Santander Kaggle Competition
– predicting unsatisfied customers

1. Project Overview

Customer satisfaction has been the key to business success. Companies have spent millions of dollar conducting surveys to improve their customer service. However, dissatisfied customers rarely voice their dissatisfaction before leaving. Nowadays the increasing amount of data is generated with the use of internet service which provides another way for identifying the dissatisfied customers using the tools of machine learning. Through this approach, companies can identify dissatisfied customers at an early stage and take a proactive approach to prevent the customers from leaving.

Santander Bank has recently held a competition on Kaggle, a website for companies and researchers post their data, and statisticians and data miners from all over the world compete to produce the best models. The data set includes more than 300 anonymized features and more than 70,000 entries. The competition has attracted more than 4000 teams over the world.

In this project, I will use the Kaggle data set to predict the dissatisfied customers. The steps are as followed:

- Firstly, the performance metric will be defined.
- Secondly, different algorithms are tired and compared based on the performance metric. The best algorithm will be selected as the baseline model.
- Thirdly, we will select a few of the most important features through the selected model and conduct exploratory data analysis which will help me to better understand the data.
- Finally, the algorithm will be fine-tuned to further improve its performance. The final model will be validated through cross validation and its learning curve will be studied.

2 Problem Statement

2.1 Metrics

The problem of predicting dissatisfied customers falls into the realm of supervised learning because the data is labeled. The label of the data is binary: the dissatisfied customer is labeled as "1" and the satisfied customer is label as "0". Since the data label is binary, the AUROC (Area Under the Receiver Operating Characteristic) is chosen as the performance metric, which is often used to illustrates the performance of a binary classifier system as the threshold varies

- Receiver Operating Characteristic (roc) curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. There is a trade-off between the two variables. The advantage of the roc curve is that it explores all possible setting of the threshold. An illustration of the method is shown in Fig.1.
- Area Under the Curve (auc) is a single value to represent the performance. The value range from 0.5 (random classifier) to 1 (perfect classifier).The higher the score, the better.

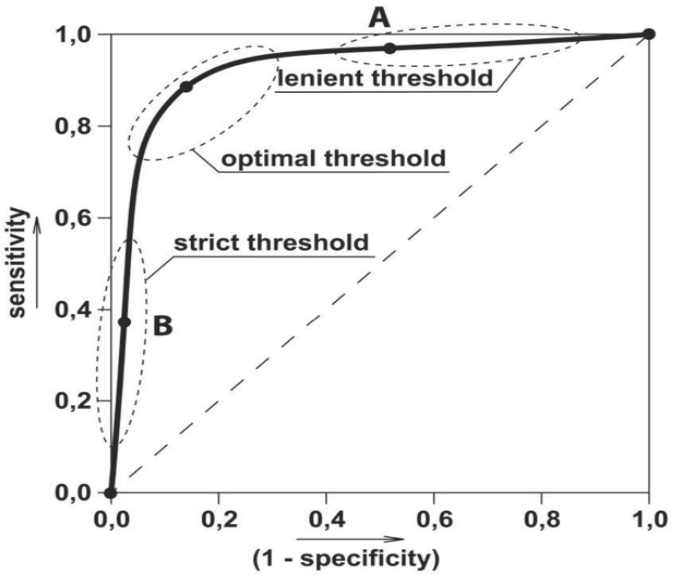


Fig. 1 Illustration of AUROC metric

3. Preprocess

3.1 Data Exploration

The data set contains 369 anonymized features and 76020 entries in the training set and about the same amount of entries in the testing set. Only the labels of the training set are provided. The data set is relatively clean, all the features take numerical value. However, the data set is sparse. Many of the entries are 0. A summary of the data is shown in Table. I.

Table. I Data Summary

# data in training set	76020
# data in test set	75818
# features:	369
# satisfied customer	73012
# dissatisfied customer	3008
% dissatisfied customer	4%

3.2 Data Cleaning

The next step is to clean the constant features which has a constant value in all the entries, and duplicate features, which has all the entries in the feature equal to an another feature.
Through data auditing, I find:

- 34 constant features. For example, all the values of the feature ‘ind_var2_0’ are 0. The name of the constant feature is listed in the appendix.
- 29 duplicate features. For example 'ind_var18_0' and 'ind_var18' have the same values. The name of the features with duplicated value are listed in the appendix.

