Classification Methods for Bank Marketing

Yanxi Cai

Xiao Song

abstract

This paper will perform various of classification algorithms in bank marketing field. The class to be predicted is binary, positive response or negative response. We will consider normalizing the dataset and using sampling method to balance the data. Then we choose Naïve Bayes, Random Forest, Adaboost and SVM to build classifier model, and use accuracy, sensitivity, ROC curve, AUC and complexity to evaluate the performance of each model, and finally get the best one among them.

Key words: undersampling, Naïve Bayes, Random Forest, Adaboost, SVM, evaluation, ROC, AUC, sensitivity, accuracy

Introduction

In the real world, during bank marketing campaigns, usually small portion of customers will actually buy the products. Due to limited resources, it's very important for the bank to find the customers who are most likely to buy the product, which will significantly increase bank's efficiency. So, the question is how to predict which customers tend to give a positive response? The target feature is binary, 'yes' or 'no'.

Therefore, the motivation of this paper is to find a classification algorithm with best predictive capability. We will first consider the basic classifier, Naïve Bayes. Then ensemble learnings, Random Forest and Adaboost, as well as SVM algorithm are introduced. Finally, we will use various methods to evaluate the pattern.

Related works

Haya and Mohammad (2016) proposed a new approach, a Hybrid of Data-level and Algorithmic-level solutions (HybridDA) to deal with imbalanced data. He used the data from marketing campaigns of a Portuguese bank. He combined random oversampling technique SMOTE and random sampling, and then he optimized the parameters in SVM using a grid search. Finally, he proposed a model which performed better than other models on the same dataset.

Olatunji Apampa (2016) also performed classification algorithms in bank marketing field. He used Logistic Regression, Naïve Bayes, Decision Tree, and Random Forest as classifiers. His study showed that the features duration, poutcome, contact, month and housing were the most important features that contribute to the success of the bank customer marketing campaign for deposit subscription. It also dressed that balanced data could improve all the performance of algorithms he used.

Anatoli Nachev(2015) performed data mining modeling in direct marketing. To avoid the issue that random sampling may create some 'lucky' training sets that significantly better than others, they combined cross validation, multiple runs on random selection of the folds and initial weights, and multiple runs on random selection of partitions. They also compared several algorithms

including logistic regression, Naïve Bayes, linear and quadratic discriminant analysis and neural nets, and neural nets performed best. They evaluated the models using ROC curve and AUC.

Elsalamony (2014) used data mining models to identify the most important features that affect the response from customers in marketing campaigns. He performed Multilayer Perception Neural Network, Logistic Regression, C5.0 Decision Tree and Tree Augmented Naïve-Bayes to build the model. He proposed that C5.0 Decision Tree is the best performer.

Wisaeng (2013) compared different classification algorithms for bank direct marketing. He tested 4 algorithms, J48-graft algorithm, SVM, LAD tree algorithm and radial basis function network. Finally, he evaluated the pattern using sensitivity, specificity, accuracy, mean absolute error and root mean squared error. The result is SVM performed better than the others.

Moro, Laureano and Cortez (2011) used data mining algorithms to find potential buying customers in bank marketing campaigns. They collected data from a Portuguese bank and performed the Cross Industry Standard Process for Data Mining methodology. They used Naïve Bayes, Decision Tree and SVM as classifiers and SVM was proven to be the best one. They also dressed that 'duration' is the most important variable.

Generally, the related studies show that SVM is a good classifier, and balancing the data can improve the performance of the algorithms.

Data description

Our dataset is collected from bank marketing campaigns. We have 41188 instances, 20 features, as shown in Table 1, including numeric features, such as age, consumer price index, and categorical features, such as education, job and so forth. The target feature y (has the client subscribed a term deposit?) is binary, 'yes' or 'no'. The data is very imbalanced with 4640 positive instance and 36548 instances.

