EE-559 Mathematical Pattern Recognition

Spring 2018

Project

Name: Jinpeng Qi & Shuang Yuan

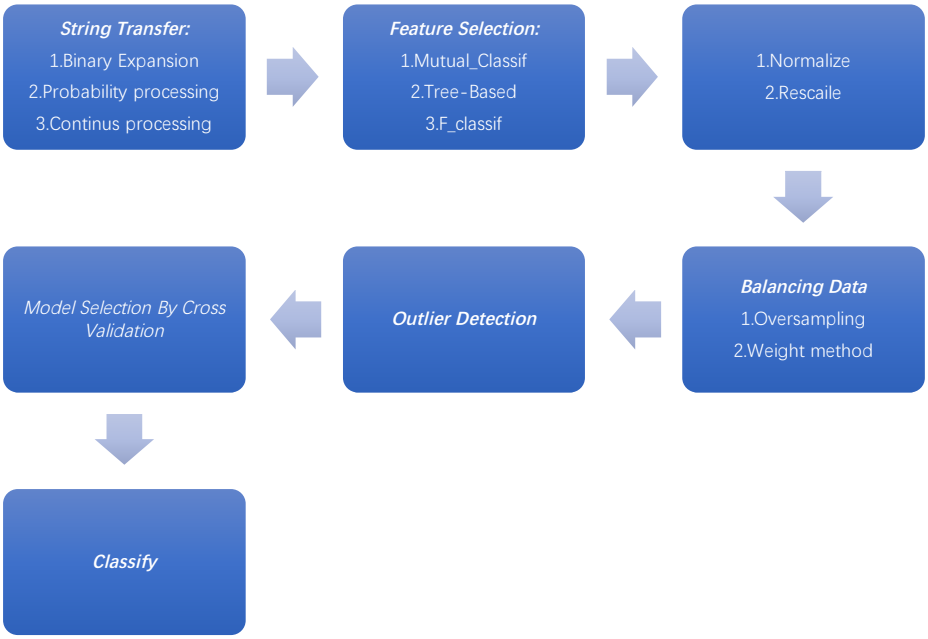
Email: jinpengq@usc.edu

shuangy@usc.edu

04/30/2018

Abstract

The possibility of a client being inclined to subscribe to a long-term deposit after a call depends on a lot of factors such as job, education, loan and so on. It is clear to see that there are only two choices will the client make- yes or no, indeed leading to a two-class classifier problem. To begin with, the bank marketing data set is split to two parts by selecting ten percentages data as test dataset and rest dataset as training dataset. Notice we would not touch the test dataset and make all the processing procedure on the trainset. Before training dataset being sent to the classifier model, it needs a substantial preprocessing work including string transfer, treatment of missing data point, data balancing, normalization or rescaling, feature dimension adjustment and outlier detection. Then the processed training dataset will be sent to six different classifiers (Naïve Bayes, SVM, KNN, RF, LR, Adaboost) to train and applying on test dataset. It was found that KNN, SVM and AdaBoost could give the optimal results after model selection based on different processing techniques. The approximate optimal result is given on **F1_score of 0.87** and **AUC of 0.79.**



Pre-processing of dataset

1.1 Generating test dataset

Randomly pick up 10% percentage of the raw dataset as the test set. The bank-additional dataset whose size is 4119x63 will be split to two datasets-the train dataset(3669x63) and test dataset(451x63). Only preprocess the training dataset and apply it to different pattern recognition systems. The test set will remain untouched through the whole process since in reality, we do not know anything about the test dataset. When performing the test classifying, we will use the result of the all the processing on train model to apply to the test model.

1.2 Recast representation of data and missing point treatment

It can be seen from this dataset that there are two feature types which are numerical and categorical. Firstly, we need to unify the data types as values so we replace the strings with values. We take three measures to do this:

1.2.1 Categorical Feature Expansion

For each feature, there are several variables for each categorical feature and expanding every variable as a new feature (including the unknown feature) using numerical to binary conversion. This processing will help in highlighting hidden information and we get 62 features (not including the label column). With this procedure, we assume all feature matters and do not analyze each feature' s importance or correlation with other features and replace the unknown data (in a way we add some noises if we do this) which we could keep the dataset complete. After that, we will use different feature selection techniques to select the proper features in this dataset so that uncorrelated features or highly correlated features will be deleted from the dataset. The issue is that we will attain a dimensionality expansion problem which could cause computation complexity problem but we will reduce the dimensionality by using proper selecting features method such as **ExtraTreesClassifer** which could select 20 out of 60 features with high importance in the entire dataset and will give a satisfied result as well.

The reason why not applying PCA to lower dimension is that our principle is that all

feature matters so we proceed binary string transfer processing. However, it is also possible that some features may have not that much correlation in the whole dataset (also could be explained that they are not that important in the whole dataset). By applying feature selection, we could remove them and lower the dimension as well but PCA sometimes will work badly when many uncorrelated information is included in the whole dataset which after operating PCA, it is difficult to separate these data points and will cause bad performance.

We test this phenomenon in the dataset and found that after binary feature expansion and preforming PCA to adjust the dimension, the F1 score will lowered about 10 percent. Run SVM classifier to explain this:

	Accuracy	F1 Score	AUC
PCA	0.75	0.74	0.64
N_PCA	0.84	0.83	0.73

We will explain this more thoroughly later.

1.2.2 Continuous processing

We find different unique string on each feature and set continuous value to it. As to the 'unknown' and 'nonexistent' string which we think it as **missing data point**, we fill up in the corresponding classwise feature means by using **imputer** in sklearn. After processing, we get 19 features (not including the label column). For some special features, we observe such as poutcome feature, this feature had more than 93% of nonexistent data which we may think of this feature hold irrelevant information, but if the data is success then it is must class1 data and failure it is must class 2 data. Thus, there are also some relevant information in this feature and we could not delete it. Thus, we still incline to let the feature selection technique to handle this.

1.2.3 Probability processing

Based on above processing thoughts, we put forward this method to give a better classification pre-processing. For each feature, we calculate the ratio of each variable with labeled yes on that feature and replace each string with corresponding ratio. For example, for a two-class problem with one feature owning just two variables x_1 and x_2. We will replace:

$$x_1 = \frac{number\ of\ x1\ labeld\ yes}{number\ of\ x_1}$$

Then the same method applies to x_2. This replacement means the probability of variables of each feature will have the influence of long deposit. Repeat this process for each feature.

