1. Definition:

1.1. Project Overview:

Supervised learning is the machine learning task of inferring a function from labeled training data. The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

Nowadays, machine learning has been used vastly in insurance companies in order to recommend decent contracts to their clients based on the past information of them. For example, buying a new car and its insurance need a precise evaluation on both side client and insurance company to make a decent contract. Nothing ruins the thrill of buying a brand new car more quickly than seeing your new insurance bill. The sting's even more painful when you know you're a good driver. It doesn't seem fair that you have to pay so much if you've been cautious on the road for years. Therefore, having a robust predictive model is a key factor for successful companies.

Fig. 1 shows the road map of the project. The project includes four main parts: data analysis, feature engineering, modeling and prediction. In the data analysis part, we applied several statistical techniques to investigate the data in order to get better understanding about our data and prepare a good sample for our analysis. In the feature engineering section, we will deal with missing data, creating interaction variables, making new features by frequency encoding and binary encoding, calculating feature importance and finally select the importance feature and to reduce the size of the features. In the modeling part we will use several machine learning algorithms and stack them in three levels to make a prediction. Finally, by the prediction is made of the mean of the two last prediction models.

The main file that we can run the project is: *project.ipynb*

1.2. Problem- Statement

Porto Seguro, one of Brazil's largest auto and homeowner insurance companies. Inaccuracies in car insurance company's claim predictions raise the cost of insurance for good drivers and reduce the price for bad ones. The goal of this project is to build a model that predicts the probability that a driver will initiate an auto insurance claim in the next year.

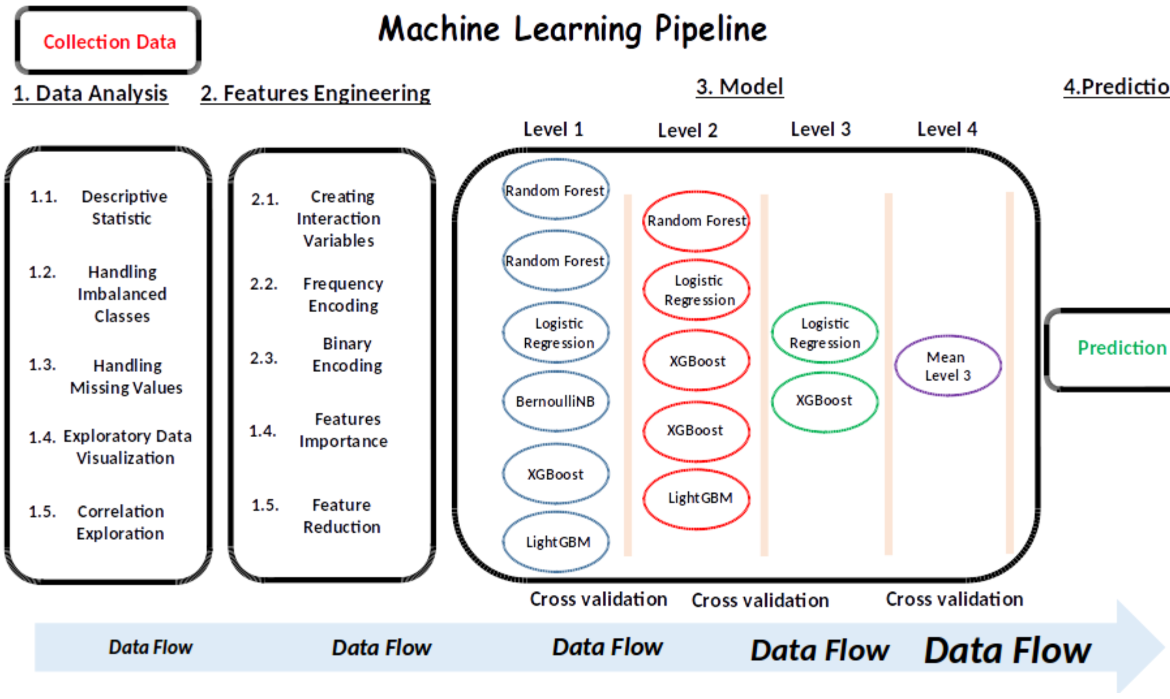


Fig.1. the road map of the project.

An accurate prediction will allow them to further tailor their prices, and hopefully make auto insurance coverage more accessible to more drivers. Thus, I will predict the probability that an auto insurance policy holder files a claim. The dataset is available dataset.

1.3. Metrics

code: crossValidation/gini.py

Our problem is binary classification. For binary classifiers, this option reports the AUC (area under curve) and Gini coefficient evaluation metrics. Both of these evaluation metrics are calculated together for each binary model. The values of the metrics are reported in a table in the analysis output browser.

The AUC evaluation metric is calculated as the area under an ROC (receiver operator characteristic) curve, and is a scalar representation of the expected performance of a classifier. The AUC is always between 0 and 1, with a higher number representing a better classifier. A diagonal ROC curve between the coordinates (0,0) and (1,1) represents a random classifier, and has an AUC of 0.5. Therefore, a realistic classifier will not have and AUC of less than 0.5.

The Gini coefficient evaluation metric is sometimes used as an alternative evaluation metric to the AUC, and the two measures are closely related. The Gini coefficient is calculated as twice the area between the ROC curve and the diagonal, or as Gini = 2AUC - 1. The Gini coefficient is always between 0 and 1, with a higher number representing a better classifier. The Gini coefficient is negative in the unlikely event that the ROC curve is below the diagonal.

The Gini coefficient measures the inequality among values of a frequency distribution (for example, levels of income). A Gini coefficient of zero expresses perfect equality, where all values are the same (for example, where everyone has the same income). A Gini coefficient of 1 (or 100%) expresses maximal inequality among values (e.g., for a large number of people, where only one person has all the income or consumption, and all others have none, the Gini coefficient will be very nearly one). However, a value greater than one may occur if some persons represent negative contribution to the total (for example, having negative income or wealth).

The metric used for this Kaggle competition is the Normalized Gini Coefficient. During scoring, observations are sorted from the largest to the smallest predictions. Predictions are only used for ordering observations; therefore, the relative magnitude of the predictions are not used during scoring. The scoring algorithm then compares the cumulative proportion of positive class observations to a theoretical uniform proportion.

2. Analysis:

2.1. Data Exploration

The dataset contains 2 .csv files with information necessary to make a prediction. In the train and test data, features that belong to similar groupings are tagged as such in the feature names (e.g., ind, reg, car, calc). In addition, feature names include the postfix bin to indicate binary features and cat to indicate categorical features. Features without these designations are either continuous or ordinal. Values of -1 indicate that the feature was missing from the observation. The shape of train is: (595212, 59) and memory usage: 267.9 MB and test set shape is (892816, 28). The column id shows the 'id' od customers and the target column is 'target'. There is no NaN in the data set.

2.1.1 Meta Data:

Analysis/featureAnalysis/fetureAnalysis.py

To facilitate the data management, we'll store meta-information about the variables in a data frame. This will be helpful when we want to select specific variables for analysis, visualization, modeling, etc.. We can also apply the describe method on the data frame. However, it doesn't make much sense to calculate the mean, std, etc. on categorical variables and the id variable. We'll explore the categorical variables visually later. Thanks to our meta file we can easily select the variables on which we want to compute the descriptive statistics. To keep things clear, we'll do this per data type.

The meta data frame includes this information:

Feature, type of feature, number unique, number of duplicate, 1st frequency of data, number of 1st frequency of data, 2nd frequency of data, number of 2nd frequency of data, 3rd frequency of data, number of 3rd frequency of data, mean, standard deviation, min, max, 25%, 50%, 75% of the data information. If the data does not include these info, the NaN is replaced in the data frame cell. Thus, based on the analysis, the 'id' is unique and the target is classified to 0 or 1. Therefore, the problem is binary classification. The brief summary of the this analysis is:

reg variables:

- only ps_reg_03 has missing values

- the range (min to max) differs between the variables. We could apply scaling (e.g. StandardScaler), but it depends on the classifier we will want to use.

car variables:

- ps_car_12 and ps_car_15 have missing values

- again, the range differs and we could apply scaling.

calc variables:

- no missing values

- this seems to be some kind of ratio as the maximum is 0.9

- all three _calc variables have very similar distributions

Ordinal variables:

- Only one missing variable: ps_car_11

- We could apply scaling to deal with the different ranges

Binary variables :

- A priori in the train data is 3.645%, which is strongly imbalanced.

