individual assignment

student ID:474084

(a)

```
library(ISLR)
attach(OJ)
set.seed(1)
train=sample(1:nrow(OJ),800)
train.x=OJ[train,]
test.x=OJ[-train,]
```

(b)

```
library(e1071)
svm.fit=svm(Purchase~.,data=train.x, kernel = "linear", cost=0.01,scale = FALSE)
summary(svm.fit)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = train.x, kernel = "linear",
##     cost = 0.01, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##
## Number of Support Vectors:  615
##
##  ( 309 306 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

According to the result of summary, we can see that there are 615 support vectors. And there are 309 from class CH and 306 from class MM.

(c)

```
train.pred=predict(svm.fit,train.x)
test.pred=predict(svm.fit,test.x)

table(predict=train.pred,truth=train.x$Purchase)
```

```
##        truth
## predict  CH  MM
##      CH 420 105
##      MM  65 210
```

```
mean(train.pred!=train.x$Purchase)
```

```
## [1] 0.2125
```

```
table(predict=test.pred,truth=test.x$Purchase)
```

```
##        truth
## predict  CH  MM
##      CH 148  43
##      MM  20  59
```

```
mean(test.pred!=test.x$Purchase)
```

```
## [1] 0.2333333
```

The training error rate is 0.2125 and the test error rate is 0.23333.

(d)

```
set.seed(2)
tune.out=tune(svm,Purchase~.,data = train.x, kernel = "linear", ranges = list(cost = c(10^seq(-2, 1, by
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##      cost
##  1.778279
##
## - best performance: 0.1675
##
## - Detailed performance results:
##           cost   error dispersion
## 1   0.01000000 0.17625 0.04059026
## 2   0.01778279 0.17625 0.04348132
## 3   0.03162278 0.17125 0.04604120
## 4   0.05623413 0.17000 0.04005205
## 5   0.10000000 0.17125 0.04168749
## 6   0.17782794 0.17000 0.04090979
## 7   0.31622777 0.17125 0.04411554
## 8   0.56234133 0.17125 0.04084609
## 9   1.00000000 0.17000 0.04090979
## 10  1.77827941 0.16750 0.03782269
## 11  3.16227766 0.16750 0.03782269
## 12  5.62341325 0.16750 0.03545341
## 13 10.00000000 0.17000 0.03736085
```

```
bestmod=tune.out$best.model
```

In this chunk, we can see that the optimal cost is 1.77827941 , which brings the lowest error 0.16750.

(e)

```
ypred.train=predict(bestmod,train.x)
table(predict=ypred.train,truth=train.x$Purchase)
```

```
##         truth
## predict  CH  MM
##      CH 423  69
##      MM  62 246
```

```
mean(ypred.train!=train.x$Purchase)
```

```
## [1] 0.16375
```

```
ypred.test=predict(bestmod,test.x)
table(predict=ypred.test,truth=test.x$Purchase)
```

```
##         truth
## predict  CH  MM
##      CH 156  29
##      MM  12  73
```

```
mean(ypred.test!=test.x$Purchase)
```

```
## [1] 0.1518519
```

After changing cost to the tuned model, we can see that the train and test error rates are 0.16375 and 0.1518519 respectively. Both of the error rates are decreasing.

(f)using "radial"

```
#step 1
svm.machine=svm(Purchase~.,data=train.x, kernel = "radial", cost=0.01,scale = FALSE)
summary(svm.machine)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = train.x, kernel = "radial",
##     cost = 0.01, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.01
##
## Number of Support Vectors:  642
##
##  ( 327 315 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
train.pred2=predict(svm.machine,train.x)
test.pred2=predict(svm.machine,test.x)

