Problem 3

```r
library(e1071)
library(ISLR)
attach(OJ)
```

a)

```r
set.seed(2017)

train = sample(1:nrow(OJ), 535)
test = setdiff(1:nrow(OJ), train)
```

b)

```r
svm.fit <- svm(Purchase~.,data=OJ[train,],kernel="linear",cost=1)
summary(svm.fit)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ[train, ], kernel = "linear",
##     cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  202
##
##  ( 102 100 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

The SVM machine has 202 support vectors with 102 and 100 in the classes.

c) What are the training and test error rates?

```r
yhat <- predict(svm.fit)

table(predict=yhat, truth=OJ$Purchase[train])
```

```
##        truth
## predict  CH  MM
##      CH 293  40
##      MM  37 165
```

The training error rate is the count of errors made over the tootal count or (40+37)/(201+80+132+122) = 0.14

```r
table(predict=yhat, truth=OJ$Purchase[test])
```

```
##        truth
## predict  CH  MM
##      CH 201 132
##      MM 122  80
```

The test error rate is the count of errors made over the count of cases or (132+122)/(201+80+132+122) = 0.47.

d)

```r
tune.out <- tune(svm,Purchase~.,data=OJ[train,],kernel="linear",ranges=list(cost=c(0.01,0.05,0.1,0.5,0.7
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.5
##
## - best performance: 0.1531447
##
## - Detailed performance results:
##     cost     error dispersion
## 1   0.01 0.1626485 0.04984485
## 2   0.05 0.1607268 0.07040721
## 3   0.10 0.1607268 0.06642371
## 4   0.50 0.1531447 0.06044108
## 5   0.70 0.1549965 0.05928271
## 6   0.75 0.1549965 0.05928271
## 7   0.85 0.1549965 0.05928271
## 8   0.95 0.1549965 0.05928271
## 9   1.00 0.1549965 0.05928271
## 10  5.00 0.1643606 0.05659160
## 11  7.50 0.1662823 0.06116933
## 12  8.00 0.1681342 0.05957297
## 13 10.00 0.1681342 0.05695706
```

An optimal cost is 0.75.

e)

```r
bestmod <- tune.out$best.model
summary(bestmod)
```

```
##
## Call:
## best.tune(METHOD = svm, train.x = Purchase ~ ., data = OJ[train,
##     ], ranges = list(cost = c(0.01, 0.05, 0.1, 0.5, 0.7, 0.75, 0.85,
##     0.95, 1, 5, 7.5, 8, 10)), kernel = "linear")
##
```

```
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.5
##
## Number of Support Vectors:  207
##
##  ( 103 104 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```r
yhat <- predict(bestmod)

table(predict=yhat, truth=OJ$Purchase[train])
```

```
##        truth
## predict  CH  MM
##      CH 294  39
##      MM  36 166
```

The training error rate is the count of errors made over the total count or (40+36)/535 = 0.14

```r
table(predict=yhat, truth=OJ$Purchase[test])
```

```
##        truth
## predict  CH  MM
##      CH 201 132
##      MM 122  80
```

The test error rate is the count of errors made over the tootal count or (133+122)/535 = 0.48

(f) Repeat parts (b) through (e) using a support vector machine with a radial kernel. Use the default value for gamma.

```r
svmrad.fit <- svm(Purchase~.,data=OJ[train,],kernel="radial")
summary(svmrad.fit)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ[train, ], kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  260
##
##  ( 127 133 )
##
##
## Number of Classes:  2
```

```
##
## Levels:
##  CH MM
```

The SVM machine has 260 support vectors with 127 and 133 in the classes.