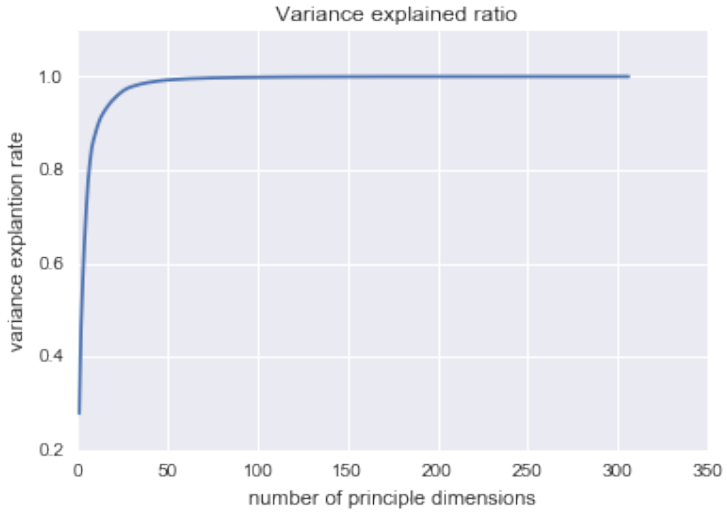
The constant and duplicate features are removed because they don't add any information. Finally, I removed the column 'ID' which are the number of the customer and does not add useful information. The column 'Target' is the label of the data set.

- The cleaned data (training set) set has 306 features and 76020 entries.

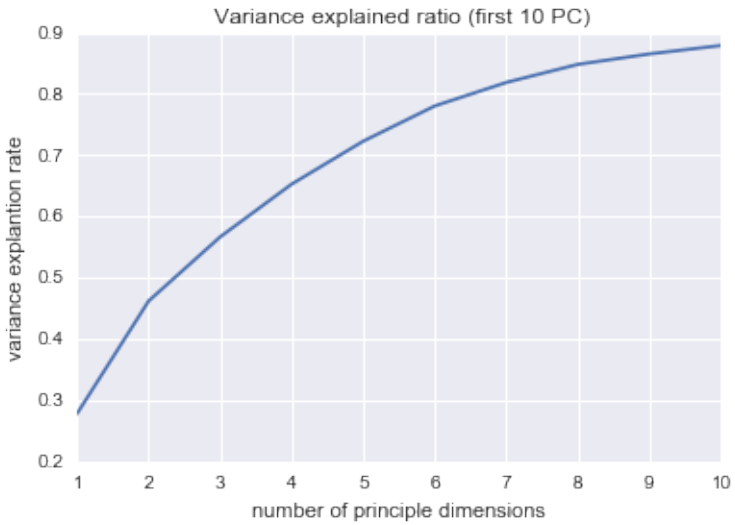
3.3 PCA

To get a better a glance of the data set. Principal Component Analysis (PCA) is conducted. The PCA preserves the maximum variance in the feature.

- Max-min scaling was conducted to normalize the features, since they have very different values. After normalization the feature will have value ranges for 0 to 1.
- Fig 2 (a) shows the accumulated variance explanation rate as the number of principal components increase. Though the data set has 360 dimensions, the variance explanation rate reaches close to 1 in about 50 dimensions.
- Fig. 2 (b) shows the accumulated variance explanation rate of the first 10 principal components. They explained almost 90% of the all the variance in the data.



(a) all principle components



(b) The first 10 Principle Components

Fig. 2 The accumulative variance explaining rate

The scatter plot of the data projected on the first 2 principal components is shown in Fig. 3. I find:

- The satisfied and dissatisfied data are largely overlapped. It indicates separating the two classes can be a hard problem.
- The data is not evenly distributed in the entire space. It seems to have several cluster. Some places have more data that other.

Fig. 3 Scatter plot of the data set projected on 2 principle components

4. Algorithms and Techniques

4.1 Model selection

In this section, I compared the performance of the different classifiers. 7 different algorithms is compared using the same procedures:

- The classifiers are in their default setting and not fine-tuned.
- The performance metric uses the AUROC score.
- 3-fold cross-validation is used. The total data, which contains 76020 entries, was randomly divided into 3 folds. One fold of the data set is chosen as the test set, and the rest is chosen training set. The process is repeated 3 times and the average score is reported.
- The running time includes the total time of training and testing time in cross-validation process.

The data set is relatively large so scalability will be an important aspect when to consider the algorithms. For example, we also attempted to use SVM, however, the running time of the algorithm is too long for this data set.

Table II shows the score and time of 7 different classifiers.

