Step1: read data

```
1. x_train = pd.read_csv(r'C:/Users/wangyicheng/Desktop/6000Bproject/project1/traindata.csv')
2. y_train = pd.read_csv(r'C:/Users/wangyicheng/Desktop/6000Bproject/project1/trainlabel.csv')
3. x_test = pd.read_csv(r'C:/Users/wangyicheng/Desktop/6000Bproject/project1/testdata.csv')
```

Then we print the train_data and the test data we know thattrain data is 3219 rows x 57column and test_data has 1379 rows x 57 column, and the label of train data is 3219 rows x 1column.

	0.0	0.0.1	0.0.2	0.0.3	0.0.4	0.0.5	0.0.6	0.0.7	0.0.8	0.0.9	...	0.0.29	0.0.30	0.267	0.133	0.133.1	0.0.31	0.0.32	2.607	13.0	73.0
0	0.09	0.00	0.27	0.00	0.36	0.09	0.00	0.18	0.09	0.00	...	0.00	0.015	0.047	0.031	0.252	0.031	0.031	3.816	69.0	542.0
1	0.00	0.00	0.13	0.00	0.26	0.00	0.00	0.65	0.13	0.00	...	0.13	0.000	0.105	0.000	0.000	0.052	0.000	2.165	20.0	446.0
2	0.00	1.28	0.00	0.00	0.64	0.00	0.00	0.00	0.00	1.28	...	0.00	0.104	0.418	0.000	0.209	0.000	0.000	1.888	22.0	102.0
3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.00	...	0.00	0.000	0.682	0.000	0.000	0.000	0.000	2.705	11.0	46.0
4	0.00	1.42	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.42	...	0.00	0.000	0.000	0.000	0.000	0.000	0.000	3.555	21.0	96.0
5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	0.00	0.000	0.453	0.181	0.000	0.000	0.090	4.037	28.0	214.0
6	0.00	0.00	1.11	0.00	1.11	0.00	0.74	0.00	0.00	0.00	...	0.00	0.000	0.000	0.000	0.368	0.552	0.000	1.580	7.0	79.0
7	0.00	0.00	0.54	0.00	1.08	0.00	0.00	0.00	0.00	0.00	...	0.00	0.029	0.178	0.000	0.416	0.000	0.000	1.373	6.0	92.0

Step2: Data preprocessing

From the attribution of train data we find that some individual characteristic are too large and do not follow Gaussian normal distribution, as a result we need Data preprocessing. First we need transform train dataset and test dataset to z-score Standardization. The calculation function is subtracting the eigenvalue from the mean and the divide the standard deviation. The aim of Standardization or Mean Removal and Variance Scaling is to zoom and shrink data to a fit range.

```
1. #data prprocessing
2. train_scaled = preprocessing.scale(x_train)
3. test_scaled = preprocessing.scale(x_test)
4. y_train = np.ravel(y_train)
```

```
In [3]: train_scaled = preprocessing.scale(x_train)
        test_scaled = preprocessing.scale(x_test)

In [7]: train_scaled.shape,x_train.shape
Out[7]: ((3219, 57), (3219, 57))
```

Eliminate warning:

```
1. y_train = np.ravel(y_train)
```

After data preprocessing, we got the z-score standardization data as shown below:

```
In [13]: train_scaled = pd.DataFrame(data = train_scaled)
```

```
In [17]: train_scaled.head()
```

Out[17]:

	0	1	2	3	4	5	6	7	8	9	...	47	48	49	50
0	-0.054547	-0.168661	-0.013427	-0.047065	0.089533	-0.004007	-0.29659	0.176241	-0.005201	-0.371977	...	-0.128946	-0.098221	-0.385544	0.121163
1	-0.345466	-0.168661	-0.298172	-0.047065	-0.069997	-0.356878	-0.29659	1.287797	0.132584	-0.371977	...	0.363211	-0.160474	-0.131574	-0.155740
2	-0.345466	0.865904	-0.562578	-0.047065	0.536217	-0.356878	-0.29659	-0.249461	-0.315215	1.554353	...	-0.128946	0.271145	1.238986	-0.155740
3	-0.345466	-0.168661	-0.562578	-0.047065	-0.484775	-0.356878	-0.29659	-0.249461	-0.315215	2.637913	...	-0.128946	-0.160474	2.394986	-0.155740
4	-0.345466	0.979060	-0.562578	-0.047065	-0.484775	-0.356878	-0.29659	-0.249461	-0.315215	1.765045	...	-0.128946	-0.160474	-0.591347	-0.155740

Step3: Model select

I contrast some classifier such as svm, linear regression and logistic regression, in the end I choose svm for my classifier, because svm is effective in high dimensional spaces, different Kernel functions can be specified for the decision function, effective in cases where number of dimensions is greater than the number of samples. And then I use cross-validation to proof my hypothesis.

```
1. #clf = svm.SVR()
2. #clf = BernoulliNB()
3. #clf = GaussianNB().fit(train_scaled,y_train)
4. clf = SVC(probability=True, kernel='rbf')
5. clf.fit(train_scaled,y_train)
6. scores = cross_validation.cross_val_score(clf, train_scaled, y_train, cv =
   5)
```

```
1.SVR: [ 0.73344942  0.74087002  0.73715897  0.75911146  0.70977685]
2.BernoulliNB: [ 0.90232558  0.90993789  0.89440994  0.89269051  0.90357698]
3.GaussianNB: [ 0.85116279  0.85714286  0.82919255  0.83048212  0.81337481]
4.Svc: [ 0.9255814   0.93322981  0.93012422  0.9377916   0.91912908]
```

From the cross-validation_val_score we find that svc has the best performance, as a result I select SVM as my classifier.

Step4: Train svm model

```
1. clf = SVC(probability=True, kernel='rbf')
2. clf.fit(x_train,y_train)
```

Step5: Predict result

```
1. predictions = clf.predict(x_test)
```

Then we get the array of predictions.

```
In [43]: predictions = clf.predict(test_scaled)
```

```
In [44]: predictions
```

```
Out[44]: array([ 0.,  1.,  1., ...,  0.,  0.,  0.])
```

Step6: Output and save the forecast result

```
1. np.savetxt("C:/Users/wangyicheng/Desktop/testdatay.csv", predictions, delimi
   ter=",")
2. print(predictions)
```