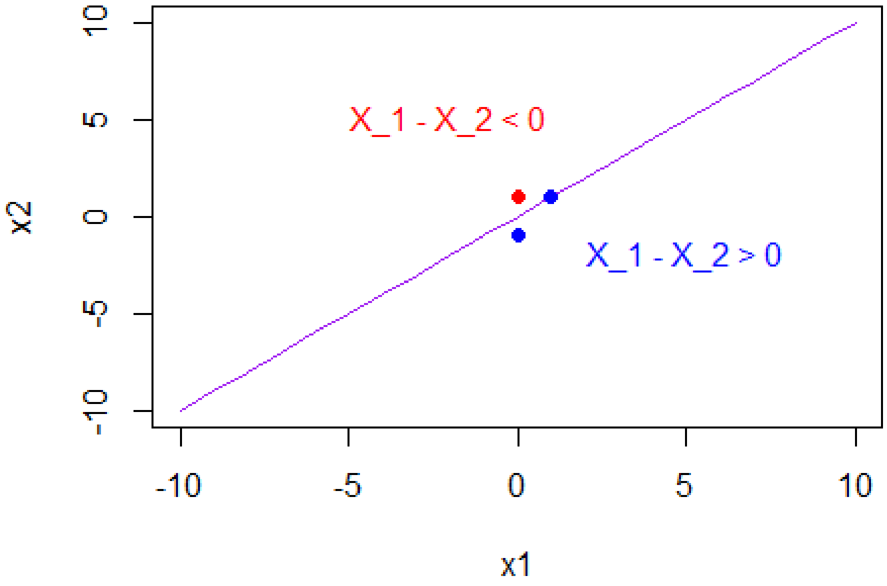
HW3-MATH4323

anthonycastillo ID:1670011

2022-10-03

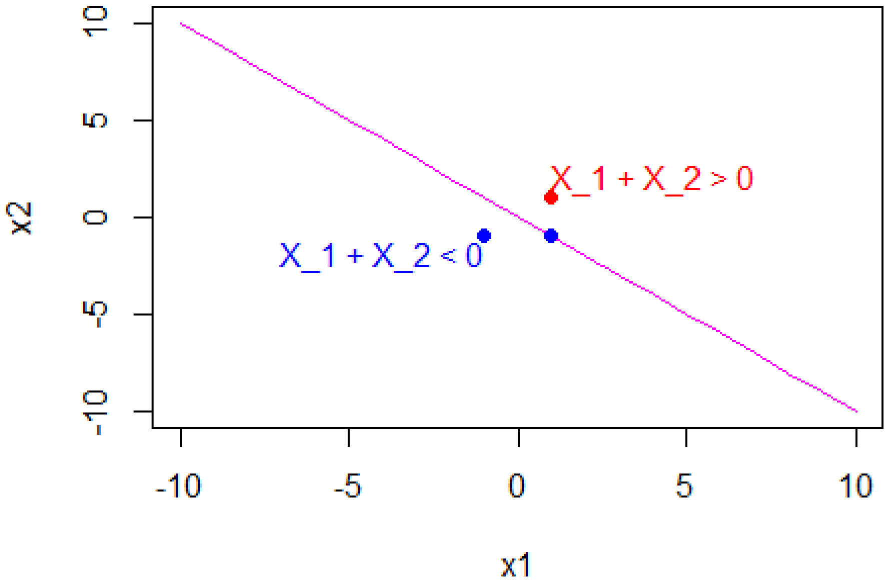
Question 1 (a):

```
x1 <- -10:10
x2 <- x1
plot(x1,x2, col = "purple",type = "l")
points(1,1, pch = 19, col = "blue")
points(0,-1,pch = 19,col = "blue")
points(0,1, pch = 19,col = "red")
text(-2,5,"X_1 - X_2 < 0", col = "red")
text(5,-2,"X_1 - X_2 > 0", col = "blue")
```



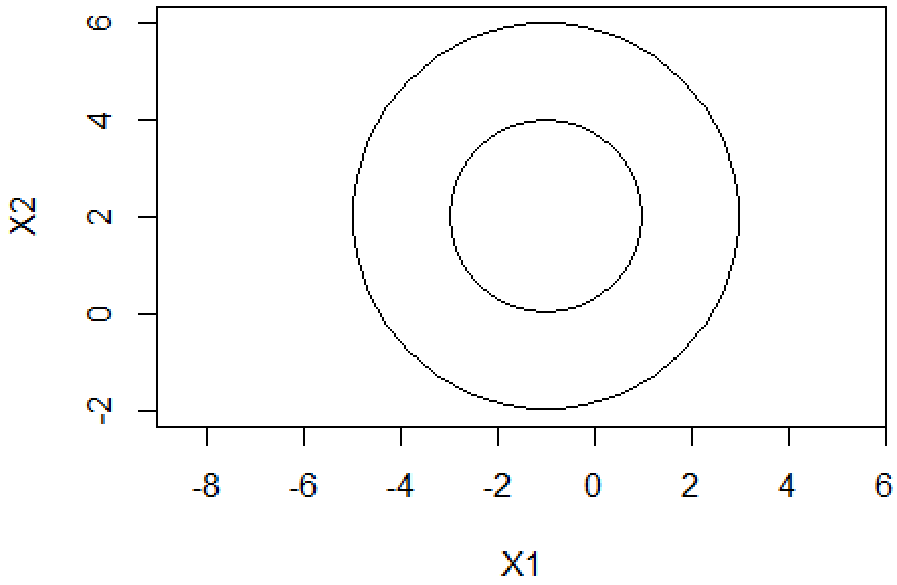
Question 1 (b):

```
x1 <- -10:10
x2 <- -x1
plot(x1,x2, type = "l", col = "magenta")
points(1,1, pch = 19, col = "red")
points(-1,-1,pch = 19,col = "blue")
points(1,-1, pch = 19,col = "blue")
text(-4,-2,"X_1 + X_2 < 0", col = "blue")
text(4,2,"X_1 + X_2 > 0", col = "red")
```