- From the means we can conclude that for most variables the value is zero in most cases.

Target :

- A priori in the train data is 3.645%, which is strongly imbalanced.

- From the means we can conclude that for most variables the value is zero in most cases.

2.2. Exploratory Visualization

The dataset includes binary, categorical and numerical features. The plots of all figures are shown in the Jupyter Notebook. The binary and categorical features are plotted by histogram methods. The numerical features are plot by histogram and boxplot methods in order to find out their distributions and the outliers. However, the dataset does not suffer from outliers problem. The correlation of numerical variables are obtained and plot by the heat map plot. Besides, the relation between numerical data is obtain by the seaborn linear regression library package for the target.

The target distribution is shown on Fig. 2. It can be seen that the target has imbalance data.

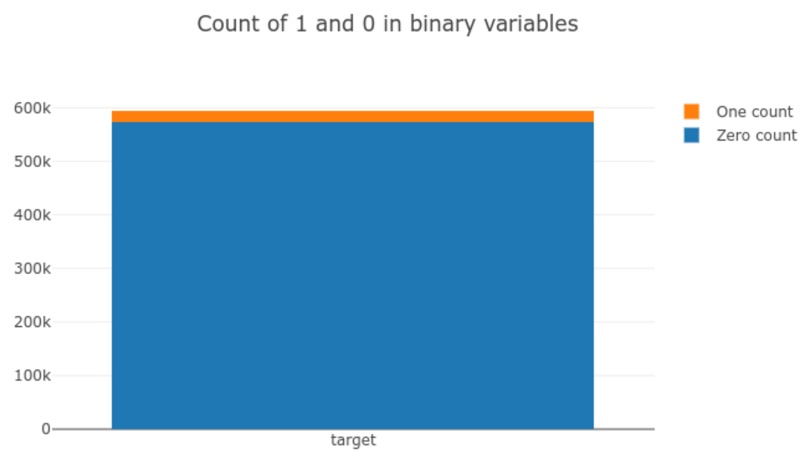


Fig.2: target distribution.

Hence, we need to treat the imbalance target in the project. The distribution of target in binary features is shown Fig.3 after sampling.

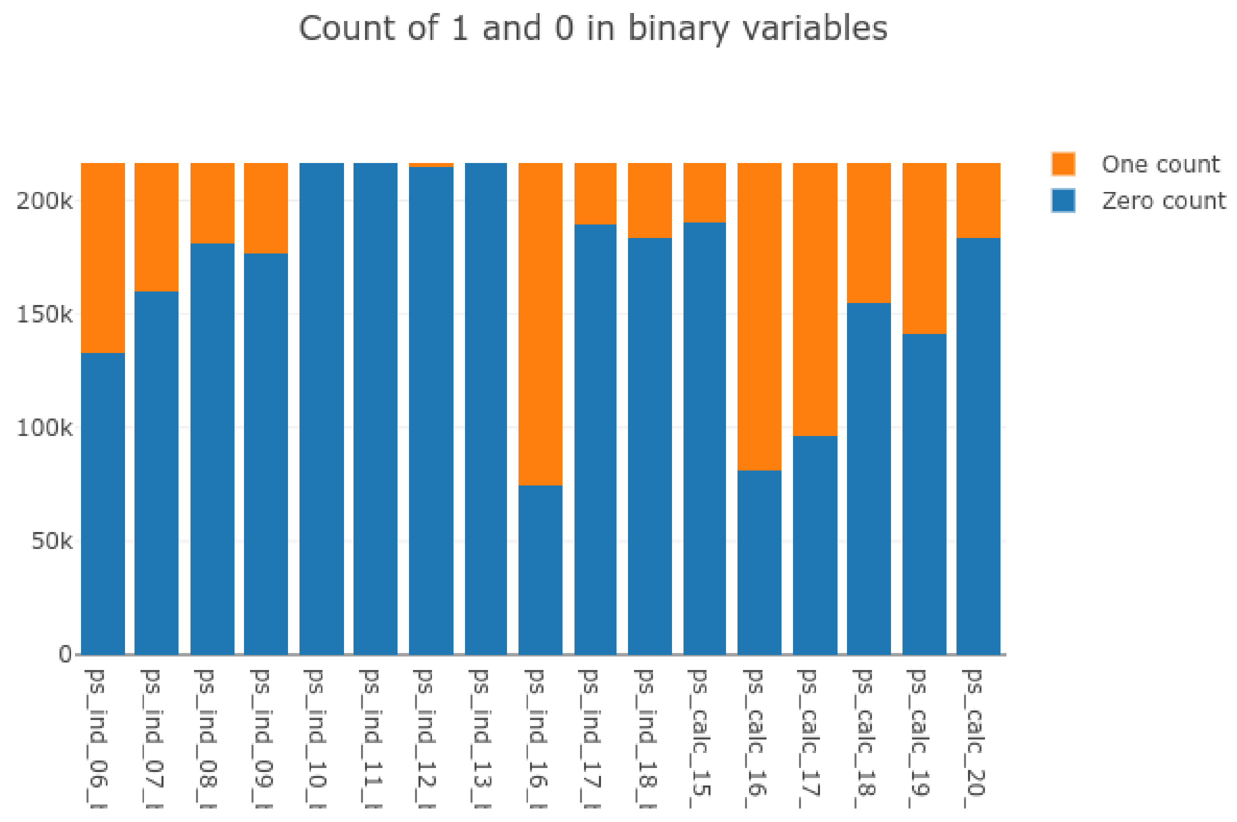
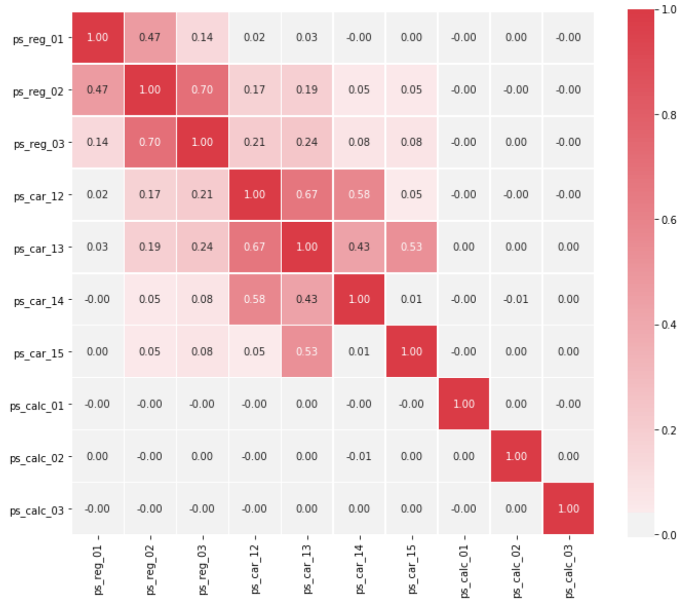


Fig. 3: The distribution of target in binary features.