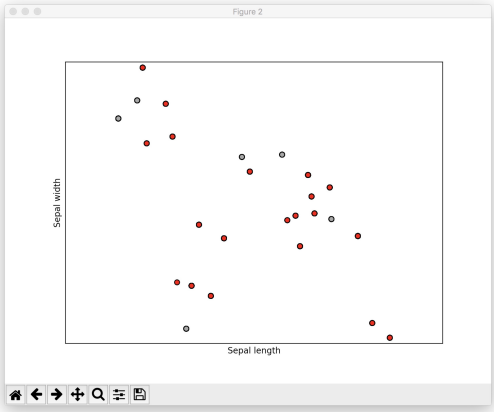
This method gives a better performance than the continuous processing since it is much proper to assign the probability of customers will choose to deposit after receiving a marketing call to the string of each feature which could more reflect the meaning of each variable presenting rather than assign the continuous number to each string. The second advantage is that the discriminative level in one feature column will increase which could better represent each variable' s influence level. And this method gives a great performance compared to the other two. In result part, the comparison will be given.

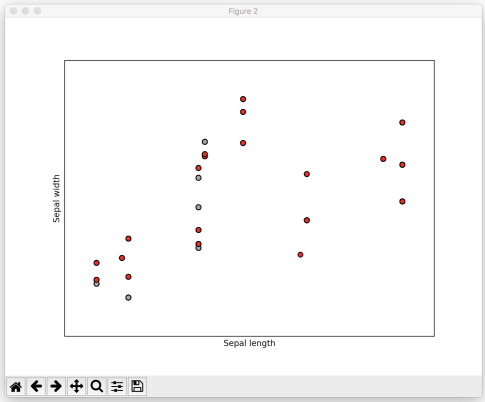
1.3 Feature-space dimensionality adjustment

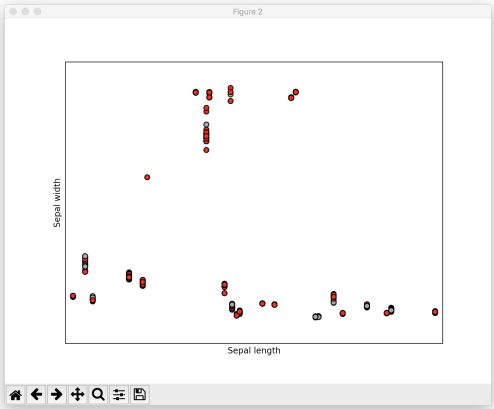
1.3.1 PCA

After we applying feature expansion method for recast the representation of the data and we get 62 features. We could use Principle Components Analysis to project it to a lower dimensional space. We set the parameter whiten = true which makes it possible to project the data onto the singular space while scaling each component to unit variance. However, PCA does not work very well for all kinds of models which we project the dataset into a dimension which does not properly separate two class data points very well. We could calculate the covariance matrix and find the best line which the dataset will project into it which is kind of complicated. We take advantage of other method which is use the **scatter** in sklearn to see each two columns' distribution. In our dataset, if we perform binary expansion method, PCA does not perform very well for after applying it, the accuracy and f1 score decreases 0.2 for Bayes classifier and 0.1 for SVM and RF. Thus, we do not prefer to apply PCA in the trainset model, we just perform feature selection.

By using scatter function, pick 2 features at a time, we found that after binary expansion, some features could not be separated by observing their distribution as following (from left to right: 57th and 58th feature, 58th and 59th feature, 60th and 61th feature):







However, for probability continuous preprocessing, we could utilize PCA to lower the dimension to 10 and it gives a satisfied result since in this case the assumption based on the first preprocessing method does not exist at all.

1.3.2 LinearDiscriminantAnalysis dimensionality reduction

LDA is also another method to reduce the feature by projecting the dataset to the most discriminative directions. However, for our dataset, there are only two classes and using LDA we will only reduce the dimension to C-1 = 1 dimension which is improper and inappropriate for this dataset. So, we will drop this method.

1.3.3 Univariate feature selection

This kind feature dimensionality reduction depends on the dependence among different features, if some features are almost independent to other features, we assume these features are uncorrelated so we will remove them from the dataset.

We use this technique to select the best features based on univariate statistical tests. In this project, we use the **mutual_info_classif** method in **sklearn** to estimate mutual

information of the dataset for a discrete target variable. Mutual information between two random variables means the **dependency** between the variables. Provided it equals to zeros only if two random variables are independent which means low correlation and higher value means high dependency. By utilizing **mutual_info_classif**, we obtain a series of mutual information value and delete all those features which owns the value of mutual information is zero which means high uncorrelated level in the whole dataset. In the dataset, we rearrange the dimensionality of the feature by reducing to about **40** features. This is the first method we implement and it still owns a high dimensional feature space so we also introduce the following feature selection method.

1.3.4 Feature Selection using SelectFromModel in sklearn

Besides mutual_info_classif method, we also tried the Tree-based feature selection model which Tree-based estimators can be used to compute feature importance, which in turn can be used to discard irrelevant features. By using **ExtraTreesClassifer,** it will delete corresponding irrelevant features based on the evaluation of importance of features on classifiers. After selecting, the number of features which will be remained fluctuate in the range of 18 to 25. In this way, by reducing some number of features, it also achieves the aim of adjusting the dimensionality.

1.3.5 F_classif feature selection

The methods based on F-test estimate the degree of linear dependency between different features. Applying this method, it takes an input a scoring function that returns univariate scores and p-values which manifests the probability for a given model that the statistical summary would be the same as or greater magnitude than the actual observed results. Provided the p-value of one feature is greater than 0.5 which means the statistical significance of evidence is small, it will be deleted from the dataset.

When finishing probability string transformation, by trying above three feature selection methods, it could be found that the third one will perform better in most classifiers. The features have been deleted are default and ‘day of weeks’. As to default feature, we could see that most of the variables are ‘unknown’ and ‘no’ which provide little information about the classifying which we name this kind of feature highly uncorrelated.

1.4 Data Unbalanced Problem

1.4.1 Oversampling

Bank Marketing Data Set is inherently a 2-class problem but there is unbalanced problem which most data points belong to class 1 (3669 data points) and very few data points belong to class 2 (451 data points). Thus, the ratio of class1 to class2 is about 8.1. Actually, many classifiers are sensitive to unbalance in the predictor class. For example, a machine learning model that has been trained and tested on such a dataset could now predict label 1 for all samples and still give a high accuracy which an unbalanced dataset is biased toward the more common class.