Table 1

	Feature	Description	Type
1	age	customer's age	numeric
2	job	type of job	categorical
3	marital	marital status	categorical
4	education	Education level	categorical
5	default	has credit in default?	categorical
6	housing	has housing loan?	categorical
7	loan	has personal loan?	categorical
8	contact	contact communication type	categorical
9	month	last contact month of year	categorical
10	day_of_week	last contact day of the week	categorical
11	duration	last contact duration, in seconds	numeric
12	campaign	number of contacts performed during this campaign and for this client	numeric

13	pdays	number of days that passed by after the client was last contacted from a previous campaign	numeric
14	previous	number of contacts performed before this campaign and for this client	numeric
15	poutcome	outcome of the previous marketing campaign	categorical
16	emp.var.rate	employment variation rate	numeric
17	cons.price.idx	consumer price index	numeric
18	cons.conf.idx	consumer confidence index	numeric
19	euribor3m	euribor 3 month rate	numeric
20	nr.employed	number of employees	numeric

Methodology

Data preprocessing

We first discarded feature 'duration' because it is not known before a call is performed. Also, after the end of the call y is obviously known. So, it can't be used in predictive model. Also, feature 'default' has 21% missing values. To obtain more data, we also delete it. Then removed all the missing values. After that we check outliers for the data and there is no obvious outliers. Then we normalized the data using standard scaler and created dummy variables for the categorical features. So far, we have 38245 instances left, and 55 features, and target y has 4258 'yes' and 33987'no'.

Because undersampling has been proposed to be a good technique balancing the data, also in our case, we have 4258 minority class, which means we have sufficient number of instances to build the classifier model, we decide to use undersampling to balance the data. We build a basic decision tree model and choose accuracy and sensitivity to compare balanced and imbalanced data. Results shown as Figure 1, Balanced data performs significantly better than imbalanced data.

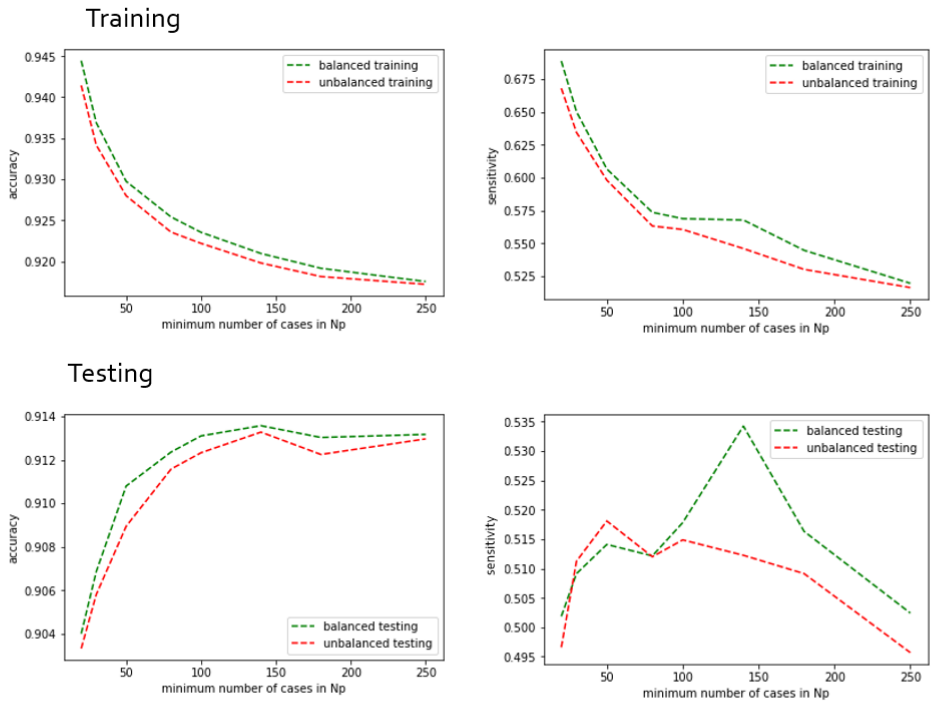


Figure 1

We also did 2-components PCA to visualize the data. After balacing the data, positive responses (red plots) ralively stand out, which means it's much easier for us to detect positive responses.

