collinearity among features is also a worth-explore point. The figure below can show that some features have high correlation with each other. Based on the data non-linearity and high collinearity among some features, tree based machine learning algorithm should be a fit for this value prediction project. Since there are thousands of features available, the dimension reduction is the next step, which will be explained in detail in the feature engineering session.



Feature Engineering:

Because of the extreme high dimensional data, a good feature engineering method is essential to the success of this prediction project. PCA algorithm has been tried but the result is not promising, no single feature can largely explain the variance of the data. The detailed result is in appendix for report completeness.

Random forest algorithm consists of a number of decision trees. Every node in the decision trees is a condition on a single feature, designed to split the dataset into two so that similar response values end up in the same set. The measure based on which the (locally) optimal condition is chosen is called impurity, for regression trees it is variance. Thus when training a tree, it can be computed how much each feature decreases the weighted impurity in a tree. For a forest, the impurity decrease from each feature can be averaged and the features are ranked according to this measure. This is the feature selection by random forest algorithm and it shows relatively good result in our case. Total 277 features are selected based on the embedded feature importance function and the top part of which is showed in the below figure. The details will be explain in the model implement session.



Model Implementation Baseline:

Data split: We have a training set as well as a testing set. But the testing set doesn't come with the target value. Thus, all models implemented here are trained in the training set only. And for every train, 20 % of training set is randomly split as a hold-out set. All the parameter tune and cross validation only implemented in the 80% training set, I will use name training set refer to it in the rest of this report. The hold out set will serve as 'new data' to evaluate the performance.

Hyper-parameter tune: Since a lot of parameters needed to be tuned, use grid search directly has huge computational cost, thus two methods of tuning are implemented according to different models.

a). Combination of random search and grid search

1). Set a relatively wide range of parameter space

2). Set number of total training times, random search would randomly pick combination from parameter space until total number reached.

3). Fitting and hyper-tune model

4). 5-fold cross validation with RMSE as score method is implemented to give best parameter combination

5). Start from best parameter combination from random search, set a relatively narrow range of parameter space.

6). Repeat 3 & 4.

b). Bayesian optimization:

1). Build a surrogate probability model of the objective function

2). Find the hyperparameters that perform best on the surrogate

3). Apply these hyper-parameters to the true objective function

4). Update the surrogate model incorporating the new results

5). Repeat steps 2–4 until max iterations or time is reached

Thanks to the python package hyperopt, I don't need to write script to do this procedure from scratch.

Result Analysis:

Model base line: Use the data split stated above, and all the valid features after EDA, a random forest model with default parameters in Sklearn library is trained and the result on hold-out set is as follows.

```
Model Performance
Hold-out set RMSE = 1.4671
* Model are evaluated by the 20 % hold-out part
```

Random forest: Same as the base model, the difference is that this random forest model are tuned to have a better performa

The hyper-parameter tune method used in this model is a) combination of random search and grid search. The key parameter are as follows,

Parameters	**Tuned Value**
max_depth	None
max_features	Sqrt
min_samples_leaf	2
n_estimators	200

* Since we have a large number of features, the maximum depth of this random forest are much deeper than usual model, so lift up the limitation on it gives best result.

```
Model Performance
Hold-out set RMSE = 1.4077
* Model are evaluated by the 20 % hold-out part
```

XGBoost: Same as the base model, the model used 20% of training set are hold out as validation set. From the random forest model, features are signed with an importance attribute which derived from how much purity decrease without it. Feature with higher than average importance are selected to fed into this XGBoost model. The hyper-parameter tune method used is b) Bayesian Optimization. The key parameter are as follows,

Parameters	**Tuned Value**
objective	reg:squarederror
learning_rate	0.05
max_depth	10
n_estimators	177

```
Model Performance
Hold-out set RMSE = 1.4077
```

Light GBM: Same as the XGBoost model, the model used 20% of training set are hold out as validation set. From the random forest model, features are signed with an importance attribute which derived from how much purity decrease without it. Feature with higher than average importance are selected to fed into this light GBM model. The hyper-parameter tune method used is b) Bayesian Optimization. The key parameter are as follows,