We need to address this problem by balancing all the data to the same number level which is oversampling. With oversampling, we randomly duplicate samples from the class with fewer instances and we add **10 percent** standardization Gaussian noise to each duplicate sample, so as to match the number of samples in each class.

For balancing, one problem is that we should not just duplicate the number of data point which is labeled class 2 so we add the Gaussian noise to give the whole expanded dataset a reasonable distribution but also there is risk of blurring the boundary by adding many messy data points. Thus, we could run multiple of times to average the result and use the **outlier detection** to eliminate the outlier data point.

1.4.2 Weighted Samples

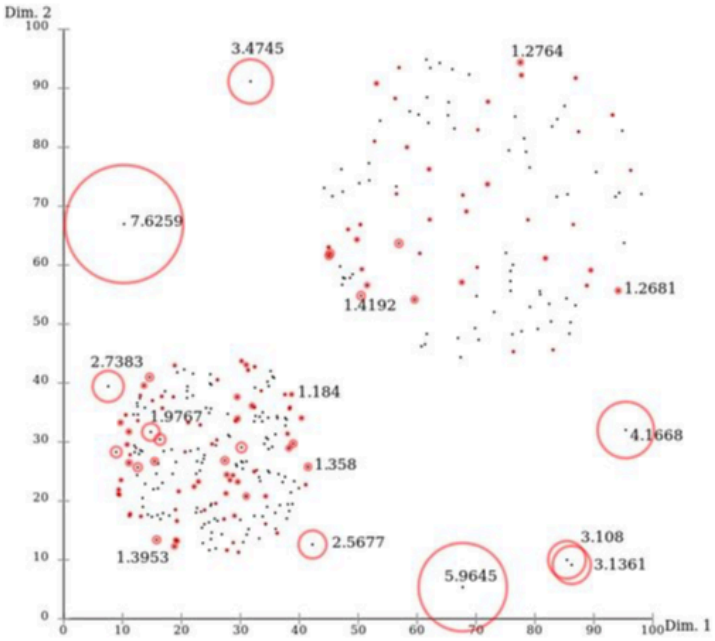
Since our data points are unbalanced, set some weight to let the classifier puts more emphasize on getting data points right. The ratio of class 1 data points and class 2 data points are almost 8:1. Therefore, for some classifiers such as Bayes, SVM, Logistic Regression, Random Forest, we could set the corresponding weight to each class which in this way we do not need to oversample the data. After testing, we find the result is similar with applying oversampling method.

1.5 Outlier Detection

LOF (Local Outlier Factor) is a density-based algorithm whose core part is the description of the density of data points.

- For each data point, calculate its distance from all other points and sort them from near to far;
- 2. For each data point, find its k-nearest-neighbor and calculate the LOF score.

  According to the definition of local anomaly factor, if the LOF score of data point p is close to 1, it means that the local density of data point p is similar to that of its neighbors; if the LOF score of data point p is less than 1, it means that data point p is in a relative position. The dense area is not like an anomaly; if the LOF score of the data point p is much larger than 1, it means that the data point p is more distant from other points, which is probably an abnormal point.



1.6 Normalizing and scaling data

Notice dataset features with different orders of magnitude so we want to make it easier to interpret results and take advantage of **standardization** in sklearn. Preprocessing to transform the data to center it by removing the mean value of each feature, then scale it by dividing non-constant features by their standard deviation. Normalizing is better suitable for non-statistical classifier such as KNN, RF.

For statistical classifier, such as Bayes, we prefer using scaling technique to obtain better performance since using normal technique will influence the dependence between each feature which will influence the performance a lot. In this project, we also find that for SVM scaling dataset will have better performance.

1.7 Cross Validation

In order to avoid the problem of overfitting, we use cross validation for which we held out part of the available dataset which is the validation set and training on other parts then repeat the procedure for holding out every part of the available data.

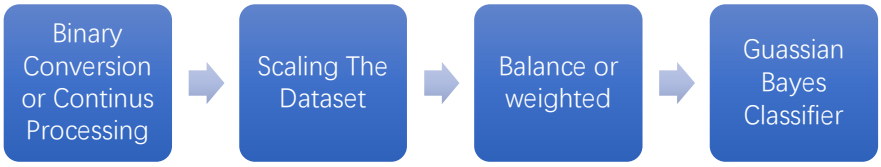
Also, for the aim of testing set could not peeking into the model, we define the validation set which training proceeds on the training set, after which evaluation is done on the validation set, and when the experiment seems to be successful. To elaborate, in our project, we use cross validation to pick up good estimate of parameters for different classifiers, performing the error estimation and model selection. Then we apply the result of validation set to the test dataset.

For the parameter selection, we use cross validation to find the **C and gamma** for SVM, **C** for Logistic Regression, **n_estimators** for Random Forest and **neighbors** for KNN, **n_estimators and learning rate** for AdaBoost.

Classifiers

This project implements five classifiers: Naïve Bayes, SVM, KNN, Random Forest, Logistic Regression and Adaptive Boosting. Cross validation is used for diverse classifiers to choose good parameter.

1. Naïve Bayes classifier



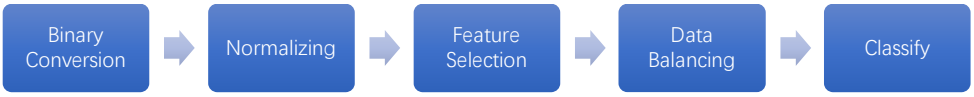
Naïve Bayes methods belong to supervised learning algorithms and make assumption of independence for each pair of features. Thus, if each pair features are more independent, then the classifier will own a better performance. Notice that standardization of a dataset might influence the dependence between each feature so for statistical classifiers such as Naïve Bayes, it is rational to dealing with dataset by

rescaling to keep the dependence between each feature unaffected.

Next, since Naïve Bayes model process based on the probability density function and the different naïve Bayes classifiers differ mainly by the assumptions they make regarding the distribution methods such as 'Gaussian', 'Multinomial', 'Bernoulli' were tried to train the dataset and it is found 'Gaussian' is a little better than other two which could achieve weighted **F1 score of 0.86 and AUC of 0.71.**

It has been found that without much preprocessing such as feature selection, data balancing, Bayes classifier could give above results which means that this dataset incorporate a high independence between each feature.