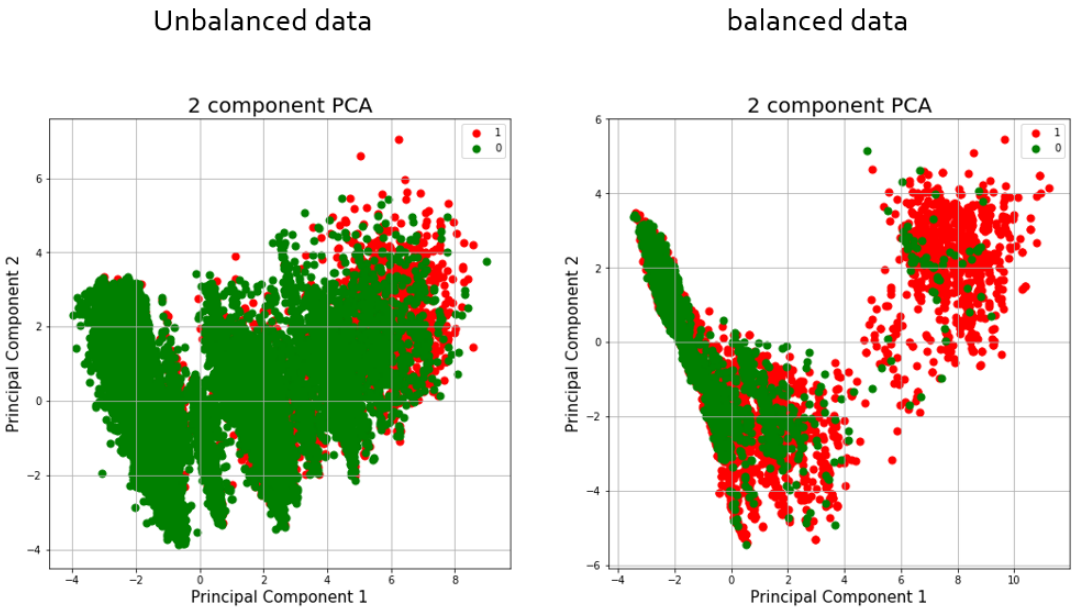


Figure 2

Pattern discovery

Naïve Bayes:

The Naïve Bayes classification algorithms entails the process of determining classification based on Bayes theorem of posterior probability. It is assumed that the values of each features in the train datasets are independent of one another. Bayes method learns the conditional probability of each attribute given the class label from the training data and then computes the probability of a class value given the particular instance and predicting the class value with the highest probability.

Random forest:

Random forests is an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training set and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

AdaBoost:

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

Support Vector Machine:

SVM is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well (look at the below snapshot).

Result

Naïve Bayes:

We repeat 100 times on holdout data partition process and record the mean of the accuracy, confidence intervals and sensitivity, and running time on training and testing data. The parameter we tuned for optimization is number of iterations. And we get two graphs for number of iterations to accuracy and sensitivity on training and testing data.

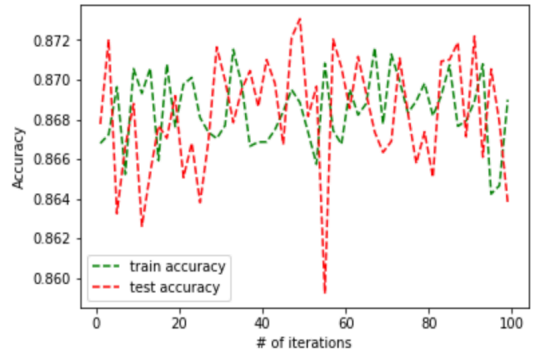


Figure 3

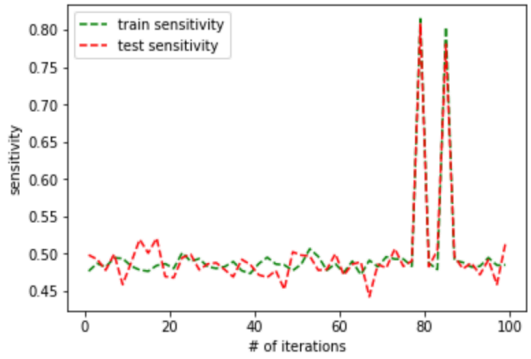


Figure 4

As figure 3 and figure 4 shows, there is not clear pattern in the iterations to accuracy but there is a big improvement in the chart of number of iterations to sensitivity when the value of iterations equals 83. And when number of iterations equals 83, the performance on the accuracy chart is good as well. So we decide that when the number of iterations reach 83, we captured the optimal performance of naïve bayes classifier. We then run the model and record scores: mean of accuracy on training data: 0.8686. CI:

[0.8686, 0.8686]. The mean of accuracy on testing data: 0.8681. CI: [0.8681, 0.8681]. The sensitivity on the training set is: 0.4835 and on the testing se is:0.4802. The running time of this algorithm is: 2.0114 seconds.

Random Forest:

There are three parameters we take into account when we apply the random forest algorithm:

number of trees: bigger the better but may increase complexity; min_samples_split: start with sqrt of number of samples; min_samples_leaf: half of "min_samples_split". First, we fix the min_samples_split as sqrt of number of samples which equals 92. We also fix the number of the min_samples_leaf as 46. Then we only changing the number of trees. From the graph: number of trees to error rate on training and testing sets we found out that when number of trees equal 173, both error rates on training set and testing sets are pretty low and they are close to each other. We choose 173 as our parameter for number of trees.

