STAT 6340 Statistical and Machin Learning

Bonus Project

Buddhika Jayawardana

Section 1

Problem 1

a) support vector classifier with linear kernel was fitted and the optimal cost parameter was found using 10-fold cross-validation as 0.1. There were 4 misclassifications on the training set and only 1 misclassification on the test set.

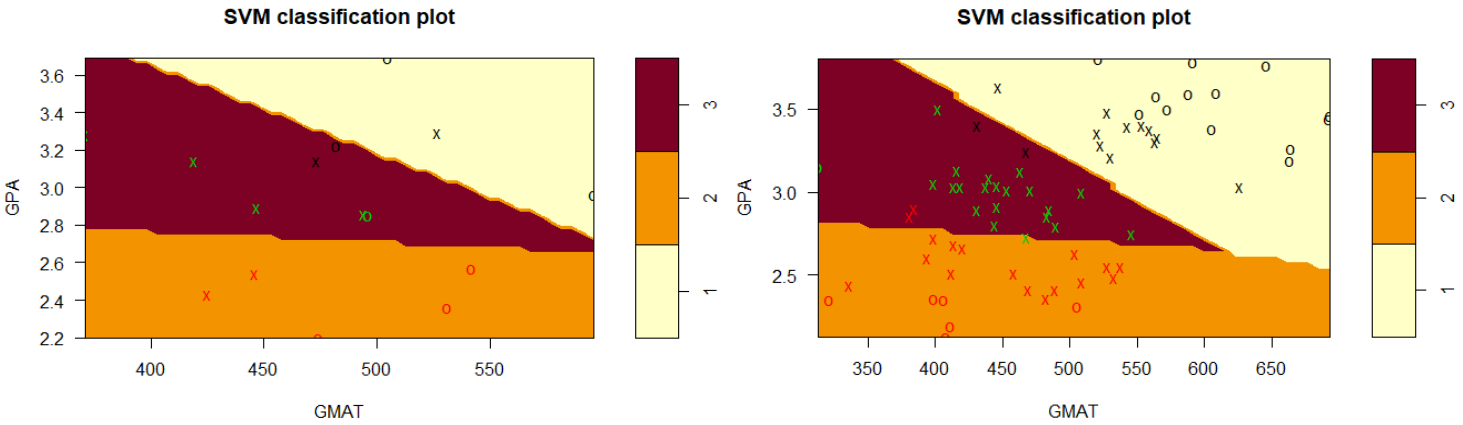
10-fold CV error = 0.05714286

Training set confusion matrix

```
     Pred
true  1  2  3
   1 25  0  1
   2  0 23  0
   3  1  2 18
```

Test set confusion matrix

```
    pred
true 1 2 3
   1 4 0 1
   2 0 5 0
   3 0 0 5
```



b) support vector classifier with polynomial degree 2 kernel was fitted and the optimal cost parameter was found using 10-fold cross-validation as 73.3. There were 21 misclassifications on the training set and 6 misclassifications on the test set.

