R Capability – ISLR Practice

Aishwarya Pawar

8/9/2019

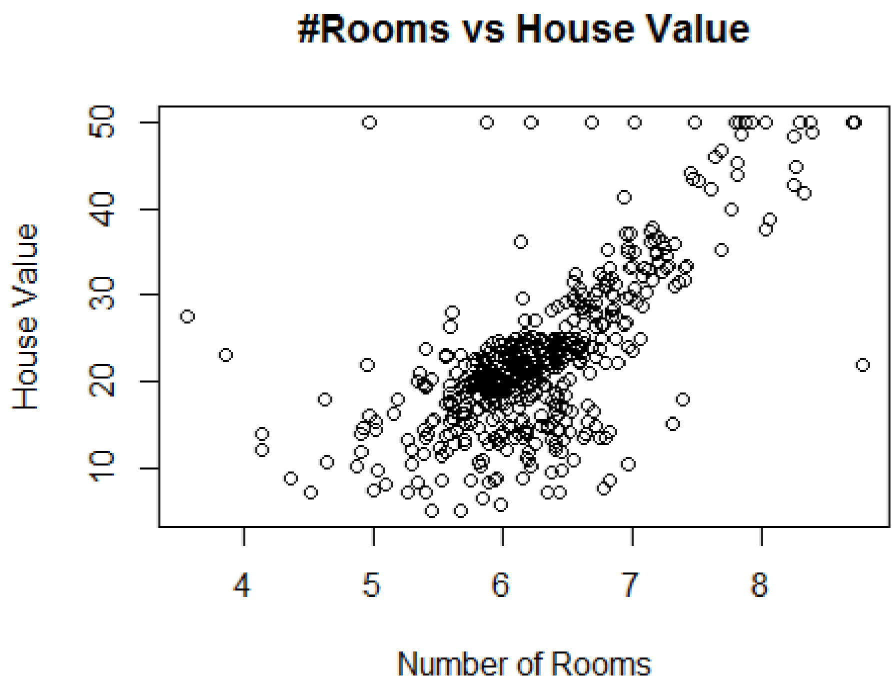
Chapter 2 : Question 10 :

Question a:

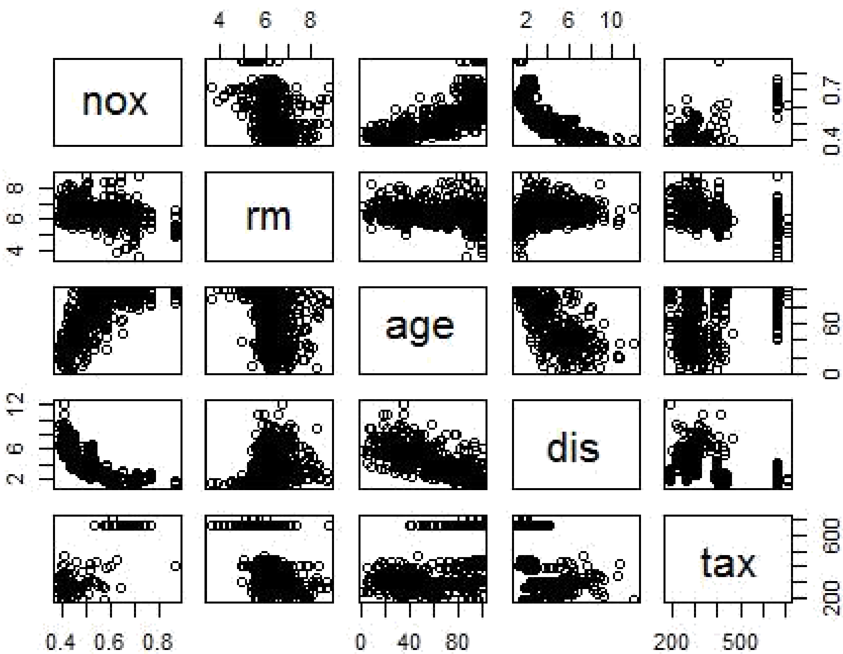
```
library(MASS)
#Number of Rows
no_rows= nrow(Boston)
#no_rows
print(paste("Number of Rows of Boston Dataset:",no_rows))

## [1] "Number of Rows of Boston Dataset: 506"

no_cols=ncol(Boston)
print(paste("Number of Columns of Boston Dataset:",no_cols))

## [1] "Number of Columns of Boston Dataset: 14"

print("The columns specify the suburbs of boston and the columns represent di
fferent attributes of the locality Ex.- Crime rate , nitrogen concentration e
tc. ")

## [1] "The columns specify the suburbs of boston and the columns represent
d ifferent attributes of the locality Ex.- Crime rate , nitrogen
concentration etc. "
```

Question b:

```
#Relationship between Number of Rooms per dwelling and median value of owner-
occupied home

x1=Boston$rm
y1=Boston$medv

plt =plot(x1, y1, main = "#Rooms vs House Value", xlab = "Number of Rooms",
ylab = "House Value")
require(lattice)

## Loading required package: lattice
```




```
require(ggplot2)

## Loading required package: ggplot2

bstn_plt=Boston[c(1,5:8,10:14)]

pairs(bstn_plt[2:6], pch = 21)
```

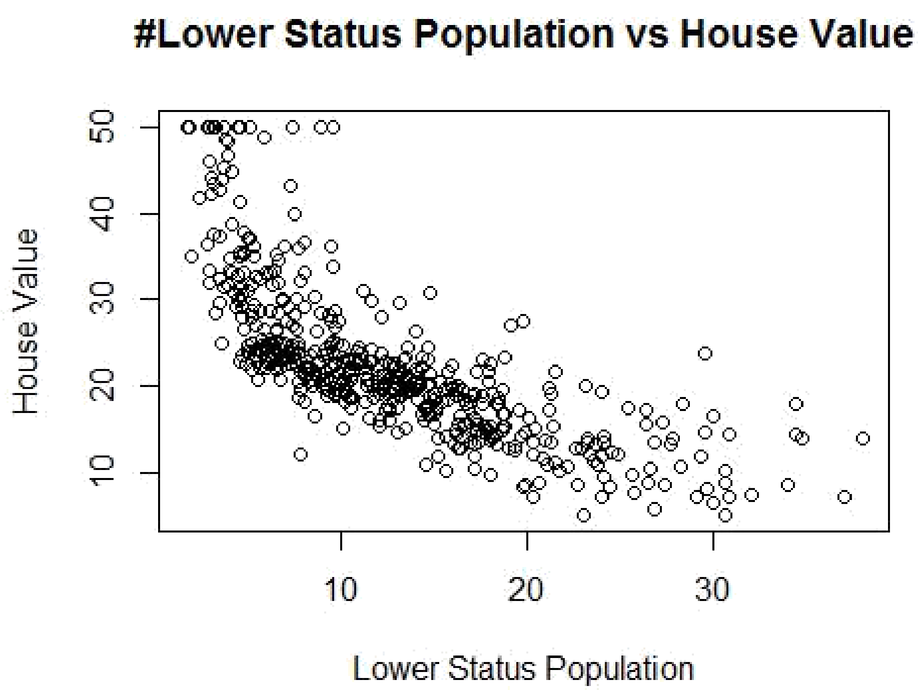


Finding: -

As the number of rooms increase in the dwelling, the median value #of owner-occupied homes increases i.e. positive correlation - There seems to be a positive correlation between nitrogen oxide and age of the units - The scatter plot shows that as the distance from Boston employment center increases, there seems to be a slight decrease in nitrogen oxide concentration

Question b:

```
#Relationship between Number of Rooms per dwelling and median value of owner-occupied home

x1=Boston$lstat
y1=Boston$medv

plt =plot(x1, y1, main = "#Lower Status Population vs House Value", xlab = " Lower Status Population", ylab = "House Value")
```