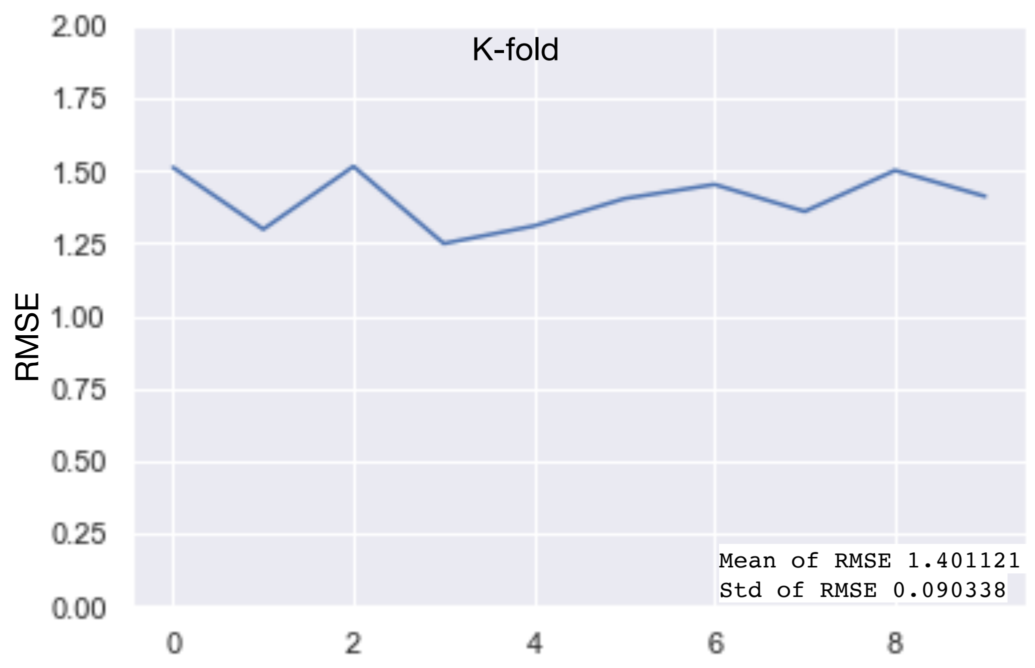
Parameters	**Tuned Value**
objective	regression
learning_rate	0.08
max_depth	18

```
Model Performance
Hold-out set RMSE = 1.4198

* Model are evaluated by the 20 % hold-out part
```

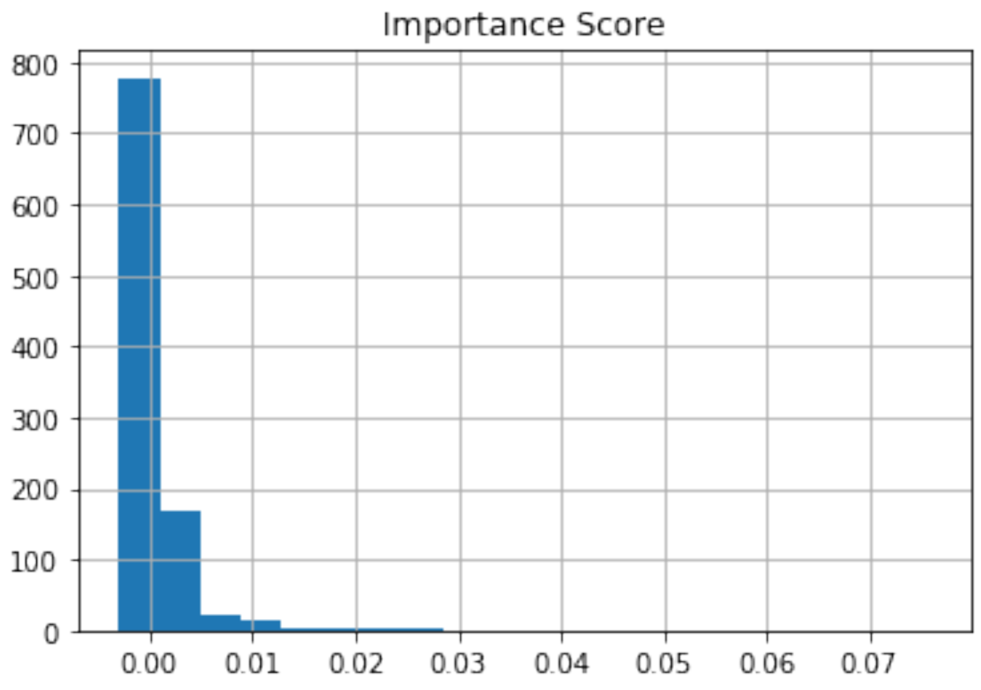
```
Combining Three Tree based model:
```