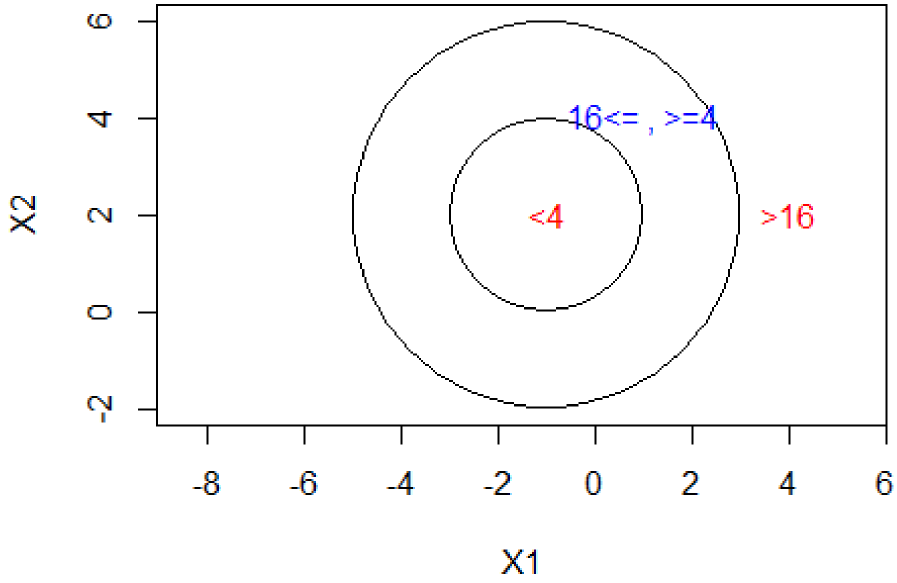
Question 2 (a):

```
plot(NA, NA, type = "n", xlim = c(-2, -1), ylim = c(-2, 6), asp = 1, xlab =
"X1", ylab = "X2")
symbols(c(-1), c(2), circles = c(2), add = TRUE, inches = FALSE)
symbols(c(-1), c(2), circles = c(4), add = TRUE, inches = FALSE)
```



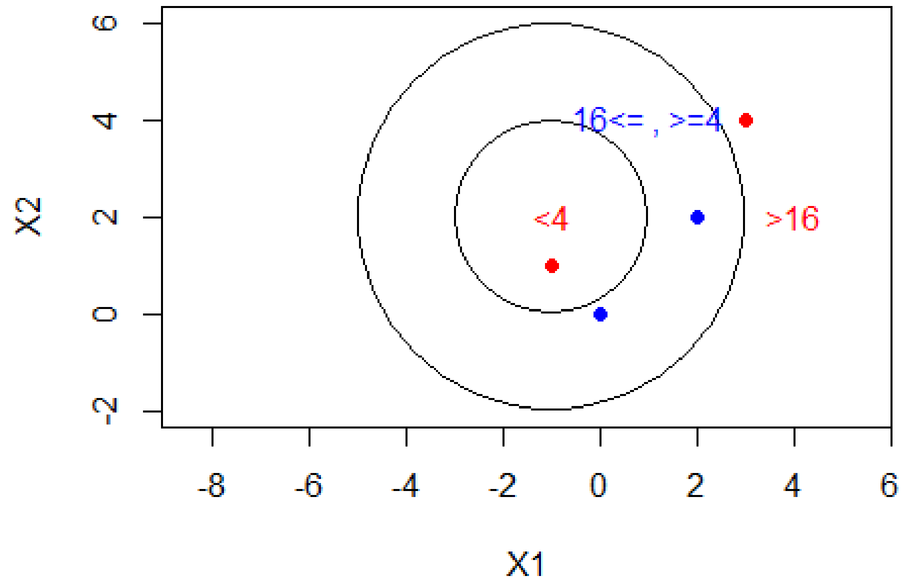
Question 2 (b):

```
plot(NA, NA, type = "n", xlim = c(-2, -1), ylim = c(-2, 6), asp = 1, xlab =
"X1", ylab = "X2")
symbols(c(-1), c(2), circles = c(2), add = TRUE, inches = FALSE)
symbols(c(-1), c(2), circles = c(4), add = TRUE, inches = FALSE)
text(4,2,">16", col = "red")
text(-1,2,"<4", col = "red")
text(1,4,"16<= , >=4", col = "blue")
```



