Apurva Shrivastava
925009508
ISEN 613 - 600

Assignment #7

Problem 1

In this problem, you will use support vector approaches to predict whether a given car gets high or low gas mileage based on the Auto data set in the ISLR package.

(a) Create a binary variable that takes on a 1 for cars with gas mileage above the median, and a 0 for cars with gas mileage below the median. Use this variable as response in the following analysis.

```
> library(ISLR)
> var <- ifelse(Auto$mpg > median(Auto$mpg), 1, 0)
> Auto$mpglevel <- as.factor(var)
>
```

(b) Fit a support vector classifier to the data with various values of cost, to predict whether a car gets high or low gas mileage. Report the cross-validation errors associated with different values of this parameter. Comment on your results.

```
> library(e1071)
> set.seed(1)
> svm.out <- tune(svm, mpglevel ~ ., data = Auto, kernel = "linear", ranges = list(cost = c(0.01, 0.1, 1, 5, 10, 100, 1000)))
> summary(svm.out)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
    1

- best performance: 0.01275641

- Detailed performance results:
   cost      error dispersion
1 1e-02 0.07403846 0.05471525
2 1e-01 0.03826923 0.05148114
3 1e+00 0.01275641 0.01344780
4 5e+00 0.01782051 0.01229997
5 1e+01 0.02038462 0.01074682
6 1e+02 0.03820513 0.01773427
7 1e+03 0.03820513 0.01773427

>
```

Comment: As we can see from the output, cost of 1 performs the best.

Apurva Shrivastava
925009508
ISEN 613 - 600

(c) Now repeat (b), this time using SVMs with radial and polynomial kernels, with different values of gamma, degree and cost. Comment on your results.

```
> set.seed(1)
> svm.out <- tune(svm, mpglevel ~ ., data = Auto, kernel = "polynomial", ranges = list(cost = c(0.01, 0.1, 1, 5, 10, 100), degree = c(2, 3, 4)))
> summary(svm.out)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost degree
  100      2

- best performance: 0.3013462

- Detailed performance results:
    cost degree     error dispersion
1  1e-02      2 0.5611538 0.04344202
2  1e-01      2 0.5611538 0.04344202
3  1e+00      2 0.5611538 0.04344202
4  5e+00      2 0.5611538 0.04344202
5  1e+01      2 0.5382051 0.05829238
6  1e+02      2 0.3013462 0.09040277
7  1e-02      3 0.5611538 0.04344202
8  1e-01      3 0.5611538 0.04344202
9  1e+00      3 0.5611538 0.04344202
10 5e+00      3 0.5611538 0.04344202
11 1e+01      3 0.5611538 0.04344202
12 1e+02      3 0.3322436 0.11140578
13 1e-02      4 0.5611538 0.04344202
14 1e-01      4 0.5611538 0.04344202
15 1e+00      4 0.5611538 0.04344202
16 5e+00      4 0.5611538 0.04344202
17 1e+01      4 0.5611538 0.04344202
18 1e+02      4 0.5611538 0.04344202

>
```

For a **polynomial kernel**, the lowest cross-validation error is obtained for a **degree of 2 and a cost of 100.**

```
> set.seed(1)
> svm.out <- tune(svm, mpglevel ~ ., data = Auto, kernel = "radial", ranges = list(cost = c(0.01, 0.1, 1, 5, 10, 100), gamma = c(0.01, 0.1, 1, 5, 10, 100)))
> summary(svm.out)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
  100  0.01

- best performance: 0.01532051

- Detailed performance results:
    cost gamma      error dispersion
1  1e-02 1e-02 0.56115385 0.04344202
2  1e-01 1e-02 0.09185897 0.03862507
3  1e+00 1e-02 0.07147436 0.05103685
4  5e+00 1e-02 0.04326923 0.04975032
5  1e+01 1e-02 0.02551282 0.03812986
6  1e+02 1e-02 0.01532051 0.01788871
7  1e-02 1e-01 0.19153846 0.07612945
8  1e-01 1e-01 0.07916667 0.05201159
9  1e+00 1e-01 0.05608974 0.05092939
10 5e+00 1e-01 0.03064103 0.02637448
```