- The simple classifiers such as “decision tree” “naive bayes” and logistic regression perform poorly. It further indicates that the problem is difficult.
- The last 4 methods (marked in red) belong to a group of methods called ensemble methods which combined a group of “weak learners” to achieve better performance. In this case, we use decision tree based weak learner. The ensemble methods have much better performance score.

Table II. The performance of different models

classifier	auc score	Running time (sec)
decision Tree	0.567	7
naïve bayes	0.513	2
logistic regression	0.604	24
adaboost	0.826	21
random forest	0.680	5
bagging	0.694	43
gradient boosting	0.833	224

Note: the "Running time" includes training and test using 3-fold cross-validation

4.2 Benchmark

Gradient boosting and AdaBoost have the highest score. The score for gradient boosting is a little higher, but the algorithm requires 10 times in training. Since in this competition we are more concerned about the performance than training time, we choose gradient boosting as the algorithm.

The benchmark for a untune gradient boosting classifier is :

- Auc score: 0.833
- Parameters:

```
(init=None, learning_rate=0.1, loss='deviance',
max_depth=3, max_features=None, max_leaf_nodes=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, n_estimators=100,
presort='auto', random_state=None, subsample=1.0, verbose=0,
warm_start=False)
```

4.3 Feature Importance

The data set contains 306 features, however not all the features are equally important to the classifier. It is possible that some features are irrelevant. The Gradient Boosting provides a way to determine the importance to each feature. The Gradient Boosting classifier is based on decision trees. Individual decision trees intrinsically perform feature selection by selecting appropriate split points. This information can be used to measure the importance of each feature. The more often a feature is used in the split points of a tree, the more important that feature is.