2. KNN classifier



This model proceeds based on the **KNeighborsClassifier** in **Python Nearest Neighbors Classification toolbox.** Basically**,** it is computed from a simple majority vote of the nearest neighbors of each point which that point is assigned the class that has the most representatives within the nearest neighbors of point.

In the project, assigning weight to the contributions of neighbors is a useful way to improve the performance so that the nearer neighbors contribute more to the average than the more distant ones. In **KNeighborsClassifier,** we set the **weights** to **'distance'**, performing a weighting scheme consists in giving each neighbor a weight of inverse of the distance.

Next, to validate such a classifier, it is rational to use cross validation to find the optimal parameter which is **n_estimator** (**Number of K nearest neighbors).** Theoretically, the number of neighbors could be given by $K_n = \sqrt{n}$. After oversampling, the size of the dataset will become about 6000x63 hence $K_n = \sqrt{6000} = 77.45$. Using cross validation, it is run for different values of K neighbors ranging from 5 to 100 over step of 5 and

found that a value of choosing about 50 to 70 nearest neighbors will give best result in classification. By applying this model to train the dataset and attaining predicted labels then test it on the true test labels. An **F1 score(weighted) of 0.87 and an auc_score of 0.72 was achieved.**

3. SVM classifier

Support vector machines are supervised learning models based on the principle that mapping training data points in the feature space so that diverse categories are divided by a gap as wide as possible. Additional, SVM can take advantage of kernel trick to map the inputs into high dimensional feature spaces in order to efficiently perform a non-linear classification. We will try to applying 'linear' kernel and 'rbf' kernel functions in this classifier which 'rbf' kernel method is better and find the optimal penalty parameter C and Kernel efficient parameter gamma by using cross validation.

For data preprocessing part, we will choose to perform probability replacement method transfer string to values. Instead of normalizing data but using rescaling technique to shrink data into a given range would give a better classifying performance. Next, oversampling the dataset to achieve the balance between two classes. F_classif feature selection method in sklearn is used to obtain the univariate score and p-value for each feature for performing feature selection. Based on the assumption that features with returning p-value greater than 0.05 contribute less to the entire dataset, we reduce these features to finish the feature selection. As for SVM, this feature selection method is better than the other two which is Mutual_classif and ExtraTreeBasis feature selection methods. Moreover, Principle components analysis is used to adjust the original feature space to **nine** dimensions.

After preprocessing, we use cross validation to proceed model selection and attain the

result: **F1 score of 0.89 and auc_score of 0.74.**

4. Adaboost classifier

Adaptive boosting method could be used in conjunction with many other types of learning algorithms to improve performance. Its adaptation consists in that the samples of previous basic classifier are improved and the weighted samples are again used to train the next basic classifier. In the meantime, a new weak classifier is added in each round until a certain enough small error rate is reached or a predetermined maximum number of iterations is reached. The whole algorithm procedure could be summarized as three steps: Firstly, initialize the weight distribution of training data. Secondly, train the weak classifiers. Thirdly, combining weak classifiers obtained by each training into strong classifiers.

In the procedure of training, there are two important parameters: **n_estimators** which means the max iteration times and learning rate which shrinks the contribution of each classifier. The data set is incline to be under fitting if n_estimators is too small and overfitting if too large. Normally, we will do model selection by combining both of two parameters. By cross validation, we found that set n_estimators = 100 and learning rate = 0.7 will give a satisfied result: **0.89 of F1_score and 0.79 of AUC.**

5. Logistic Regression

Logistic regression model actually is based on the linear regression, applying a logic function. Its principle includes: Firstly, finding a suitable hypothesis function, which aims to predict the input data' s judgment result. Secondly, construct a cost function (loss function) that represents the deviation between the predicted output and the training data category. Thirdly, applying gradient descent to find the optimal solution to the cost function (the minimum value).

We use LogisticRegression model in sklearn.linear.model and select the optimal parameter C by cross validation. We get the result of **F1 score of 0.84 and auc_score of 0.73.**

6. Random Forest

1) The concept of information, entropy, and information gain

These three basic concepts are the root of the decision tree, and they are the basis for determining the order of feature selection when the decision tree uses features to classify. Understand them, you will know about the decision tree.

In the words of Shannon, information is used to eliminate random uncertainties. Of course, although this sentence is classic, it is still very difficult to figure out what kind of thing this kind of thing is in the end. It may be that in different places, it means something different. For the decision tree in machine learning, if the set of things with classification can be divided into multiple categories, the information of a class (x_i) can be defined as follows:

$$I(X = x_i) = -log_2 p(x)$$

$I(X)$ is used to represent the information of a random variable and $p(x_i)$ is the probability when xi occurs.

Entropy is used to measure uncertainty. The larger the entropy, the greater the uncertainty of $X = x_i$, and vice versa. For classification problems in machine learning, the greater the entropy, the greater the uncertainty of this category, and vice versa.

Information gain is used to select features in the decision tree algorithm. The greater the information gain, the better the selectivity of this feature.

2) Decision Tree

A decision tree is a tree structure in which each internal node represents a test on an attribute, each branch represents a test output, and each leaf node represents a category. The common decision tree algorithms are C4.5, ID3, and CART.

3) Integrated learning

Integrated learning solves single prediction problems by establishing several model combinations. It works by generating multiple classifiers/models and learning and making predictions independently of each other. These predictions are finally combined into single predictions and are therefore superior to any single classification in making predictions.

We use **RandomForestClassifier** model in **sklearn.linear.ensemble** and select the

optimal parameter n_estimators by cross validation.