Comparing to the baseline set up by the naive (un-tuned) Random forest model, our three models don't have a significantly improved. Combining all three model in a weight average manner, which means that add up weighted perditions from the three model, gives a result of 1.3810, slightly improved but still not satisfying. A 10-fold cross validation is also implemented, results are as follows.



Further analysis:

Based on the prediction from above, the feature engineering seems not very helpful. The permutation feature is a model inspection technique, can inspect the feature importance in a different perspective. The permutation feature importance is defined to be the decrease in a model score when a single feature value is randomly shuffled. This procedure breaks the relationship between the feature and the target, thus the drop in the model score is indicative of how much the model depends on the feature. Use this method to scan the feature importance of Random forest model, the result is show in the below figure. It is clear that most features has no effect to the model, some features are even slightly less than 0. This means that further data explore and feature engineering are needed. This part of work will be discussed in the next session.



Further Data Explore Analysis — Data Pattern reveal:

Since the data consists a large number of zeros, it is a worth probing direction. Sorted data with number of zeros from columns and rows, a pattern can be revealed. With some manually adjustment, a time-series relationship among certain features and target can be confirmed. A simple example is showed as follows.

ID	target	f190486d6	58e2e02e6	9fd594eec	eeb9cd3aa	6eef030c1	15ace8c9f	fb0f5dbfe	20aa07010
6726fff18	115636.36	1015000.00	1563411.76	1563411.76	1563411.76	1563411.76	1563411.76	1563411.76	1563411.76
d94655f86	834800.00	540000.00	1015000.00	1563411.76	1563411.76	1563411.76	1563411.76	1563411.76	1563411.76
1df3ca92e	296444.44	115636.36	540000.00	1563411.76	1015000.00	1563411.76	1563411.76	1563411.76	1563411.76
d8e48b069	247166.66	834800.00	115636.36	1015000.00	540000.00	1563411.76	1563411.76	1563411.76	1563411.76
24204cd10	550000.00	296444.44	834800.00	540000.00	115636.36	1015000.00	1563411.76	1563411.76	1563411.76

bf6c2d1ef	3076666.66	247166.66	296444.44	115636.36	834800.00	540000.00	1015000.00	1563411.76	1563411.76
eacc7ab9e	440000.00	550000.00	247166.66	834800.00	296444.44	115636.36	540000.00	1015000.00	1563411.76
4a5425356	1600000.00	440000.00	3076666.66	247166.66	550000.00	296444.44	834800.00	115636.36	1015000.00

This is the starting point, the next steps is to find more group of patterns like this. A simple brute force algorithm is implemented and 100 groups are found in training and testing set. In the training set, 3886 out of 4459 records are with this time series pattern and 7830 out of 49342 records can be found with this pattern in testing set. To explain it in more detail, the 100 groups all have at least 40 features which have the time series pattern as the above table in certain rows. This means that the target values in the rows which have this pattern can be predict using feature values directly, because the features are the time series lag value to the target.

To improve the model prediction ability, finding more pattern groups is one way, engineering further on features is another way. Finding more groups may required algorithm more effective than brute force, I will try that in the future. In the next session, further feature engineering is discussed.

Feature Engineering with Discovered Data Pattern:

In training set, 3886 out of 4459 rows can be predicted directly by data pattern. However, this project's ultimate goal is to predict target value in testing set which only has 7830 out of 49342 rows can be predicted by data pattern.

Thus, a way to leverage the information from data pattern without direct prediction is needed. From the data pattern revealed in the Further Data Explore Analysis session, the features meaning is also revealed, which is nothing but former time series data of the target value. Based on this information, row based aggregate features are more suitable to leverage data pattern information. Since 100 groups are found and each of them has at least 40 columns, most columns in the data are covered.