Question 2 (c):

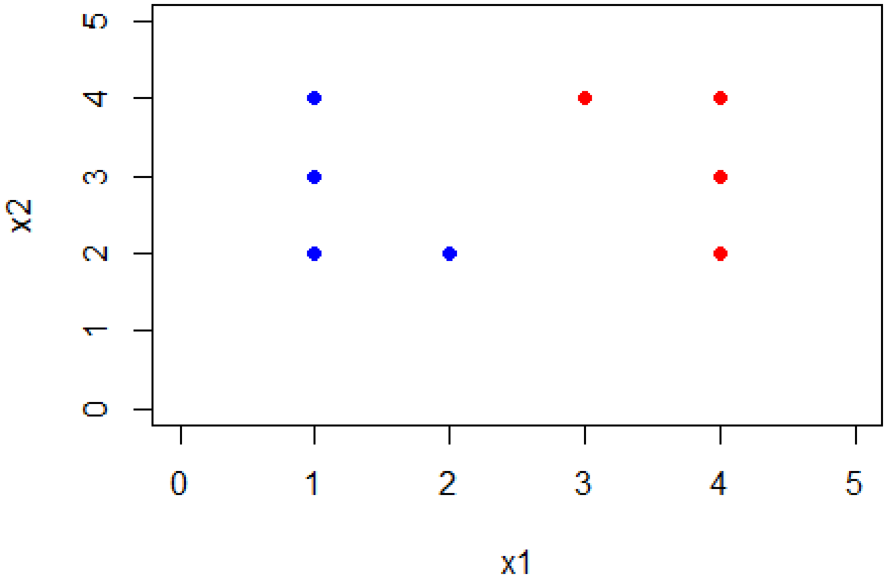
```
plot(NA, NA, type = "n", xlim = c(-2, -1), ylim = c(-2, 6), asp = 1, xlab =
"X1", ylab = "X2")
symbols(c(-1), c(2), circles = c(2), add = TRUE, inches = FALSE)
symbols(c(-1), c(2), circles = c(4), add = TRUE, inches = FALSE)
text(4,2,">16", col = "red")
text(-1,2,"<4", col = "red")
text(1,4,"16<= , >=4", col = "blue")
points(0,0,pch=19,col = "blue")
points(-1,1,pch=19,col = "red")
points(2,2,pch=19,col = "blue")
points(3,4,pch=19,col = "red")
```



Question 2 (d):
$(1+X_1)^2 + (2-X_2)^2 = 4$ is linear by expanding it $1 + X_1 + X_1^2 + X_2 + X_2^2 = 0$

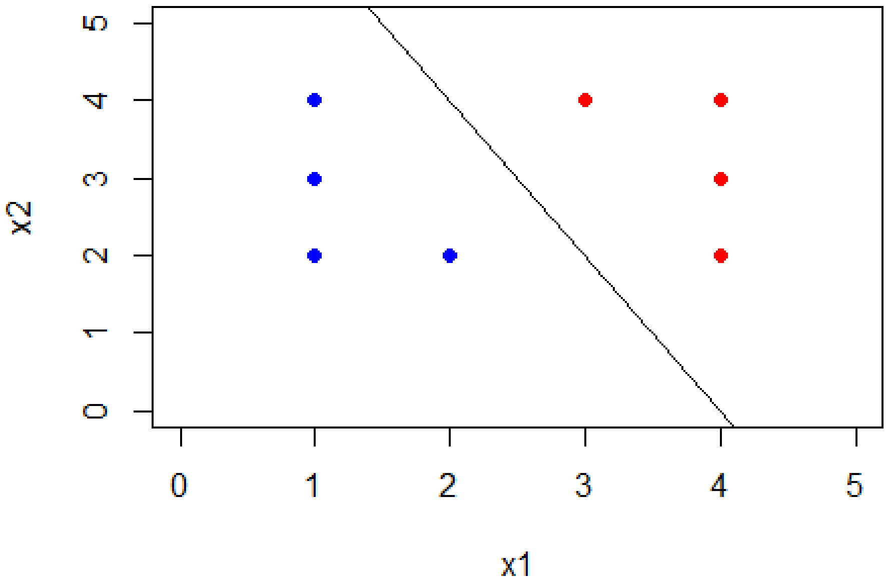
Question 3 (a):

```
x1 <- c(1,3,4,2,4,4,1,1)
x2 <- c(4,4,3,2,2,4,2,3)
plot.color <- c("blue","red","red","blue","red","red","blue","blue")
plot(x1,x2, col = plot.color, xlim = c(0,5), ylim = c(0,5), pch = 19)
```