Performance Evaluation

1.Binary Expansion Preprocessing

1.1 Tree-Based Feature Selection

Classifiers	Feature Selection	Balance	Accuracy	F1 Score	AUC
Bayes	NO	Oversampling	0.86619	0.86766	0.71943
RF	Tree-Based	Oversampling	0.90056	0.87985	0.63451
LR	Tree-Based	Oversampling	0.79021	0.82041	0.73412
Adaboost	Tree-Based	Oversampling	0.91476	0.90656	0.82186
KNN	Tree-Based	Oversampling	0.82897	0.84543	0.72481
SVM	Tree-Based	Oversampling	0.84219	0.85153	0.72562

Obtaining above result based on following model selection

Model selection result

Classifiers	RF	LR	Adaboost	KNN	SVM
Paramater1	N_estimators=20	C=8	N_estimators=61	Neighbors=63	C=1.2
Paramater2	Max_depth=8	none	Learning rate=0.8	none	Gamma=0.5

F1 score report and ROC curve (the optimal result which is given by Bayes, Adaboost, KNN and SVM):

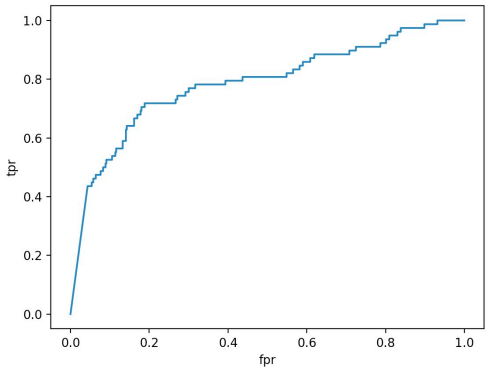
Bayes:

```
acc_test =  0.8661971830985915
             precision    recall  f1-score   support

    class 1       0.93      0.92      0.92       493
    class 2       0.49      0.52      0.51        75

avg / total       0.87      0.87      0.87       568

f1 score = 0.8676623711906692
auc = 0.7194320486815415
```



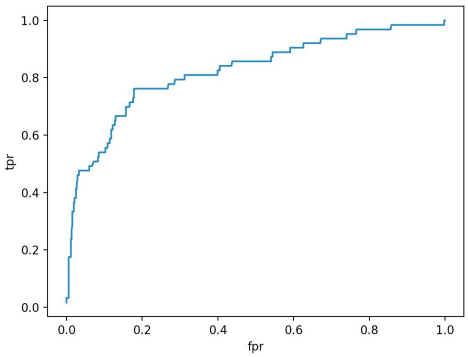
Adaboost:

```
acc_test =  0.9136442141623489
             precision    recall  f1-score   support

    class 1       0.94      0.97      0.95       516
    class 2       0.64      0.46      0.54        63

avg / total       0.90      0.91      0.91       579

f1 score = 0.9071880911259148
auc =  0.8236280300233788
```



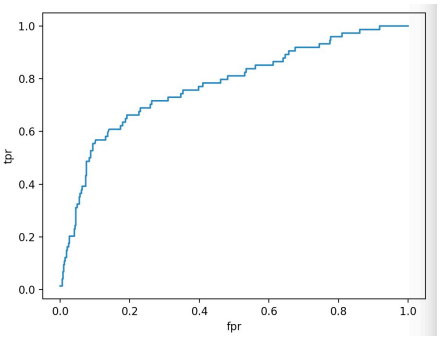
SVM:

```
acc_test =  0.8421985815602837
             precision    recall  f1-score   support

    class 1       0.93      0.88      0.91       490
    class 2       0.42      0.57      0.49        74

avg / total       0.86      0.84      0.85       564

f1 score = 0.8515349545606059
auc =  0.7256205184776614
```



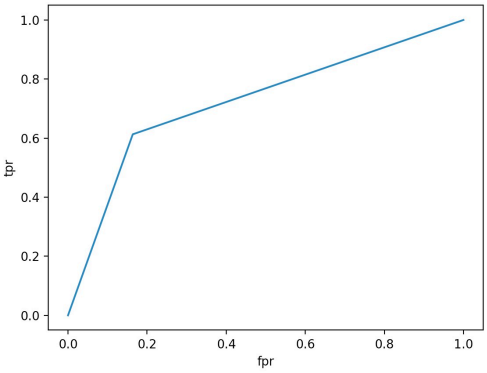
KNN:

```
acc_test =  0.8289703315881326
             precision    recall  f1-score   support

    class 1       0.94      0.86      0.90       505
    class 2       0.36      0.59      0.45        68

avg / total       0.87      0.83      0.85       573

f1 score = 0.8454376347206941
auc =  0.7248107163657542
```



1.2 Mutual_Classif Feature Selection

Classifiers	Feature Selection	Balance	Accuracy	F1 Score	AUC
Bayes	NO	Oversampling	0.87766	0.87819	0.70772
RF	Mutual_Classif	Oversampling	0.90906	0.88834	0.60491
LR	Mutual_Classif	Oversampling	0.81572	0.84063	0.73286
Adaboost	Mutual_Classif	Oversampling	0.91854	0.90823	0.76903
KNN	Mutual_Classif	Oversampling	0.82842	0.85019	0.72084
SVM	Mutual_Classif	Oversampling	0.84413	0.86712	0.73516

Model selection result

Classifiers	RF	LR	Adaboost	KNN	SVM
Paramater1	N_estimators=24	C=6	N_estimators=54	Neighbors=69	C=1.9
Paramater2	Max_depth=12	none	Learning rate=0.6	none	Gamma=0.7

F1 score report and ROC curve (the optimal result which is given by Adaboost, KNN and SVM):

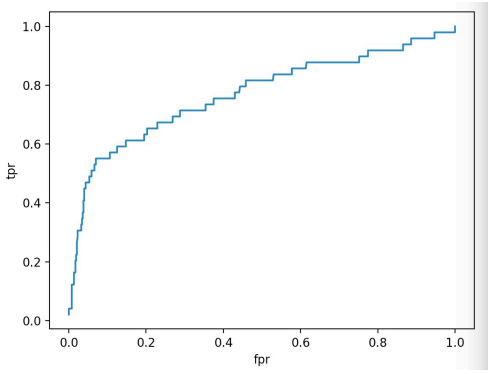
Adaboost:

```
acc_test =  0.9185441941074524
             precision    recall  f1-score   support

    class 1       0.94      0.98      0.96       528
    class 2       0.54      0.31      0.39        49

avg / total       0.90      0.92      0.91       577

f1 score = 0.9082307215963675
auc =  0.7690940012368583
```



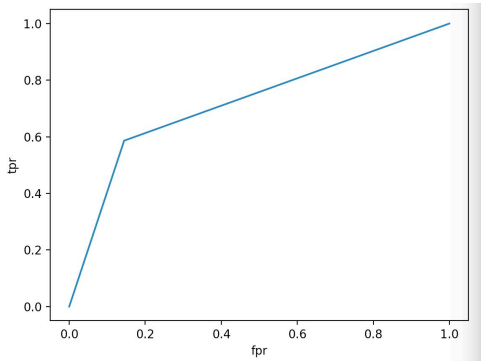
KNN:

```
acc_test =  0.82842287694974
             precision    recall  f1-score   support

    class 1       0.95      0.86      0.90       519
    class 2       0.31      0.59      0.41        58

avg / total       0.88      0.83      0.85       577

f1 score = 0.8501889353677426
auc =  0.7208491130157465
```