Finding: -

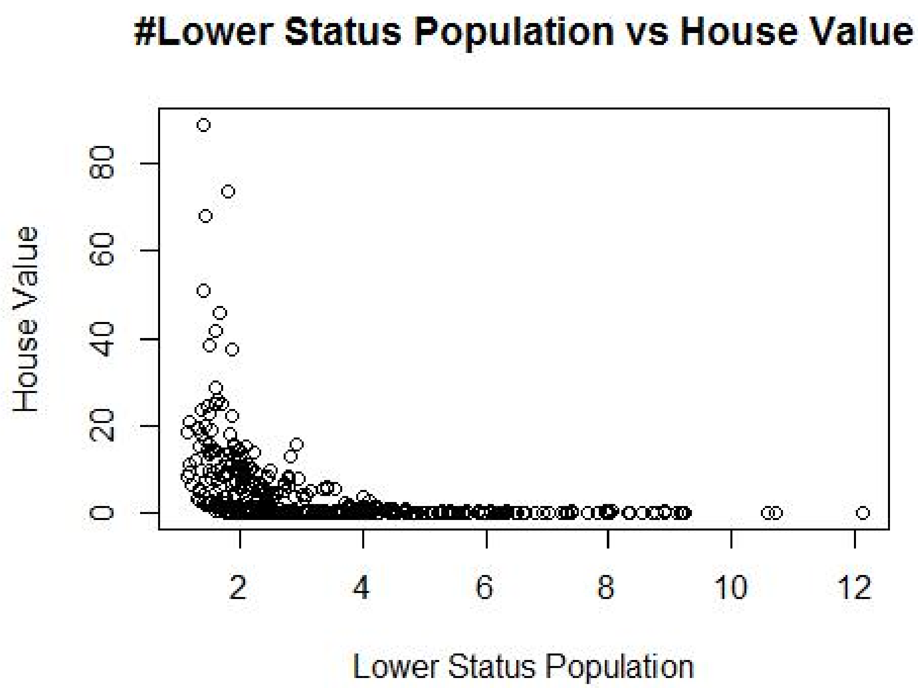
As the lower status population increases in the locality, the median value of owner-occupied homes decreases i.e. negative correlation

Question c:

```
#Are any of the predictors associated with per capita crime rate? If so,
expl ain the relationship.

correlation= cor(Boston[-1],Boston$crim)

print("Correlation of Crime Rate with other factors:")

## [1] "Correlation of Crime Rate with other factors:"

print(correlation)

##                 [,1]
## zn       -0.20046922
## indus     0.40658341
## chas     -0.05589158
## nox       0.42097171
## rm       -0.21924670
## age       0.35273425
## dis      -0.37967009
## rad       0.62550515
## tax       0.58276431
```

```
## ptratio  0.28994558
## black   -0.38506394
## lstat0.45562148
## medv-0.38830461
```

```r
x2=Boston$dis
y2=Boston$crim
plt =plot(x2, y2, main = "#Lower Status Population vs House Value", xlab = "
Lower Status Population", ylab = "House Value")
```



Finding:

- Index of accessibility to radial highways and property tax seems to have positive correlation of 0.62 and 0.58 with crime rate i.e. as the tax and distance from highway increase the crime rate decreases.
- Distance from 5 boston employment centres seems to have negative correlation of 0.37 and the scatter plot also shows the decrease

Question d:

```r
#Do any of the suburbs of Boston appear to have particularly high crime rates
? Tax rates? Pupil-teacher ratios? Comment on the range of each predictor.


max_crim=Boston[Boston$crim==max(Boston$crim),]
print("381st suburb has the highest crime rate")
```

```
## [1] "381st suburb has the highest crime rate"
```

```r
## range calculation
range_crim=range(Boston$crim)
range_crim_f=range_crim[2]-range_crim[1]
print(paste0("Range of crime rate is : ",range_crim_f))
```

```
## [1] "Range of crime rate is : 88.96988"
```

```r
max_tax=Boston[Boston$tax==max(Boston$tax),]
print("489-493 suburbs have the highest crime rate")
```

```
## [1] "489-493 suburbs have the highest crime rate"
```

```r
## range calculation
range_tax=range(Boston$tax)
range_tax_f=range_tax[2]-range_tax[1]
print(paste0("Range of tax rate is : ",range_tax_f))
```

```
## [1] "Range of tax rate is : 524"
```

```r
max_ptratio=Boston[Boston$ptratio==max(Boston$ptratio),]
print("355& 356 suburbs have the highest crime rate of 22")
```

```
## [1] "355& 356 suburbs have the highest crime rate of 22"
```

```r
## range calculation
range_ptratio=range(Boston$ptratio)
range_ptratio_f=range_ptratio[2]-range_ptratio[1]
print(paste0("Range of pupil-teacher ratio is : ",range_ptratio_f))
```

```
## [1] "Range of pupil-teacher ratio is : 9.4"
```

Question e:

```r
#How many of the suburbs in this data set bound the Charles
  river? #sum of chas flags would give us number of suburbs
tot_chas=sum(Boston$chas)
print(paste0("Number of suburbs bound the Charles river :",tot_chas))
```

```
## [1] "Number of suburbs bound the Charles river :35"
```

Question f:

```r
#What is the median pupil-teacher ratio among the towns in this data set?

med_ptratio=median(Boston$ptratio)
print(paste("The median pupil-teacher ration of Boston is :",med_ptratio))
```

```
## [1] "The median pupil-teacher ration of Boston is : 19.05"
```

Question g:

```r
#Which suburb of Boston has lowest median value of owneroccupied homes? What are
the values of the other predictors for that suburb, and how do those valu es
compare to the overall ranges for those predictors? Comment on your findin
```

```
gs.

min_medv=Boston[(Boston$medv==min(Boston$medv)),]

print(min_medv)

##         crim zn indus chas   nox    rm age    dis rad tax ptratio  black
## 399  38.3518  0  18.1    0 0.693 5.453 100 1.4896  24 666    20.2 396.90
## 406  67.9208  0  18.1    0 0.693 5.683 100 1.4254  24 666    20.2 384.97
##     lstat medv
 ## 399 30.59    5
 ## 406 22.98    5
```

Findings:

- suburb 399 and 406 are the suburbs with lowest median value of owner-occupied homes of $5000
- These suburbs have very high crime rate; high proportion of non-retail business acres per town; above average nitrogen oxide concentration; Below average number of rooms per dwelling; high proportion of blacks; higher % of lower status population. Also, these suburbs seems to have very old units build prior to 1940

Question h:

```
# In this data set, how many of the suburbs average more than seven rooms
per dwelling? More than eight rooms per dwelling? Comment on the suburbs
that ave rage more than eight rooms per dwelling.

mr_7=nrow(Boston[(Boston$rm>7),])
print(paste0("Number of suburbs with more than 7 rooms per dwelling on an
ave rage : ",mr_7))

## [1] "Number of suburbs with more than 7 rooms per dwelling on an average :
64"

mr_8=nrow(Boston[(Boston$rm>8),])
mr_8_set=Boston[(Boston$rm>8),]
print(paste0("Number of suburbs with more than 8 rooms per dwelling on an
ave rage : ",mr_8))

## [1] "Number of suburbs with more than 8 room per dwelling on an average :
13"
```

Findings:

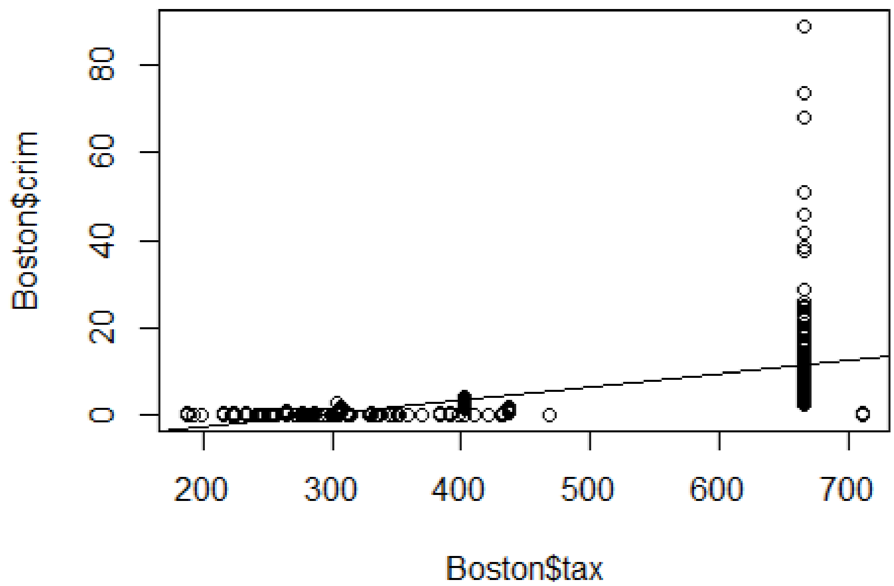
The suburbs with more than 8 rooms per dwelling have below average lower status population % ; above average median value of owner-occupied homes ; above average proportion of owner-occupied units built prior 1940 and mostly below average crime rate