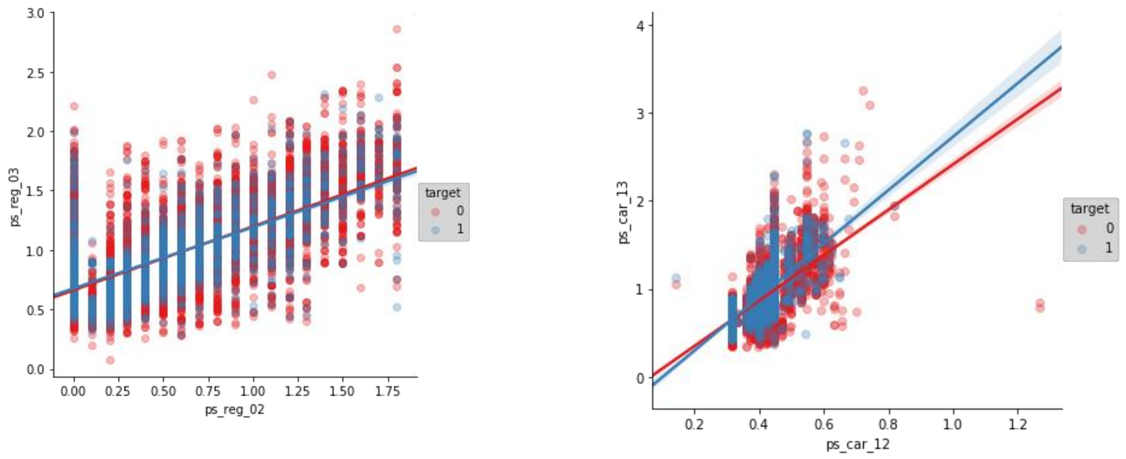
The correlation among the numerical values is plotted Fig.4. The relation between the numerical values is shown in Fig. 5

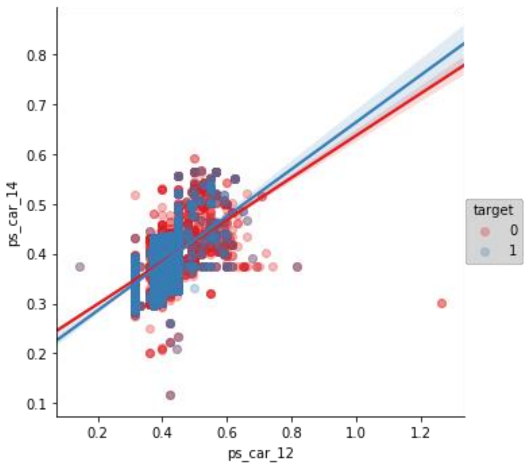
Fig.4: Heat Map

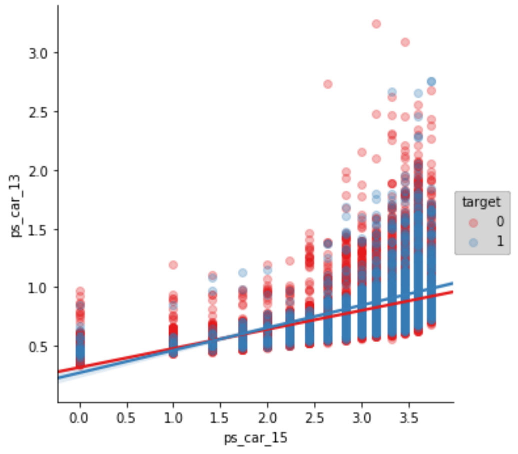


	ps_reg_01	ps_reg_02	ps_reg_03	ps_car_12	ps_car_13	ps_car_14	ps_car_15	ps_calc_01	ps_calc_02	ps_calc_03
ps_reg_01	1.00	0.47	0.14	0.02	0.03	-0.00	0.00	-0.00	0.00	-0.00
ps_reg_02	0.47	1.00	0.70	0.17	0.19	0.05	0.05	-0.00	-0.00	-0.00
ps_reg_03	0.14	0.70	1.00	0.21	0.24	0.08	0.08	-0.00	0.00	-0.00
ps_car_12	0.02	0.17	0.21	1.00	0.67	0.58	0.05	-0.00	-0.00	-0.00
ps_car_13	0.03	0.19	0.24	0.67	1.00	0.43	0.53	0.00	0.00	0.00
ps_car_14	-0.00	0.05	0.08	0.58	0.43	1.00	0.01	-0.00	-0.01	0.00
ps_car_15	0.00	0.05	0.08	0.05	0.53	0.01	1.00	-0.00	0.00	0.00
ps_calc_01	-0.00	-0.00	-0.00	-0.00	0.00	-0.00	-0.00	1.00	0.00	-0.00
ps_calc_02	0.00	-0.00	0.00	-0.00	0.00	-0.01	0.00	0.00	1.00	0.00
ps_calc_03	-0.00	-0.00	-0.00	-0.00	0.00	0.00	0.00	-0.00	0.00	1.00

Fig.5: regression relation







2.3. Algorithm and Techniques

In this section I will explain the five algorithms that I use in this project: logistic regression, BernoulliNB, Random Forest, XGBoost and Light GBM.

logistic regression model:

Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes).

BernoulliNB model:

BernoulliNB implements the naive Bayes training and classification algorithms for data that is distributed according to multivariate Bernoulli distributions; i.e., there may be multiple features but each one is assumed to be a binary-valued (Bernoulli, boolean) variable. Therefore, this class requires samples to be represented as binary-valued feature vectors; if handed any other kind of data, a BernoulliNB instance may binarize its input (depending on the binarize parameter).

The decision rule for Bernoulli naive Bayes is based on

P(x_i|y) = P(i|y)x_i + (1-P(i|y))(1-x_i)

which differs from multinomial NB’s rule in that it explicitly penalizes the non-occurrence of a feature i that is an indicator for class y, where the multinomial variant would simply ignore a non-occurring feature.

Ensemble methods:

The goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator.

Two families of ensemble methods are usually distinguished:

In averaging methods, the driving principle is to build several estimators independently and then to average their predictions. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced.

Example: Bagging methods, Forests of randomized trees, ...

By contrast, in boosting methods, base estimators are built sequentially and one tries to reduce the bias of the combined estimator. The motivation is to combine several weak models to produce a powerful ensemble.

Examples: AdaBoost, Gradient Tree Boosting, …

The sklearn.ensemble module includes two averaging algorithms based on randomized decision trees: the RandomForest algorithm and the Extra-Trees method.

Random Forests:

In random forests , each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. In addition, when splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. As a result of this randomness, the bias of the forest usually slightly increases (with respect to the bias of a single non-random tree) but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default).

XGBoost

XGBoost stands for eXtreme Gradient Boosting. The name xgboost, though, actually refers to the engineering goal to push the limit of computations resources for boosted tree algorithms. Which is the reason why many people use xgboost.

Algorithm Use by XGBoost

The XGBoost library implements the gradient boosting decision tree algorithm. This algorithm goes by lots of different names such as gradient boosting, multiple additive regression trees, stochastic gradient boosting or gradient boosting machines. Boosting is an ensemble technique where new models are added to correct the errors made by existing models. Models are added sequentially until no further improvements can be made. A popular example is the AdaBoost algorithm that weights data points that are hard to predict.

Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models. This approach supports both regression and classification predictive modeling problems.

1. The XGBoost Advantage

a. Regularization: Standard GBM implementation has no regularization like XGBoost, therefore it also helps to reduce overfitting. In fact, XGBoost is also known as 'regularized boosting' technique.
b. Parallel Processing: XGBoost implements parallel processing and is blazingly faster as compared to GBM.
c. High Flexibility: XGBoost allow users to define custom optimization objectives and evaluation criteria. This adds a whole new dimension to the model and there is no limit to what we can do.
d. Handling Missing Values: XGBoost has an in-built routine to handle missing values. User is required to supply a different value than other observations and pass that as a parameter. XGBoost tries different things as it encounters a missing value on each node and learns which path to take for missing values in future.
e. Tree Pruning: A GBM would stop splitting a node when it encounters a negative loss in the split. Thus it is more of a greedy algorithm. XGBoost on the other hand make splits

upto the max_depth specified and then start pruning the tree backwards and remove splits beyond which there is no positive gain. Another advantage is that sometimes a split of negative loss say -2 may be followed by a split of positive loss +10. GBM would stop as it encounters -2. But XGBoost will go deeper and it will see a combined effect of +8 of the split and keep both.

f. Built-in Cross-Validation: XGBoost allows user to run a cross-validation at each iteration of the boosting process and thus it is easy to get the exact optimum number of boosting iterations in a single run. This is unlike GBM where we have to run a grid-search and only a limited values can be tested.
g. Continue on Existing Model: User can start training an XGBoost model from its last iteration of previous run. This can be of significant advantage in certain specific applications.

1. Light GBM

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. So when growing on the same leaf in Light GBM, 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. Also, it is surprisingly very fast, hence the word 'Light'.