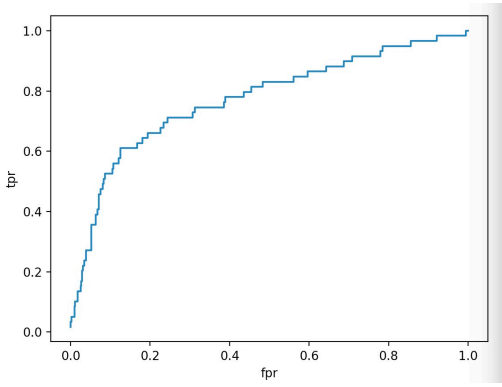
SVM:

```
acc_test =  0.8441330998248686
             precision    recall  f1-score   support

    class 1       0.96      0.87      0.91       523
    class 2       0.29      0.60      0.39        48

avg / total       0.90      0.84      0.87       571

f1 score = 0.8671766567208249
auc =  0.7351617272147863
```



1.3 F_Classif Feature Selection

Classifiers	Feature Selection	Balance	Accuracy	F1 Score	AUC
Bayes	NO	Oversampling	0.87766	0.87819	0.70772
RF	F_Classif	Oversampling	0.90067	0.88357	0.62149
LR	F_Classif	Oversampling	0.85269	0.86827	0.72717
Adaboost	F_Classif	Oversampling	0.90674	0.89069	0.82299
KNN	F_Classif	Oversampling	0.82137	0.84669	0.79014
SVM	F_Classif	Oversampling	0.84163	0.86285	0.77959

Model selection result

Classifiers	RF	LR	Adaboost	KNN	SVM

Paramater1	N_estimators=24	C=6	N_estimators=54	Neighbors=59	C=1.1
Paramater2	Max_depth=12	none	Learning rate=0.7	none	Gamma=0.2

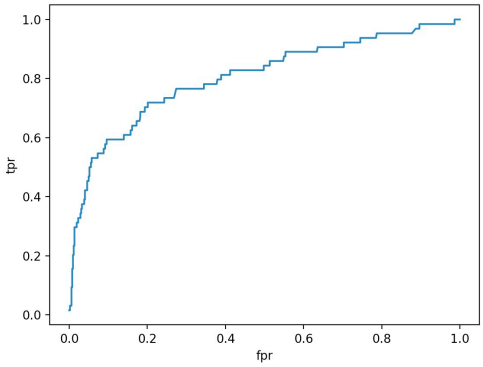
Adaboost:

```
acc_test =  0.9067357512953368
             precision    recall  f1-score   support

    class 1       0.92      0.98      0.95       510
    class 2       0.74      0.33      0.46        69

avg / total       0.90      0.91      0.89       579

f1 score = 0.8906904218537274
auc =  0.8229894856493323
```



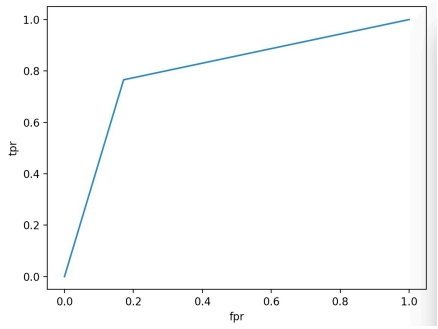
KNN:

```
acc_test =  0.8213660245183888
             precision    recall  f1-score   support

    class 1       0.97      0.83      0.89       507
    class 2       0.36      0.77      0.49        64

avg / total       0.90      0.82      0.85       571

f1 score = 0.8466933639720238
auc =  0.7970136834319527
```



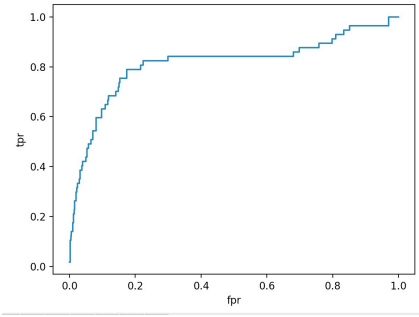
SVM：

```
acc_test =  0.8416370106761566
             precision    recall  f1-score   support

    class 1       0.96      0.86      0.91       505
    class 2       0.36      0.70      0.47        57

avg / total       0.90      0.84      0.86       562

f1 score = 0.8628459453657993
auc =  0.7795900642695849
```



2.Probability Replacement Preprocessing

2.1 Tree-Based Feature Selection

Classifiers	Feature Selection	Balance	Accuracy	F1 Score	AUC
Bayes	NO	Oversampling	0.86212	0.87103	0.73115
RF	Tree-Based	Oversampling	0.87115	0.86063	0.64551
LR	Tree-Based	Oversampling	0.705575	0.74297	0.68713
Adaboost	Tree-Based	Oversampling	0.90243	0.87962	0.79530
KNN	Tree-Based	Oversampling	0.83888	0.85952	0.80005
SVM	Tree-Based	Oversampling	0.81546	0.84204	0.66884

Model selection result

Classifiers	RF	LR	Adaboost	KNN	SVM
Paramater1	N_estimators=24	C=11	N_estimators=57	Neighbors=54	C=2.069
Paramater2	Max_depth=9	non	Learning rate=0.9	non	Gamma=0.45

(the optimal result which is given by Bayes, Adaboost, KNN and SVM):

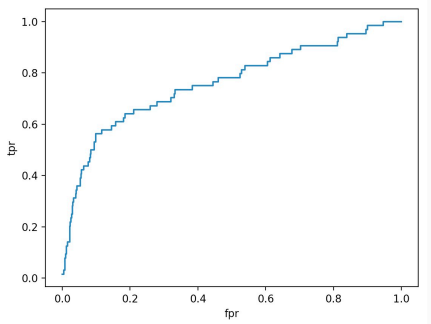
Bayes:

```
acc_test =  0.8621291448516579
              precision    recall  f1-score   support

     class 1       0.94      0.90      0.92       509
     class 2       0.41      0.56      0.48        64

avg / total       0.88      0.86      0.87       573

f1 score = 0.8710357608459992
auc = 0.731151768172888
```