Apurva Shrivastava
925009508
ISEN 613 - 600

```
11 1e+01 1e-01 0.02551282 0.02076457
12 1e+02 1e-01 0.02807692 0.01458261
13 1e-02 1e+00 0.56115385 0.04344202
14 1e-01 1e+00 0.56115385 0.04344202
15 1e+00 1e+00 0.06634615 0.06187383
16 5e+00 1e+00 0.06128205 0.06186124
17 1e+01 1e+00 0.06128205 0.06186124
18 1e+02 1e+00 0.06128205 0.06186124
19 1e-02 5e+00 0.56115385 0.04344202
20 1e-01 5e+00 0.56115385 0.04344202
21 1e+00 5e+00 0.49224359 0.03806832
22 5e+00 5e+00 0.48967949 0.03738577
23 1e+01 5e+00 0.48967949 0.03738577
24 1e+02 5e+00 0.48967949 0.03738577
25 1e-02 1e+01 0.56115385 0.04344202
26 1e-01 1e+01 0.56115385 0.04344202
27 1e+00 1e+01 0.51775641 0.04471079
28 5e+00 1e+01 0.51012821 0.03817175
29 1e+01 1e+01 0.51012821 0.03817175
30 1e+02 1e+01 0.51012821 0.03817175
31 1e-02 1e+02 0.56115385 0.04344202
32 1e-01 1e+02 0.56115385 0.04344202
33 1e+00 1e+02 0.56115385 0.04344202
34 5e+00 1e+02 0.56115385 0.04344202
35 1e+01 1e+02 0.56115385 0.04344202
36 1e+02 1e+02 0.56115385 0.04344202

>
```

For a **radial kernel**, the lowest cross-validation error is obtained for a **gamma of 0.01 and a cost of 100**.

Problem 2

This problem uses the OJ data set in the ISLR package.

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

```
> set.seed(1)
> train <- sample(nrow(OJ), 800)
> train_data <- OJ[train, ]
> test_data <- OJ[-train, ]
>
```

Apurva Shrivastava
925009508
ISEN 613 - 600

(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.

```
> svm.linear <- svm(Purchase ~ ., data = train_data, kernel = "linear", cost = 0.01)
> summary(svm.linear)

Call:
svm(formula = Purchase ~ ., data = train_data, kernel = "linear", cost = 0.01)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.01
      gamma:  0.05555556

Number of Support Vectors:  432

 ( 215 217 )


Number of Classes:  2

Levels:
 CH MM


> |
```

Support vector classifier creates **432 support vectors out of 800 training points**. Out of these, **217** belong to level **MM** and remaining **215** belong to level **CH**.

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

Training error rate:

```
> train.pred <- predict(svm.linear, train_data)
> table(train_data$Purchase, train.pred)
    train.pred
      CH  MM
  CH 439  55
  MM  78 228
> |
```

From the table above:

Rate = (78 + 55) / (78 + 55 + 228 + 439) = **0.16625**

Apurva Shrivastava
925009508
ISEN 613 - 600

Test error rate:

```
> test.pred <- predict(svm.linear, test_data)
> table(test_data$Purchase, test.pred)
    test.pred
      CH  MM
  CH 141  18
  MM  31  80
>
```

From the table above: Rate = (31 + 18) / (31 + 18 + 141 + 80) = **0.1814**

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

```
> set.seed(2)
> svm.out <- tune(svm, Purchase ~ ., data = train_data, kernel = "linear", ranges = list(cost = 10^seq(-2, 1, by = 0.25)))
> summary(svm.out)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
  0.1

- best performance: 0.1625

- Detailed performance results:
         cost   error dispersion
1   0.01000000 0.17125 0.05172376
2   0.01778279 0.16500 0.05197489
3   0.03162278 0.16625 0.04604120
4   0.05623413 0.16500 0.04594683
5   0.10000000 0.16250 0.04787136
6   0.17782794 0.16250 0.04249183
7   0.31622777 0.16875 0.04379958
8   0.56234133 0.16625 0.03998698
9   1.00000000 0.16500 0.03670453
10  1.77827941 0.16625 0.03682259
11  3.16227766 0.16500 0.03717451
12  5.62341325 0.16500 0.03525699
13 10.00000000 0.16750 0.03917553

>
```