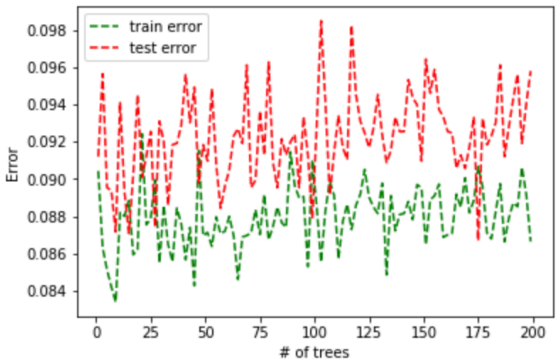


Figure 5

Then we fix number of trees as 173 and changing the value of min_samples_split from 50 to 92 and test the performance.

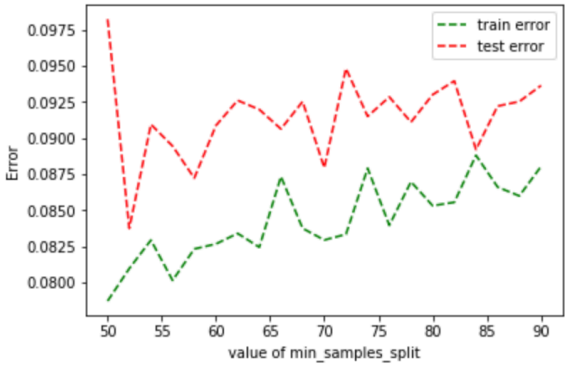


Figure 6

From the graph we can see that when min_samples_split equals 53. Both error rates on training and testing sets are low and are pretty close to each other. Then we choose 53 and 26 as our parameters for min_samples_split and min_samples_leaf, respectively. We record optimized combination and compute accuracy, sensitivity and complexity on training and testing sets. The accuracy on the training set is:

0.9050. The accuracy on the testing set is: 0.9039. The sensitivity on the training set is: 0.1863 and on the testing set is: 0.1927. The running time of the algorithm is 1.4572 seconds.

AdaBoost:

We set the base estimator as decision stump. The parameters we test on are iterations and learning rates. The learning rate is the contribution of each model to the weights. The reducing of the learning rate will mean the weights will tend to be a small degree, forcing the model train slower. We fix the iterations as a constant and test the learning rate first and get the output.

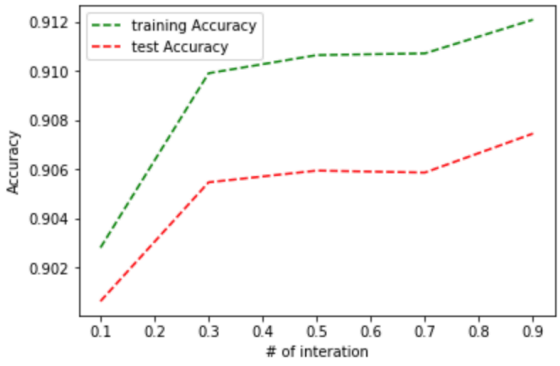


Figure 7

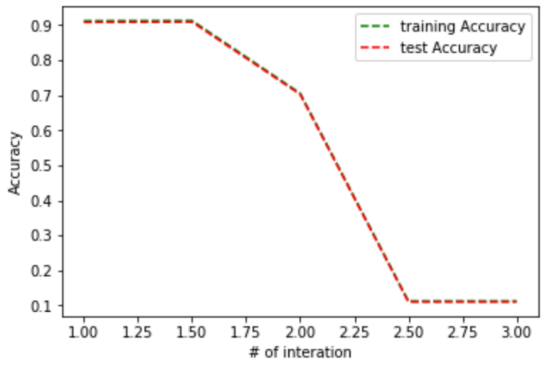


Figure 8

From the output it's easy to see that there is a positive relationship between learning rates and accuracies on training and testing data when learning rate's range in (0, 1) and a negative relationship when learning rate larger than 1. So, we pick 1 as our optimal parameter as learning rate and next test on the iterations. From the graph we can see that the accuracies on the training and testing data climb very fast when the value of number of iterations increase.