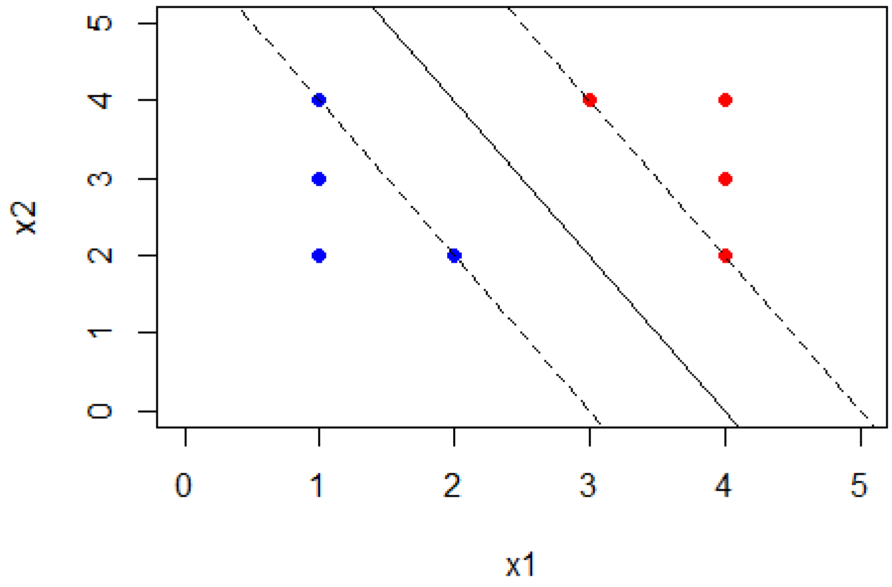
Question 3 (b):

```
plot(x1,x2, col = plot.color, xlim = c(0,5), ylim = c(0,5), pch = 19)
abline(8,-2)
```



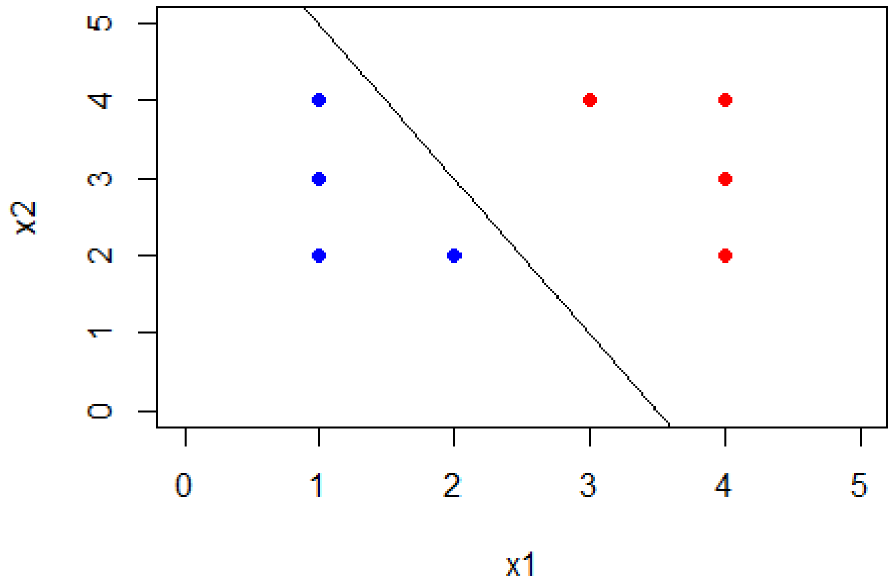
equation for hyperplane: x1+x2+8 = 0
Question 3 (c):
classify to blue if x1+x2+1 < 0 and classify to red if x1+x2+1 > 0
Question 3 (d):

```
plot(x1,x2, col = plot.color, xlim = c(0,5), ylim = c(0,5), pch = 19)
abline(8,-2)
abline(6.0,-2.0, lty = 2)
abline(10.0,-2.0, lty = 2)
```



