Stat 897 Fall 2017 Data Analysis Assignment 9

Penn State

Due November 12, 2017

In this assignment we will use the OJ data found in the ISLR library.

```
library(ISLR)
library(e1071)
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(knitr)

data("OJ")
str(OJ)
```

```
## 'data.frame':    1070 obs. of  18 variables:
##  $ Purchase      : Factor w/ 2 levels "CH","MM": 1 1 1 2 1 1 1 1 1 1 ...
##  $ WeekofPurchase: num  237 239 245 227 228 230 232 234 235 238 ...
##  $ StoreID       : num  1 1 1 1 7 7 7 7 7 7 ...
##  $ PriceCH       : num  1.75 1.75 1.86 1.69 1.69 1.69 1.69 1.75 1.75 1.75 ...
##  $ PriceMM       : num  1.99 1.99 2.09 1.69 1.69 1.99 1.99 1.99 1.99 1.99 ...
##  $ DiscCH        : num  0 0 0.17 0 0 0 0 0 0 0 ...
##  $ DiscMM        : num  0 0.3 0 0 0 0 0.4 0.4 0.4 0.4 ...
##  $ SpecialCH     : num  0 0 0 0 0 0 1 1 0 0 ...
##  $ SpecialMM     : num  0 1 0 0 0 1 1 0 0 0 ...
##  $ LoyalCH       : num  0.5 0.6 0.68 0.4 0.957 ...
##  $ SalePriceMM   : num  1.99 1.69 2.09 1.69 1.69 1.99 1.59 1.59 1.59 1.59 ...
##  $ SalePriceCH   : num  1.75 1.75 1.69 1.69 1.69 1.69 1.69 1.75 1.75 1.75 ...
##  $ PriceDiff     : num  0.24 -0.06 0.4 0 0 0.3 -0.1 -0.16 -0.16 -0.16 ...
##  $ Store7        : Factor w/ 2 levels "No","Yes": 1 1 1 1 2 2 2 2 2 2 ...
##  $ PctDiscMM     : num  0 0.151 0 0 0 ...
##  $ PctDiscCH     : num  0 0 0.0914 0 0 ...
##  $ ListPriceDiff : num  0.24 0.24 0.23 0 0 0.3 0.3 0.24 0.24 0.24 ...
##  $ STORE         : num  1 1 1 1 0 0 0 0 0 0 ...
```

We see that Purchase has 2 levels and we proceed to apply SVM on the data set.

(a) Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

```
set.seed(35)
train=sample(nrow(OJ),800)
OJ.train = OJ[train,]
OJ.test = OJ[-train,]
```

(b) Fit a support vector classifier to the training data using cost=0.01, with Purchase as the response and the other variables as predictors. Use the summary() function to produce summary statistics, and describe the results obtained.

A support vector classifier would mean that we run svm with kernel=linear

```
svm_l = svm(Purchase~.,data=OJ.train,kernel="linear",cost=0.01)
summary(svm_l)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "linear",
##     cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##       gamma:  0.05555556
##
## Number of Support Vectors:  426
##
##  ( 214 212 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

The summary tells us that the model selects 426 out of 800 observations as support points. The two levels contribute 214 and 212 data points.

(c) What are the training and test error rates?

```
# Training Error
pred.train = predict(svm_l, newdata=OJ.train)
table(pred=pred.train, truth=OJ.train$Purchase)
```

```
##     truth
## pred  CH  MM
##   CH 433  73
##   MM  59 235
```

```
cm=confusionMatrix(data = pred.train, reference = OJ.train$Purchase)
cm$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.8800813            0.7629870            0.8557312
##       Neg Pred Value            Precision               Recall
##            0.7993197            0.8557312            0.8800813
##                   F1           Prevalence       Detection Rate
##            0.8677355            0.6150000            0.5412500
```

```
## Detection Prevalence    Balanced Accuracy
##            0.6325000            0.8215342
```

```r
mean(pred.train == OJ.train$Purchase)
```

```
## [1] 0.835
```

```r
# Test Error
pred.test = predict(svm_l,newdata=OJ.test)
table(pred=pred.test, truth=OJ.test$Purchase)
```