==

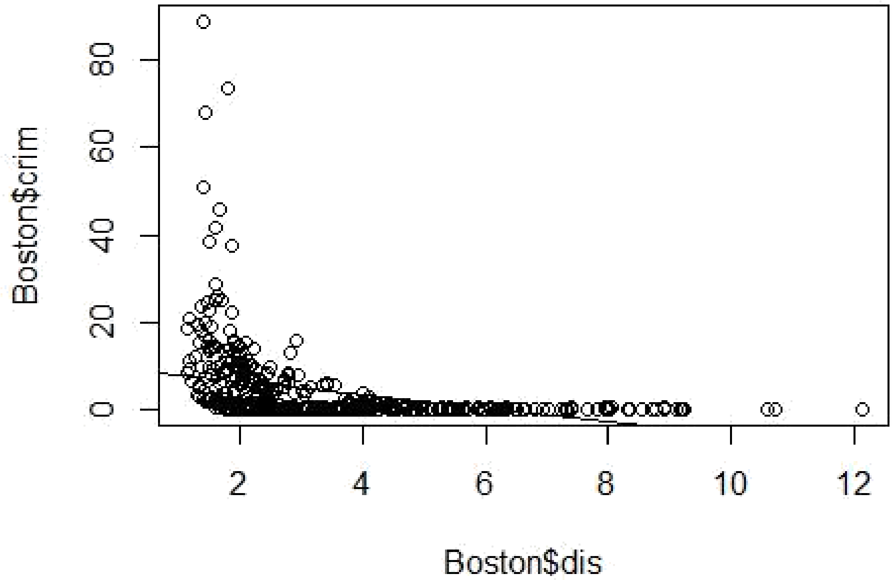
Chapter 3 : Question 15 :

a)For each predictor, fit a simple linear regression model to predict the response. Describe your results. In which of the models is there a statistically significant association between the predictor and the response? Create some plots to back up your assertions.

```
#Load the data set
library(MASS)
# Y = crim

# Function to regress each variable with crim

var_names=as.array(colnames(Boston))
var_names=var_names[-1]

reg_fin_tab=data.frame(coeff=double(),
                     std_err=double(),
                     t_stat=double(),
                     p_val=double(),
                     r_sqr=double(),
                     adj_r_sqr=double())

linear_reg_func=function(x)
{
  fm = as.formula(paste("crim~",x))
  bost_reg=lm(fm,data=Boston)
  return(summary(bost_reg))

}


##bost_reg=lm(crim~tax,data=Boston)

for (i in 1:length(var_names))
{
  m=linear_reg_func(var_names[i])
  reg_fin_tab=rbind(reg_fin_tab,c(m$coefficients[2,],m$r.squared,m$adj.r.squa
red))
}
reg_fin_tab=cbind(reg_fin_tab,var_names)
colnames(reg_fin_tab)=c('coeff','std_err','t_stat','p_val','r-sqrd','adj_r_sq
r','variable')

# Relationship between crime rate and tax
##Plots reg=lm(crim~tax,data=Boston)
plot(Boston $tax,Boston$crim) abline(reg)
```




```
# Relationship between crime rate and distance to employment
center reg1=lm(crim~dis,data=Boston)
plot(Boston$dis,Boston$crim)
abline(reg1)
```



Findings:

1. All the variables (zn, indus,chas, nox, rm,age,dis,rad,tax,pratio,black,lstat,mdev) are significant i.e. with low p-values and standard errors. T-stats are also high except for charles river dummy variable

2. Accessibility to radial highways and full-value property-tax rate showcases highest explainability of the variation i.e. r-square values (39% and 34% resp)

```
# Fit a multiple regression model to predict the response using all of the
pr edictors. Describe your results. For which predictors can we reject the
null hypothesis H0 : βj = 0?

library(MASS)
model=lm(crim~zn+indus+chas+nox+rm+age+dis+rad+tax+ptratio+black+lstat+medv,
data=Boston)

mod_sum=summary(model)
mod_sum

##
## Call:
## lm(formula = crim ~ zn + indus + chas + nox + rm + age + dis +
##     rad + tax + ptratio + black + lstat + medv, data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
```

```
## -9.924 -2.120 -0.353  1.019 75.051
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  17.033228   7.234903   2.354 0.018949 *
## zn            0.044855   0.018734   2.394 0.017025 *
## indus        -0.063855   0.083407  -0.766 0.444294
## chas         -0.749134   1.180147  -0.635 0.525867
## nox         -10.313535   5.275536  -1.955 0.051152 .
## rm            0.430131   0.612830   0.702 0.483089
## age           0.001452   0.017925   0.081 0.935488
## dis          -0.987176   0.281817  -3.503 0.000502 ***
## rad           0.588209   0.088049   6.680 6.46e-11 ***
## tax          -0.003780   0.005156  -0.733 0.463793
## ptratio      -0.271081   0.186450  -1.454 0.146611
## black        -0.007538   0.003673  -2.052 0.040702 *
## lstat         0.126211   0.075725   1.667 0.096208 .
## medv         -0.198887   0.060516  -3.287 0.001087 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.439 on 492 degrees of freedom
## Multiple R-squared:  0.454,  Adjusted R-squared:  0.4396
## F-statistic: 31.47 on 13 and 492 DF,  p-value: < 2.2e-16
```

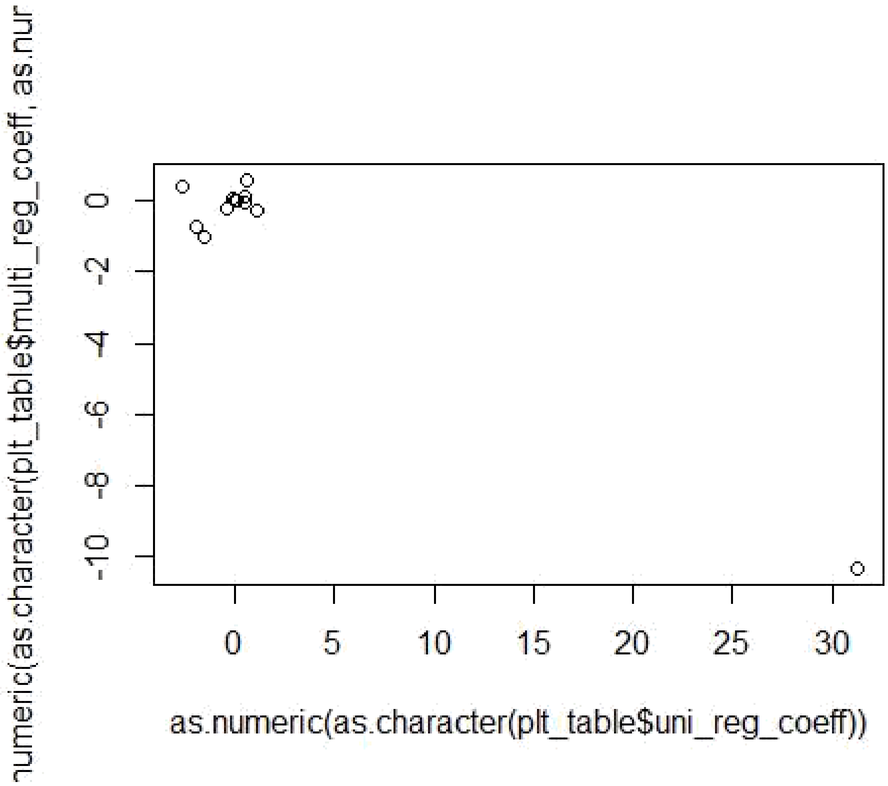
Findings:

1. The multiple regression model has standard error of 6.439 and can explain ~44% of the crime rate variation.

2. Weighted distance to employment centres, accessibility to radial highways and median value of homes have significant effect on the crime rate of the suburbs.