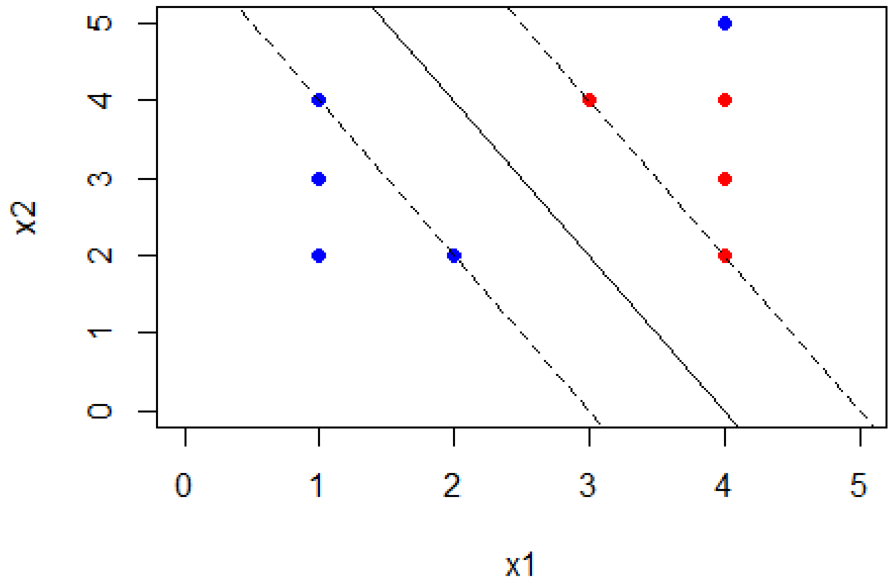
Question 3 (e):
The support vectors are (2,2),(1,4),(4,2),and (3,4)
Question 3 (f):
The 7th obs is (1,2) which is not a support vector which will not change the maximal margin hyperplane
Question 3 (g):

```
plot(x1,x2, col = plot.color, xlim = c(0,5), ylim = c(0,5), pch = 19)
abline(7,-2)
```



equation for hyperplane: x1+x2+7 = 0
Question 3 (h):

```
plot(x1,x2, col = plot.color, xlim = c(0,5), ylim = c(0,5), pch = 19)
abline(8,-2)
abline(6.0,-2.0, lty = 2)
abline(10.0,-2.0, lty = 2)
points(4,5, pch = 19, col = "blue")
```



Question 4 (a):

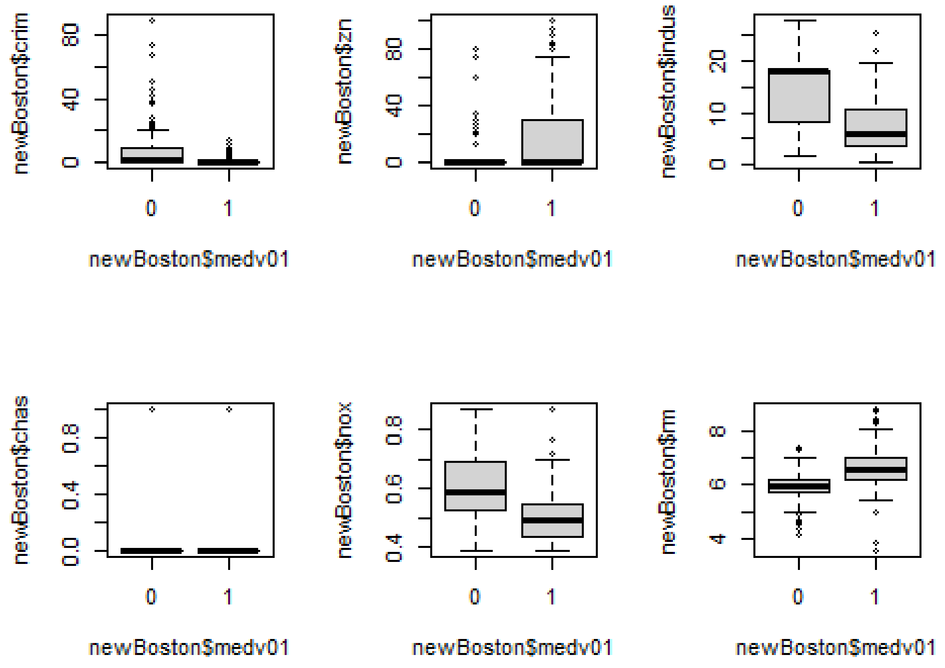
```
library(MASS)
library(ISLR)

## Warning: package 'ISLR' was built under R version 4.1.3

library(class)
library(caret)

## Warning: package 'caret' was built under R version 4.1.3

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.1.3

## Loading required package: lattice

newBoston <- data.frame(Boston)
medv.med <- median(Boston$medv)
newBoston$medv01 <- ifelse(Boston$medv > medv.med, yes = 1, no = 0)
newBoston$medv = NULL
```

Question 4 (b):