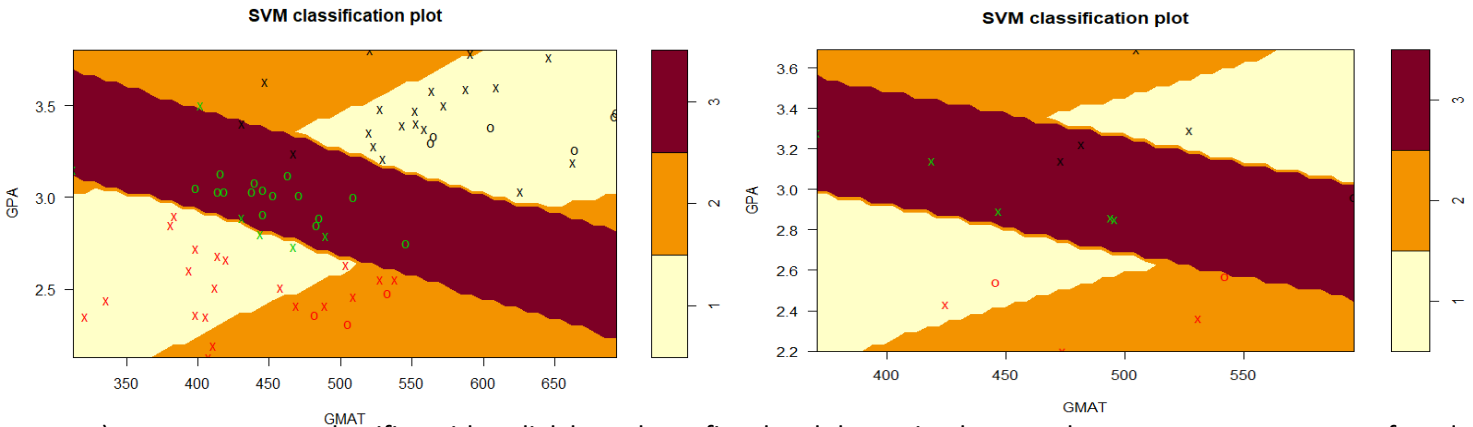
10-fold CV error = 0.04285714

Training set confusion matrix

```
     Pred
true  1  2  3
   1 21  3  2
   2 13 10  0
   3  2  1 18
```

Test set confusion matrix

```
    pred
true 1 2 3
   1 1 1 3
   2 2 3 0
   3 0 0 5
```



c) support vector classifier with radial kernel was fitted and the optimal cost and gamma parameters were found using 10-fold cross-validation as 12.5 and 0.8. This method classified the training set perfectly and there was only 1 misclassification in the test set.

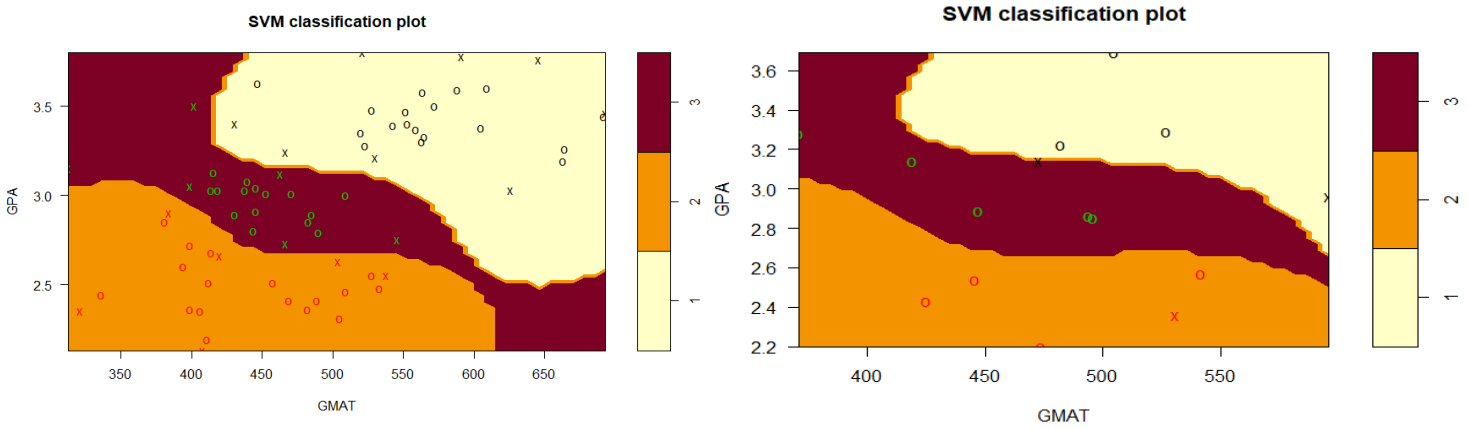
10-fold CV error = 0.2164154

Training set confusion matrix

```
     Pred
true  1  2  3
   1 26  0  0
   2  0 23  0
   3  0  0 21
```

Test set confusion matrix

```
    pred
true 1 2 3
   1 4 0 1
   2 0 5 0
   3 0 0 5
```



d) In project 2, we saw QDA had the lowest test and training error rates. It had 2 misclassifications for the training set and only 1 misclassification for the test set.

Method	Training Misclassification error rate	Test Misclassification error rate	10-fold CV error
SVM linear	4/70	1/15	0.05714286
SVM polynomial degree 2	21/70	6/15	0.3142857
SVM radial	0	1/15	0.04285714
QDA	2/70	1/15	

All three methods SVM linear, radial and QDA have the same lowest test error rate and by comparing test error rate, best method out of them is SVM radial. Also the cross validation error was lower for SVM radial too.

