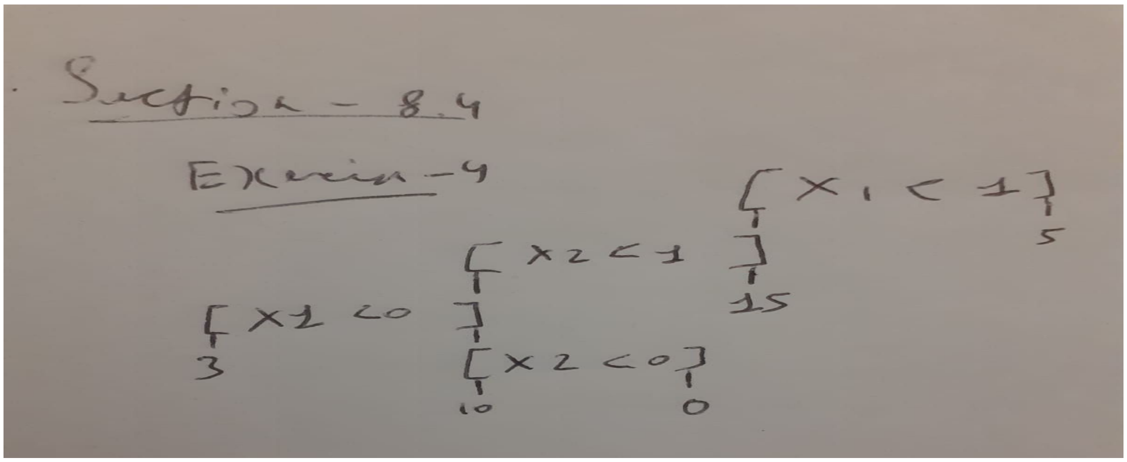
Homework-5

tg289@njit.edu

1. Exercises 4, 8, and 10 from Section 8.4 of our textbook.

Q4)

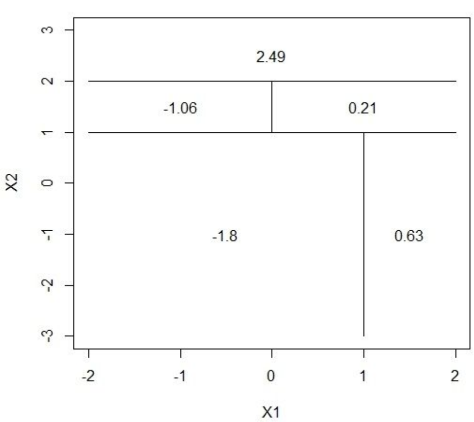
A)



B) Code-

```
> par(xpd=NA)
> plot(NA,NA,type='n',xlim=c(-2,2),ylim=c(-3,3),xlab='X1',ylab='X2')
> #X2 < 1
> lines(x=c(-2,2),y= c(1,1))
> #X1<1 with X2<1
> lines(x=c(1,1),y=c(-3,1))
> text(x=(-2+1)/2, y=-1, labels=c(-1.8))
> text(x=1.5, y=-1, labels=c(0.63))
> #X2<2 with X2>=1
> lines(x=c(-2,2),y=c(2,2))
> text(x=0,y=2.5, labels= c(2.49))
> #X1<0 with X2<2 and X2>=1
> lines(x= c(0,0),y=c(1,2))
> text(x=-1,y=1.5, labels=c(-1.06))
> text(x=1,y=1.5,labels=c(0.21))
```

Output-



Q8)

A) code-

```
library(ISLR)
set.seed(1)
train<- sample(1:nrow(Carseats),nrow(Carseats)/2)
carseats.train<- Carseats[train, ]
carseats.test<- Carseats[-train,
```

B) Regression Tree->

code-

```
library(tree)
tree.carseats<- tree(Sales~ ., data= carseats.train)
summary(tree.carseats)
```