```r
yhat <- predict(svmrad.fit)

table(predict=yhat, truth=OJ$Purchase[train])
```

```
##        truth
## predict  CH  MM
##      CH 306  49
##      MM  24 156
```

The training error rate is the count of errors made over the total count or (49+24)/535 = 0.136

```r
table(predict=yhat, truth=OJ$Purchase[test])
```

```
##        truth
## predict  CH  MM
##      CH 220 135
##      MM 103  77
```

The test error rate is the count of errors made over the total count or (135+103)/535 = 0.445

```r
tunerad.out <- tune(svm,Purchase~.,data=OJ[train,],kernel="radial",ranges=list(cost=c(0.01,0.05,0.1,0.5
summary(tunerad.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     5
##
## - best performance: 0.1682739
##
## - Detailed performance results:
##    cost     error dispersion
## 1  0.01 0.3829490 0.04681606
## 2  0.05 0.3232006 0.04590639
## 3  0.10 0.2017470 0.04827853
## 4  0.50 0.1831936 0.05047356
## 5  1.00 0.1776380 0.04857852
## 6  5.00 0.1682739 0.05086527
## 7 10.00 0.1701607 0.04657436
```

Best model is at a cost of 5.

```r
bestmodrad <- tunerad.out$best.model
summary(bestmodrad)
```

```
##
## Call:
## best.tune(METHOD = svm, train.x = Purchase ~ ., data = OJ[train,
##     ], ranges = list(cost = c(0.01, 0.05, 0.1, 0.5, 1, 5, 10)), kernel = "radial")
```

```
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  5
##
## Number of Support Vectors:  218
##
##  ( 105 113 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

This SVM has 218 support vectors wiht 105 and 113 in each class.

```r
yhat <- predict(bestmodrad)

table(predict=yhat, truth=OJ$Purchase[train])
```

```
##        truth
## predict  CH  MM
##      CH 306  42
##      MM  24 163
```

The training error rate is the count of errors made over the total count or $(42+24)/535 = 0.123$

```r
table(predict=yhat, truth=OJ$Purchase[test])
```

```
##        truth
## predict  CH  MM
##      CH 216 132
##      MM 107  80
```

The test error rate is the count of errors made over the total count or $(132+107)/535 = 0.447$

g)

```r
svmply.fit <- svm(Purchase~.,data=OJ[train,],kernel="polynomial",degree=2)
summary(svmply.fit)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ[train, ], kernel = "polynomial",
##     degree = 2)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  2
##      coef.0:  0
##
```

```
## Number of Support Vectors:  315
##
##  ( 153 162 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

The SVM now has 315 support vectors with 153 and 162 in the classes.

```r
yhat <- predict(svmply.fit)

table(predict=yhat, truth=OJ$Purchase[train])
```

```
##        truth
## predict  CH  MM
##      CH 310  80
##      MM  20 125
```

The training error rate is the count of errors made over the total count or (80+20)/535 = 0.187

```r
table(predict=yhat, truth=OJ$Purchase[test])
```

```
##        truth
## predict  CH  MM
##      CH 238 152
##      MM  85  60
```

The test error rate is the count of errors made over the total count or (152+85)/535 = 0.443

```r
tuneply.out <- tune(svm,Purchase~.,data=OJ[train,],kernel="polynomial",ranges=list(cost=c(0.01,0.05,0.1
summary(tuneply.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.1722572
##
## - Detailed performance results:
##    cost     error dispersion
## 1  0.01 0.3630678 0.08121140
## 2  0.05 0.3405311 0.06305619
## 3  0.10 0.3199161 0.06900220
## 4  0.50 0.1984277 0.05271605
## 5  1.00 0.1946191 0.05263791
## 6  5.00 0.1816212 0.04802431
## 7 10.00 0.1722572 0.04681925
```

The best model is at a cost of 10.