As we can see from the result, optimum cost is **0.1.**

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

```
> svm_fit <- svm(Purchase ~ ., kernel = "linear", data = train_data, cost = tune.out$best.parameter$cost)
> train.pred <- predict(svm_fit, train_data)
> table(train_data$Purchase, train.pred)
    train.pred
      CH  MM
  CH 440  54
  MM  73 233
>
```

Train error rate:

(73 + 54) / (73 + 54 + 440 + 233) = **0.15875**

```
> test.pred <- predict(svm_fit, test_data)
> table(test_data$Purchase, test.pred)
    test.pred
      CH  MM
  CH 140  19
  MM  33  78
>
```

Apurva Shrivastava
925009508
ISEN 613 - 600

Test error rate:

(33 + 19) / (33 + 19 + 140 + 78) = **0.19259**

(f) Repeat parts (b) through (e) using a support vector machine with a radial kernel. Use the tune() function to select an optimal cost and gamma.

```
> svm.radial <- svm(Purchase ~ ., kernel = "radial", data = train_data)
> summary(svm.radial)

Call:
svm(formula = Purchase ~ ., data = train_data, kernel = "radial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.05555556

Number of Support Vectors:  379

 ( 188 191 )


Number of Classes:  2

Levels:
 CH MM


>
```

```
> train.pred <- predict(svm.radial, train_data)
> table(train_data$Purchase, train.pred)
    train.pred
      CH  MM
  CH 455  39
  MM  77 229
>
```

Train error rate: (77 + 39) / (77 + 39 + 455 + 229) = **0.145**

```
> test.pred <- predict(svm.radial, test_data)
> table(test_data$Purchase, test.pred)
    test.pred
      CH  MM
  CH 141  18
  MM  28  83
>
```

Test error rate: (28 + 18) / (28 + 18 + 141 + 83) = **0.170**

Radial kernel with default gamma creates **379 support vectors**, out of which, **188** belong to level **CH** and remaining **191** belong to level **MM**. The classifier has a training error of **14.5%** and a test error of **17%** which is a slight improvement over linear kernel.

We now use tune to find optimal cost and gamma:

Apurva Shrivastava
925009508
ISEN 613 - 600

```
> set.seed(2)
> tune.out <- tune(svm, Purchase ~ ., data = train_data, kernel = "radial", ranges = list(cost = c(.01,.02,.05,.1,.2,.5,1,2,5,10), gamma = c(0.00
1,.002,.005,.01,.02,.05,.1,.2,.5,1,2,5,10)) )
> summary(tune.out)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
    1  0.05

- best performance: 0.16125


> svm.radial <- svm(Purchase ~ ., kernel = "radial", data = train_data, cost = tune.out$best.parameter$cost, gamma = tune.out$best.parameter$gamm
a)
> summary(svm.radial)

Call:
svm(formula = Purchase ~ ., data = train_data, kernel = "radial", cost = tune.out$best.parameter$cost, gamma = tune.out$best.parameter$gamma)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.05

Number of Support Vectors:  378

 ( 187 191 )


Number of Classes:  2

Levels:
 CH MM
```

Train error rate:

```
> train.pred <- predict(svm.radial, train_data)
> table(train_data$Purchase, train.pred)
    train.pred
      CH  MM
  CH 455  39
  MM  76 230
>
```

Error rate = (76 + 39) / (455 + 230 + 76 + 39) = **0.145**

Test error rate:

Apurva Shrivastava
925009508
ISEN 613 - 600

```
> test.pred <- predict(svm.radial, test_data)
> table(test_data$Purchase, test.pred)
    test.pred
      CH  MM
  CH 141  18
  MM  30  81
>
```

Test error rate: (30 + 18) / (30 +18 + 141 + 81) = **0.177**

Tuning does not reduce train and test error rates as we already used the optimal cost of 1 and gamma 0f 0.05.