Output-

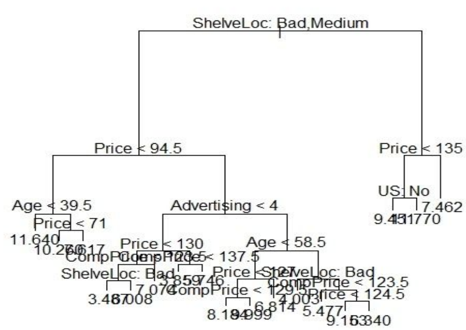
```
Regression tree:
tree(formula = Sales ~ ., data = carseats.train)
Variables actually used in tree construction:
[1] "ShelveLoc"   "Price"
[3] "Age"         "Advertising"
[5] "CompPrice"   "US"
Number of terminal nodes:  18
Residual mean deviance:  2.167 = 394.3 / 182
Distribution of residuals:
   Min.  1st Qu.   Median     Mean
-3.88200 -0.88200 -0.08712  0.00000
 3rd Qu.     Max.
 0.89590  4.09900
```

Plotting the tree

Code-

```
plot(tree.carseats)
text(tree.carseats, pretty= 0)
```

output-



Error rate-

Code-

```
> errorrate<- predict(tree.carseats, newdata= carseats.test)
> mean((errorrate- carseats.test$Sales)^2)
```

output-

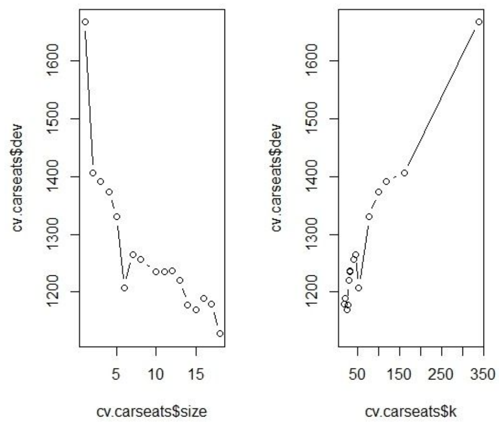
```
[1] 4.922039
```

The test MSE is about **4.93**

C) **code-**

```
cv.carseats= cv.tree(tree.carseats, FUN= prune.tree)
par(mfrow= c(1,2))
plot(cv.carseats$size, cv.carseats$dev, type='b')
plot(cv.carseats$k, cv.carseats$dev, type='b')
```

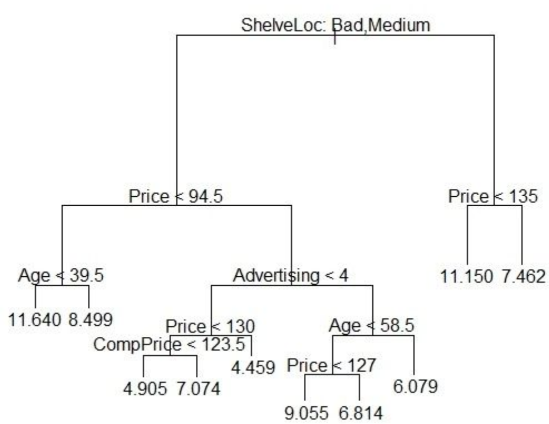
Output-



Code- (best size for tree-9)

```
pruned.carseats<- prune.tree(tree.carseats, best=9)
par(mfrow= c(1,1))
plot(pruned.carseats)
text(pruned.carseats, pretty=0)
```

Output-



code-(prune tree)

```
pred.prune<- predict(pruned.carseats,carseats.test)
mean((carseats.test$Sales- pred.prune)^2)
```

Output-

[1] 5.113254

When we prune the tree the **test MSE rises to about 5.12**

D) **Code-**

```
library(randomForest)
bagging.carseats<- randomForest(Sales~ ., data = carseats.train, mtry= 10, ntree = 500,      importance= TRUE)
bagging.pred<- predict(bagging.carseats,carseats.test)
mean((carseats.test$Sales- bagging.pred)^2)
```