3. With the multiple regression model, the variables which were significant separately, like tax rate, pupil-teacher ratio etc, are now not very significant.

```
##### Question c
mult_reg_coeff=mod_sum$coefficients[-1,]
plt_table=data.frame(cbind(reg_fin_tab[,7],reg_fin_tab[,1],mult_reg_coeff[,1]
))

colnames(plt_table)=c('Predictor','uni_reg_coeff' ,'multi_reg_coeff')

#Coefficient Plot
plot(as.numeric(as.character(plt_table$uni_reg_coeff)),as.numeric(as.characte
r(plt_table$multi_reg_coeff,as.numeric)))
```



Finding:

- Most of the coefficients are in the range of -1 to 1 for both univariant and multi-variant regression, except for nitrogen oxide concentration, which has high positive univariant regression coefficient but negative multi-variant coefficient. This indicates that in the presence of other variables, the impact of nitrogen oxide concentration changes completely while explaining the variation in crime rate

```
##### Question d
#Is there evidence of non-linear association between any of the predictors
an d the response?

non_linear_reg_func=function(x)
{
  fmn = as.formula(paste("crim~",x,"+ I(",x,"^2)+I(",x,"^3)"))
  bost_reg_non=lm(fmn,data=Boston)
  return(summary(bost_reg_non))
}

for (i in 1:length(var_names))
{
  m=non_linear_reg_func(var_names[i])
  #print(m)
}
```

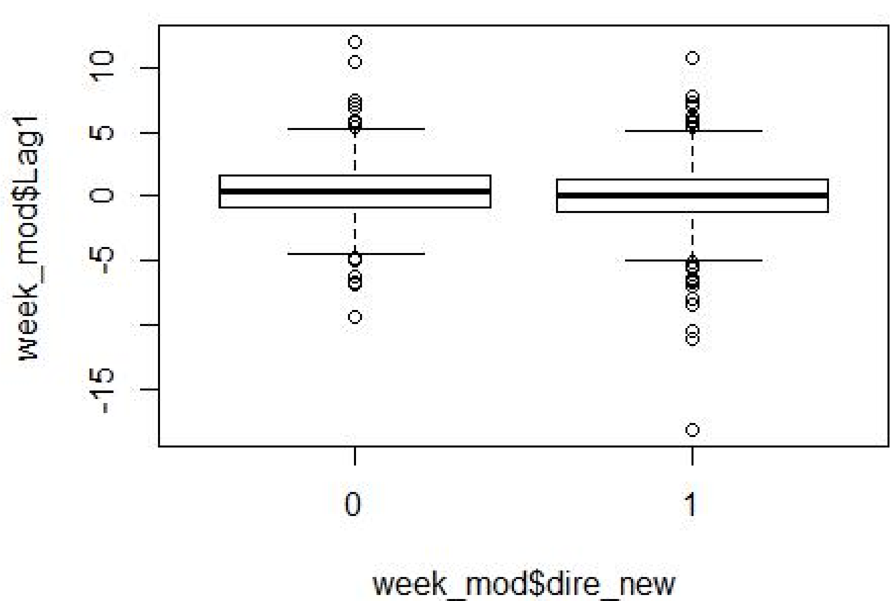
Findings: -

1. Median home values showcase the most significant non-linear relationship with crime rate with large t-statistics
2. Proportion of non-retail business area, median value of the house (medv), nitrogen oxide concentration, distance from employment centre and home value seems to have significant non-linear relationship with crime rate.

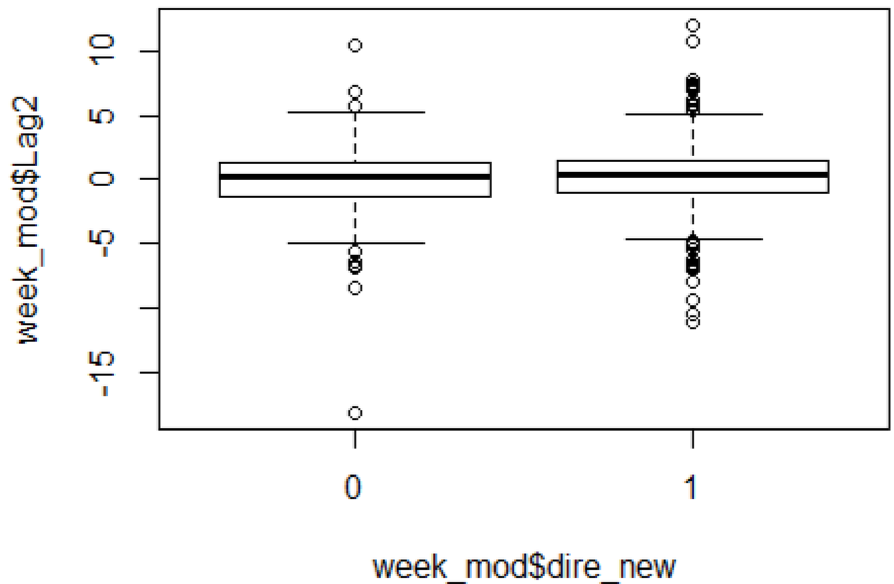
==

Chapter 4 : Question 10

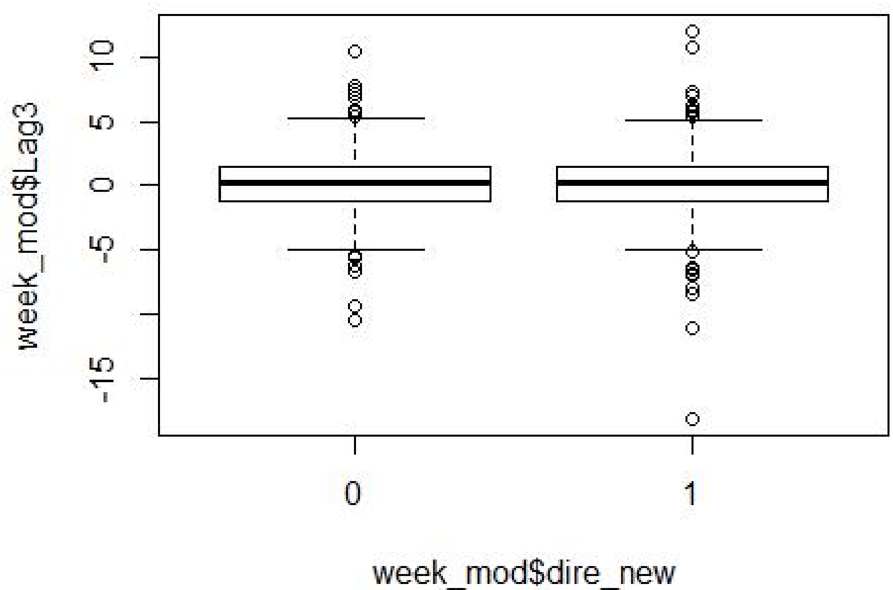
```
#Produce some numerical and graphical summaries of the Weekly data. Do
there appear to be any patterns?

library(ISLR)
#summary(Weekly)
#head(Weekly)

week_mod=Weekly # Copy for transformations
week_mod$dire_new=ifelse(Weekly$Direction == "Up", 1,0)

#Drop Direction column - which has
strings week_mod=week_mod[,-9]

# Data Distribution Plots

boxplot(week_mod$Lag1~week_mod$dire_new, data=week_mod)
```



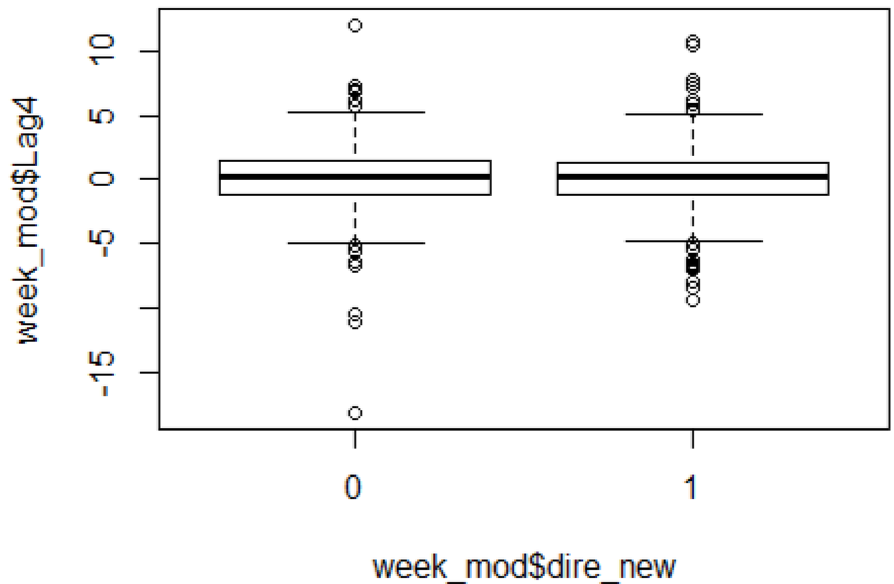
```r
boxplot(week_mod$Lag2~week_mod$dire_new, data=week_mod)
```