Problem 2

a) support vector classifier with linear kernel was fitted and the optimal cost parameter was found using 10-fold cross-validation as 0.1. However this method was not able to classify the training data set as it only clustered all the data as bad quality.
10-fold CV error = 0.2164154
Misclassification error rate = 0.2164149

Training set confusion matrix

```
     pred
true    0    1
   0 3838    0
   1 1060    0
```

b) support vector classifier with polynomial degree 2 kernel was fitted and the optimal cost parameter was found using 10-fold cross-validation as 8.9.
10-fold CV error = 0.1872213
Misclassification error rate = 0.1817068

Training set confusion matrix

```
     pred
true    0    1
   0 3766   72
   1  818  242
```

c) support vector classifier with radial kernel was fitted and the optimal cost and gamma parameters were found using 10-fold cross-validation as 16 and 1. This method classified the training set with only 3 misclassifications.
10-fold CV error = 0.1235216
Misclassification error rate = 0.0006124949

Training set confusion matrix

```
     pred
true    0    1
   0 3836    2
   1    1 1059
```

d) In project 4, we saw Ridge regression method had the lowest test and training error rates.

Method	**Misclassification error rate**	**10-fold CV error**
SVM linear	0.2164149	0.2164154
SVM polynomial degree 2	0.1817068	0.1872213
SVM radial	0.0006124949	0.1235216
Full set	0.1976317	0.2000789
Best Subset	0.1984483	0.1990572
Backward	0.1984483	0.1990572
Forward	0.1982442	0.1994666
Ridge	0.1965238	0.1976317
Lasso	0.1969356	0.1978359

Out of all the methods, SVM radial has the lowest 10-fold CV error and it preformed better at clustering the data set with only 3 misclassifications. Thus, the recommended method is SVM radial method.