Before is a diagrammatic representation by the makers of the Light GBM to explain the difference clearly.

2. Advantages of Light GBM

a. Faster training speed and higher efficiency: Light GBM use histogram based algorithm i.e it buckets continuous feature values into discrete bins which fasten the training procedure.
b. Lower memory usage: Replaces continuous values to discrete bins which result in lower memory usage.
c. Better accuracy than any other boosting algorithm: It produces much more complex trees by following leaf wise split approach rather than a level-wise approach which is the main factor in achieving higher accuracy. However, it can sometimes lead to overfitting which can be avoided by setting the max_depth parameter.
d. Compatibility with Large Datasets: It is capable of performing equally good with large datasets with a significant reduction in training time as compared to XGBOOST.

e. Parallel learning supported.

2.4. Benchmark

In this section, I show the result of a logistic regression model which is applied on this problem. Our metric is Gini which is related to Gini = 2AUC – 1. For the simple logistic regression on the sampled data, we obtained Gini = 0.19 score. I am going to improve the result of the prediction. Although this calculation shows an acceptable gini score, the prediction cannot predict the real data because the train data is unbalance, hence, our prediction has overfit problem. Therefore, to establish a robust model we have to overcome the unbalance data problem in our data set.

3. Methodology

3.1. Data Preprocessing

3.1.1 Handling Imbalanced Classes

Preparing/sampleing.py

What can we do when we have imbalanced data? Mainly three things: 1) Ignoring the problem. 2) Undersampling the majority class. 3) Oversampling the minority class.

IGNORING THE PROBLEM:

Building a classifier using the data as it is, would in most cases give us a prediction model that always returns the majority class. The classifier would be biased. Let's build the models:

UNDERSAMPLING THE MAJORITY CLASS:

One of the most common and simplest strategies to handle imbalanced data is to undersample the majority class. While different techniques have been proposed in the past, typically using more advanced methods did not bring any improvement with respect to simply selecting samples at random. So, for this analysis I will simply select n samples at random from the majority class, where n is the number of samples for the minority class, and use them during training phase, after excluding the sample to use for validation.

OVERSAMPLING THE MINORITY CLASS

Oversampling the minority class can result in overfitting problems if we oversample before cross-validating. What is wrong with oversampling before cross-validating? Let’s consider the simplest oversampling method ever, as an example that clearly explains this point.

The easiest way to oversample is to re-sample the minority class, i.e. to duplicate the entries, or manufacture data which is exactly the same as what we have already. Now, if we do so before cross-validating, i.e. before we enter the leave one participant out cross-validation loop, we will be training the classifier using N-1 entries, leaving 1 out, but including in the N-1 one or more instances that are exactly the same as the one being validated. Thus, defeating the purpose of cross-validation altogether.

As we have a rather large training set, we can go for undersampling. Therefore, the percentage of 0 in train 0.96, the percentage of 1 in train 0.03, the percentage of 0 in the sample train 0.9, the percentage of 1 in the sample train 0.1

3.1.1 Handling Missing Value in the Sample dataset

analysis/missingvalue/missingvalue.py, Analysis/traintest/traintest.py

The missing values for each feature are:

Variable ps_ind_02_cat has 216 records (0.04%) with missing values.

Variable ps_ind_04_cat has 83 records (0.01%) with missing values.

Variable ps_ind_05_cat has 5809 records (0.98%) with missing values.

Variable ps_reg_03 has 107772 records (18.11%) with missing values.

Variable ps_car_01_cat has 107 records (0.02%) with missing values.

Variable ps_car_02_cat has 5 records (0.00%) with missing values.

Variable ps_car_03_cat has 411231 records (69.09%) with missing values.

Variable ps_car_05_cat has 266551 records (44.78%) with missing values.

Variable ps_car_07_cat has 11489 records (1.93%) with missing values.

Variable ps_car_09_cat has 569 records (0.10%) with missing values.

Variable ps_car_11 has 5 records (0.00%) with missing values.

Variable ps_car_12 has 1 records (0.00%) with missing values.

Variable ps_car_14 has 42620 records (7.16%) with missing values.

In total, there are 13 variables with missing values. Thus, we deal with missing values as follows:

ps_car_03_cat and ps_car_05_cat have a large proportion of records with missing values. Remove these variables.

For the other categorical variables with missing values, we can leave the missing value -1 as such.

ps_reg_03 (continuous) has missing values for 18% of all records. Replace by the mean.

ps_car_11 (ordinal) has only 5 records with misisng values. Replace by the mode.

ps_car_12 (continuous) has only 1 records with missing value. Replace by the mean.

ps_car_14 (continuous) has missing values for 7% of all records. Replace by the mean.

3.1.2. Features Engineering

CREATING INTERACTION VARIABLE:

Preparing/categoricalEncoding.py

She klearn.preporocessing library is used for fload data to make new features from interaction between features with PolynomialFeaures(2). Before creating interactions, we have 57 variables. After creating interactions, we have 112 variables. Before creating interactions, we have 56 variables. After creating interactions, we have 111 variable.

FREQUENCY ENCODING:

Preparing/categoricalEncoding.py

For the categorical data, ps_ind_02_cat','ps_car_04_cat', 'ps_car_09_cat','ps_ind_05_cat', 'ps_car_01_cat', 'ps_car_11_cat', the number of repetition data for each feature is counted and the results is saved into a new data frame column.

BINARY ENCODING:

Preparing/categoricalEncoding.py

Additionally, for the categorical data with low number of category, ps_ind_02_cat','ps_car_04 _cat', 'ps_car_09_cat', 'ps_ind_05_cat', 'ps_car_01_cat', 'ps_car_11_cat', I did binary encoding.

Finally, train shape is: (216940, 135) and test shape is: (892816, 134).

3.2. Implementation

The first step of the project is data analysis. I determine the type of the problem, features, missing value and imbalance data in this section. Then, since the target is imbalance, I used undersampling model to produce more balance data. After that, in the data engineering section, the new features are added to base dataset. The interaction between numerical features are obtained by polynomial method. The categorical features are treated in two ways: binary encoding and frequency encoding. The important features in the data are extract by random forest feature importance model.

In the modeling part three level of machine learning algorithms are stacked together. In the first level, Logistic Regression, BernouliNB, Random Forest, XGBoost and Light GBM are applied. The result of the five models are stacked and it is used as the input of the level 2. In the level 2, Logistic Regression, Random Forest, XGBoost and Light GBM algorithms are used. Finally, in the last level we applied Logistic Regression, XGBoost models and their result are avarged to make the final prediction.