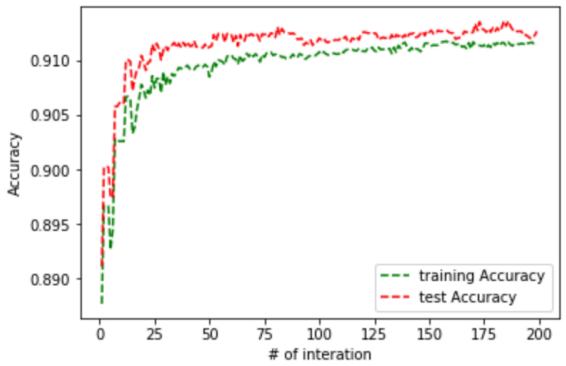


Figure 9

However, when the value of number of iterations exceeds 50, there isn't significant improvement anymore, so we don't need to pick a very larger number in case to add more complexity on the algorithm. Finally, we choose 162 as our optimal parameter as number of iterations.

We record optimized combination and compute accuracy, sensitivity and complexity on training and testing sets. The accuracy on the training set is: 0.9146 and on the testing set is 0.9088. The sensitivity on the training set is: 0.4210 and on the testing set is: 0.4020. The running time on the algorithm is: 3.0089 seconds.

SVM:

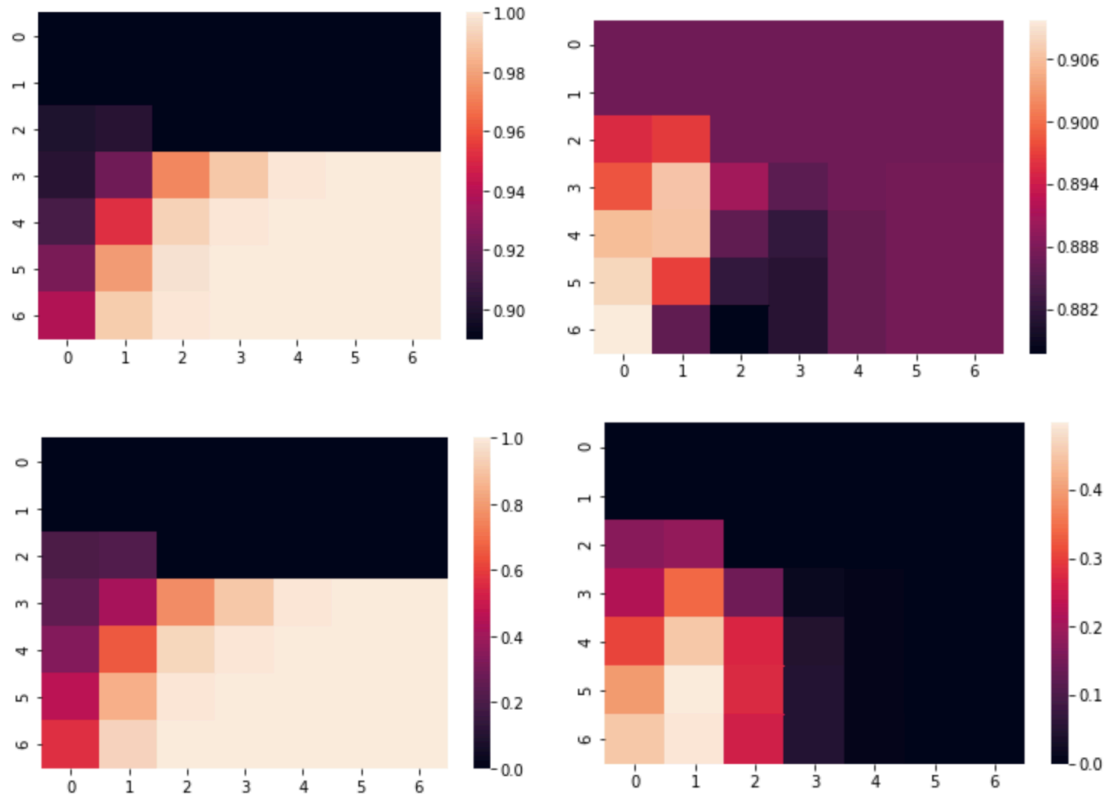
We test a linear and a non-learn Support Vector Machine on both training set and testing set. We control C as the parameter for both linear and non-linear SVM. Another parameter is gamma for non-linear SVM. The table shows the comparison of performances on the linear and non-linear SVMs.

Table 2

Classifier	**Accuracy on training set**	**Accuracy on testing set**	**Sensitivity on training set**	**Sensitivity on testing set**
Linear-SVM	0.9045	0.9013	0.2938	0.2751
Nonlinear-SVM	0.9730	0.8905	0.7747	0.1443

So, based on the performance we can see that non-linear SVM is better than linear SVM. Then we assign same ranges to C and gamma(1e-3,1e-2,0.1,1,10,100,1e3) for parameters tuning.