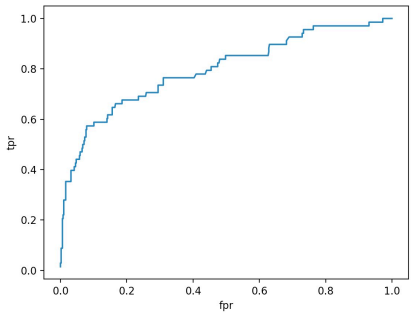
Adaboost:

```
acc_test =  0.9024390243902439
             precision    recall  f1-score   support

    class 1       0.91      0.99      0.95       506
    class 2       0.77      0.25      0.38        68

avg / total       0.89      0.90      0.88       574

f1 score = 0.8796276658418759
auc =  0.7953092304115321
```



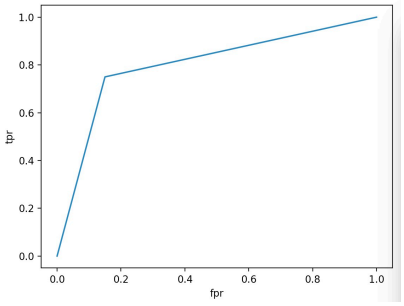
KNN:

```
acc_test =  0.8388791593695272
             precision    recall  f1-score   support

    class 1       0.96      0.85      0.90       507
    class 2       0.39      0.75      0.51        64

avg / total       0.90      0.84      0.86       571

f1 score = 0.8595232387089728
auc =  0.8000493096646942
```



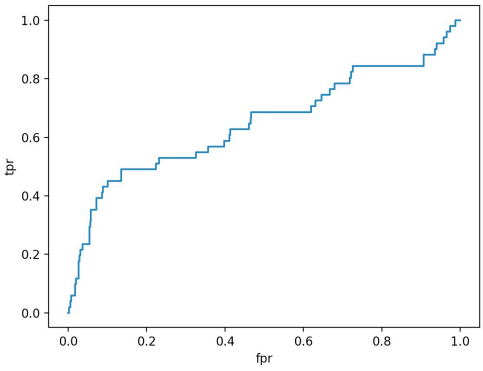
SVM:

```
acc_test =  0.8154657293497364
             precision    recall  f1-score   support

    class 1       0.94      0.85      0.89       518
    class 2       0.24      0.49      0.32        51

avg / total       0.88      0.82      0.84       569

f1 score = 0.8420404083548103
auc =  0.66884321296086
```



2.2 Mutual_Classif Feature Selection

Classifiers	Feature Selection	Balance	Accuracy	F1 Score	AUC
Bayes	NO	Oversampling	0.86135	0.87018	0.70321
RF	Mutual_Classif	Oversampling	0.90601	0.89088	0.63464

LR	Mutual_Classif	Oversampling	0.83778	0.85852	0.74921
Adaboost	Mutual_Classif	Oversampling	0.90718	0.88994	0.80343
KNN	Mutual_Classif	Oversampling	0.85962	0.87454	0.78124
SVM	Mutual_Classif	Oversampling	0.84864	0.86531	0.79133

Model selection result

Classifiers	RF	LR	Adaboost	KNN	SVM
Paramater1	N_estimators=26	C=5	N_estimators=51	Neighbors=64	C=1.4
Paramater2	Max_depth=7	non	Learning rate=0.5	non	Gamma=0.43

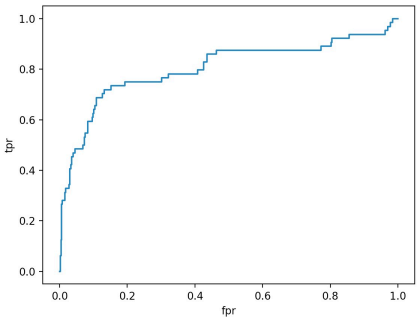
AdaBoost:

```
acc_test =  0.9071803852889667
             precision    recall  f1-score   support

    class 1       0.92      0.98      0.95       507
    class 2       0.70      0.30      0.42        64

avg / total       0.89      0.91      0.89       571

f1 score = 0.889944300793113
auc =  0.803439349112426
```



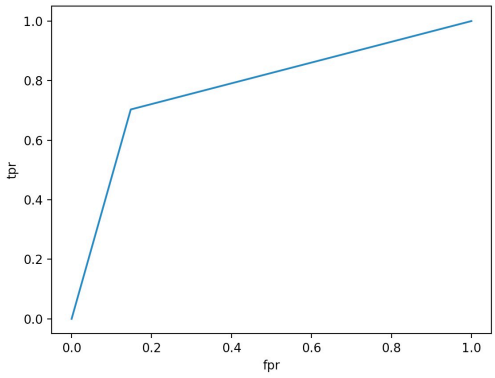
KNN：

```
acc_test =  0.8353765323992994
             precision    recall  f1-score   support

    class 1       0.96      0.85      0.90       507
    class 2       0.38      0.70      0.49        64

avg / total       0.89      0.84      0.86       571

f1 score = 0.8556163877923421
auc =  0.77759800295858
```



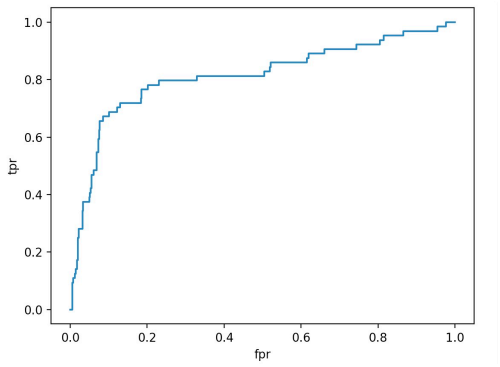
SVM:

```
acc_test =  0.8476357267950964
             precision    recall  f1-score   support

    class 1       0.96      0.86      0.91       507
    class 2       0.40      0.72      0.51        64

avg / total       0.90      0.85      0.87       571

f1 score = 0.8653066837668062
auc =  0.7913276627218935
```