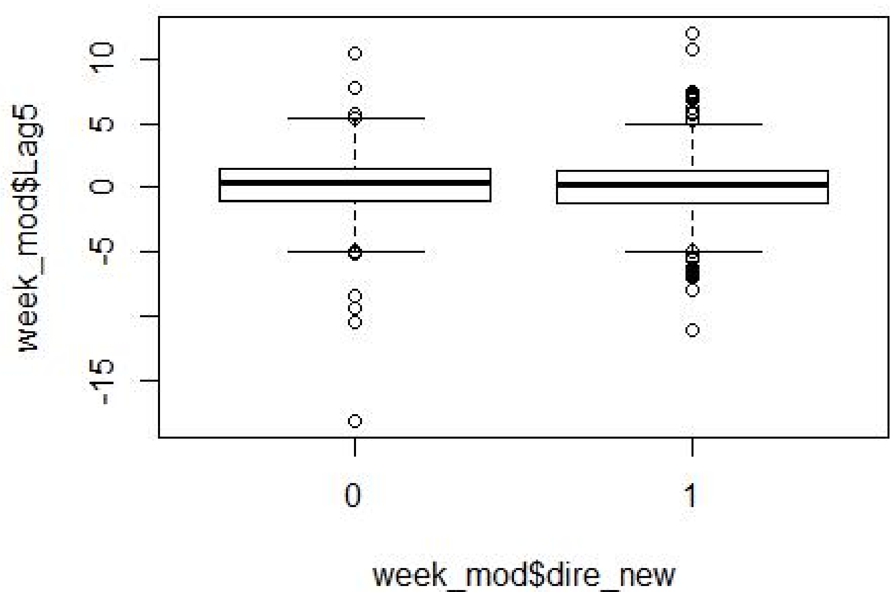

```r
boxplot(week_mod$Lag3~week_mod$dire_new, data=week_mod)
```



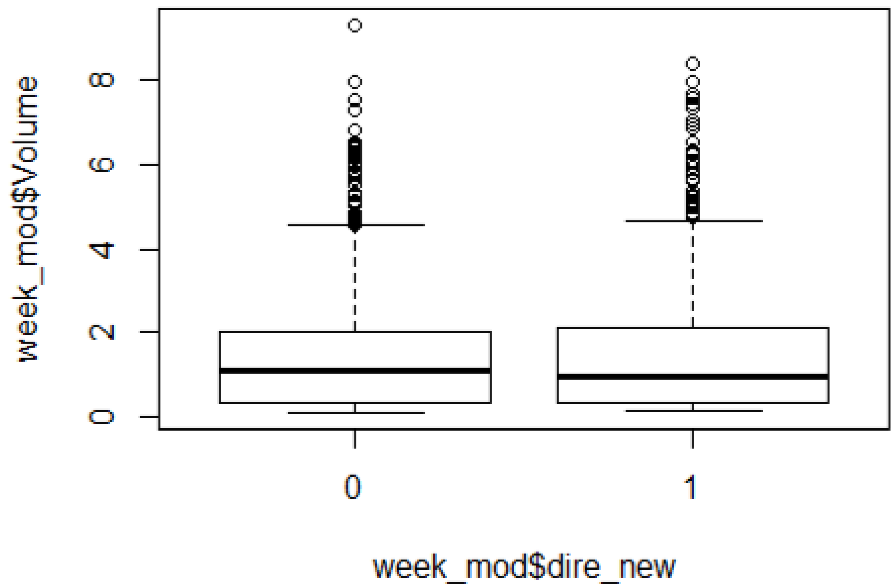
```r
boxplot(week_mod$Lag4~week_mod$dire_new, data=week_mod)
```



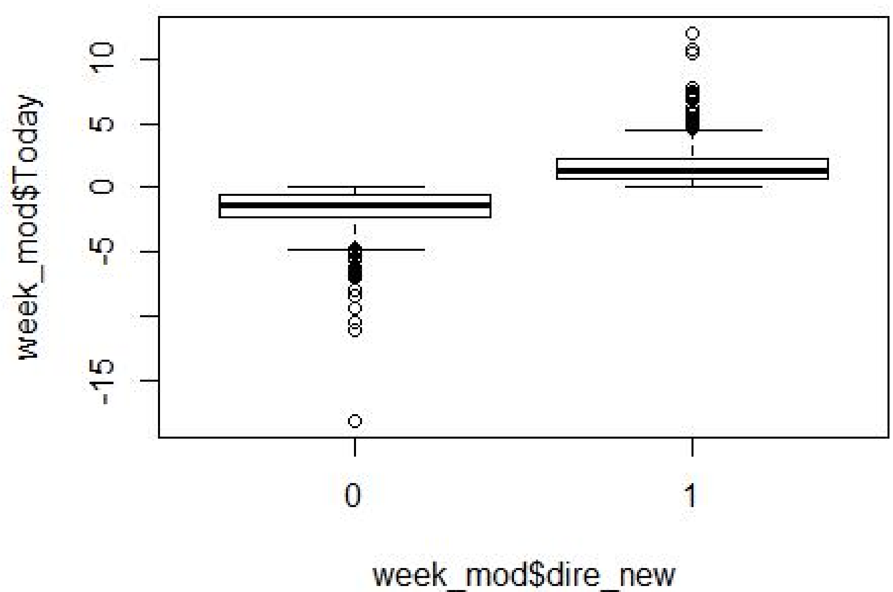

```r
boxplot(week_mod$Lag5~week_mod$dire_new, data=week_mod)
```



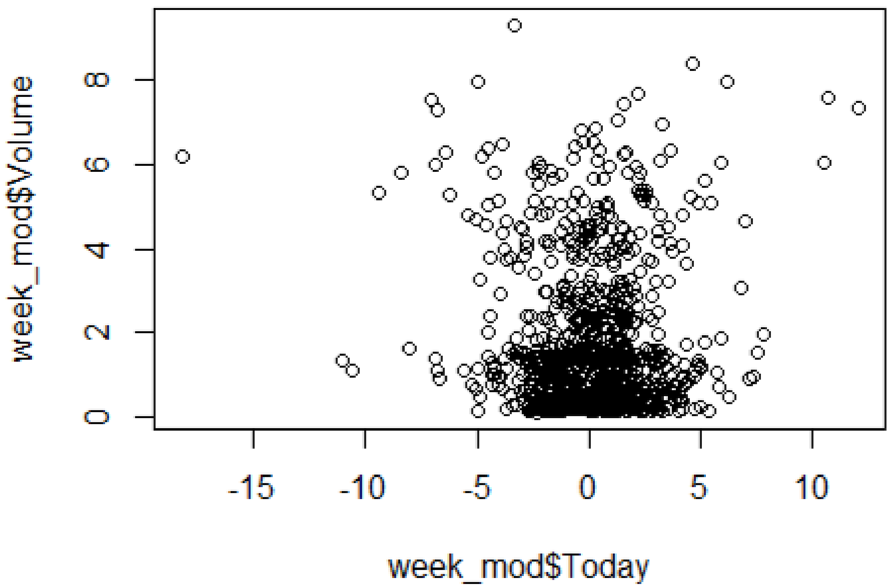
```r
boxplot(week_mod$Volume~week_mod$dire_new, data=week_mod)
```




```r
boxplot(week_mod$Today~week_mod$dire_new, data=week_mod)
```



```r
plot(week_mod$Today, week_mod$Volume, data=week_mod)
```




```r
cor_matrix=data.frame(cor(week_mod[-1]))
cor_matrix
```

```
##                  Lag1        Lag2        Lag3         Lag4         Lag5
## Lag1      1.000000000 -0.07485305  0.05863568 -0.071273876 -0.008183096
## Lag2     -0.074853051  1.00000000 -0.07572091  0.058381535 -0.072499482
## Lag3      0.058635682 -0.07572091  1.00000000 -0.075395865  0.060657175
## Lag4     -0.071273876  0.05838153 -0.07539587  1.000000000 -0.075675027
## Lag5     -0.008183096 -0.07249948  0.06065717 -0.075675027  1.000000000
## Volume   -0.064951313 -0.08551314 -0.06928771 -0.061074617 -0.058517414
## Today    -0.075031842  0.05916672 -0.07124364 -0.007825873  0.011012698
## dire_new -0.050003804  0.07269634 -0.02291281 -0.020549456 -0.018168272
##               Volume        Today    dire_new
## Lag1     -0.06495131 -0.075031842 -0.05000380
## Lag2     -0.08551314  0.059166717  0.07269634
## Lag3     -0.06928771 -0.071243639 -0.02291281
## Lag4     -0.06107462 -0.007825873 -0.02054946
## Lag5     -0.05851741  0.011012698 -0.01816827
## Volume    1.00000000 -0.033077783 -0.01799521
## Today    -0.03307778  1.000000000  0.72002470
## dire_new -0.01799521  0.720024704  1.00000000
```