3.3. Refinement

for the logistic regression tuning parameters:

```
//\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
Tuning logistic regression penalty = l1

/////////////////////////////////////////////////
Tuning logistic regression c = 0.1

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
sklearn: logit0
--------------------
fold cv 0 AUC score is 0.600559, Gini_Norm score is 0.201117
fold cv 1 AUC score is 0.596478, Gini_Norm score is 0.192955
fold cv 2 AUC score is 0.590134, Gini_Norm score is 0.180268
fold cv 3 AUC score is 0.597373, Gini_Norm score is 0.194746
fold cv 4 AUC score is 0.604743, Gini_Norm score is 0.209487
cv AUC score is 0.597826, Gini_Norm score is 0.195652
it takes 241.420 seconds to perform cross validation

/////////////////////////////////////////////////
Tuning logistic regression c = 0.2
```

```
-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
sklearn: logit1
--------------------
fold cv 0 AUC score is 0.600362, Gini_Norm score is 0.200724
fold cv 1 AUC score is 0.596654, Gini_Norm score is 0.193308
fold cv 2 AUC score is 0.589995, Gini_Norm score is 0.179990
fold cv 3 AUC score is 0.597331, Gini_Norm score is 0.194662
fold cv 4 AUC score is 0.604741, Gini_Norm score is 0.209482
cv AUC score is 0.597783, Gini_Norm score is 0.195566
it takes 313.016 seconds to perform cross validation

//////////////////////////////////////////////////
Tuning logistic regression c = 0.5

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
sklearn: logit2
--------------------
fold cv 0 AUC score is 0.600152, Gini_Norm score is 0.200304
fold cv 1 AUC score is 0.596733, Gini_Norm score is 0.193466
fold cv 2 AUC score is 0.589920, Gini_Norm score is 0.179840
fold cv 3 AUC score is 0.597286, Gini_Norm score is 0.194572
fold cv 4 AUC score is 0.604642, Gini_Norm score is 0.209285
cv AUC score is 0.597711, Gini_Norm score is 0.195423
it takes 447.866 seconds to perform cross validation

//////////////////////////////////////////////////
Tuning logistic regression c = 0.6

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
sklearn: logit3
--------------------
fold cv 0 AUC score is 0.600121, Gini_Norm score is 0.200242
fold cv 1 AUC score is 0.596735, Gini_Norm score is 0.193469
fold cv 2 AUC score is 0.589916, Gini_Norm score is 0.179831
fold cv 3 AUC score is 0.597286, Gini_Norm score is 0.194572
fold cv 4 AUC score is 0.604630, Gini_Norm score is 0.209261
cv AUC score is 0.597702, Gini_Norm score is 0.195404
it takes 464.058 seconds to perform cross validation
```

```
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
Tuning logistic regression penalty = l2

////////////////////////////////////////////////
Tuning logistic regression c = 0.1

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
sklearn: logit4
--------------------
fold cv 0 AUC score is 0.600112, Gini_Norm score is 0.200223
fold cv 1 AUC score is 0.596640, Gini_Norm score is 0.193279
fold cv 2 AUC score is 0.589793, Gini_Norm score is 0.179587
fold cv 3 AUC score is 0.597297, Gini_Norm score is 0.194594
fold cv 4 AUC score is 0.604579, Gini_Norm score is 0.209159
cv AUC score is 0.597649, Gini_Norm score is 0.195299
it takes 25.757 seconds to perform cross validation

////////////////////////////////////////////////
Tuning logistic regression c = 0.2

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
sklearn: logit5
--------------------
fold cv 0 AUC score is 0.600042, Gini_Norm score is 0.200083
fold cv 1 AUC score is 0.596696, Gini_Norm score is 0.193391
fold cv 2 AUC score is 0.589805, Gini_Norm score is 0.179610
fold cv 3 AUC score is 0.597295, Gini_Norm score is 0.194589
fold cv 4 AUC score is 0.604579, Gini_Norm score is 0.209158
cv AUC score is 0.597647, Gini_Norm score is 0.195295
it takes 26.582 seconds to perform cross validation

////////////////////////////////////////////////
Tuning logistic regression c = 0.5

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
sklearn: logit6
--------------------
fold cv 0 AUC score is 0.599978, Gini_Norm score is 0.199956
fold cv 1 AUC score is 0.596734, Gini_Norm score is 0.193467
fold cv 2 AUC score is 0.589819, Gini_Norm score is 0.179638
fold cv 3 AUC score is 0.597287, Gini_Norm score is 0.194574
fold cv 4 AUC score is 0.604585, Gini_Norm score is 0.209169
cv AUC score is 0.597644, Gini_Norm score is 0.195288
it takes 28.731 seconds to perform cross validation

////////////////////////////////////////////////
Tuning logistic regression c = 0.6

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
sklearn: logit7
--------------------
fold cv 0 AUC score is 0.599970, Gini_Norm score is 0.199941
fold cv 1 AUC score is 0.596736, Gini_Norm score is 0.193472
fold cv 2 AUC score is 0.589823, Gini_Norm score is 0.179645
fold cv 3 AUC score is 0.597287, Gini_Norm score is 0.194575
fold cv 4 AUC score is 0.604584, Gini_Norm score is 0.209169
cv AUC score is 0.597643, Gini_Norm score is 0.195287
it takes 30.719 seconds to perform cross validation
```

Based on the calculation it seems that the first model: penalty = 0.1 and l1 has the best gini = 0.1956 score.

Random Forest://\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/

```
Tuning Random Forest = 80

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
sklearn: rf0
--------------------
fold cv 0 AUC score is 0.601040, Gini_Norm score is 0.202080
fold cv 1 AUC score is 0.598763, Gini_Norm score is 0.197527
fold cv 2 AUC score is 0.594034, Gini_Norm score is 0.188068
fold cv 3 AUC score is 0.597612, Gini_Norm score is 0.195225
fold cv 4 AUC score is 0.607661, Gini_Norm score is 0.215321
cv AUC score is 0.599656, Gini_Norm score is 0.199311
it takes 51.777 seconds to perform cross validation

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
sklearn: rf1
```

```
--------------------
fold cv 0 AUC score is 0.603028, Gini_Norm score is 0.206055
fold cv 1 AUC score is 0.606140, Gini_Norm score is 0.212280
fold cv 2 AUC score is 0.600661, Gini_Norm score is 0.201323
fold cv 3 AUC score is 0.604827, Gini_Norm score is 0.209654
fold cv 4 AUC score is 0.611461, Gini_Norm score is 0.222922
cv AUC score is 0.605103, Gini_Norm score is 0.210207
it takes 98.799 seconds to perform cross validation

//\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
Tuning Random Forest = 100

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
sklearn: rf2
--------------------
fold cv 0 AUC score is 0.600872, Gini_Norm score is 0.201744
fold cv 1 AUC score is 0.599315, Gini_Norm score is 0.198630
fold cv 2 AUC score is 0.594168, Gini_Norm score is 0.188336
fold cv 3 AUC score is 0.597708, Gini_Norm score is 0.195416
fold cv 4 AUC score is 0.607792, Gini_Norm score is 0.215583
cv AUC score is 0.599804, Gini_Norm score is 0.199608
it takes 74.736 seconds to perform cross validation

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
sklearn: rf3
--------------------
fold cv 0 AUC score is 0.603556, Gini_Norm score is 0.207113
fold cv 1 AUC score is 0.605790, Gini_Norm score is 0.211580
fold cv 2 AUC score is 0.601267, Gini_Norm score is 0.202535
fold cv 3 AUC score is 0.605055, Gini_Norm score is 0.210110
fold cv 4 AUC score is 0.612224, Gini_Norm score is 0.224448
cv AUC score is 0.605451, Gini_Norm score is 0.210903
it takes 139.322 seconds to perform cross validation

//\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
Tuning Random Forest = 150

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
sklearn: rf4
--------------------
fold cv 0 AUC score is 0.600858, Gini_Norm score is 0.201716
fold cv 1 AUC score is 0.599301, Gini_Norm score is 0.198601
fold cv 2 AUC score is 0.593954, Gini_Norm score is 0.187908
fold cv 3 AUC score is 0.597705, Gini_Norm score is 0.195411
fold cv 4 AUC score is 0.607628, Gini_Norm score is 0.215255
cv AUC score is 0.599696, Gini_Norm score is 0.199391
it takes 123.802 seconds to perform cross validation

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
sklearn: rf5
--------------------
fold cv 0 AUC score is 0.604579, Gini_Norm score is 0.209158
fold cv 1 AUC score is 0.605835, Gini_Norm score is 0.211670
fold cv 2 AUC score is 0.601992, Gini_Norm score is 0.203984
fold cv 3 AUC score is 0.604994, Gini_Norm score is 0.209988
fold cv 4 AUC score is 0.612246, Gini_Norm score is 0.224491
cv AUC score is 0.605819, Gini_Norm score is 0.211637
it takes 193.951 seconds to perform cross validation

//\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
Tuning Random Forest = 200

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
sklearn: rf6
--------------------
fold cv 0 AUC score is 0.600722, Gini_Norm score is 0.201444
fold cv 1 AUC score is 0.599271, Gini_Norm score is 0.198542
fold cv 2 AUC score is 0.593835, Gini_Norm score is 0.187671
fold cv 3 AUC score is 0.597710, Gini_Norm score is 0.195421
fold cv 4 AUC score is 0.607909, Gini_Norm score is 0.215818
cv AUC score is 0.599694, Gini_Norm score is 0.199389
it takes 136.169 seconds to perform cross validation

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
sklearn: rf7
--------------------
fold cv 0 AUC score is 0.605047, Gini_Norm score is 0.210095
fold cv 1 AUC score is 0.605762, Gini_Norm score is 0.211524
fold cv 2 AUC score is 0.602105, Gini_Norm score is 0.204210
fold cv 3 AUC score is 0.604953, Gini_Norm score is 0.209905
fold cv 4 AUC score is 0.613117, Gini_Norm score is 0.226233
cv AUC score is 0.606069, Gini_Norm score is 0.212139
it takes 250.441 seconds to perform cross validation
```

The calculation shoes that the seven model: n_estimators = 200 and max_depth = 10 has the highest score.