Compared to simply apply aggregation function row-wise, which is basically apply to all columns in data. It is better to divide features into several partitions. In this way, we can better leverage the information from the data pattern. The partition I chose is as follows,

Partition	**Explanation**
All features	All columns in the data except ID and Target
Features not in any pattern group	The columns not found in any pattern group
Features per pattern group	For each pattern group, apply row-wise aggregate function to the 40 columns in it. 100 groups in total
Time features	For each group, there are at least 40 time-series features, this partition are built by the features on the same time lag. That means we have 100 features in a group and has 40 groups.
Features in pattern group	All the features are found in pattern groups

The aggregate function used are the common statistics function like max, min, mean, median, sum, std, first_non_zero and last_non_zero. For each of the partition in the above table, these aggregate functions are applied to rows of the columns in it. More functions can be tried here, but for now, these functions have caught enough information from the data and are fed into our final model which is discussed in the next session.

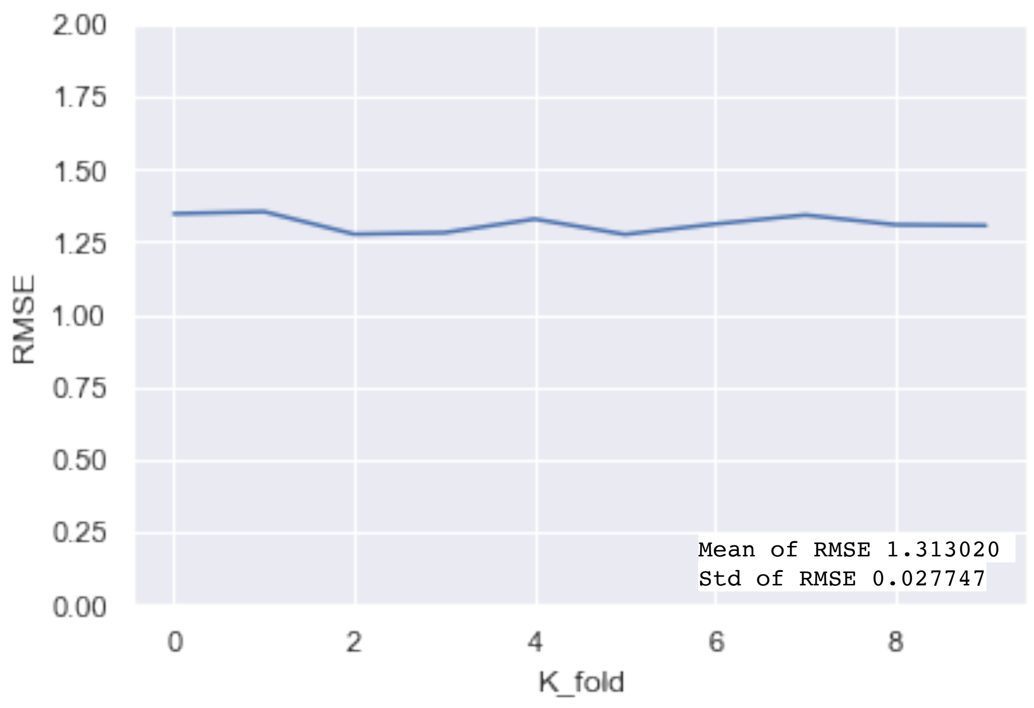
Final Model Implementation and Result:

Since there are 7830 out of 49342 records can be found in data pattern, it will be better to make training data from the original training plus the pattern in test data. So now we have 12289 rows in training set and 1287 features. 20 % of the new training set is split as hold-out set.