Section 2

```
# problem 1

library(e1071) # for SVM

admission.data <- read.csv('admission.csv', header = T)
attach(admission.data)
str(admission.data)

test <- rbind(admission.data[Group==1,][1:5,],
              admission.data[Group==2,][1:5,],
              admission.data[Group==3,][1:5,])

train <- rbind(admission.data[Group==1,][-(1:5),],
               admission.data[Group==2,][-(1:5),],
               admission.data[Group==3,][-(1:5),])

plot(train[,1:2], xlab = "GPA", ylab = "GMAT",
     col = train$Group, pch=19)

# part a)
set.seed(1)
tune.out.a <- tune(svm, as.factor(Group) ~ ., data = train, kernel = "linear",
                   cross = 10, ranges = list(cost = seq(0.01,0.5,0.01)))

summary(tune.out.a)
# Parameter tuning of 'svm':
#
#   - sampling method: 10-fold cross validation
#
# - best parameters:
#   cost
# 0.1
#
# - best performance: 0.05714286

bestmod.a <- tune.out.a$best.model
summary(bestmod.a)
# Call:
#   best.tune(method = svm, train.x = as.factor(Group) ~ ., data = train, ranges =
list(cost = seq(0.01,0.5,0.01)), kernel = "linear",
#             cross = 10)
#
#
# Parameters:
#   SVM-Type:  C-classification
# SVM-Kernel:  linear
# cost:  0.1
#
# Number of Support Vectors:  50
#
# ( 13 17 20 )
#
#
# Number of Classes:  3
#
# Levels:
#   1 2 3
#
```

```
# 10-fold cross-validation on training data:
#
#   Total Accuracy: 91.42857
# Single Accuracies:
#   100 100 85.71429 100 85.71429 100 71.42857 85.71429 100 85.71429

plot(bestmod.a, train, main = 'SVM classification on training set')
table(true = train$Group, pred = predict(bestmod.a, newdata = train[,1:2]))
#    pred
# true  1  2  3
#    1 24  0  2
#    2  0 21  2
#    3  0  0 21


plot(bestmod.a, test, main = 'SVM classification on test set')
table(true = test$Group, pred = predict(bestmod.a, newdata = test[,1:2]))
#   pred
# true 1 2 3
#    1 4 0 1
#    2 0 5 0
#    3 0 0 5

# part b)
set.seed(1)
tune.out.b <- tune(svm, as.factor(Group) ~ ., data = train, kernel = "polynomial", degree
= 2,
                   cross = 10, ranges = list(cost = seq(70,80,0.1)))

summary(tune.out.b)
# Parameter tuning of 'svm':
#
#   - sampling method: 10-fold cross validation
#
# - best parameters:
#   cost
# 73.3
#
# - best performance: 0.3142857

bestmod.b <- tune.out.b$best.model
summary(bestmod.b)
# Call:
#   best.tune(method = svm, train.x = as.factor(Group) ~ ., data = train, ranges =
list(cost = seq(0.1, 100, 0.1)), kernel = "polynomial",
#             degree = 2, cross = 10)
#
#
# Parameters:
#   SVM-Type:  C-classification
# SVM-Kernel:  polynomial
# cost:  73.3
# degree:  2
# coef.0:  0
#
# Number of Support Vectors:  46
#
# ( 20 20 6 )
#
#
```

```
# Number of Classes:  3
#
# Levels:
#   1 2 3
#
# 10-fold cross-validation on training data:
#
#   Total Accuracy: 65.71429
# Single Accuracies:
#   57.14286 57.14286 85.71429 71.42857 57.14286 85.71429 42.85714 85.71429 57.14286
57.14286

plot(bestmod.b, train, main = 'SVM classification on training set')
table(true = train$Group, pred = predict(bestmod.b, newdata = train[,1:2]))
#     pred
# true  1  2  3
#    1 21  3  2
#    2 13 10  0
#    3  2  1 18


plot(bestmod.b, test, main = 'SVM classification on test set')
table(true = test$Group, pred = predict(bestmod.b, newdata = test[,1:2]))
#   pred
# true 1 2 3
#    1 1 1 3
#    2 2 3 0
#    3 0 0 5

# part c)
set.seed(1)
tune.out.c <- tune(svm, as.factor(Group) ~ ., data = train, kernel = "radial",
                   cross = 10, ranges = list(cost = seq(10,15,0.1), gamma =
seq(0.1,1,0.1)))

summary(tune.out.c)
# Parameter tuning of 'svm':
#
#   - sampling method: 10-fold cross validation
#
# - best parameters:
#   cost gamma
# 12.5   0.8
#
# - best performance: 0.04285714

bestmod.c <- tune.out.c$best.model
summary(bestmod.c)
# Call:
#   best.tune(method = svm, train.x = as.factor(Group) ~ ., data = train, ranges =
list(cost = seq(10, 15, 0.1), gamma = seq(0.1,
#
1, 0.1)), kernel = "radial", cross = 10)
#
#
# Parameters:
#   SVM-Type:  C-classification
# SVM-Kernel:  radial
# cost:  12.5
#
```

```
# Number of Support Vectors:  20
#
# ( 8 6 6 )
#
#
# Number of Classes:  3
#
# Levels:
#   1 2 3
#
# 10-fold cross-validation on training data:
#
#   Total Accuracy: 95.71429
# Single Accuracies:
#   100 100 100 85.71429 85.71429 85.71429 100 100 100 100

plot(bestmod.c, train, main = 'SVM classification on training set')