BN:

```
//\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
Tuning BN alpha = 0.7

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
sklearn: nb0
--------------------
fold cv 0 AUC score is 0.585148, Gini_Norm score is 0.170296
fold cv 1 AUC score is 0.584793, Gini_Norm score is 0.169587
fold cv 2 AUC score is 0.575421, Gini_Norm score is 0.150841
fold cv 3 AUC score is 0.578943, Gini_Norm score is 0.157885
fold cv 4 AUC score is 0.589558, Gini_Norm score is 0.179115
cv AUC score is 0.582661, Gini_Norm score is 0.165322
it takes 12.986 seconds to perform cross validation

//\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
Tuning BN alpha = 0.8

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
sklearn: nb1
--------------------
fold cv 0 AUC score is 0.585148, Gini_Norm score is 0.170295
fold cv 1 AUC score is 0.584793, Gini_Norm score is 0.169587
fold cv 2 AUC score is 0.575420, Gini_Norm score is 0.150841
fold cv 3 AUC score is 0.578943, Gini_Norm score is 0.157885
fold cv 4 AUC score is 0.589557, Gini_Norm score is 0.179115
cv AUC score is 0.582661, Gini_Norm score is 0.165321
it takes 13.001 seconds to perform cross validation

//\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
Tuning BN alpha = 0.9

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
sklearn: nb3
--------------------
fold cv 0 AUC score is 0.585148, Gini_Norm score is 0.170295
fold cv 1 AUC score is 0.584793, Gini_Norm score is 0.169587
fold cv 2 AUC score is 0.575420, Gini_Norm score is 0.150841
fold cv 3 AUC score is 0.578942, Gini_Norm score is 0.157885
fold cv 4 AUC score is 0.589557, Gini_Norm score is 0.179115
cv AUC score is 0.582661, Gini_Norm score is 0.165321
```

```
it takes 12.997 seconds to perform cross validation

//\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
Tuning BN alpha = 1.0

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
sklearn: nb4
--------------------
fold cv 0 AUC score is 0.585148, Gini_Norm score is 0.170296
fold cv 1 AUC score is 0.584793, Gini_Norm score is 0.169586
fold cv 2 AUC score is 0.575420, Gini_Norm score is 0.150840
fold cv 3 AUC score is 0.578942, Gini_Norm score is 0.157885
fold cv 4 AUC score is 0.589557, Gini_Norm score is 0.179115
cv AUC score is 0.582661, Gini_Norm score is 0.165321
it takes 12.974 seconds to perform cross validation
```

The result for different alpha does not change.

XGBoost:

```
//\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
Tuning xgb max_depth = 3

////////////////////////////////////////////////
Tuning xgb eta = 0.05

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
xgb: xgb0
--------------------
fold cv 0 AUC score is 0.615142, Gini_Norm score is 0.230284
fold cv 1 AUC score is 0.613793, Gini_Norm score is 0.227587
fold cv 2 AUC score is 0.612151, Gini_Norm score is 0.224301
fold cv 3 AUC score is 0.616587, Gini_Norm score is 0.233173
fold cv 4 AUC score is 0.622668, Gini_Norm score is 0.245335
cv AUC score is 0.616017, Gini_Norm score is 0.232034
it takes 82.737 seconds to perform cross validation

////////////////////////////////////////////////
Tuning xgb eta = 0.1

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
xgb: xgb1
--------------------
fold cv 0 AUC score is 0.615780, Gini_Norm score is 0.231561
fold cv 1 AUC score is 0.613078, Gini_Norm score is 0.226157
fold cv 2 AUC score is 0.611121, Gini_Norm score is 0.222242
fold cv 3 AUC score is 0.614833, Gini_Norm score is 0.229665
fold cv 4 AUC score is 0.621485, Gini_Norm score is 0.242970
cv AUC score is 0.615189, Gini_Norm score is 0.230378
it takes 57.457 seconds to perform cross validation

////////////////////////////////////////////////
Tuning xgb eta = 0.5

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
xgb: xgb2
--------------------
fold cv 0 AUC score is 0.612897, Gini_Norm score is 0.225794
fold cv 1 AUC score is 0.609494, Gini_Norm score is 0.218988
fold cv 2 AUC score is 0.608960, Gini_Norm score is 0.217921
fold cv 3 AUC score is 0.613851, Gini_Norm score is 0.227702
fold cv 4 AUC score is 0.617259, Gini_Norm score is 0.234518
cv AUC score is 0.612189, Gini_Norm score is 0.224378
it takes 22.986 seconds to perform cross validation

//\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
Tuning xgb max_depth = 4

////////////////////////////////////////////////
Tuning xgb eta = 0.05

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
xgb: xgb3
--------------------
fold cv 0 AUC score is 0.615595, Gini_Norm score is 0.231190
fold cv 1 AUC score is 0.613870, Gini_Norm score is 0.227741
fold cv 2 AUC score is 0.612001, Gini_Norm score is 0.224002
fold cv 3 AUC score is 0.615283, Gini_Norm score is 0.230565
fold cv 4 AUC score is 0.621745, Gini_Norm score is 0.243491
cv AUC score is 0.615551, Gini_Norm score is 0.231103
it takes 63.491 seconds to perform cross validation

////////////////////////////////////////////////
Tuning xgb eta = 0.1

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
xgb: xgb4
--------------------
```

```
fold cv 0 AUC score is 0.615941, Gini_Norm score is 0.231883
fold cv 1 AUC score is 0.613240, Gini_Norm score is 0.226480
fold cv 2 AUC score is 0.611552, Gini_Norm score is 0.223105
fold cv 3 AUC score is 0.613225, Gini_Norm score is 0.226450
fold cv 4 AUC score is 0.621063, Gini_Norm score is 0.242126
cv AUC score is 0.614959, Gini_Norm score is 0.229918
it takes 43.116 seconds to perform cross validation

//////////////////////////////////////////////////
Tuning xgb eta = 0.5

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
xgb: xgb5
--------------------
fold cv 0 AUC score is 0.610521, Gini_Norm score is 0.221042
fold cv 1 AUC score is 0.608286, Gini_Norm score is 0.216573
fold cv 2 AUC score is 0.606437, Gini_Norm score is 0.212874
fold cv 3 AUC score is 0.609278, Gini_Norm score is 0.218555
fold cv 4 AUC score is 0.616090, Gini_Norm score is 0.232180
cv AUC score is 0.610019, Gini_Norm score is 0.220039
it takes 23.814 seconds to perform cross validation

//\/\/\/\/\/\/\/\/\/\/\/\/\
Tuning xgb max_depth = 5

//////////////////////////////////////////////////
Tuning xgb eta = 0.05

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
xgb: xgb6
--------------------
fold cv 0 AUC score is 0.614052, Gini_Norm score is 0.228104
fold cv 1 AUC score is 0.614145, Gini_Norm score is 0.228289
fold cv 2 AUC score is 0.611498, Gini_Norm score is 0.222996
fold cv 3 AUC score is 0.614662, Gini_Norm score is 0.229324
fold cv 4 AUC score is 0.621993, Gini_Norm score is 0.243987
cv AUC score is 0.615154, Gini_Norm score is 0.230307
it takes 69.842 seconds to perform cross validation

//////////////////////////////////////////////////
Tuning xgb eta = 0.1

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
xgb: xgb7
```

```
--------------------
fold cv 0 AUC score is 0.614715, Gini_Norm score is 0.229429
fold cv 1 AUC score is 0.612743, Gini_Norm score is 0.225485
fold cv 2 AUC score is 0.610731, Gini_Norm score is 0.221462
fold cv 3 AUC score is 0.612769, Gini_Norm score is 0.225538
fold cv 4 AUC score is 0.621248, Gini_Norm score is 0.242497
cv AUC score is 0.614377, Gini_Norm score is 0.228755
it takes 47.964 seconds to perform cross validation

////////////////////////////////////////////////
Tuning xgb eta = 0.5

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
xgb: xgb8
--------------------
fold cv 0 AUC score is 0.609041, Gini_Norm score is 0.218083
fold cv 1 AUC score is 0.602736, Gini_Norm score is 0.205473
fold cv 2 AUC score is 0.604474, Gini_Norm score is 0.208949
fold cv 3 AUC score is 0.607535, Gini_Norm score is 0.215071
fold cv 4 AUC score is 0.612751, Gini_Norm score is 0.225501
cv AUC score is 0.604381, Gini_Norm score is 0.208763
it takes 24.708 seconds to perform cross validation
```