With the new aggregate features, one light GBM model is trained with the data split strategy mentioned above. Other two models trained in the above session are not necessary because the result of this single model is good enough. The hyper-parameter tune method used is b) Bayesian Optimization and the key parameters are as follows,

Parameters	**Tuned Value**
objective	regression
learning_rate	0.05
max_depth	-1

A 10-fold cross validation is also implemented, results are as follows,



Compared to the base model with a RMSE of 1.4671, the Light GBM model, has a mean RMSE of 1.313, with aggregate features has a significant improvement in prediction ability.
Now move to the final goal of this project, predict the target values in the testing set. A submission has been made to Kaggle and the result is as follows,

```
Model Performance
RMSE = 0.48123
* The model is evaluated in the actual testing set.
```

Conclusion:

The outcome of this project provides strong prediction ability to identify the value of transactions for each potential customer. In this project, I first did data explore analysis in a traditional correlation based method. Although some insight can be found by this method, the machine learning models built based on it doesn't provide a satisfied result. With further data explore, especially more focus on the data itself, I successfully revealed the pattern behind zeros and anonymous feature names. With the basically identical method of building the machine learning models, correct data explore makes a huge difference of prediction power. It may be a cliché, but this project shows again how important the data explore can be and we should always analyze the data itself as much as we can before jump into all the advance techniques.

future steps:

Based on the data patterns found in the data, I think each group is associated with one or certain clients. In this project, I only looked for the group with at least 40 timestamp long. However, there are more similar groups to search and shorter timestamp groups may also be very useful.

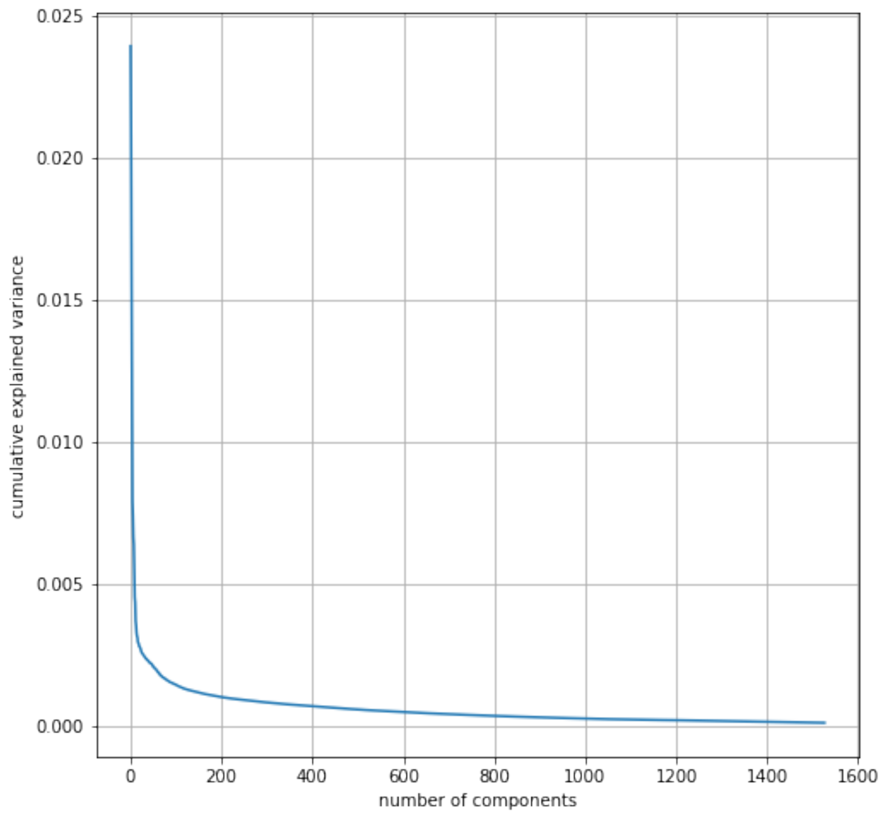
Besides, Santander's final goal is to provide personalized service to all clients, so instead of building one model and combining the data pattern predictions, it would be more logical to build prediction model for each client identified by the data pattern.

Appendix:

A PCA data transformation algorithm is implemented with 0.95 total variance explained. The result is showed in the below figure, the maximum explained variance for a single principle component is less than 0.025 and the most principal components have explained variance less than 0.005. This means that the PAC algorithm doesn't have a good performance on our training data.

In our case, the data has large sparsity and populated with large portions of zeros, which may led the PCA components all have similar variance. As a result, there may be no 'good' universal stopping rule to discard some exact 'k' Principal Components and this could make PCA fails to perform.

Another reason may lead to the failure of PAC algorithm is linearity. PCA assumes the subspace of useful data is linear, but there is no significant linearity in our data.


cumulative explained variance
number of components