Findings:

- Percentage of return this week is observed to be a deciding factor for direction of the return on a given week
- From the correlation matrix, no strong correlation has been observed amongst all the variables except for Direction and Today

```
##
## Call:
## glm(formula = dire_new ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
##     family = binomial, data = week_mod)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181
## Lag2         0.05844    0.02686   2.175   0.0296 *
## Lag3        -0.01606    0.02666  -0.602   0.5469
## Lag4        -0.02779    0.02646  -1.050   0.2937
## Lag5        -0.01447    0.02638  -0.549   0.5833
## Volume      -0.02274    0.03690  -0.616   0.5377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
## Number of Fisher Scoring iterations: 4

## [1] 0.4371

## [1] "There is a mis-clasification error of : 43.71% with the current model
"
```

```
##     0   1
## 03123
## 1 453 582

## [1] 0.9619835

## [1] 0.06404959
```

Finding: -

1. Only Lag2, seems to be significant with standard error of 0.02686 and z-stats of 2.175

2. The model has overall accuracy of 56.3%

3. The model sensitivity is ~96% i.e. the model is capable of predicting actual Ups in the direction as Up. However, it has very low sensitivity (6.4%) i.e. the actual Down in the direction are not correctly predicted by the model

4. The model is able to predict the true positives well. However, many of the 'Down' values are predicted incorrectly by the model

```
#Logistic reg with train and test

train_log=week_mod[which(week_mod$Year<=2008),]
test_log=week_mod[which(week_mod$Year>2008),]

log_reg_2=glm(dire_new~Lag2, data=train_log,family=binomial)

summary(log_reg_2)

##
## Call:
## glm(formula = dire_new ~ Lag2, family = binomial, data = train_log)
##
## Deviance Residuals:
##    Min       1Q   Median     3Q     Max
## -1.536   -1.264   1.021   1.091    1.368
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.20326    0.06428   3.162   0.00157 **
## Lag2          0.05810    0.02870   2.024   0.04298 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1354.7  on 984  degrees of freedom
## Residual deviance: 1350.5  on 983  degrees of freedom
## AIC: 1354.5
##
## Number of Fisher Scoring iterations: 4

# predict for the entire data set
predicted <- predict(log_reg_2, test_log, type="response")

#Decide on optimal cut-off
library(InformationValue)
optCutOff <- optimalCutoff(test_log$dire_new, predicted)[1]
optCutOff
```

```
## [1] 0.4953567

#0.4953

conf_mat=confusionMatrix(test_log$dire_new, predicted, threshold = optCutOff)
conf_mat

##    0  1
## 0  8  4
## 13557

sensitivity(test_log$dire_new, predicted, threshold = optCutOff)

## [1] 0.9344262

# Sensitivity= 0.9344

specificity(test_log$dire_new, predicted, threshold =

optCutOff) ## [1] 0.1860465

# specificity = 0.1860

misClassError(test_log$dire_new, predicted, threshold = optCutOff)

## [1] 0.375

#Mis classification is 37.5%
```

Findings: -

- The model has ~ 63.5% of fraction of correct predictions
- Sensitivity = 93.44% i.e. 93.44% of Ups are predicted as Ups by the model
- Specificity = 18.6% i.e. 18.6% of Downs are predicted as Ups by the

model <u>Question 4 : KNN</u>

```
library(class)

set.seed(11)


knn_pred=knn( data.frame(train_log$Lag2),data.frame(test_log$Lag2),train_log$d
ire_new ,k=1)
df=table(knn_pred ,test_log$dire_new)

pred_accuracy=(df[1,1]+df[2,2])/(df[1,1]+df[1,2]+df[2,1]+df[2,2])
print(paste("Prediction accuracy with KNN is : ", pred_accuracy*100,"%"))

## [1] "Prediction accuracy with KNN is :  50 %"
```

Finding: -

Logistic regression is observed to be producing better prediction accuracy compared to knn when k=1 (Accuracy of logistic reg=63.5% and accuracy of knn=50%)

```
pred_accuracy2=list()

set.seed(11)
for (i in 1:50)
{
  knn_pred=knn(data.frame(train_log$Lag2),data.frame(test_log$Lag2),train_log
$dire_new ,k=i)
  df=table(knn_pred ,test_log$dire_new)

  pred_accuracy2[i]=(df[1,1]+df[2,2])/(df[1,1]+df[1,2]+df[2,1]+df[2,2])

  print(paste("Prediction accuracy is : ", pred_accuracy2[i],"for k=", i))
}

## [1] "Prediction accuracy is :  0.5 for k= 1"
## [1] "Prediction accuracy is :  0.596153846153846 for k= 2"
## [1] "Prediction accuracy is :  0.538461538461538 for k= 3"
## [1] "Prediction accuracy is :  0.605769230769231 for k= 4"
## [1] "Prediction accuracy is :  0.528846153846154 for k= 5"
## [1] "Prediction accuracy is :  0.557692307692308 for k= 6"
## [1] "Prediction accuracy is :  0.548076923076923 for k= 7"
## [1] "Prediction accuracy is :  0.567307692307692 for k= 8"
## [1] "Prediction accuracy is :  0.548076923076923 for k= 9"
## [1] "Prediction accuracy is :  0.548076923076923 for k= 10"
## [1] "Prediction accuracy is :  0.548076923076923 for k= 11"
## [1] "Prediction accuracy is :  0.596153846153846 for k= 12"
## [1] "Prediction accuracy is :  0.596153846153846 for k= 13"
## [1] "Prediction accuracy is :  0.567307692307692 for k= 14"
## [1] "Prediction accuracy is :  0.586538461538462 for k= 15"
## [1] "Prediction accuracy is :  0.538461538461538 for k= 16"
## [1] "Prediction accuracy is :  0.596153846153846 for k= 17"
## [1] "Prediction accuracy is :  0.576923076923077 for k= 18"
## [1] "Prediction accuracy is :  0.557692307692308 for k= 19"
## [1] "Prediction accuracy is :  0.567307692307692 for k= 20"
## [1] "Prediction accuracy is :  0.557692307692308 for k= 21"
## [1] "Prediction accuracy is :  0.567307692307692 for k= 22"
## [1] "Prediction accuracy is :  0.567307692307692 for k= 23"
## [1] "Prediction accuracy is :  0.548076923076923 for k= 24"
## [1] "Prediction accuracy is :  0.548076923076923 for k= 25"
## [1] "Prediction accuracy is :  0.538461538461538 for k= 26"
## [1] "Prediction accuracy is :  0.528846153846154 for k= 27"
## [1] "Prediction accuracy is :  0.548076923076923 for k= 28"
## [1] "Prediction accuracy is :  0.548076923076923 for k= 29"
```

```
## [1] "Prediction accuracy is :  0.519230769230769 for k= 30"
## [1] "Prediction accuracy is :  0.538461538461538 for k= 31"
## [1] "Prediction accuracy is :  0.557692307692308 for k= 32"
## [1] "Prediction accuracy is :  0.548076923076923 for k= 33"
## [1] "Prediction accuracy is :  0.557692307692308 for k= 34"
## [1] "Prediction accuracy is :  0.576923076923077 for k= 35"
## [1] "Prediction accuracy is :  0.557692307692308 for k= 36"
## [1] "Prediction accuracy is :  0.567307692307692 for k= 37"
## [1] "Prediction accuracy is :  0.576923076923077 for k= 38"
## [1] "Prediction accuracy is :  0.567307692307692 for k= 39"
## [1] "Prediction accuracy is :  0.557692307692308 for k= 40"
## [1] "Prediction accuracy is :  0.557692307692308 for k= 41"
## [1] "Prediction accuracy is :  0.557692307692308 for k= 42"
## [1] "Prediction accuracy is :  0.557692307692308 for k= 43"
## [1] "Prediction accuracy is :  0.567307692307692 for k= 44"
## [1] "Prediction accuracy is :  0.548076923076923 for k= 45"
## [1] "Prediction accuracy is :  0.596153846153846 for k= 46"
## [1] "Prediction accuracy is :  0.615384615384615 for k= 47"
## [1] "Prediction accuracy is :  0.586538461538462 for k= 48"
## [1] "Prediction accuracy is :  0.586538461538462 for k= 49"
## [1] "Prediction accuracy is :  0.576923076923077 for k= 50"
```