Output-

```
[1] 2.622982
```

We can see that when we use bagging the test **MSE decreases to 2.62**

Code-(for importance)-

```
importance(bagging.carseats)
```

Output-

```
                %IncMSE      IncNodePurity
CompPrice   25.8083065    176.750788
Income       4.3976727     90.755860
Advertising 12.2530540     96.396343
Population  -3.1644545     55.382276
Price       56.2028471    501.877790
ShelveLoc   46.4048838    377.701755
Age         16.0014582    155.431845
Education   -0.2585088     44.103966
Urban       -0.3479058      8.679922
US           5.5939322     18.643493
```

We can conclude from this - **Price, ShelveLoc and CompPrice are three most important variables**

E) Code-

```
random_f<- randomForest(Sales ~., data= carseats.train, mtry=3,ntree=500,importance=TRUE)
random_pred<-predict(random_f,carseats.test)
mean((carseats.test$Sales- random_pred)^2)
```

Output-

```
[1] 3.070883
```

Code-

```
importance(random_f)
```

Output-

```
                %IncMSE      IncNodePurity
CompPrice   15.3620526    156.79771
Income       3.1692502    120.90880
Advertising  7.8046269    108.57608
Population  -2.5512335    100.56722
Price       36.5947531    396.36763
ShelveLoc   35.9522295    290.53078
Age         11.4111604    174.34911
Education    0.4647345     72.66570
Urban        1.4222554     15.61579
US           6.3063458     34.71705
```

We again see that price shevlock and compprice are the most important variables

Q10)

A) Code-

```
library(ISLR)
Hitters<- na.omit(Hitters)
Hitters$Salary<- log(Hitters$Salary)
```

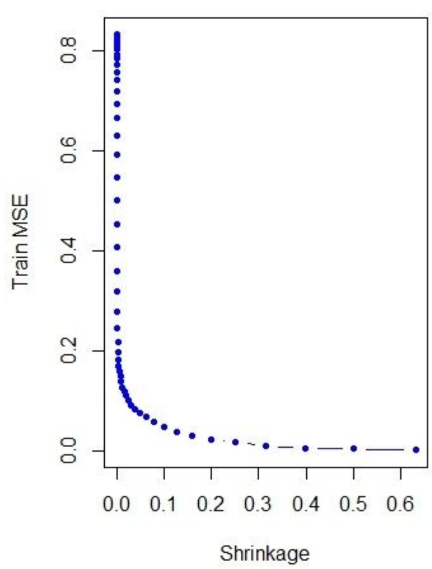
B) Code-

```
train<- 1:200
Hitters.train= Hitters[train,]
Hitters.test= Hitters[-train,]
```

C) Code-

```
install.packages('gbm')
library(gbm)
set.seed(103)
pows=seq(-10, -0.2, by=0.1)
lambdas= 10^pows
length.lambdas= length(lambdas)
train.errors= rep(NA, length.lambdas)
test.errors= rep(NA,length.lambdas)
for(i in 1:length.lambdas){
 boost.hitters= gbm(Salary~ ., data= Hitters.train, distribution = 'gaussian',
 n.trees= 1000, shrinkage= lambdas[i])
 train.pred= predict(boost.hitters, Hitters.train, n.trees= 1000)
 test.pred= predict(boost.hitters,Hitters.test, n.trees= 1000)
 train.errors[i]= mean((Hitters.train$Salary- train.pred)^2)
 test.errors[i]= mean((Hitters.test$Salary- test.pred)^2)
}
plot(lambdas, train.errors, type='b',xlab='Shrinkage',ylab='Train MSE',col='blue',pch=20)
```