Max_depth = 3 and eta = 0.05 has the highest score

LightGBM:

```
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
Tuning lgb max_depth = 4

////////////////////////////////////////////////
Tuning lgb learning rate= 0.05

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
lgb: lgb0
--------------------
fold cv 0 AUC score is 0.614950, Gini_Norm score is 0.229900
fold cv 1 AUC score is 0.613113, Gini_Norm score is 0.226226
fold cv 2 AUC score is 0.611873, Gini_Norm score is 0.223747
fold cv 3 AUC score is 0.615055, Gini_Norm score is 0.230109
fold cv 4 AUC score is 0.621566, Gini_Norm score is 0.243131
cv AUC score is 0.615190, Gini_Norm score is 0.230379
it takes 54.471 seconds to perform cross validation
```

```
////////////////////////////////////////////////
Tuning lgb learning rate= 0.1

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
lgb: lgb1
--------------------
fold cv 0 AUC score is 0.614581, Gini_Norm score is 0.229162
fold cv 1 AUC score is 0.612085, Gini_Norm score is 0.224171
fold cv 2 AUC score is 0.609464, Gini_Norm score is 0.218928
fold cv 3 AUC score is 0.611263, Gini_Norm score is 0.222526
fold cv 4 AUC score is 0.619412, Gini_Norm score is 0.238824
cv AUC score is 0.612916, Gini_Norm score is 0.225832
it takes 27.766 seconds to perform cross validation

////////////////////////////////////////////////
Tuning lgb learning rate= 0.5

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
lgb: lgb2
--------------------
fold cv 0 AUC score is 0.607664, Gini_Norm score is 0.215327
fold cv 1 AUC score is 0.605388, Gini_Norm score is 0.210776
fold cv 2 AUC score is 0.599278, Gini_Norm score is 0.198556
fold cv 3 AUC score is 0.606970, Gini_Norm score is 0.213939
fold cv 4 AUC score is 0.615770, Gini_Norm score is 0.231540
cv AUC score is 0.606844, Gini_Norm score is 0.213688
it takes 13.643 seconds to perform cross validation

//\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
Tuning lgb max_depth = 5

////////////////////////////////////////////////
Tuning lgb learning rate= 0.05

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
lgb: lgb3
--------------------
fold cv 0 AUC score is 0.615866, Gini_Norm score is 0.231732
fold cv 1 AUC score is 0.611064, Gini_Norm score is 0.222128
fold cv 2 AUC score is 0.612053, Gini_Norm score is 0.224105
fold cv 3 AUC score is 0.612527, Gini_Norm score is 0.225054
fold cv 4 AUC score is 0.621569, Gini_Norm score is 0.243138
cv AUC score is 0.613458, Gini_Norm score is 0.226916
```

```
it takes 46.082 seconds to perform cross validation

/////////////////////////////////////////////////
Tuning lgb learning rate= 0.1

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
lgb: lgb4
--------------------
fold cv 0 AUC score is 0.614035, Gini_Norm score is 0.228070
fold cv 1 AUC score is 0.610871, Gini_Norm score is 0.221743
fold cv 2 AUC score is 0.607883, Gini_Norm score is 0.215765
fold cv 3 AUC score is 0.612194, Gini_Norm score is 0.224388
fold cv 4 AUC score is 0.618460, Gini_Norm score is 0.236920
cv AUC score is 0.610806, Gini_Norm score is 0.221613
it takes 26.670 seconds to perform cross validation

/////////////////////////////////////////////////
Tuning lgb learning rate= 0.5

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
lgb: lgb5
--------------------
fold cv 0 AUC score is 0.607051, Gini_Norm score is 0.214103
fold cv 1 AUC score is 0.602126, Gini_Norm score is 0.204252
fold cv 2 AUC score is 0.598851, Gini_Norm score is 0.197701
fold cv 3 AUC score is 0.604940, Gini_Norm score is 0.209880
fold cv 4 AUC score is 0.611881, Gini_Norm score is 0.223762
cv AUC score is 0.596599, Gini_Norm score is 0.193198
it takes 13.958 seconds to perform cross validation

//\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
Tuning lgb max_depth = 6

/////////////////////////////////////////////////
Tuning lgb learning rate= 0.05

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
lgb: lgb6
--------------------
fold cv 0 AUC score is 0.615854, Gini_Norm score is 0.231709
fold cv 1 AUC score is 0.611241, Gini_Norm score is 0.222481
fold cv 2 AUC score is 0.611273, Gini_Norm score is 0.222546
fold cv 3 AUC score is 0.615087, Gini_Norm score is 0.230175
fold cv 4 AUC score is 0.619785, Gini_Norm score is 0.239571
```

```
cv AUC score is 0.613211, Gini_Norm score is 0.226421
it takes 49.677 seconds to perform cross validation

////////////////////////////////////////////////
Tuning lgb learning rate= 0.1

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
lgb: lgb7
--------------------
fold cv 0 AUC score is 0.614036, Gini_Norm score is 0.228071
fold cv 1 AUC score is 0.612551, Gini_Norm score is 0.225101
fold cv 2 AUC score is 0.608329, Gini_Norm score is 0.216657
fold cv 3 AUC score is 0.610588, Gini_Norm score is 0.221176
fold cv 4 AUC score is 0.619877, Gini_Norm score is 0.239753
cv AUC score is 0.612752, Gini_Norm score is 0.225504
it takes 30.125 seconds to perform cross validation

////////////////////////////////////////////////
Tuning lgb learning rate= 0.5

-*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*- -*-
lgb: lgb8
--------------------
fold cv 0 AUC score is 0.607873, Gini_Norm score is 0.215746
fold cv 1 AUC score is 0.604575, Gini_Norm score is 0.209151
fold cv 2 AUC score is 0.600133, Gini_Norm score is 0.200266
fold cv 3 AUC score is 0.604147, Gini_Norm score is 0.208293
fold cv 4 AUC score is 0.610733, Gini_Norm score is 0.221467
cv AUC score is 0.602417, Gini_Norm score is 0.204834
it takes 14.403 seconds to perform cross validation
```

max_depth = 4 and learbing rate = 0.05 has the best result

We can also do more tuning on other parameters and levels; however, because of my laptop limitation I will skip more precise tuning.

The stacking technique in three levels is applied to improve the result of the calculation.

3. Model

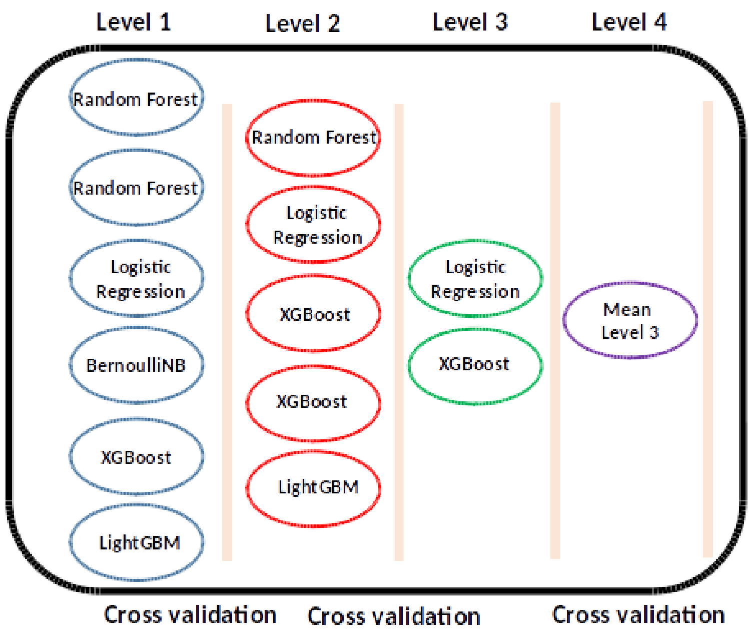


Fig.6: The stacking model to improve the prediction.