```r
bestmodply <- tuneply.out$best.model
summary(bestmodply)
```

```
##
## Call:
## best.tune(METHOD = svm, train.x = Purchase ~ ., data = OJ[train,
##     ], ranges = list(cost = c(0.01, 0.05, 0.1, 0.5, 1, 5, 10)), kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  10
##      degree:  3
##      coef.0:  0
##
## Number of Support Vectors:  215
##
##  ( 108 107 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

This SVM has 215 support vectors with 108 and 108 in the classes.

```r
yhat <- predict(bestmodply)

table(predict=yhat, truth=OJ$Purchase[train])
```

```
##        truth
## predict  CH  MM
##      CH 313  43
##      MM  17 162
```

The training error rate is the count of errors made over the total count or (43+17)/535 = 0.112

```r
table(predict=yhat, truth=OJ$Purchase[test])
```

```
##        truth
## predict  CH  MM
##      CH 221 135
##      MM 102  77
```

The test error rate is the count of errors made over the total count or (135+102)/535 = 0.443

(h) Repeat parts (b) through (e) using a linear support vector machine, applied to an expanded feature set consisting of linear and all possible quadratic terms for the predictors. How does this compare to the polynomial kernel both conceptually and in terms of the results for this problem?

The results are not as good. This model is simpler in that it is a linear model; only the predictors have been transformed.

```r
svmexp.fit <- svm(Purchase~.+poly(PriceDiff,2)+poly(ListPriceDiff,2)+poly(DiscMM,2)+poly(LoyalCH,2)+pol
summary(svmexp.fit)
```

```
##
## Call:
## svm(formula = Purchase ~ . + poly(PriceDiff, 2) + poly(ListPriceDiff,
##     2) + poly(DiscMM, 2) + poly(LoyalCH, 2) + poly(SalePriceMM, 2) +
##     poly(SalePriceCH, 2) + poly(PctDiscMM, 2) + poly(PctDiscCH, 2),
##     data = OJ[train, ], kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  200
##
##  ( 97 103 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```r
yhat <- predict(svmexp.fit)

table(predict=yhat, truth=OJ$Purchase[train])
```

```
##        truth
## predict  CH  MM
##      CH 295  43
##      MM  35 162
```

The training error rate is the count of errors made over the total count or (49+24)/535 = ****

```r
table(predict=yhat, truth=OJ$Purchase[test])
```

```
##        truth
## predict  CH  MM
##      CH 206 132
##      MM 117  80
```

The test error rate is the count of errors made over the total count or (49+24)/535 = ****

```r
tuneexp.out <- tune(svm,Purchase~.+poly(PriceDiff,2)+poly(ListPriceDiff,2)+poly(DiscMM,2)+poly(LoyalCH,
summary(tuneexp.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     5
##
## - best performance: 0.1701957
##
```

```
## - Detailed performance results:
##    cost     error dispersion
## 1  0.01 0.1737945 0.03041157
## 2  0.05 0.1776380 0.04102085
## 3  0.10 0.1720475 0.04234664
## 4  0.50 0.1739693 0.04862185
## 5  1.00 0.1758211 0.04625692
## 6  5.00 0.1701957 0.04552464
## 7 10.00 0.1720825 0.04251418
```

```r
bestmodexp <- tuneexp.out$best.model
summary(bestmodexp)
```

```
##
## Call:
## best.tune(METHOD = svm, train.x = Purchase ~ . + poly(PriceDiff,
##     2) + poly(ListPriceDiff, 2) + poly(DiscMM, 2) + poly(LoyalCH,
##     2) + poly(SalePriceMM, 2) + poly(SalePriceCH, 2) + poly(PctDiscMM,
##     2) + poly(PctDiscCH, 2), data = OJ[train, ], ranges = list(cost = c(0.01,
##     0.05, 0.1, 0.5, 1, 5, 10)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  5
##
## Number of Support Vectors:  197
##
##  ( 98 99 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```r
yhat <- predict(bestmodexp)

table(predict=yhat, truth=OJ$Purchase[train])
```

```
##        truth
## predict  CH  MM
##      CH 297  43
##      MM  33 162
```

The training error rate is the count of errors made over the total count or (42+36)/535 = 0.146

```r
table(predict=yhat, truth=OJ$Purchase[test])
```

```
##        truth
## predict  CH  MM
##      CH 208 132
##      MM 115  80
```

The test error rate is the count of errors made over the total count or (130+117)/535 = 0.462

(i) Overall, which approach seems to give the best results on this data? The polynomial model has the best fit for the test data.