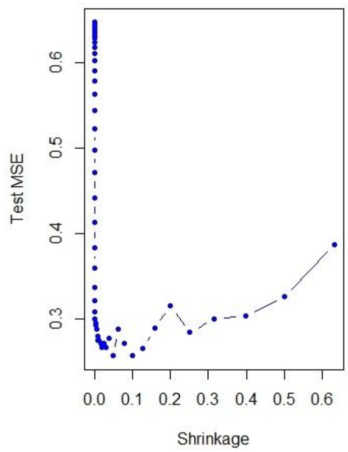
Output-



D) Code-

```
plot(lambdas,test.errors, type='b',xlab='Shrinkage',ylab='Test MSE',col='blue',pch=20 )
```

Output-



Code- (For MSE)

```
min(test.errors)
```

Output-

[1] 0.008058291

Code-(For lambda)

```
lambdas[which.min(test.errors)]
```

Output-

[1] 0.05011872

The minimun test error is 0.008 and corresponding lambda value is 0.05

E) Code-

```
library(glmnet)
lm.fit= lm(Salary~ ., data= Hitters.train)
lm.pred= predict(lm.fit, Hitters.test)
mean((Hitters.test$Salary- lm.pred)^2)
```

Output-

[1] 0.01496996

Code-

```
set.seed(1)
x= model.matrix(Salary ~., data= Hitters.train)
y= Hitters.train$Salary
x.test= model.matrix(Salary~ ., data= Hitters.test)
lassso.fit= glmnet(x,y,alpha=1)
lasso.pred= predict(lassso.fit, s=0.01, newx= x.test)
mean((Hitters.test$Salary- lasso.pred)^2)
```

Output-

[1] 0.01377199

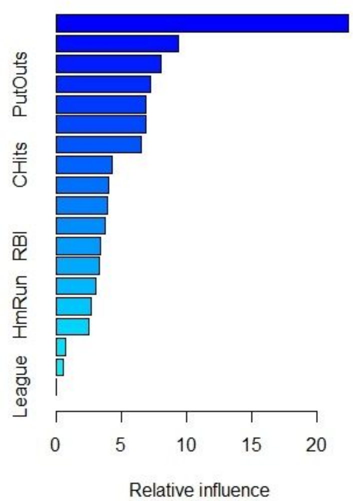
Both **linear model and regularization like Lasso have higher test MSE than boosting**

F) Code-

```
library(gbm)
boosting.best<- gbm(Salary~ ., data= Hitters.train, distribution = 'gaussian',
n.trees= 1000, shrinkage= lambdas[which.min(test.errors)])
summary(boosting.best)
```

Output-

```
                    var    rel.inf
CAtBat       CAtBat   20.1133386
CRuns         CRuns   12.4546958
CRBI           CRBI     10.9286391
PutOuts     PutOuts  7.0217036
CWalks       CWalks  6.8343354
Years          Years      6.3477290
Walks          Walks     5.7228669
CHmRun   CHmRun  5.4150877
RBI              RBI       4.6494594
Hits             Hits      4.4069403
Assists      Assists  3.4818646
AtBat          AtBat   2.9389697
HmRun     HmRun  2.6833097
CHits         CHits  2.6574682
Errors       Errors  1.6588080
Runs           Runs  1.2466717
NewLeague NewLeague  0.6987026
Division   Division  0.5333779
League       League  0.2060317
```



G) Code-

```
library(randomForest)
set.seed(25)
rf.hitters<- randomForest(Salary~ ., data= Hitters.train, ntree=500, mtry=19)
rf.pred<- predict(rf.hitters, Hitters.test)
mean((Hitters.test$Salary- rf.pred)^2)
```

Output-

[1] 0.007193393

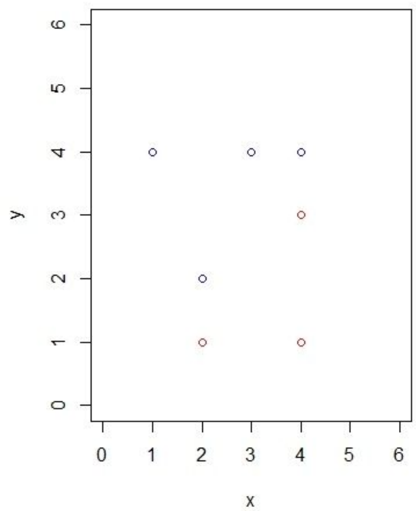
2. Exercises 3 and 7 from Section 9.7 of our textbook.