The result of the improvement in gibi score shown in the Fig.6 is listed in the Table 1. We keep the same order as Fig.6 to report in table.

Without tuning hyper parameters:

Table 1.

Level 1	Level 2	Level 3	Level 4
0.212			
0.199	0.227		
0.195	0.232	0.230	
0.165	0.105	0.227	0.229
0.232	0.232		
0.230	0.232		

With tuning hyper parameters

Table 2.

Level 1	Level 2	Level 3	Level 4
0.207			
0.199	0.226		
0.195	0.228	0.232	
0.165	0.114	0.244	0.230
0.228	0.228		
0.221	0.229		

4. Results

4.1. Model Evaluation and Validation

In this project, in order to improve our prediction in comparison a single model which is explained in Benchmark, we tried Four ways:

1) Tuning each model by its hyper-parameter. I tune hyper parameters. Without tuning I obtain gini = 0.228 and with tuning we obtain gini = 0.230

2) In the project2.py, I change the random value from 37 to 2018 in sampling model to obtain new sample as well as changing random in modeling from 2017 to 2018. Then, I tune the hyper parameters for the new model. I obtain a new gini = 0.224 score

3) using different models to predict because usually model such as logistic regression tends to under fitting and modeled such as xgboost and lightbm tend to over fitting. Therefore, we stack these model to overcome the under fitting and over fitting problem. For example in the level two we can see we use different method in xgboost model, with raking without ranking in order to try different model in our prediction and different gini score.

4) The models are stacked together in three deferent levels in order to improve the results. Therefore, we obtain the higher gini score after three levels.

Finally by considering the above model and discussions, at the level four which is mean of level three output, (logistic regression, xgboost) the gini score = 0.230 is obtained.

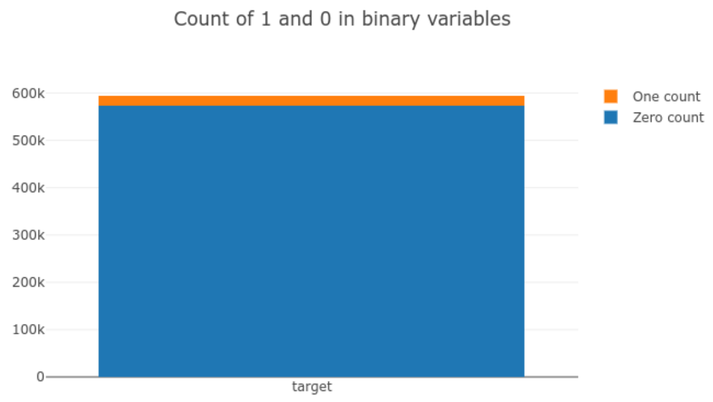
4.2. Justification

In comparison in our benchmark model, simple logistic regression gini = 0.19 score, we obtained less gini = 0.23 which shows the improvement on our modeling. The submission results in 0.27947 leader board score in competition. Top 40 %.

5. Conclusion

5.1. Free-From Visualization

one of the main problem of this classification challenge is imbalance target. Fig. 7 shows that the undersampling approach can solve the imbalance sample.



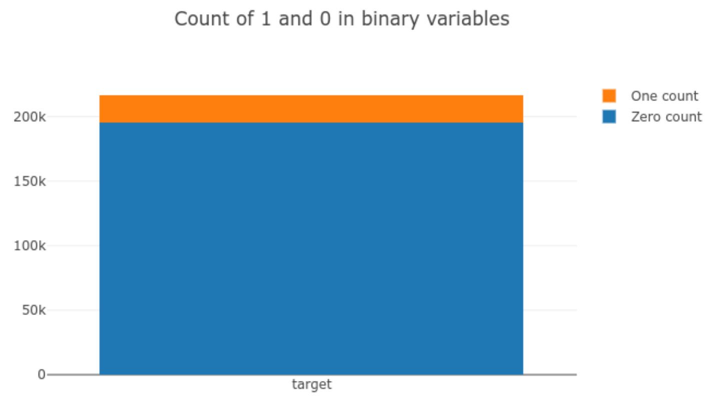


Fig.7: Figures show the target values before and after undersampling respectively.

The main approach to solve this binary classification challenge is stacking of different machine learning algorithms. Therefore, it is necessary to investigate whether the gini score improves thought this approach. Therefore, we shows how the mean of the gini score changes for each level in Fig. 8. It is seen that the mean of gini score is increasing by increasing the level. It means that the prediction is getting better in level 3.

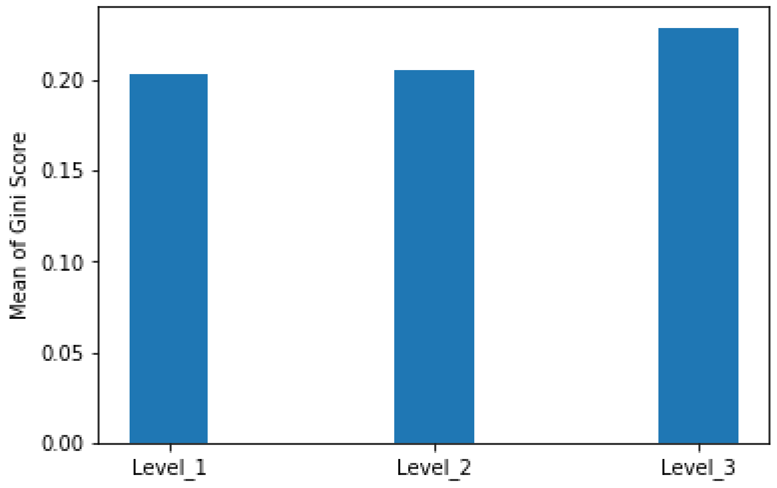


Fig.8: Figures show the target values before and after undersampling respectiv*ely.*

5.2. Reflection

In this project, I work on the dataset from kaggle competition. The dataset is provided by the Porto Seguro company. Porto Seguro, one of Brazil's largest auto and homeowner insurance companies. Inaccuracies in car insurance company's claim predictions raise the cost of insurance for good drivers and reduce the price for bad ones. The goal of this project was to build a model that predicts the probability that a driver will initiate an auto insurance claim in the next year. After the target exploration, I figure out that the dataset target is classified into 0 and 1. Thus, the problem is binary classification. Since the problem is binary classification, the gini metric is used for the calculation of the cross validation score. The interesting aspect of this binary problem is found after the statical exploration of the target. The exploration shows that the target values are imbalance. Therefore, the undersampling method is applied to over come this problem as well as to obtain the new sampling from the dataset. The data analysis shows that we have binary, categorical, numerical features. We use binary encoding and frequency encoding to extract the new features from the categorical values. The new features from numerical features are obtained by using regression model with polynomial kernels. The data engineer part of the sample was the most difficult part of the project because I had to find new relevant features to improve the prediction score. After data analysis and feature engineering, several machine learning algorithms, Logistic Regression, Random Forest, Brenoulli, Xgboost, Light GBM are stacked togther in three levels to make prediction for this classification challenge. Therefore, we reached the gini = 0.23 score for predicting the target value. Additionally, I also apply the whole machine learning pipeline with different random state and I obtained gini = 0.224.

Therefore, the machine learning pipeline is design for solving the binary classification problem which can be used for the similar classification problem.

5.3. Improvement

The result of the calculation can be improved by using target encoding and stacking more diverse machine learning algorithm. Moreover, the stacking model also can be improved by giving the wight for each model and then tuning that wight. In the other word, every model has wight = 1 now but it is possible to tune this wight too.