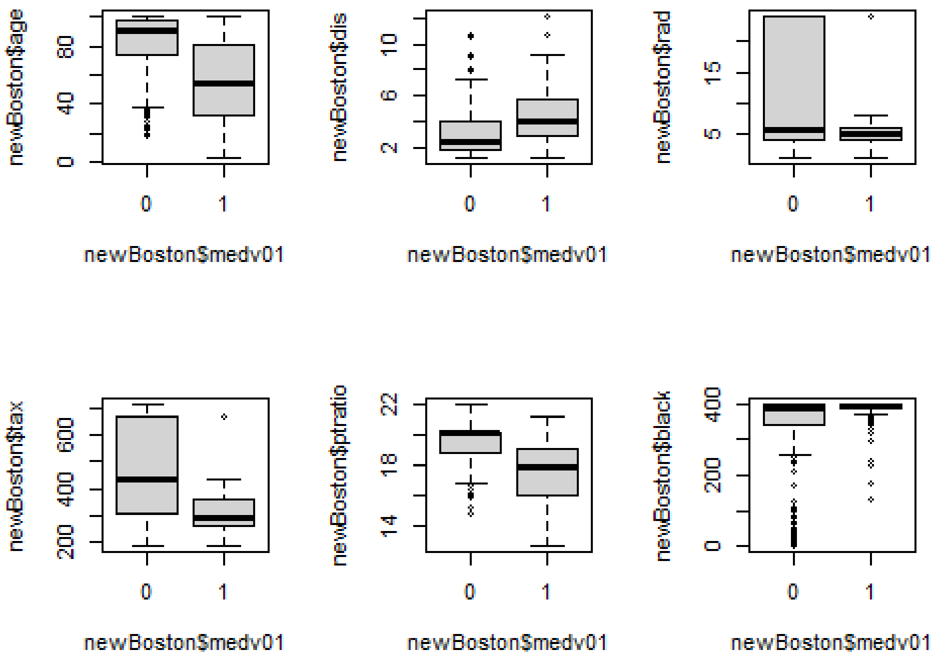
```
cor(newBoston)
```

```
##                crim          zn       indus         chas         nox
## crim     1.00000000 -0.20046922  0.40658341 -0.055891582  0.42097171
## zn      -0.20046922  1.00000000 -0.53382819 -0.042696719 -0.51660371
## indus    0.40658341 -0.53382819  1.00000000  0.062938027  0.76365145
## chas    -0.05589158 -0.04269672  0.06293803  1.000000000  0.09120281
## nox      0.42097171 -0.51660371  0.76365145  0.091202807  1.00000000
## rm      -0.21924670  0.31199059 -0.39167585  0.091251225 -0.30218819
## age      0.35273425 -0.56953734  0.64477851  0.086517774  0.73147010
## dis     -0.37967009  0.66440822 -0.70802699 -0.099175780 -0.76923011
## rad      0.62550515 -0.31194783  0.59512927 -0.007368241  0.61144056
## tax      0.58276431 -0.31456332  0.72076018 -0.035586518  0.66802320
## ptratio  0.28994558 -0.39167855  0.38324756 -0.121515174  0.18893268
## black   -0.38506394  0.17552032 -0.35697654  0.048788485 -0.38005064
## lstat    0.45562148 -0.41299457  0.60379972 -0.053929298  0.59087892
## medv01  -0.31397528  0.33050237 -0.47164023  0.135646949 -0.45000549
##                  rm         age         dis          rad          tax
ptratio
## crim    -0.21924670  0.35273425 -0.37967009  0.625505145  0.58276431
0.2899456
## zn       0.31199059 -0.56953734  0.66440822 -0.311947826 -0.31456332 -
0.3916785
## indus   -0.39167585  0.64477851 -0.70802699  0.595129275  0.72076018
0.3832476
## chas     0.09125123  0.08651777 -0.09917578 -0.007368241 -0.03558652 -
0.1215152
## nox     -0.30218819  0.73147010 -0.76923011  0.611440563  0.66802320
0.1889327
## rm       1.00000000 -0.24026493  0.20524621 -0.209846668 -0.29204783 -
0.3555015
## age     -0.24026493  1.00000000 -0.74788054  0.456022452  0.50645559
0.2615150
## dis      0.20524621 -0.74788054  1.00000000 -0.494587930 -0.53443158 -
0.2324705
## rad     -0.20984667  0.45602245 -0.49458793  1.000000000  0.91022819
0.4647412
## tax     -0.29204783  0.50645559 -0.53443158  0.910228189  1.00000000
0.4608530
## ptratio -0.35550149  0.26151501 -0.23247054  0.464741179  0.46085304
1.0000000
## black    0.12806864 -0.27353398  0.29151167 -0.444412816 -0.44180801 -
0.1773833
## lstat   -0.61380827  0.60233853 -0.49699583  0.488676335  0.54399341
0.3740443
## medv01   0.50961542 -0.47493222  0.30185475 -0.359152204 -0.45097795 -
0.4601715
##               black      lstat     medv01
## crim    -0.38506394  0.4556215 -0.3139753
## zn       0.17552032 -0.4129946  0.3305024
## indus   -0.35697654  0.6037997 -0.4716402
## chas     0.04878848 -0.0539293  0.1356469
## nox     -0.38005064  0.5908789 -0.4500055
## rm       0.12806864 -0.6138083  0.5096154
## age     -0.27353398  0.6023385 -0.4749322
## dis      0.29151167 -0.4969958  0.3018547
## rad     -0.44441282  0.4886763 -0.3591522
## tax     -0.44180801  0.5439934 -0.4509779
## ptratio -0.17738330  0.3740443 -0.4601715
## black    1.00000000 -0.3660869  0.3029397
## lstat   -0.36608690  1.0000000 -0.6633267
## medv01   0.30293974 -0.6633267  1.0000000
```