```
##     truth
## pred  CH  MM
##   CH 147  37
##   MM  14  72
```

```r
cm=confusionMatrix(data = pred.test, reference = OJ.test$Purchase)
cm$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.9130435            0.6605505            0.7989130
##       Neg Pred Value            Precision               Recall
##            0.8372093            0.7989130            0.9130435
##                   F1           Prevalence       Detection Rate
##            0.8521739            0.5962963            0.5444444
## Detection Prevalence    Balanced Accuracy
##            0.6814815            0.7867970
```

```r
mean(pred.test == OJ.test$Purchase)
```

```
## [1] 0.8111111
```

In summary we see the following results for linear kernel:

```
            Training        Test
```

Mean 0.835 0.8111111 Sensitivity 0.8800813 0.9130435 Specificity 0.7629870 0.6605505 Precision 0.8557312 0.7989130

We see that while the sensitivity is slightly higher, the overall accuracy drops for the test data as compared to the training data.

(d) Use the tune() function to select an optimal cost. Consider values in the range 0.01 to 10.

```r
set.seed(35)
cost_range = c(.01, .02, .03, .05, .2, .4, .7, 1,2,3,4,5,6,7,8,9,10)

tune.out = tune(svm, Purchase~., data=OJ.train, kernel="linear",  ranges=list(cost=cost_range))
summary(tune.out)
```

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

```
##  0.03
##
## - best performance: 0.16
##
## - Detailed performance results:
##     cost   error dispersion
## 1   0.01 0.16750 0.03641962
## 2   0.02 0.16625 0.03438447
## 3   0.03 0.16000 0.03106892
## 4   0.05 0.16375 0.03653860
## 5   0.20 0.16375 0.03653860
## 6   0.40 0.16375 0.03653860
## 7   0.70 0.16375 0.03972562
## 8   1.00 0.16500 0.03944053
## 9   2.00 0.16875 0.04093101
## 10  3.00 0.16875 0.03784563
## 11  4.00 0.16750 0.03593976
## 12  5.00 0.17000 0.04090979
## 13  6.00 0.17000 0.04090979
## 14  7.00 0.16875 0.03919768
## 15  8.00 0.16875 0.03919768
## 16  9.00 0.16875 0.03919768
## 17 10.00 0.16875 0.03919768
```

From the results we can say that the optimal results are achieved when cost=0.03. We get error: 0.16

(e) Compute the training and test error rates using this new value for cost.

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

```
##
## Call:
## best.tune(method = svm, train.x = Purchase ~ ., data = OJ.train,
##     ranges = list(cost = cost_range), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.03
##       gamma:  0.05555556
##
## Number of Support Vectors:  368
##
##  ( 185 183 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```r
# Training Error
pred.train = predict(bestmod, newdata=OJ.train)
table(pred=pred.train, truth=OJ.train$Purchase)
```

```
##     truth
## pred  CH  MM
##   CH 434  70
##   MM  58 238
```

```r
cm=confusionMatrix(data = pred.train, reference = OJ.train$Purchase)
cm$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.8821138            0.7727273            0.8611111
##       Neg Pred Value            Precision               Recall
##            0.8040541            0.8611111            0.8821138
##                   F1           Prevalence       Detection Rate
##            0.8714859            0.6150000            0.5425000
## Detection Prevalence    Balanced Accuracy
##            0.6300000            0.8274205
```

```r
mean(pred.train == OJ.train$Purchase)
```

```
## [1] 0.84
```

```r
# Test Error
pred.test = predict(bestmod,newdata=OJ.test)
table(pred=pred.test, truth=OJ.test$Purchase)
```

```
##     truth
## pred  CH  MM
##   CH 146  36
##   MM  15  73
```

```r
cm=confusionMatrix(data = pred.test, reference = OJ.test$Purchase)
cm$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.9068323            0.6697248            0.8021978
##       Neg Pred Value            Precision               Recall
##            0.8295455            0.8021978            0.9068323
##                   F1           Prevalence       Detection Rate
##            0.8513120            0.5962963            0.5407407
## Detection Prevalence    Balanced Accuracy
##            0.6740741            0.7882785
```

```r
mean(pred.test == OJ.test$Purchase)
```

```
## [1] 0.8111111
```

In summary we see the following results for bestmodel that uses linear kernel:

```
            Training        Test
```

Mean 0.84 0.8111111 Sensitivity 0.8821138 0.9068323 Specificity 0.7727273 0.6697248 Precision 0.8611111 0.8021978

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

```r
# b. SVM with radial kernel and cost=0.01
svm_r = svm(Purchase~.,data=OJ.train,kernel="radial",cost=0.01)
summary(svm_r)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "radial",
##     cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.01
##       gamma:  0.05555556
##
## Number of Support Vectors:  617
##
##  ( 309 308 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

The summary tells us that the model selects 617 out of 800 observations as support points. The two levels contribute 309 and 308 data points.

```r
# c. Training and test error rates
# Training Error
pred.train = predict(svm_r, newdata=OJ.train)
table(pred=pred.train, truth=OJ.train$Purchase)
```

```
##     truth
## pred  CH  MM
##   CH 492 308
##   MM   0   0
```

```r
cm=confusionMatrix(data = pred.train, reference = OJ.train$Purchase)
cm$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            1.0000000            0.0000000            0.6150000
##       Neg Pred Value            Precision               Recall
##                  NaN            0.6150000            1.0000000
##                   F1           Prevalence       Detection Rate
##            0.7616099            0.6150000            0.6150000
## Detection Prevalence    Balanced Accuracy
##            1.0000000            0.5000000
```

```r
mean(pred.train == OJ.train$Purchase)
```

```
## [1] 0.615
```

```
# Test Error
pred.test = predict(svm_r,newdata=OJ.test)
table(pred=pred.test, truth=OJ.test$Purchase)
```

```
##     truth
## pred  CH  MM
##   CH 161 109
##   MM   0   0
```

```
cm=confusionMatrix(data = pred.test, reference = OJ.test$Purchase)
cm$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            1.0000000            0.0000000            0.5962963
##       Neg Pred Value            Precision               Recall
##                  NaN            0.5962963            1.0000000
##                   F1           Prevalence       Detection Rate
##            0.7470998            0.5962963            0.5962963
## Detection Prevalence    Balanced Accuracy
##            1.0000000            0.5000000
```

```
mean(pred.test == OJ.test$Purchase)
```

```
## [1] 0.5962963
```

In summary we see the following results for radial kernel:

```
            Training        Test
```

Mean 0.615 0.5962963 Sensitivity 1.0000000 1.0000000 Specificity 0.0000000 0.0000000 Precision 0.6150000 0.5962963

The results have become worse.

```
# d. tune on radial
cost_range = c(.01, .02, .03, .05, .2, .4, .7, 1,2,3,4,5,6,7,8,9,10)
set.seed(35)
tune.out = tune(svm, Purchase~., data=OJ.train, kernel="radial",  ranges=list(cost=cost_range))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.1625
##
## - Detailed performance results:
##     cost   error dispersion
## 1   0.01 0.38500 0.04632314
## 2   0.02 0.38500 0.04632314
## 3   0.03 0.36750 0.05957022
## 4   0.05 0.19875 0.02791978
```

```
## 5   0.20 0.17500 0.03632416
## 6   0.40 0.17125 0.03283481
## 7   0.70 0.16500 0.03525699
## 8   1.00 0.16250 0.03486083
## 9   2.00 0.16375 0.03356689
## 10  3.00 0.17000 0.03917553
## 11  4.00 0.17250 0.03670453
## 12  5.00 0.17125 0.03537988
## 13  6.00 0.17250 0.03525699
## 14  7.00 0.17375 0.03304563
## 15  8.00 0.17500 0.03227486
## 16  9.00 0.17750 0.03162278
## 17 10.00 0.17750 0.03162278
```