mean(train.pred2!=train.x$Purchase)
```

```
## [1] 0.39375
```

```r
mean(test.pred2!=test.x$Purchase)
```

```
## [1] 0.3777778
```

```r
#step 2
set.seed(10)
tune.out2=tune(svm,Purchase~.,data = train.x, kernel = "radial", ranges = list(cost = c(10^seq(-2, 1, by
summary(tune.out2)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##       cost
##  0.5623413
##
## - best performance: 0.17
##
## - Detailed performance results:
##           cost   error dispersion
## 1   0.01000000 0.39375 0.05344065
## 2   0.01778279 0.39375 0.05344065
## 3   0.03162278 0.35375 0.07360980
## 4   0.05623413 0.19625 0.05894029
## 5   0.10000000 0.18875 0.04839436
## 6   0.17782794 0.17875 0.04372023
## 7   0.31622777 0.18125 0.03547789
## 8   0.56234133 0.17000 0.04174992
## 9   1.00000000 0.17000 0.04495368
## 10  1.77827941 0.17375 0.04543387
## 11  3.16227766 0.17875 0.04566256
## 12  5.62341325 0.18125 0.04218428
## 13 10.00000000 0.18625 0.03972562
```

```r
bestmod2=tune.out2$best.model
tune.out2$best.parameters
```

```
##        cost
## 8 0.5623413
```

```r
#step  3
ypred.train2=predict(bestmod2,train.x)
mean(ypred.train2!=train.x$Purchase)
```

```
## [1] 0.14875
```

```r
ypred.test2=predict(bestmod2,test.x)
mean(ypred.test2!=test.x$Purchase)
```

```
## [1] 0.1777778
```

After using "radial" with default gamma, we can see that there are 642 support vectors and 327 of them are CH, 315 of them are MM. The training and test error rates before tuned is 0.39375 and 0.3777778. The best model from tune.out has the cost of 0.56234133, which produces error of 0.17000. After tuning, the error rates are 0.14875 and 0.1777778.

(g)using " polynomial"

```
#step 1
svm.poly=svm(Purchase~., data = train.x, kernel ="polynomial", degree = 2, cost = 0.01,scale = FALSE)
summary(svm.poly)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = train.x, kernel = "polynomial",
##     degree = 2, cost = 0.01, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  0.01
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  333
##
##  ( 166 167 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
train.pred3=predict(svm.poly,train.x)
test.pred3=predict(svm.poly,test.x)

mean(train.pred3!=train.x$Purchase)
```

```
## [1] 0.165
```

```
mean(test.pred3!=test.x$Purchase)
```

```
## [1] 0.1592593
```

```
#step 2
set.seed(100)
tune.out3=tune(svm,Purchase~.,data = train.x, kernel = "polynomial", degree = 2,ranges = list(cost = c(
bestmod3=tune.out3$best.model
summary(tune.out3)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.17375
##
```

```
## - Detailed performance results:
##           cost   error dispersion
## 1   0.01000000 0.39375 0.03875224
## 2   0.01778279 0.37125 0.04450733
## 3   0.03162278 0.36000 0.04958158
## 4   0.05623413 0.34500 0.05177408
## 5   0.10000000 0.32625 0.04730589
## 6   0.17782794 0.25250 0.04594683
## 7   0.31622777 0.21000 0.03525699
## 8   0.56234133 0.20625 0.03830162
## 9   1.00000000 0.20125 0.03606033
## 10  1.77827941 0.19375 0.03919768
## 11  3.16227766 0.18125 0.04299952
## 12  5.62341325 0.18125 0.03547789
## 13 10.00000000 0.17375 0.03793727
```

```
#step  3
ypred.train3=predict(bestmod3,train.x)
mean(ypred.train3!=train.x$Purchase)
```

```
## [1] 0.15
```

```
ypred.test3=predict(bestmod3,test.x)
mean(ypred.test3!=test.x$Purchase)
```

```
## [1] 0.1888889
```

After using "polynomial" with degree=2, we can see that there are 333 support vectors and 166 of them are CH, 167 of them are MM. The training and test error rates before tuned is 0.165 and 0.1592593. The best model from tune.out has the cost of 10, which produces error of 0.17375. After tuning, the error rates are 0.15 and 0.1888889.

(h) We can find that compared with error rates, the SVM with "polynomial" has the best results before tuning. And after tuning, the SVM with "radial" method has lower test error rate of 0.1777778.