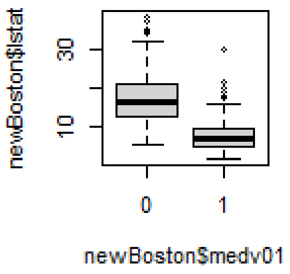
```
par(mfrow=c(2,3))
boxplot(newBoston$crim~newBoston$medv01)
boxplot(newBoston$zn~newBoston$medv01)
boxplot(newBoston$indus~newBoston$medv01)
boxplot(newBoston$chas~newBoston$medv01)
boxplot(newBoston$nox~newBoston$medv01)
boxplot(newBoston$rm~newBoston$medv01)
```



```
boxplot(newBoston$age~newBoston$medv01)
boxplot(newBoston$dis~newBoston$medv01)
boxplot(newBoston$rad~newBoston$medv01)
boxplot(newBoston$tax~newBoston$medv01)
boxplot(newBoston$ptratio~newBoston$medv01)
boxplot(newBoston$black~newBoston$medv01)
```



```
boxplot(newBoston$lstat~newBoston$medv01)
```



I will use the k-values 1,5,10

my three subset predictors will be all predictors,(nox, ptratio, lstat), and (tax,age,rm)

Question 4 (c):

#LOOCV

```
set.seed(1)
n <-nrow(newBoston)
boston.train <- cbind(newBoston$nox,
                      newBoston$ptratio,
                      newBoston$lstat)
boston.y.train <- newBoston$medv01

set.seed(1)
for(i in c(1,5,10)){
  boston.knn.cv <- knn.cv(train = boston.train,cl = boston.y.train,k=i)
  print(mean(boston.knn.cv != boston.y.train))
}
```

```
## [1] 0.229249
## [1] 0.173913
## [1] 0.1798419
```

```
set.seed(1)
n <-nrow(newBoston)
boston.train <- newBoston[,-14]
boston.y.train <- newBoston[,"medv01"]

set.seed(1)
for(i in c(1,5,10)){
  boston.knn.cv <- knn.cv(train = boston.train,cl = boston.y.train,k=i)
  print(mean(boston.knn.cv != boston.y.train))
}
```

```
## [1] 0.1936759
## [1] 0.1758893
## [1] 0.215415
```

```
set.seed(1)
n <-nrow(newBoston)
boston.train <- cbind(newBoston$tax,
                      newBoston$age,
                      newBoston$rm)
boston.y.train <- newBoston$medv01

set.seed(1)
for(i in c(1,5,10)){
  boston.knn.cv <- knn.cv(train = boston.train,cl = boston.y.train,k=i)
  print(mean(boston.knn.cv != boston.y.train))
}
```

```
## [1] 0.2549407
## [1] 0.2509881
## [1] 0.243083
```

The model with the predictor nox,ptratio, and lstat and k-value = 5 had the best results of all the models
Question 4 (d):
The variable in the Boston data set are not in the same scale. You can divide the data into training and test set and scale them
Question 4 (e):

```
set.seed(1)
n <-nrow(newBoston)
boston.train <- scale(cbind(newBoston$nox,
                      newBoston$ptratio,
                      newBoston$lstat))
boston.y.train <- newBoston$medv01

set.seed(1)
for(i in c(1,5,10)){
  boston.knn.cv <- knn.cv(train = boston.train,cl = boston.y.train,k=i)
  print(mean(boston.knn.cv != boston.y.train))
}
```

```
## [1] 0.1660079
## [1] 0.1778656
## [1] 0.1620553
```

```
set.seed(1)
n <-nrow(newBoston)
boston.train <- scale(newBoston[,-14])
boston.y.train <- newBoston[,"medv01"]

set.seed(1)
for(i in c(1,5,10)){
  boston.knn.cv <- knn.cv(train = boston.train,cl = boston.y.train,k=i)
  print(mean(boston.knn.cv != boston.y.train))
}
```

```
## [1] 0.1383399
## [1] 0.1264822
## [1] 0.1324111
```

```
set.seed(1)
n <-nrow(newBoston)
boston.train <- scale(cbind(newBoston$tax,
                      newBoston$age,
                      newBoston$rm))
boston.y.train <- newBoston$medv01
```

```
set.seed(1)
for(i in c(1,5,10)){
  boston.knn.cv <- knn.cv(train = boston.train,cl = boston.y.train,k=i)
  print(mean(boston.knn.cv != boston.y.train))
}
```

```
## [1] 0.2134387
## [1] 0.1758893
## [1] 0.1660079
```

The test error rate were lower for all the predictor subsets and k-values
Question 5 (a):

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.1.3
```

```
mpg.median <- median(Auto$mpg)
newAuto <- data.frame(Auto)
newAuto$mpg01 <- ifelse(newAuto$mpg > mpg.median,1,0)
newAuto$mpg01 = as.factor(newAuto$mpg01)
newAuto$mpg = NULL
```

Question 5 (b):

