Mini Project #5

Name ： Xi Cui xxc161130

a) Data preparing.

b) R OUTPUT:

```
summary(tune.out)
Parameter tuning of ‘svm’:
- sampling method: 10-fold cross validation
- best parameters:
 cost
   10
- best performance: 0.154023
- Detailed performance results:
   cost    error dispersion
1 1e-03 0.3137931 0.05365353
2 1e-02 0.1597701 0.03886576
3 1e-01 0.1586207 0.04224996
4 1e+00 0.1574713 0.04092628
5 5e+00 0.1551724 0.04205843
6 1e+01 0.1540230 0.04062026
7 1e+02 0.1586207 0.04395289
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  10
      gamma:  0.05555556
Number of Support Vectors:  340
 ( 170 170 )
Number of Classes:  2
Levels:
 CH MM
```

i) Performance evaluation:

The best cost parameter: 10, the best performance: 0.154023

Confusion matrix:

```
       truth
predict CH MM
     CH 87 26
     MM 18 69
```

Test error rate: (18+26)/200 = 0.22

ii) Summary:

After using the 10-fold CV, we search for best cost value among 0.001, 0.01, 0.1, 1, 5, 10, 100, The best cost parameter: 10, and the best performance: 0.154023. The number of support vectors is 340, 170 belonging to CH and 170 belonging to MM. The test error rate is 0.22.

c) R OUTPUT:

```
summary(tune.poly)
Parameter tuning of ‘svm’:
- sampling method: 10-fold cross validation
- best parameters:
 cost
    5
- best performance: 0.1712644
- Detailed performance results:
   cost    error dispersion
1 1e-03 0.3701149 0.04083651
2 1e-02 0.3712644 0.04128341
3 1e-01 0.3080460 0.05729219
4 1e+00 0.1862069 0.03706785
5 5e+00 0.1712644 0.04706566
6 1e+01 0.1712644 0.04212817
7 1e+02 0.1781609 0.04884118

Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  5
     degree:  2
      gamma:  0.05555556
     coef.0:  0
Number of Support Vectors:  380
 ( 192 188 )
Number of Classes:  2
Levels:
 CH MM
```

i) Performance evaluation:

The best cost parameter: 5, the best performance: 0.1712644

Confusion matrix:

```
       truth
predict CH MM
     CH 94 31
     MM 11 64
```

Test error rate: (11+31)/200 = 0.21

ii) Summary:

After using the 10-fold CV, with polynomial kernel of degree = 2, we search for best cost value among 0.001, 0.01, 0.1, 1, 5, 10, 100, The best cost parameter: 5, and the best performance: 0.1712644. The number of support vectors is 380, 192 belonging to CH and 188 belonging to MM. The test error rate is 0.21.

d) R OUTPUT:

```
summary(tune.out)
```

```
Parameter tuning of ‘svm’:
- sampling method: 10-fold cross validation
- best parameters:
 cost gamma
    1   0.5
- best performance: 0.2045977
- Detailed performance results:
    cost gamma     error dispersion
1  1e-01   0.5 0.2655172 0.05564137
2  1e+00   0.5 0.2045977 0.04809917
3  1e+01   0.5 0.2195402 0.03848620
4  1e+02   0.5 0.2379310 0.03019293
5  1e+03   0.5 0.2436782 0.04327976
6  1e-01   1.0 0.3183908 0.05392644
7  1e+00   1.0 0.2149425 0.04878103
8  1e+01   1.0 0.2321839 0.04011112
9  1e+02   1.0 0.2287356 0.03924165
10 1e+03   1.0 0.2471264 0.03480058
11 1e-01   2.0 0.3390805 0.04823632
12 1e+00   2.0 0.2172414 0.05348911
13 1e+01   2.0 0.2344828 0.04441801
14 1e+02   2.0 0.2367816 0.05202834
15 1e+03   2.0 0.2517241 0.05537691
16 1e-01   3.0 0.3482759 0.04662695
17 1e+00   3.0 0.2137931 0.05030697
18 1e+01   3.0 0.2275862 0.04686248
19 1e+02   3.0 0.2379310 0.05446815
20 1e+03   3.0 0.2448276 0.06833471
21 1e-01   4.0 0.3528736 0.04268206
22 1e+00   4.0 0.2218391 0.04847917
23 1e+01   4.0 0.2229885 0.04942380
24 1e+02   4.0 0.2356322 0.05288189
25 1e+03   4.0 0.2436782 0.06429480

Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.5
Number of Support Vectors:  460
 ( 246 214 )
Number of Classes:  2
Levels:
 CH MM
```

i) Performance evaluation:

The best cost parameter: 1, best gamma: 0.5, the best performance: 0.2045977

Confusion matrix:

```
       truth
predict CH MM
     CH 93 32
     MM 12 63
```

Test error rate: (12+32)/200 = 0.22

ii) Summary:

After using the 10-fold CV, with radial kernel, we search for best cost value among 0.1, 1, 10, 100, 1000, and gamma value among 0.5, 1, 2, 3, 4. The best cost parameter: 10, and the best gamma: 0.5, the best performance: 0.2045977. The number of support vectors is 460, 246 belonging to CH and 214 belonging to MM. The test error rate is 0.22.

e) Comparation:

Model	Test Error rate
SVM with Linear kernel, cost=10	0.22
SVM with polynomial kernel with degree=2, cost=5	0.21
SVM with radial kernel, cost=1, gamma=0.5	0.22
Bagging (B=1000, mtry=17)	0.18

Above comparing 4 methods, the bagging performs the best with the lowest test error rate 0.18. So, I will recommend bagging method. The SVM with polynomial kernel preforms better than liner kernel, which tells that the boundary may be non-linear.

R code:

```
install.packages("e1071")
library(e1071)
library(ISLR)
#--------------a-------------
#data preparing
attach(OJ)
data<-OJ
train<-data[1:870,]
test<-data[871:1070,]
#--------------b------------
set.seed(1)
tune.out <- tune(svm, Purchase ~ ., data = train, kernel = "linear", ranges = list(cost = c(0.001, 0.0
1, 0.1, 1, 5, 10, 100)))
summary(tune.out)
```

```
bestmod <- tune.out$best.model
summary(bestmod)
#evaluate
ypred <- predict(bestmod, test)
table(predict = ypred, truth = test$Purchase)
#test MSE
t<-table(predict = ypred, truth = test$Purchase)
(t[1,2]+t[2,1])/200
#-------------c----------------
set.seed(1)
tune.poly<- tune(svm, Purchase ~ ., data = train, kernel = "polynomial", degree=2, ranges = list(cost
 = c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
summary(tune.poly)

bestmod <- tune.poly$best.model
summary(bestmod)
#evaluate
ypred <- predict(bestmod, test)
table(predict = ypred, truth = test$Purchase)
#test MSE
t<-table(predict = ypred, truth = test$Purchase)
(t[1,2]+t[2,1])/200
#--------------d---------------
set.seed(1)
tune.out <- tune(svm, Purchase ~ ., data = train,  kernel = "radial", ranges = list(cost = c(0.1, 1, 1
0, 100, 1000), gamma = c(0.5, 1, 2, 3, 4)))
summary(tune.out)

bestmod <- tune.out$best.model
summary(bestmod)
#evaluate
ypred <- predict(bestmod, test)
table(predict = ypred, truth = test$Purchase)
#test MSE
t<-table(predict = ypred, truth = test$Purchase)
(t[1,2]+t[2,1])/200
```