Findings: -

- The model trained with all the variables on training data provides the accuracy of 41.35%
- When Today is considered while building the model along with Lag2, the model does not converge i.e. not able to solve the likelihood
- For knn with Log2 as a predictor, the best accuracy of 54.8 is obtained for k=10
- Hence the best method : Logistic regression with Lag2 as a predictor with accuracy of 63.5%

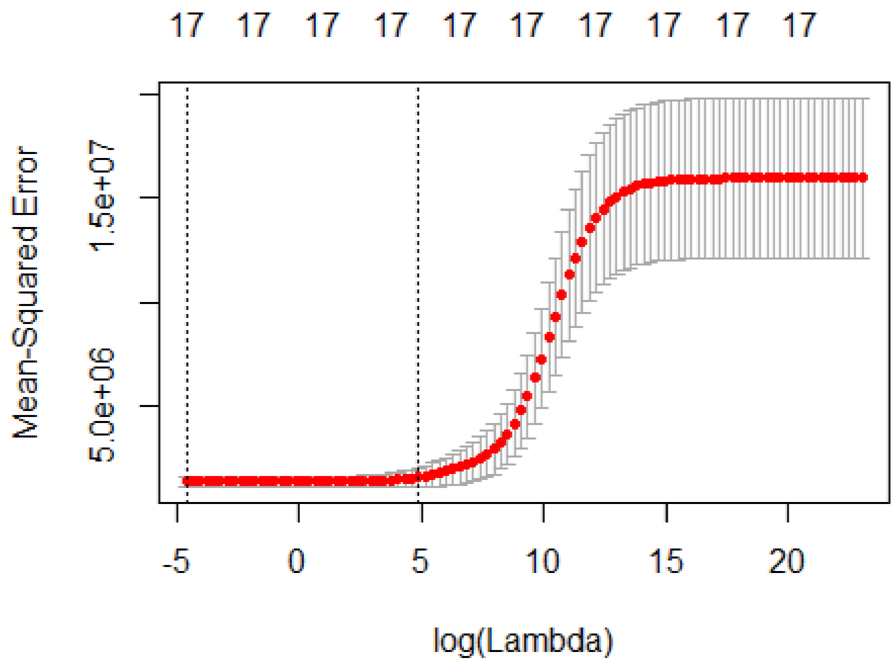
===

Chapter 6 : Question 9

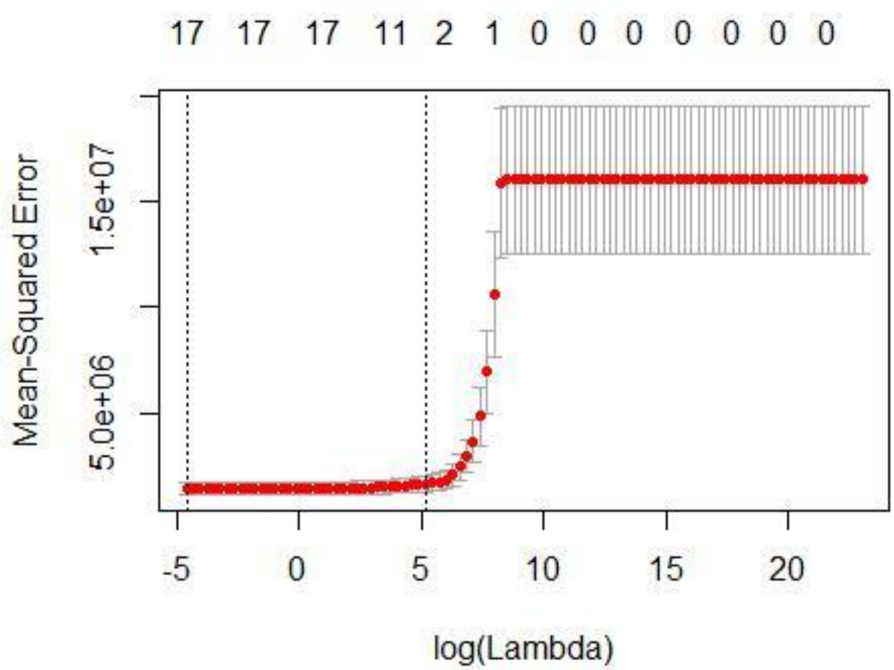
```
#Import Data

library(ISLR)
#Data Wrangling :
#Convert private flag :
college=College
college$priv_flag <- ifelse(college$Private == "Yes", 1,0)
#Drop the string Private column and use priv_flag instead
colg_mod=college[,-1]
```

```
#Divide data into test and training
set.seed (11)
train=sample (1: nrow(colg _mod), nrow(colg_mod)*0.8)
train_clg=colg_mod[train,]
test=(- train )
test_clg=colg_mod[test,]

#Regression Model : Predict #applications
ls_reg=lm(Apps~Accept+Enroll+Top10perc+Top25perc+F.Undergrad+P.Undergrad +Out
state+ Room.Board+Books+Personal+PhD +Terminal+S.F.Ratio+perc.alumni +Expend+
Grad.Rate+ priv_flag,data=train_clg)
print("Summary of Least Square Regression Model:")

## [1] "Summary of Least Square Regression Model:"

summary(ls_reg)

##
## Call:
## lm(formula = Apps ~ Accept + Enroll + Top10perc + Top25perc +
##     F.Undergrad + P.Undergrad + Outstate + Room.Board + Books +
##     Personal + PhD + Terminal + S.F.Ratio + perc.alumni + Expend +
##     Grad.Rate + priv_flag, data = train_clg)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5137.1  -423.1   -21.2   331.6  7100.9
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.784e+02  4.737e+02  -0.799  0.42469
## Accept       1.670e+00 4.701e-02  35.534  < 2e-16 ***
## Enroll      -1.221e+00 2.186e-01  -5.585 3.54e-08 ***
## Top10perc    5.959e+01 6.404e+00   9.306  < 2e-16 ***
## Top25perc   -2.044e+01 5.169e+00  -3.955 8.56e-05 ***
## F.Undergrad  8.669e-02 3.973e-02   2.182  0.02952 *
## P.Undergrad  4.757e-03 4.651e-02   0.102  0.91857
## Outstate    -9.302e-02 2.240e-02  -4.153 3.75e-05 ***
## Room.Board   1.504e-01 5.710e-02   2.634  0.00866 **
## Books       -1.482e-01 2.756e-01  -0.538  0.59099
## Personal     1.175e-01 7.729e-02   1.520  0.12907
## PhD         -9.781e+00 5.350e+00  -1.828  0.06803 .
## Terminal    -3.056e-01 5.889e+00  -0.052  0.95863
## S.F.Ratio    1.440e+01 1.494e+01   0.963  0.33573
## perc.alumni -8.579e-01  4.908e+00  -0.175  0.86131
## Expend       7.164e-02 1.344e-02   5.330  1.39e-07 ***
## Grad.Rate    9.541e+00 3.520e+00   2.710  0.00692 **
## priv_flag   -5.130e+02 1.627e+02  -3.153  0.00170 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1090 on 603 degrees of freedom
## Multiple R-squared:  0.9278, Adjusted R-squared:  0.9257
## F-statistic: 455.5 on 17 and 603 DF,  p-value: < 2.2e-16

# AIC of LS Regression
ls_reg_aic=AIC(ls_reg)

# Predictions for
Test library(caret)

##
## Attaching package: 'caret'

## The following objects are masked from 'package:InformationValue':
##
##     confusionMatrix, precision, sensitivity, specificity

app_pred=predict(ls_reg ,test_clg)

actuals_preds <- data.frame(cbind(actuals=test_clg$Apps,predicteds=app_pred))

rmse= sqrt(mean((actuals_preds$predicteds -actuals_preds$actuals)^2))
print(paste("The error obtained with linear regression is : ",rmse))

## [1] "The error obtained with linear regression is :  877.955466111169"

#summary(colg_mod$Apps)

## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-18
```




```
## [1] "RMSE of Ridge Regression:  747.181968829909"
```



```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept) -720.1372404
## Accept         1.3918057
## Enroll         .
## Top10perc     25.1615845
## Top25perc      .
## F.Undergrad    .
## P.Undergrad    .
## Outstate       .
## Room.Board     .
## Books          .
## Personal       .
```