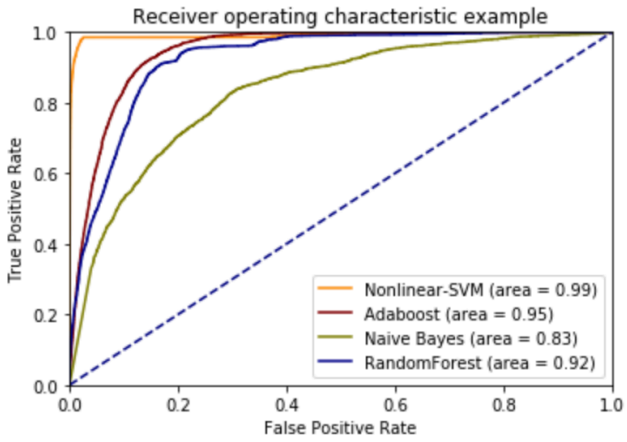
We get four heatmaps for the performance on both training and testing set as we use non-linear SVM.



The two maps in the first row represent the accuracy on the training and testing sets. The two maps in the second row represent the sensitivity on the training and testing sets. In each heatmap, the row represents C and the column represents gamma. In addition, the color in the heatmaps means the lighter the better performance on the datasets. We need to find a best spot which is in the same location in each map and it keeps us in mind that too large for C causes hard margin and too small for gamma, the model is too constrained and cannot capture the complexity or "shape" of the data. After compare four maps we think when C=1 and gamma=0.1, we could achieve optimal overall performance. Next, we record performance scores. The accuracy on training set is: 0.9730 and on testing set is: 0.8905. The sensitivity on the training set is: 0.7747 and on the testing is: 0.1443. The running time on the algorithm is: 451.0346 seconds.

Comparison:

So far, we have tested all four different types of classifiers and voted each best parameter for performance. Then let's take a look and do a comparison to vote a best classifier for this dataset. We decide to run ROC curves and compute the area under ROC curves for four classifiers.



From the ROC curves chart we can see that non-linear has the best performance and naïve bayes got the worst performance. Adaboost and random forest are competitive in the middle. We integrate auc with performance scores in a table to be convenient for comparison.

Table 3

Classifier	**Accuracy on training**	**Accuracy on testing**	**Sensitivity on training**	**Sensitivity on testing**	**Running time(seconds)**	**AUC**
NB	0.8686	0.8681	0.4835	**0.4802**	2.0114	0.83
RF	0.9050	0.9039	0.1863	0. 1927	1.4572	0.92
Adaboost	0.9146	**0.9088**	0.4210	0.4020	3.0089	0.95
NonlinearSVM	**0.9730**	0.8905	**0.7747**	0.1443	451.0346	**0.99**

So, from the table we can see that in terms of the overall performance non-linear should be voted for the best for this particular dataset. But on the other hand, the running time on the running time is horrible which can't be ignored.

Conclusion

As we know data preprocessing is a very important processing when we do data analysis. Outliers detection and normalization are what we consider usually, however, balancing data is also important. In this case, we got better performance after we applied undersampling on the training set compared to just run algorithm on dataset without balancing.

Nonlinear SVM stands out among four classification methods for this particular task with 0.9730 of the accuracy on the training set, 0.7747 of sensitivity on the training set and 0.99 of area under ROC curves. But complexity is a problem can't be ignored. For this particular dataset, especially after we applied undersampling technique, we dramatically reduce samples from 40 thousand to 8 thousand, however we still got 451 seconds of running time on the model, and this is only a binary class problem. So, what we suggest is that when you test your classifiers on the dataset, if SVM has the best performance but the worst complexity, you may choose the second better classifier as your optimal classifier. It's a trade-off but worthy to deal with large scale dataset.

Parameter tuning is essential for optimizations. Even if we have voted for the best classifiers, there are still space to optimize the performance by tuning the relevant parameters.

Future works

There are two popular sampling methods for imbalanced data. We only tried undersampling. But in the future, we should test Do SMOTE oversampling and center centroid undersampling and compare their performance. We haven't considered all parameters that are relevant to the classifiers we were dealing with. More of parameters tuning in the algorithms is necessary. In addition, feature importance can't be found in Non-linear SVM, so try to get the most important features in other algorithms.