table(true = train$Group, pred = predict(bestmod.c, newdata = train[,1:2]))
#     pred
# true  1  2  3
#    1 26  0  0
#    2  0 23  0
#    3  0  0 21

plot(bestmod.c, test, main = 'SVM classification on test set')
table(true = test$Group, pred = predict(bestmod.c, newdata = test[,1:2]))
#    pred
# true 1 2 3
#    1 4 0 1
#    2 0 5 0
#    3 0 0 5




# problem 2

library(e1071) # for SVM


wine <- read.csv("winequality-white.csv", header = T, sep=';')
wine$quality <- ifelse(wine$quality >= 7, 1, 0)
wine$quality <- as.factor(wine$quality)
attach(wine)
str(wine)


# part a)

set.seed(1)
tune.out.a <- tune(svm, quality ~ ., data = wine, kernel = "linear", cross = 10,
                   ranges = list(cost = seq(0.1,1,0.1)), scale = T)
# 0.1


summary(tune.out.a)
# Parameter tuning of 'svm':
#
#   - sampling method: 10-fold cross validation
#
# - best parameters:
```

```
#   cost
# 0.1
#
# - best performance: 0.2164154

bestmod.a <- tune.out.a$best.model
summary(bestmod.a)
# Call:
#   best.tune(method = svm, train.x = quality ~ ., data = wine, ranges = list(cost =
seq(0.1,
#
1, 0.1)), kernel = "linear", cross = 10, scale = T)
#
#
# Parameters:
#   SVM-Type:  C-classification
# SVM-Kernel:  linear
# cost:  0.1
#
# Number of Support Vectors:  2218
#
# ( 1158 1060 )
#
#
# Number of Classes:  2
#
# Levels:
#   0 1
#
# 10-fold cross-validation on training data:
#
#   Total Accuracy: 78.35851
# Single Accuracies:
#   75.86912 77.14286 78.16327 80.40816 76.73469 79.5501 78.16327 76.53061 80.40816
80.61224

table(true = quality, pred = predict(bestmod.a, newdata = wine[,-12]))
#     pred
# true    0    1
#    0 3838    0
#    1 1060    0

# Misclassification error rate
1060/(3838+1060)
# 0.2164149


# part b)

set.seed(1)
tune.out.b <- tune(svm, quality ~ ., data = wine, kernel = "polynomial", degree = 2,
                   cross = 10, ranges = list(cost = seq(8.5,9.5,0.1)), scale = T)

# 8.9

summary(tune.out.b)
# Parameter tuning of 'svm':
#
#   - sampling method: 10-fold cross validation
#
```

```
# - best parameters:
#   cost
# 8.9
#
# - best performance: 0.1872213


bestmod.b <- tune.out.b$best.model
summary(bestmod.b)
# Call:
#   best.tune(method = svm, train.x = quality ~ ., data = wine, ranges = list(cost =
seq(8.5,
#
9.5, 0.1)), kernel = "polynomial", degree = 2, cross = 10, scale = T)
#
#
# Parameters:
#   SVM-Type:  C-classification
# SVM-Kernel:  polynomial
# cost:  8.9
# degree:  2
# coef.0:  0
#
# Number of Support Vectors:  2078
#
# ( 1061 1017 )
#
#
# Number of Classes:  2
#
# Levels:
#   0 1
#
# 10-fold cross-validation on training data:
#
#   Total Accuracy: 81.4414
# Single Accuracies:
#   81.39059 82.2449 80.40816 81.22449 82.04082 78.11861 81.22449 84.89796 79.59184
83.26531

table(true = quality, pred = predict(bestmod.b, newdata = wine[,-12]))
#     pred
# true    0    1
#    0 3766   72
#    1  818  242

# Misclassification rate
(818+72)/(3766+72+818+242)
# 0.1817068


# part c)

set.seed(1)
tune.out.c <- tune(svm, quality ~ ., data = wine, kernel = "radial",
                   cross = 10, ranges = list(cost = seq(13,16), gamma = seq(1,2)))


summary(tune.out.c)
# Parameter tuning of 'svm':
```

```
#
#   - sampling method: 10-fold cross validation
#
# - best parameters:
#   cost gamma
# 16     1
#
# - best performance: 0.1235216


bestmod.c <- tune.out.c$best.model
summary(bestmod.c)
# Call:
#   best.tune(method = svm, train.x = quality ~ ., data = wine, ranges = list(cost =
seq(13,
#
16), gamma = seq(1, 3)), kernel = "radial", cross = 10)
#
#
# Parameters:
#   SVM-Type:  C-classification
# SVM-Kernel:  radial
# cost:  16
#
# Number of Support Vectors:  3334
#
# ( 2476 858 )
#
#
# Number of Classes:  2
#
# Levels:
#   0 1
#
# 10-fold cross-validation on training data:
#
#   Total Accuracy: 88.05635
# Single Accuracies:
#   87.93456 87.7551 87.95918 88.97959 87.7551 86.29857 87.7551 89.59184 89.18367
87.34694

table(true = quality, pred = predict(bestmod.c, newdata = wine[,-12]))
#     pred
# true    0    1
#    0 3836    2
#    1    1 1059

# Misclassification error rate
3/(3836+1+2+1059)
# 0.0006124949
```