Q3)

A) Code-

```
x1= c(3,2,4,1,2,4,4)
x2= c(4,2,4,4,1,3,1)
colors=(c('blue','blue','blue','blue','red','red','red'))
plot(x1,x2,col=colors,xlim=c(0,6),ylim=c(0,6))
```

Output-



B) Equation-

Section - 9.7

Exercise - 3 (b)

The maximal margin classifier has to be in b/w obs. #2, #3 and ~~#~~ 5 ~~#~~ 6

$(2,2), (4,4)$

$(2,1), (4,3)$

$\Rightarrow \left(\frac{2+2}{2}, \frac{2+1}{2}\right), \left(\frac{4+4}{2}, \frac{4+3}{2}\right)$

$= (2, 1.5), (4, 3.5)$

$b = (3.5 - 1.5)/(4-2) = \boxed{1}$

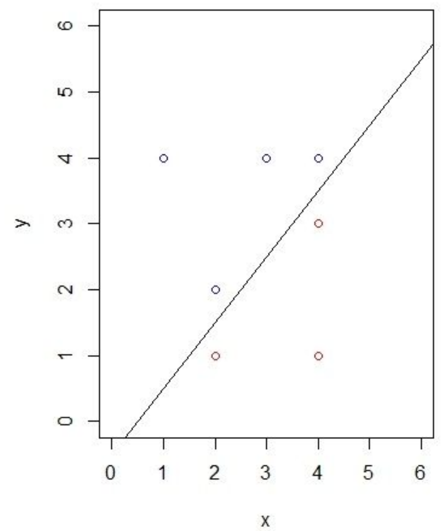
$a = X_2 - X_1 = 1.5 - 2 = \boxed{-0.5}$

Sketch-

Code-

```
plot(x1,x2,col=colors,xlim=c(0,),ylim=c(0,5))
abline(-0.5,1)
```

Output-

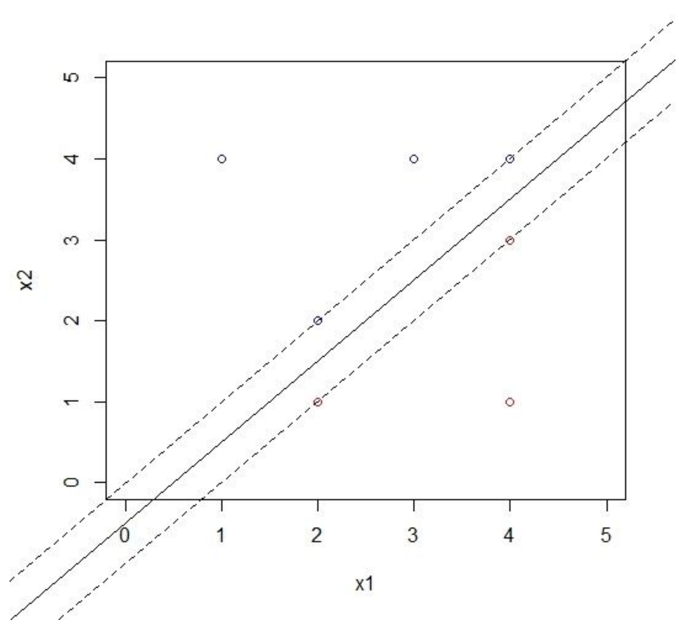


C) The classification rules are - **0.5- X1+X2>0**

D) **Code-**

```
plot(x1,x2,col=colors,xlim=c(0,5),ylim=c(0,5))
abline(-0.5,1)
abline(-1,1,lty=2)
abline(0,1,lty=2)
```