From the results we can say that the optimal results are achieved when cost=1. We get error: 0.1625

```r
# e. training and test error rates using best model
bestmod =tune.out$best.model
summary (bestmod )
```

```
##
## Call:
## best.tune(method = svm, train.x = Purchase ~ ., data = OJ.train,
##     ranges = list(cost = cost_range), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.05555556
##
## Number of Support Vectors:  360
##
##  ( 183 177 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```r
# Training Error
pred.train = predict(bestmod, newdata=OJ.train)
table(pred=pred.train, truth=OJ.train$Purchase)
```

```
##     truth
## pred  CH  MM
##   CH 451  73
##   MM  41 235
```

```r
cm=confusionMatrix(data = pred.train, reference = OJ.train$Purchase)
cm$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.9166667            0.7629870            0.8606870
##       Neg Pred Value            Precision               Recall
```

```
##             0.8514493            0.8606870            0.9166667
##                    F1           Prevalence       Detection Rate
##             0.8877953            0.6150000            0.5637500
## Detection Prevalence    Balanced Accuracy
##             0.6550000            0.8398268
```

```r
mean(pred.train == OJ.train$Purchase)
```

```
## [1] 0.8575
```

```r
# Test Error
pred.test = predict(bestmod,newdata=OJ.test)
table(pred=pred.test, truth=OJ.test$Purchase)
```

```
##     truth
## pred  CH  MM
##   CH 147  38
##   MM  14  71
```

```r
cm=confusionMatrix(data = pred.test, reference = OJ.test$Purchase)
cm$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##             0.9130435            0.6513761            0.7945946
##       Neg Pred Value            Precision               Recall
##             0.8352941            0.7945946            0.9130435
##                    F1           Prevalence       Detection Rate
##             0.8497110            0.5962963            0.5444444
## Detection Prevalence    Balanced Accuracy
##             0.6851852            0.7822098
```

```r
mean(pred.test == OJ.test$Purchase)
```

```
## [1] 0.8074074
```

In summary we see the following results for bestmodel that uses radial kernel:

```
            Training        Test
```

Mean 0.8575 0.8074074 Sensitivity 0.9166667 0.9130435 Specificity 0.7629870 0.6513761 Precision 0.8606870 0.7945946

(g) Repeat parts (b) through (e) using a support vector machine with a polynomial kernel. Set degree=2.

```r
# b. SVM with polynomial kernel and cost=0.01
svm_p = svm(Purchase~.,data=OJ.train,kernel="polynomial", degree=2,cost=0.01)
summary(svm_p)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "polynomial",
##     degree = 2, cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
```

```
##  SVM-Kernel:  polynomial
##        cost:  0.01
##      degree:  2
##       gamma:  0.05555556
##      coef.0:  0
##
## Number of Support Vectors:  620
##
##  ( 312 308 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

The summary tells us that the model selects 620 out of 800 observations as support points. The two levels contribute 312 and 308 data points.

```r
# c. Training and test error rates
# Training Error
pred.train = predict(svm_p, newdata=OJ.train)
table(pred=pred.train, truth=OJ.train$Purchase)
```

```
##     truth
## pred  CH  MM
##   CH 491 295
##   MM   1  13
```

```r
cm=confusionMatrix(data = pred.train, reference = OJ.train$Purchase)
cm$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##           0.99796748           0.04220779           0.62468193
##       Neg Pred Value            Precision               Recall
##           0.92857143           0.62468193           0.99796748
##                   F1           Prevalence       Detection Rate
##           0.76838811           0.61500000           0.61375000
## Detection Prevalence    Balanced Accuracy
##           0.98250000           0.52008764
```

```r
mean(pred.train == OJ.train$Purchase)
```

```
## [1] 0.63
```

```r
# Test Error
pred.test = predict(svm_p,newdata=OJ.test)
table(pred=pred.test, truth=OJ.test$Purchase)
```

```
##     truth
## pred  CH  MM
##   CH 161 109
##   MM   0   0
```

```r
cm=confusionMatrix(data = pred.test, reference = OJ.test$Purchase)
cm$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
```

```
##            1.0000000            0.0000000            0.5962963
##       Neg Pred Value            Precision               Recall
##                  NaN            0.5962963            1.0000000
##                   F1           Prevalence       Detection Rate
##            0.7470998            0.5962963            0.5962963
## Detection Prevalence    Balanced Accuracy
##            1.0000000            0.5000000
```

```
mean(pred.test == OJ.test$Purchase)
```

```
## [1] 0.5962963
```

In summary we see the following results for polynomial kernel:

```
            Training         Test
```

Mean 0.63 0.5962963 Sensitivity 0.99796748 1.0000000 Specificity 0.04220779 0.0000000 Precision 0.62468193 0.5962963