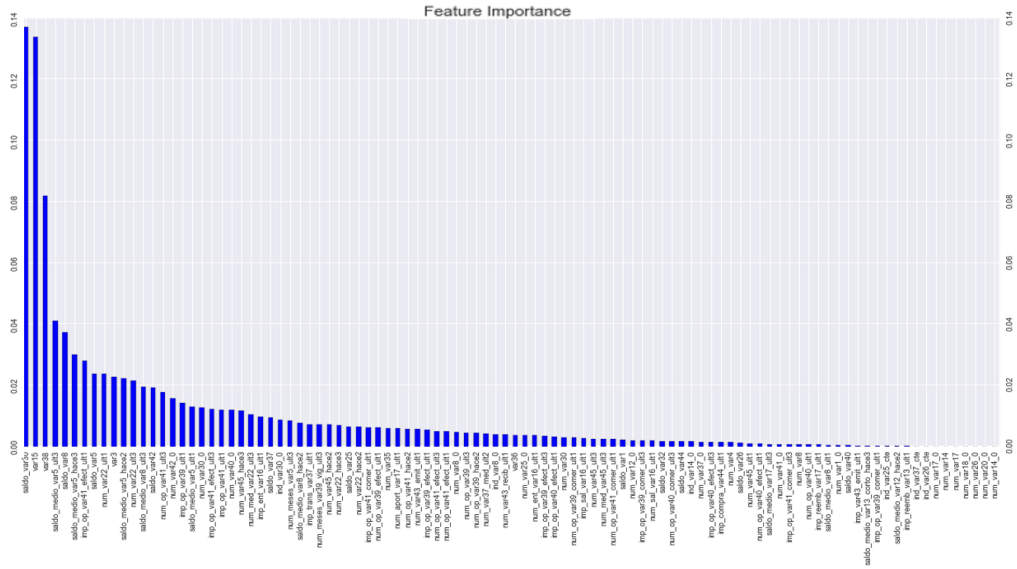


Fig.4 (a) Fist 100 most important features for the classifier

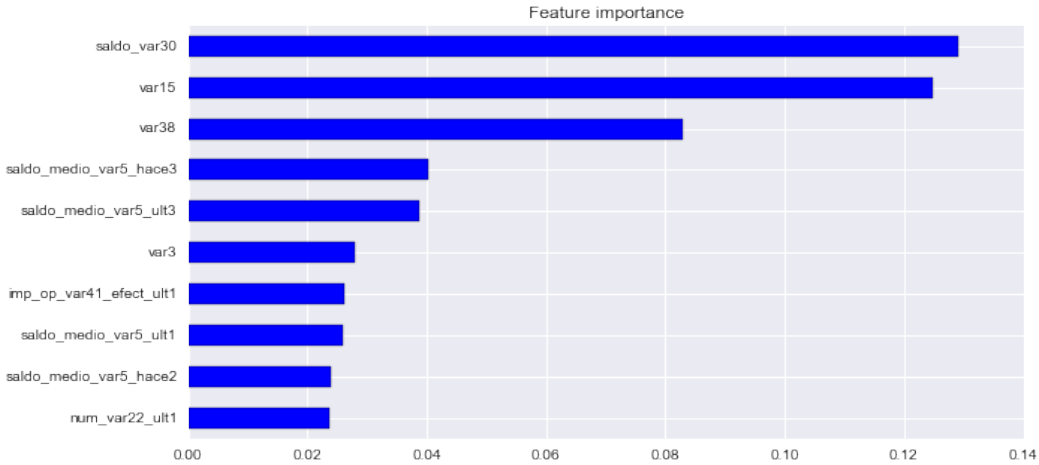


Fig.4 (b) Fist 100 most important features for the classifier

The first 100 most important features are shown in Fig 4. I found after 92 features, the feature importance drop to approximately 0. In order to accelerate the running time of the algorithm. I reduce the data set to the first 92 most important feaures.

- The reduced data set has 92 features.

4.4 Data Visualization

Since the data set is anonymized, I previously do not know any particular features to invest. However, through the Boosting algorithm we now know the feature in terms of their important.

We find the 5 features that are most important to the classifier, which will be detailly described in the later section 3.3. The 5 features are:

```
saldo_var30','var15','var38','saldo_medio_var5_ult3','saldo_medio_var5_hace3'
```

Firstly, we create a paired plot to shown the distribution of the features in Fig.5. The data from dissatisfied and satisfied classes are represented in the color green and blue, respectfully. The subplots on the diagonal show the distribution of each feature. The figures off the diagonal show the scatter plots of every combination of the 5 features and the data label. Interestingly, we find for the feature “var15” the distribution of the satisfied customer is more skewed, and the distribution of the dissatisfied customer is more normally distributed.

Since the data set is anonymized, we cannot imply more information from data visualization. However, we can get a general impression of the data set. The satisfied and dissatisfied customer data are largely overlapped. And therefore, the prediction can be a hard problem even for a human.