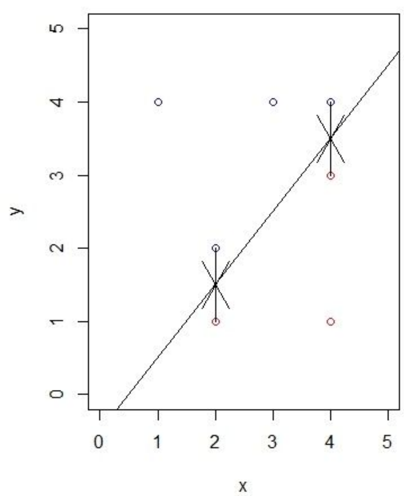
Output-



E) **Code-**

```
plot(x1,x2,col=colors,xlim=c(0,5),ylim=c(0,5))
abline(-0.5,1)
arrows(2,1,2,1.5)
arrows(2,2,2,1.5)
arrows(4,4,4,3.5)
arrows(4,3,4,3.5)
```

Output-

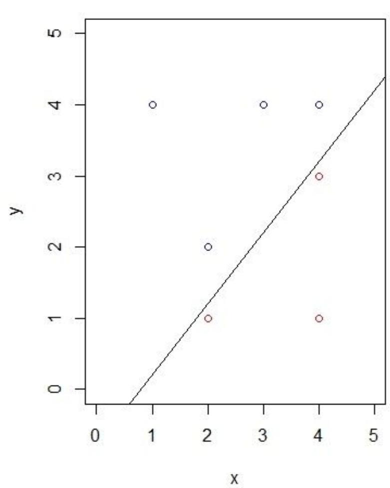


F) A slight movement of observation **#7 (4,1) red would not have an effect on the maximal margin hyperplane since its movement would be outside of the margin**.

G) Code-

```
plot(x1,x2,col=colors,xlim=c(0,5),ylim=c(0,5))
abline(-.8,1)
```

Output-

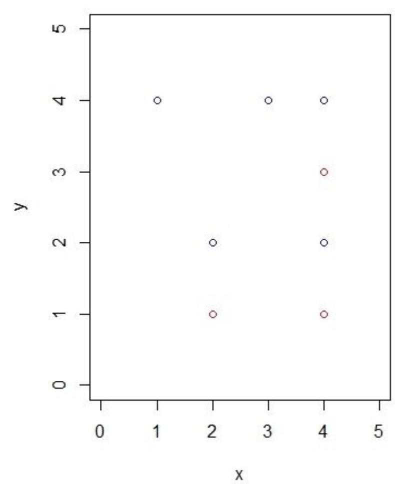


Equation of the hyperplane- (-0.8-X1+X2>0)

H) Code-

```
plot(x1,x2,col=colors,xlim=c(0,5),ylim=c(0,5))
points(c(4),c(2),col=c("blue"))
```

Output-



Q7)

A) Code-

```
library(ISLR)
gas.median= (median(Auto$mpg))
factor_med= ifelse(Auto$mpg> gas.median, 1, 0)
Auto$mpglevel= as.factor(factor_med)
```

B) Code-

```
library(e1071)
set.seed(3250)
tune.in<- tune(svm,mpglevel~ ., data=Auto, kernel='linear', ranges= list(cost= c(0.01,0.1,1,5,10,100)))
summary(tune.in)
```

Output-

```
Parameter tuning of 'svm':
- sampling method: 10-fold cross validation
- best parameters:
 cost
    1
- best performance: 0.01025641
- Detailed performance results:
   cost      error dispersion
1 1e-02 0.07410256 0.03724108
2 1e-01 0.04596154 0.02359743
3 1e+00 0.01025641 0.01792836
4 5e+00 0.01782051 0.02088734
5 1e+01 0.02038462 0.02001214
6 1e+02 0.03307692 0.02397013
```

We see that cross-validation error is minimized for **cost=1**.