2.3 F_Classif Feature Selection

Classifiers	Feature Selection	Balance	Accuracy	F1 Score	AUC
Bayes	NO	Oversampling	0.87911	0.88389	0.71882
RF	F_Classif	Oversampling	0.90189	0.89985	0.68851
LR	F_Classif	Oversampling	0.81796	0.84351	0.75369
Adaboost	F_Classif	Oversampling	0.90378	0.88778	0.82613
KNN	F_Classif	Oversampling	0.81174	0.85685	0.75514
SVM	F_Classif	Oversampling	0.84629	0.86181	0.76143

Model selection result

Classifiers	RF	LR	Adaboost	KNN	SVM
Paramater1	N_estimators=26	C=5	N_estimators=51	Neighbors=72	C=1.9
Paramater2	Max_depth=7	non	Learning rate=0.7	non	Gamma=2.7

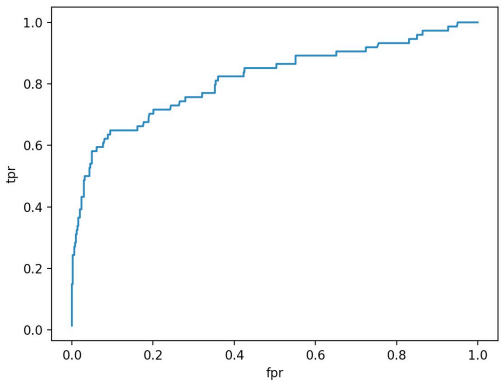
AdaBoost:

```
acc_test =  0.9037800687285223
             precision    recall  f1-score   support

    class 1       0.91      0.98      0.95       508
    class 2       0.76      0.35      0.48        74

avg / total       0.89      0.90      0.89       582

f1 score = 0.8877839101206799
auc =  0.8262130240476697
```



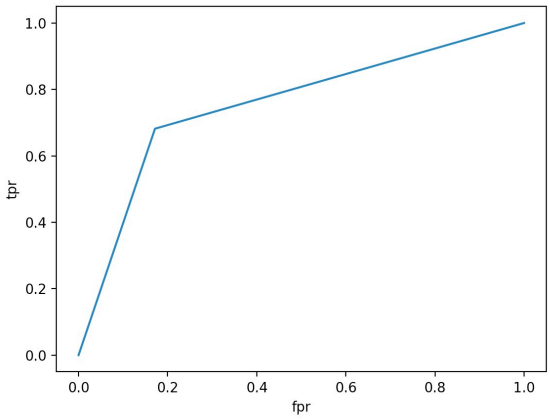
KNN:

```
acc_test =  0.8117443868739206
             precision    recall  f1-score   support

    class 1       0.95      0.83      0.89       513
    class 2       0.34      0.68      0.45        66

avg / total       0.88      0.81      0.84       579

f1 score = 0.8368594717361357
auc =  0.7551391104022682
```



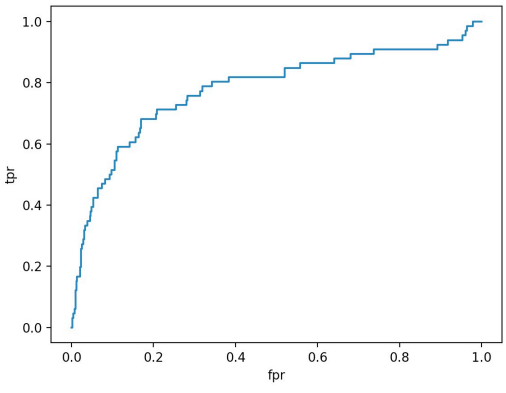
SVM:

```
acc_test =  0.846286701208981
             precision    recall  f1-score   support

    class 1       0.95      0.87      0.91       513
    class 2       0.39      0.65      0.49        66

avg / total       0.89      0.85      0.86       579

f1 score = 0.8618094881678393
auc =  0.7614300903774587
```



3. Performance

3.1 Naïve Bayes

We did not do much preprocessing work on naïve bayes classifier but only rescaling the dataset so that every time we run this classifier, the result is kind of similar.

3.2 SVM

All the three feature selection methods in binary expansion processing could give the optimal result. And for probability processing, mutual_classif and f_classif method perform better as well.

3.3 KNN

All the three feature selection methods in binary expansion processing works well for KNN. And probability processing accommodates all the three feature selection methods.

3.4 AdaBoost

This classifier gives the optimal results by applying each string transfer and feature selection method.

3.5 Random Forest

It could give good result of accuracy and f1 score but not the auc_socre.

3.6 Logistic Regression

This classifier could give some not bad results but not optimal.

Interpretation / Conclusion

- Pre-processing of data is the most important part of the entire project on improving the accuracy of the classification models. Different pre-processing
- method could lead to a better result, from this program, ***feature expansion*** and ***continuous probability*** resulted in good performance.
- Important factors that lead to deposit of customers were "***Age***", "***Job***", "***Education***", "***Campaign***", "***nr.employed***" and "***euribor3m***". These features are directly related to the possibility of making a deposit or not. And for "***default***" feature, which is redundant for classification progress.

- ***Feature selection*** and ***Outlier detection*** helped in ***increasing the accuracy by 5-10%*** of the classification. Feature selection is used for removing useless feature especially for feature expansion method and complete dimension reduction progress. Outlier detection could help to remove distant samples to improve classification accuracy and prevent overfitting problem.
- It was also seen that testing data sets in small bundles would results in better classification on the testing dataset and this led to choosing ***10 folds cross-validation*** techniques to prevent data overfitting. From above statistics, SVM, KNN and AdaBoost all could give optimal results based on different processing method. For instance, using cross-validation ***increased the performance*** of the Adaboost classifier giving an ***F1 score of 0.89, Accuracy of 0.91 and AUC of 0.80***

Work Distribution：

We equally separate the project work:

For the preprocessing part, Jinpeng Qi is responsible for string probability processing, feature dimension adjustment, normalizing and rescaling and weighted samples. Shuang Yuan is responsible for string continuous and binary expansion preprocessing, oversampling, Outlier detection and model selection.

For the Classifier part, we collaborate to work together for these six classifiers and finish the report.

Reference

[1] Online: http://scikit-learn.org/stable/modules/feature_selection.html#feature-selection

[2] Online: http://scikit-learn.org/stable/modules/preprocessing.html

[3] Online: http://scikit-learn.org/stable/modules/naive_bayes.html

[4] Online: http://scikit-learn.org/stable/supervised_learning.html

[3] ***Remove Outliers –*** M Sohrabinia

http://www.mathworks.com/matlabcentral/fileexchange/37211-remove-outliers