(g) Repeat parts (b) through (e) using a support vector machine with a polynomial kernel. Set degree=2. Use the tune() function to select an optimal cost.

```
> svm.poly <- svm(Purchase ~ ., kernel = "polynomial", data = train_data, degree = 2)
> summary(svm.poly)

Call:
svm(formula = Purchase ~ ., data = train_data, kernel = "polynomial", degree = 2)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  1
     degree:  2
      gamma:  0.05555556
     coef.0:  0

Number of Support Vectors:  454

 ( 224 230 )


Number of Classes:  2

Levels:
 CH MM
```

```
> train.pred <- predict(svm.poly, train_data)
> table(train_data$Purchase, train.pred)
    train.pred
      CH  MM
  CH 461  33
  MM 105 201
>
```

Train error rate: (105 + 33) / (105 + 33 + 461 + 201) = **0.1725**

Apurva Shrivastava
925009508
ISEN 613 - 600

```
> test.pred <- predict(svm.poly, test_data)
> table(test_data$Purchase, test.pred)
    test.pred
      CH  MM
  CH 149  10
  MM  41  70
>
```

Test error rate: (41 + 10) / (41 + 10 + 149 + 70) = **0.188**

Polynomial kernel with default gamma creates **454** support vectors, out of which, **224** belong to level **CH** and remaining **230** belong to level **MM**. The classifier has a training error of **17.2%** and a test error of **18.8%** which is **no improvement over linear kernel**.

We now use cross validation to find optimal cost:

```
> set.seed(2)
> tune.out <- tune(svm, Purchase ~ ., data = train_data, kernel = "polynomial", degree = 2, ranges = list(cost = 10^seq(-2,1, by = 0.25)))
> summary(tune.out)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
   10

- best performance: 0.18125

- Detailed performance results:
         cost   error dispersion
1   0.01000000 0.38250 0.04533824
2   0.01778279 0.36750 0.04972145
3   0.03162278 0.36500 0.05458174
4   0.05623413 0.33375 0.05070681
5   0.10000000 0.32500 0.04677072
6   0.17782794 0.25875 0.05952649
7   0.31622777 0.21250 0.06123724
8   0.56234133 0.21250 0.05743354
9   1.00000000 0.19750 0.06687468
10  1.77827941 0.19375 0.05376453
11  3.16227766 0.19625 0.05653477
12  5.62341325 0.18375 0.05434266
13 10.00000000 0.18125 0.05245699

>
```

```
> svm.poly <- svm(Purchase ~ ., kernel = "polynomial", degree = 2, data = train_data, cost = tune.out$best.parameter$cost)
> summary(svm.poly)

Call:
svm(formula = Purchase ~ ., data = train_data, kernel = "polynomial", degree = 2, cost = tune.out$best.parameter$cost)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  10
     degree:  2
      gamma:  0.05555556
     coef.0:  0

Number of Support Vectors:  342

 ( 170 172 )


Number of Classes:  2

Levels:
 CH MM
```

Apurva Shrivastava
925009508
ISEN 613 - 600

```
> train.pred <- predict(svm.poly, train_data)
> table(train_data$Purchase, train.pred)
    train.pred
      CH  MM
  CH 450  44
  MM  72 234
>
```

Train error rate: (72 + 44) / (72 + 44 + 450 + 234) = **0.145**

```
> test.pred <- predict(svm.poly, test_data)
> table(test_data$Purchase, test.pred)
    test.pred
      CH  MM
  CH 140  19
  MM  31  80
>
```

Test error rate: (31 + 19) / (31 + 19 + 140 + 80) = **0.185**

<u>Conclusion:</u> Tuning reduce train and test error rates.

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

Overall, radial model produces the best results on this data with minimum overall error rates for tuned and untuned models.