```
# d. tune on polynomial
cost_range = c(.01, .02, .03, .05, .2, .4, .7, 1,2,3,4,5,6,7,8,9,10)
set.seed(35)
tune.out = tune(svm, Purchase~., data=OJ.train, kernel="polynomial", degree=2, ranges=list(cost=cost_ra
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     9
##
## - best performance: 0.16875
##
## - Detailed performance results:
##     cost   error dispersion
## 1   0.01 0.38500 0.04632314
## 2   0.02 0.35875 0.04788949
## 3   0.03 0.35625 0.04938862
## 4   0.05 0.33500 0.04958158
## 5   0.20 0.24250 0.04571956
## 6   0.40 0.20250 0.02934469
## 7   0.70 0.20000 0.03173239
## 8   1.00 0.19250 0.02958040
## 9   2.00 0.18750 0.03908680
## 10  3.00 0.18125 0.03596391
## 11  4.00 0.17750 0.03944053
## 12  5.00 0.17125 0.03634805
## 13  6.00 0.17750 0.04241004
## 14  7.00 0.17875 0.03682259
## 15  8.00 0.16875 0.03875224
## 16  9.00 0.16875 0.03963812
## 17 10.00 0.17125 0.03866254
```

From the results we can say that the optimal results are achieved when cost=9. We get error: 0.16875

```r
# e. training and test error rates using best model
bestmod =tune.out$best.model
summary (bestmod )
```

```
##
## Call:
## best.tune(method = svm, train.x = Purchase ~ ., data = OJ.train,
##     ranges = list(cost = cost_range), kernel = "polynomial",
##     degree = 2)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  9
##      degree:  2
##       gamma:  0.05555556
##      coef.0:  0
##
## Number of Support Vectors:  338
##
##  ( 173 165 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```r
# Training Error
pred.train = predict(bestmod, newdata=OJ.train)
table(pred=pred.train, truth=OJ.train$Purchase)
```

```
##     truth
## pred  CH  MM
##   CH 450  74
##   MM  42 234
```

```r
cm=confusionMatrix(data = pred.train, reference = OJ.train$Purchase)
cm$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.9146341            0.7597403            0.8587786
##       Neg Pred Value            Precision               Recall
##            0.8478261            0.8587786            0.9146341
##                   F1           Prevalence       Detection Rate
##            0.8858268            0.6150000            0.5625000
## Detection Prevalence    Balanced Accuracy
##            0.6550000            0.8371872
```

```r
mean(pred.train == OJ.train$Purchase)
```

```
## [1] 0.855
```

```r
# Test Error
pred.test = predict(bestmod,newdata=OJ.test)
```

```r
table(pred=pred.test, truth=OJ.test$Purchase)
```

```
##     truth
## pred  CH  MM
##   CH 151  43
##   MM  10  66
```

```r
cm=confusionMatrix(data = pred.test, reference = OJ.test$Purchase)
cm$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.9378882            0.6055046            0.7783505
##       Neg Pred Value            Precision               Recall
##            0.8684211            0.7783505            0.9378882
##                   F1           Prevalence       Detection Rate
##            0.8507042            0.5962963            0.5592593
## Detection Prevalence    Balanced Accuracy
##            0.7185185            0.7716964
```

```r
mean(pred.test == OJ.test$Purchase)
```

```
## [1] 0.8037037
```

In summary we see the following results for bestmodel that uses radial kernel:

```
            Training        Test
```

Mean 0.855 0.8037037 Sensitivity 0.9146341 0.9378882 Specificity 0.7597403 0.6055046 Precision 0.8587786 0.7783505

(h) Overall, which approach seems to give the best results on this data?

We will compare the test data metrics for the best models retrieved from linear, radial and polynomial kernels.

```
            Test                  Test            Test
                    (poly, deg:2)   Radial          Linear
```

Mean 0.8037037 0.8074074 0.8111111 Sensitivity 0.9378882 0.9130435 0.9068323 Specificity 0.6055046 0.6513761 0.6697248 Precision 0.7783505 0.7945946 0.8021978 CV Error 0.16875 0.1625 0.16

We get the best mean/precision and specificity with linear kernel. The sensitivity is highest for the polynomial (degree=2) kernel. The CV error is only slightly better for the linear model.