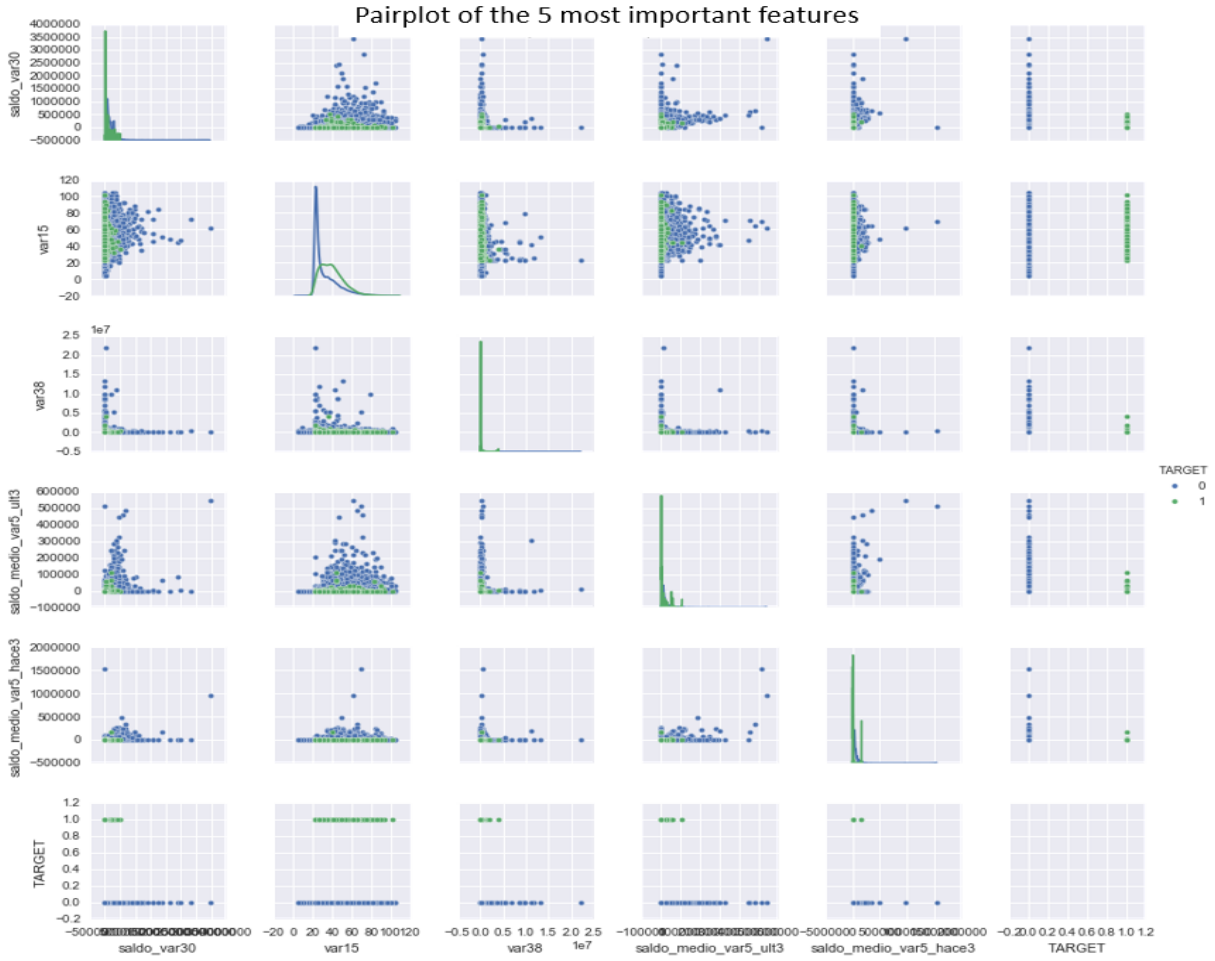


Fig. 5 Pair plot of the 5 important features and the label

Secondly, we plot the correlation between every pair of the 2 features. A relatively strong correlation can be found between variable 4 and 5, with Person's correlation factor of 0.45. The relation can be observed in Fig. 6.

Fig. 6 Correlation plot of the 5 important features

5. Model Refinement

The next step is to fine tune the classifier to improve its performance over the baseline model. I used the decision tree based weak classifier in the boosting methods. The tuning parameters can be divided into 3 categories according to [3]:

- Tree-Specific Parameters: These affect each individual tree in the model. e.g. 'min_samples_split', 'min_samples_leaf', 'max_depth'.
- Boosting Parameters: These affect the boosting operation in the model. e.g. 'learning_rate', 'n_estimators'.
- Miscellaneous Parameters: Other parameters for overall functioning. e.g. 'subsample'.

When tuning the parameter, I tried to tune the parameter with big impact first (e.g. max_features, n_estimators) and then tune the parameter with same impact such as subsample and learning_rate. The process of the parameter tuning is described as follows:

- Choose a relatively high 'learning rate'.
- Determine the optimum 'max_features'.
- Determine the optimum 'n_estimators'.
- Tune tree-specific parameters including 'min_samples_leaf' 'max_depth'.
- Tune the 'subsample' parameters.
- Fine tuning the 'learning rate'.

4-fold cross-validation is used in all the tuning steps to prevent overfitting. The results of each steps are detailed in the following sections.

5.1 Tune max_features

max_features determines the number of features to be considered while searching for the best split. The higher values can lead to over-fitting.

As a thumb-rule, squared root of the total number of features often works great. So I research max_features around 16 and found 18 has the highest performance score. Fig.7 shows the model performance under different max_features.

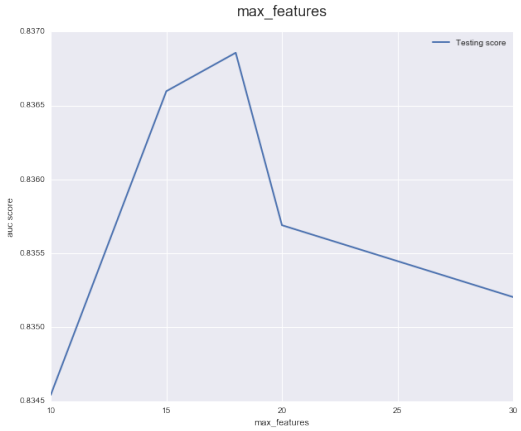


Fig.7 Tuning max_features

5.2 Tune n_estimators

Parameter n_estimators determines the number of sequential trees to be modeled, A relatively large learning_rate is selected (0.1) to reduce the training time. Fig.7 shows the model performance under different n_estimators. The curve peaks at 150 then decreases.