```
set.seed(1)
auto.tune <- tune(method = svm,
                  mpg01~., data = newAuto,
                  kernel = "linear",
                  ranges = list(cost=c(0.001,0.01,0.1,1,5,10,100)))
summary(auto.tune)
```

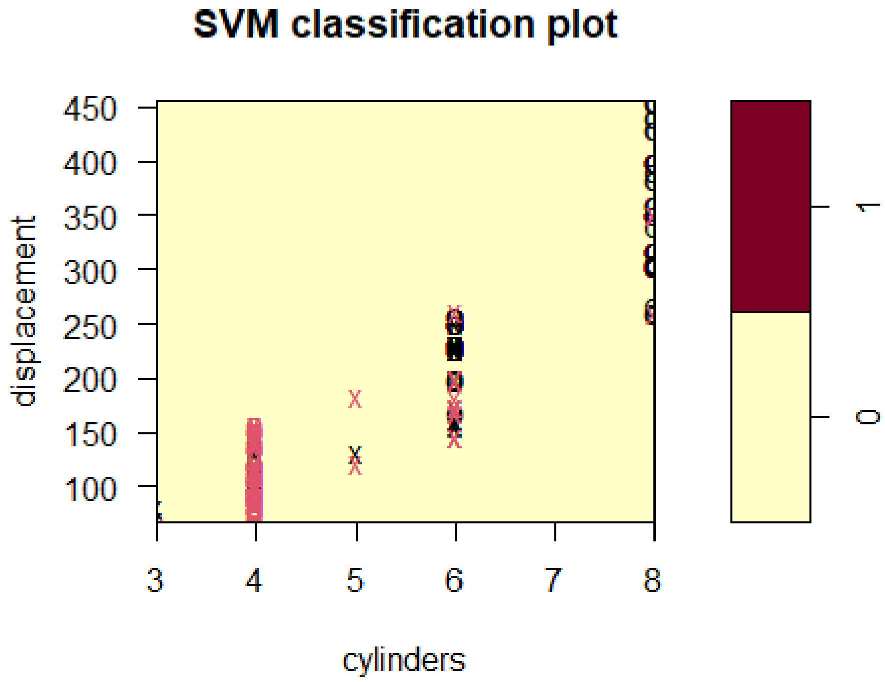
```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.08673077
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-03 0.13525641 0.05661708
## 2 1e-02 0.08923077 0.04698309
## 3 1e-01 0.08673077 0.04040897
## 4 1e+00 0.09961538 0.04923181
## 5 5e+00 0.11230769 0.05826857
## 6 1e+01 0.11237179 0.05701890
## 7 1e+02 0.11750000 0.06208951
```

Question 5 (c):

```
auto.svm <- svm(mpg01~., data = newAuto, kernel = "linear", cost = 0.1)
ypred <- predict(auto.svm,newAuto)
table(predict = ypred,truth = newAuto$mpg01)

##        truth
## predict   0   1
##       0 172   7
##       1  24 189

mean(ypred != newAuto$mpg01)

## [1] 0.07908163
```

Question 5 (d):

```
plot(auto.svm,data = newAuto, displacement~cylinders)
```



Question 6 (a):

```
library(ISLR)
set.seed(1)
n<- nrow(OJ)
oj.sample <- sample(1:n,800)
oj.train <- OJ[oj.sample,]
oj.test <- OJ[-oj.sample,]
```

Question 6 (b):

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

##
## Call:
## svm(formula = Purchase ~ ., data = oj.train, kernel = "linear", cost =
0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##
## Number of Support Vectors:  435
##
##  ( 219 216 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

There is 219 support vectors for class CH and 216 support vectors for class MM

Question 6 (c):

```
oj.train.pred <- predict(oj.svm,oj.train)
oj.test.pred <- predict(oj.svm,oj.test)
train.error <- mean(oj.train.pred != oj.train$Purchase)
test.error <- mean(oj.test.pred != oj.test$Purchase)
train.error

## [1] 0.175

test.error
```

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

Question 6 (d):

```
set.seed(1)
oj.tune <- tune(method = svm,
                Purchase~.,
                data = oj.train,
                kernel = "linear",
                ranges = list(cost=c(0.001,0.01,0.1,1,5,10,100)))
summary(oj.tune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.1725
##
## - Detailed performance results:
##    cost   error dispersion
## 1 1e-03 0.31250 0.04124790
## 2 1e-02 0.17625 0.02853482
## 3 1e-01 0.17250 0.03162278
## 4 1e+00 0.17500 0.02946278
## 5 5e+00 0.17250 0.03162278
## 6 1e+01 0.17375 0.03197764
## 7 1e+02 0.17500 0.03486083
```

Question 6 (e):

```
oj.svm <- svm(Purchase~.,
              data = oj.train,
              kernel = "linear",
              cost = 0.1)
oj.train.pred <- predict(oj.svm,oj.train)
oj.test.pred <- predict(oj.svm,oj.test)
train.error <- mean(oj.train.pred != oj.train$Purchase)
test.error <- mean(oj.test.pred != oj.test$Purchase)
train.error
```

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

```
test.error
```

```
## [1] 0.162963
```