C) Code-

```
 set.seed(22)
tune.out= tune(svm, mpglevel~ ., data=Auto, kernel= "polynomial", ranges=
list(cost=c(0.1,1,5,10),degree=c(2,3,4)))
summary(tune.out)
```

Output-

```
Parameter tuning of 'svm':
- sampling method: 10-fold cross validation
- best parameters:
 cost degree
   10      2
- best performance: 0.5510897
- Detailed performance results:
   cost degree     error dispersion
1   0.1      2 0.5741026 0.02415229
2   1.0      2 0.5741026 0.02415229
3   5.0      2 0.5741026 0.02415229
4  10.0      2 0.5510897 0.04591189
5   0.1      3 0.5741026 0.02415229
6   1.0      3 0.5741026 0.02415229
7   5.0      3 0.5741026 0.02415229
8  10.0      3 0.5741026 0.02415229
9   0.1      4 0.5741026 0.02415229
10  1.0      4 0.5741026 0.02415229
11  5.0      4 0.5741026 0.02415229
12 10.0      4 0.5741026 0.02415229
```

#The lowest cross-validation error is obtained for cost=10 and degree=2.

Code-

```
set.seed(461)
tune.out<- tune(svm, mpglevel~ ., data=Auto, kernel='radial',ranges=list(cost= c(0.1,1,5,10), gamma=
c(0.01,.1,1,5,10,100)))
summary(tune.out)
```

Output-

```
Parameter tuning of 'svm':
- sampling method: 10-fold cross validation
- best parameters:
 cost gamma
   10   0.1
- best performance: 0.02551282
- Detailed performance results:
   cost gamma      error dispersion
1   0.1 1e-02 0.08666667 0.04967643
2   1.0 1e-02 0.07391026 0.04561258
3   5.0 1e-02 0.04839744 0.03496088
4  10.0 1e-02 0.03064103 0.02651089
5   0.1 1e-01 0.07903846 0.04876296
6   1.0 1e-01 0.05346154 0.04057827
7   5.0 1e-01 0.02807692 0.02551393
8  10.0 1e-01 0.02551282 0.02702937
9   0.1 1e+00 0.57416667 0.05205090
10  1.0 1e+00 0.06352564 0.04307392
11  5.0 1e+00 0.06358974 0.03814303
12 10.0 1e+00 0.06608974 0.04146890
13  0.1 5e+00 0.57416667 0.05205090
14  1.0 5e+00 0.52814103 0.05853644
15  5.0 5e+00 0.51538462 0.05940094
16 10.0 5e+00 0.51538462 0.05940094
17  0.1 1e+01 0.57416667 0.05205090
```

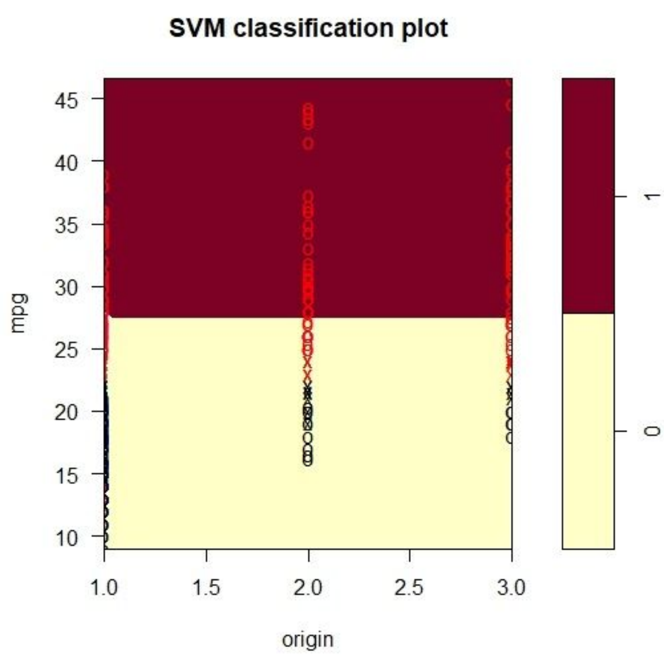
```
18  1.0 1e+01 0.53346154 0.06755706
19  5.0 1e+01 0.53089744 0.06238863
20 10.0 1e+01 0.53089744 0.06238863
21  0.1 1e+02 0.57416667 0.05205090
22  1.0 1e+02 0.57416667 0.05205090
23  5.0 1e+02 0.57416667 0.05205090
24 10.0 1e+02 0.57416667 0.05205090
```