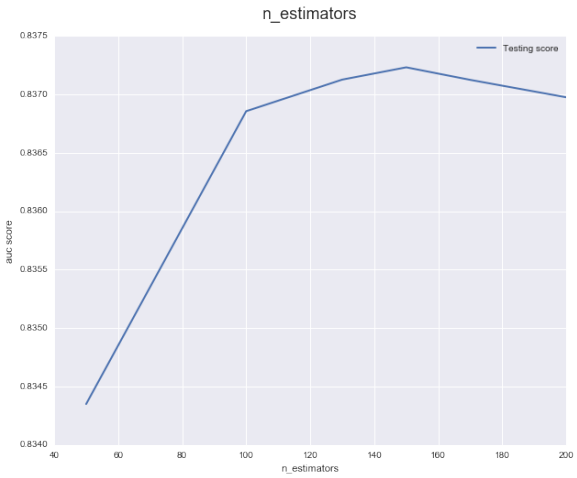


Fig.7 Tuning n_estimators

5.3 Tune tree-specific parameters

The next step is to tune the tree-specific parameters. Two of the most important parameters are max_depth, min_samples_leaf, and min_samples_split

- max_depth determines the maximum depth of a tree. The model with higher depth will tend to over-fitting because higher depth will allow the model to learn very specific to a particular sample.
- min_samples_leaf defines the minimum samples required in a terminal node. Higher values prevent a model from over-fitting the data.
- min_samples_split defines the minimum number of samples required in a node to be considered for splitting. Similar as min_samples_leaf, the higher values prevent a model from over-fitting the data. To reducing the searching space, we chose the number of min_samples_split to be 2 times the number of min_samples_leaf.

Fig.8 shows the result of tuning the tree-specific parameters. The 3 lines show the relationship of the auc score and min_samples_leaf under different max_depth. When max_depth is 3, min_samples_leaf is 5, and min_samples_split is 10, the model has the highest score.

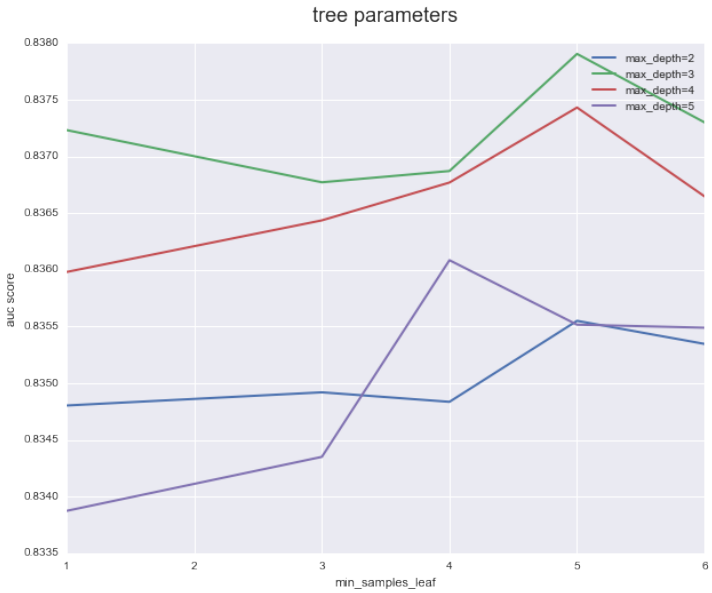


Fig.8 Tuning tree-specific parameters

5.4 Tune subsample

The next step is to tune the parameter 'subsample', which controls the fraction of observations to be selected for each tree. The selection is done by random sampling. The values slightly less than 1 make the model more robust. Fig.9 shows the result of tuning subsample. The optimal subsample: 0.95

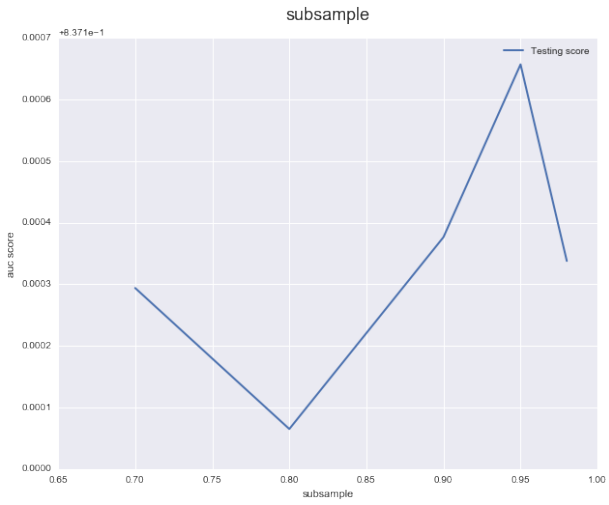


Fig.9 Tuning subsample

5.5 Fine Tune Learning Rate

The last step is to fine tune the learning rate. I decrease the learning rate and at the same time proportionally increase n_estimators. The optimal value appears to be at 0.05. Fig.10 shows the result of tuning subsample.