#Finally, for radial basis kernel, cost=10 and gamma=0.01.

D) Code-(for linear SVM)

```
svm.linear<- svm(mpglevel~ ., data=Auto,kernel="linear",cost=1)
svm.poly= svm(mpglevel~ ., data= Auto, kernel='polynomial',cost=10,degree=2)
svm.radical= svm(mpglevel~ ., data=Auto, kernel='radial',cost=10,gamma=0.01)
plotpairs<- function(fit){
  for(name in names(Auto)[!(names(Auto) %in% c('mpg','mpglevel','name'))]) {
    plot(fit,Auto,as.formula(paste('mpg~',name,sep="")))
  }
}
plotpairs(svm.linear)
```

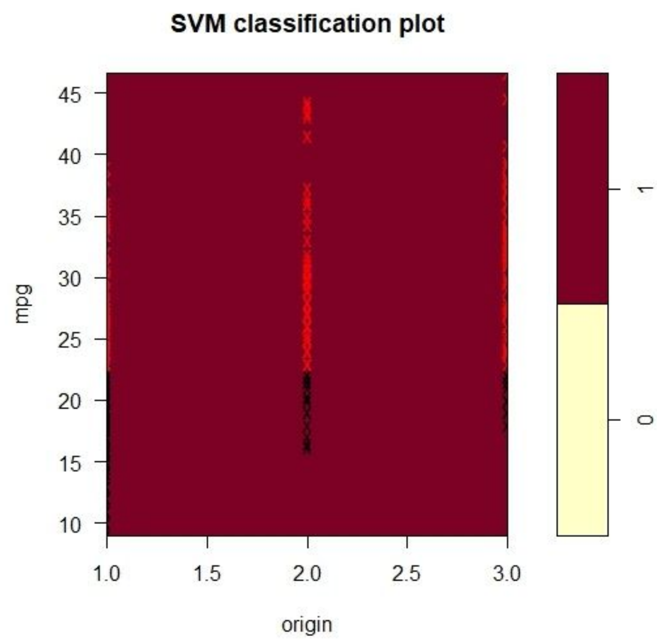
Output-



Code- (for polynimial basis)

```
svm.linear<- svm(mpglevel~ ., data=Auto,kernel="linear",cost=1)
svm.poly= svm(mpglevel~ ., data= Auto, kernel='polynomial',cost=10,degree=2)
svm.radical= svm(mpglevel~ ., data=Auto, kernel='radial',cost=10,gamma=0.01)
plotpairs<- function(fit){
  for(name in names(Auto)[!(names(Auto) %in% c('mpg','mpglevel','name'))]) {
    plot(fit,Auto,as.formula(paste('mpg~',name,sep="")))
  }
}
plotpairs(svm.poly)
```

Output-



Code-(for radial basis)

```
svm.linear<- svm(mpglevel~ ., data=Auto,kernel="linear",cost=1)
svm.poly= svm(mpglevel~ ., data= Auto, kernel='polynomial',cost=10,degree=2)
svm.radical= svm(mpglevel~ ., data=Auto, kernel='radial',cost=10,gamma=0.01)
plotpairs<- function(fit){
  for(name in names(Auto)[!(names(Auto) %in% c('mpg','mpglevel','name'))]) {
    plot(fit,Auto,as.formula(paste('mpg~',name,sep="")))
  }
}
plotpairs(svm.radial)
```

Output-