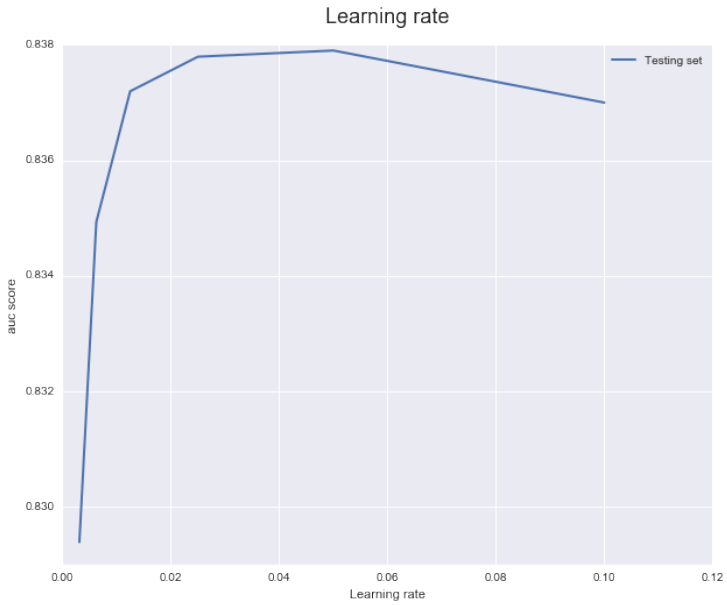


Fig.10 Fine tuning learning rate

5.6 Final Model

AUROC score: 0.838

Parameters:

- learning_rate: 0.05
- n_estimators: 300
- max_features: 18
- max_depth: 3
- min_sample_leaf: 5
- min_sample_split: 10
- subsample: 0.95

The AUROC of the final model have increased from 0.833 to 0.838 from the baseline model. Though the difference is only 0.005, in order to achieve this, I have search through a large parameter space in max_features, max_depth, min_samples_leaf, min_samples_split, subsample, and Learning_rate. The data size is relatively large. And thus, the 0.005 improvement is significant.

6. Model Validation

To validate the classifier, I plot the learning curve of the classifier using different numbers of data (Fig.9). The training set score is obtained by training and testing the model on the same data set. The testing set score is obtained using 4-fold cross validation.

As the number of data increases, the score of training set decreases, the score of the testing set increases, and difference between the two set decrease. The final difference between the training set and test set is small which

indicated the model is not overfitted. The final score of the test set is the highest among other models we compared in the previous section, so it indicates the model is not underfitted.

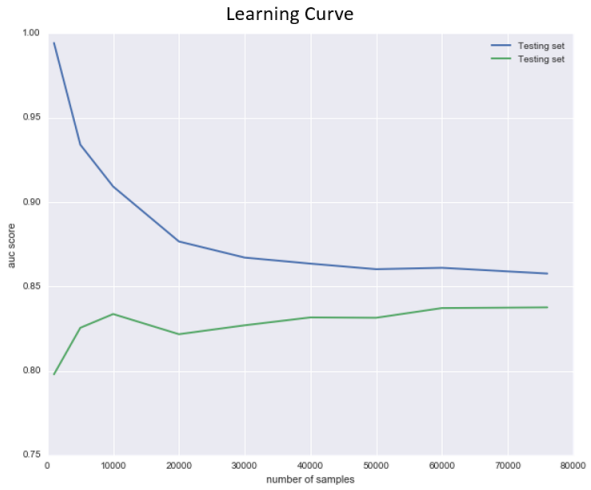


Fig. 9 the learning curve using different number of data

7. Conclusion

In this project, I have used anomalous data from Santander Bank to predict dissatisfied customers. Through exploratory data visualization, we find the problem is relatively hard because the satisfied and unsatisfied customer is overlapped. We have tried various model and find the ensemble model is the best suited for this data set. Gradient boosting is choose as the final model. The model has been fine-tuned through feature selection and parameter tuning. The model has been validated through cross-validation. By plotting the learning curve with a different number of data, it shows the model is not obviously overfitted or underfitted. The final model achieves the auc score of 0.838.

8. Reflection

- Since the data is anonymous, I have to rely more heavily on the machinery of the algorithm rather than human intuition.

- The problem is relatively hard. From the exploratory data visualization and the PCA analysis, we have found the satisfied and dissatisfied customer data is mostly overlapped. Later in model selection, the more complexed ensemble method have better performance than the simple classifiers such as “logistic regression”.

- Attending the Kaggle competition is an interesting experience. I have learned a lot through reading the problem discussed in the forum and learning from code shared by other participants.

9. Improvement

- The classifier achieved the auc score 0.838, and the best score in the Kaggle competition so far is 0.843. Some of the highest scores are obtained through the algorithm XGboosting (Extreme Gradient Boosting).
- Another way to squeeze the last drop of the performance is through parameter tuning. The Gradient Boosting algorithm has many parameters, which can take a long time to be optimized. By using more extensive and systematic tuning, I think a better performance can be achieved.

Reference

[1]Kaggle competition: https://www.kaggle.com/c/santander-customer-satisfaction

[2]Feature selection: http://scikit-learn.org/stable/modules/ensemble.html#feature-importance

[3]Guild to parameter tuning: http://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/

[4]XGboosting: http://xgboost.readthedocs.org/en/latest/model.html

Appendix

Constant Columns:

```
['ind_var2_0', 'ind_var2', 'ind_var27_0', 'ind_var28_0', 'ind_var28', 'ind_var27',
'ind_var41', 'ind_var46_0', 'ind_var46', 'num_var27_0', 'num_var28_0', 'num_var28',
'num_var27', 'num_var41', 'num_var46_0', 'num_var46', 'saldo_var28', 'saldo_var27',
'saldo_var41', 'saldo_var46', 'imp_amort_var18_hace3', 'imp_amort_var34_hace3',
'imp_reemb_var13_hace3', 'imp_reemb_var33_hace3', 'imp_trasp_var17_out_hace3',
'imp_trasp_var33_out_hace3', 'num_var2_0_ult1', 'num_var2_ult1',
'num_reemb_var13_hace3', 'num_reemb_var33_hace3', 'num_trasp_var17_out_hace3',
'num_trasp_var33_out_hace3', 'saldo_var2_ult1', 'saldo_medio_var13_medio_hace3']
```

Duplicate Columns

The columns before and after the colon have the same value. The column after the colon is removed.

```
{'delta_imp_reemb_var33_1y3': ['delta_num_reemb_var33_1y3'],
 'ind_var18_0': ['ind_var18'],
 'delta_imp_reemb_var13_1y3': ['delta_num_reemb_var13_1y3'],
 'ind_var26_0': ['ind_var26'],
 'ind_var25_0': ['ind_var25'],
 'num_var6_0': ['num_var29_0'],
 'num_var26_0': ['num_var26'],
 'ind_var40': ['ind_var39'],
 'ind_var37_0': ['ind_var37'],
 'num_var18_0': ['num_var18'],
 'delta_imp_trasp_var33_in_1y3': ['delta_num_trasp_var33_in_1y3'],
 'saldo_var13_medio': ['saldo_medio_var13_medio_ult1'],
 'num_var40': ['num_var39'],
 'num_var34_0': ['num_var34'],
 'num_var32_0': ['num_var32'],
 'ind_var13_medio_0': ['ind_var13_medio'],
 'num_var6': ['num_var29'],
 'num_var13_medio_0': ['num_var13_medio'],
 'ind_var32_0': ['ind_var32'],
 'delta_imp_reemb_var17_1y3': ['delta_num_reemb_var17_1y3'],
 'delta_imp_trasp_var17_in_1y3': ['delta_num_trasp_var17_in_1y3'],
 'saldo_var6': ['saldo_var29'],
 'ind_var34_0': ['ind_var34'],
 'num_var37_0': ['num_var37'],
  'num_var25_0': ['num_var25'],
 'ind_var6_0': ['ind_var29_0'],
 'delta_imp_trasp_var33_out_1y3': ['delta_num_trasp_var33_out_1y3'],
 'ind_var6': ['ind_var29'],
 'delta_imp_trasp_var17_out_1y3': ['delta_num_trasp_var17_out_1y3']}
```