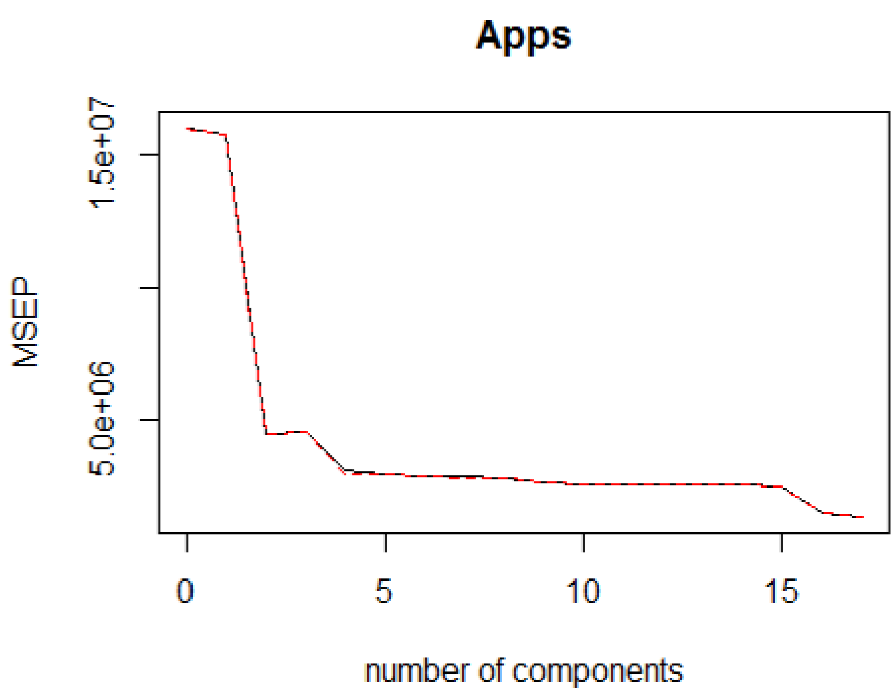
```
## [1] "RMSE of Lasso Regression:  758.358054424561"
```

Finding: -

There are 3 Non-zero coefficients: Accept, Top10perc, Expend

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:caret':
##
##     R2

## The following object is masked from 'package:stats':
##
##     loadings

## Data:X dimension: 621 17
##  Y dimension: 621 1
## Fit method: svdpc
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            4001     3973     2120     2131     1772     1722     1695
## adjCV         4001     3973     2118     2137     1715     1711     1691
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps 13 comps
## CV        1688     1676     1622      1601      1606      1610     1612
```

```
## adjCV       1683      1673      1618       1596       1602       1606      1608
##         14 comps  15 comps  16 comps  17 comps
## CV          1615       1583       1227       1173
## adjCV       1611       1572       1220       1167
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X        32.014    57.63    64.21    70.02    75.49     80.69    84.29
## Apps      1.741    72.60    72.75    82.52    82.68     83.12    83.31
##         8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X         87.61    90.45     92.81     95.00     96.84     97.90     98.74
## Apps      83.46    84.65     85.21     85.21     85.21      85.27     85.31
##        15 comps  16 comps  17 comps
## X          99.37      99.84     100.00
## Apps       88.80      92.18      92.78
```




```
## [1] 1291.631
```

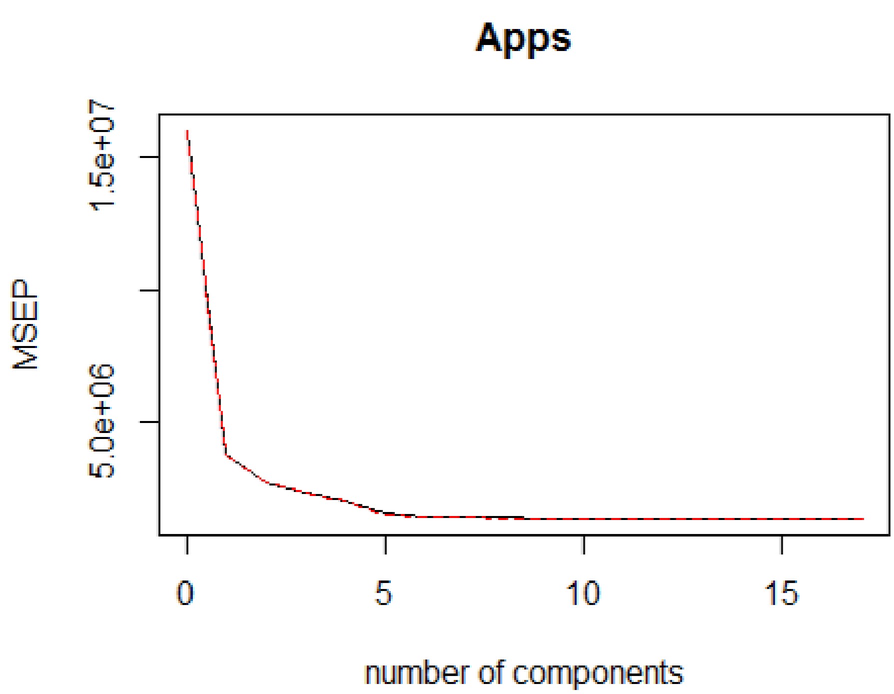
Finding: -

RMSE obtained by PCR = 1358.84 and number of components chosen = 5 as the rate of change in RMSE drops post M=5

Question Problem 3 : PLS

```
## Data:X dimension: 621 17
##  Y dimension: 621 1
## Fit method: kernelpls
```

```
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            4001     1944     1657     1540     1437     1251     1204
## adjCV         4001     1941     1653     1535     1421     1232     1197
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1193     1184     1176      1176      1179      1176      1173
## adjCV     1187     1178     1170      1170      1173      1170      1167
##        14 comps  15 comps  16 comps  17 comps
## CV         1173      1173      1173      1173
## adjCV      1167      1167      1167      1167
##
## TRAINING: % variance explained
##        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X        25.70    39.80    63.08    65.47    67.88    72.85    76.71
## Apps     77.48    84.47    86.77    90.15    92.12    92.45    92.53
##        8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X        79.96    82.24     84.85     87.43     90.90     92.62     95.10
## Apps     92.61    92.69     92.72     92.75     92.77     92.77     92.78
##        15 comps 16 comps  17 comps
## X         96.15    97.81    100.00
## Apps      92.78    92.78     92.78
```




```
## [1] "The error obtained with PLS is :  820.70070893994"
```

Findings:

College Applications could be predicted by : - Least square linear regression with the error of 930 applications

- Ridge Regression with the error of 948 applications (with 1 SE lambda)
- Lasso Regression with the error of 1034 applications
- PCR with the error of 1358 applications
- PLS with the error of 973 applications

out of all the models, Least square and ridge regression provide accurate results with error of 930 and 948 respectively.

==

Chapter 6 : Question 11

```
require(leaps)

## Loading required package: leaps

require(glmnet)
require(MASS)
data(Boston)

# Train and Test
Data set.seed(11)
samp1=sample(nrow(Boston),
nrow(Boston)*0.70) bstn_train<-
Boston[samp1,] bstn_test<-Boston[-samp1,]
```

1. OLS

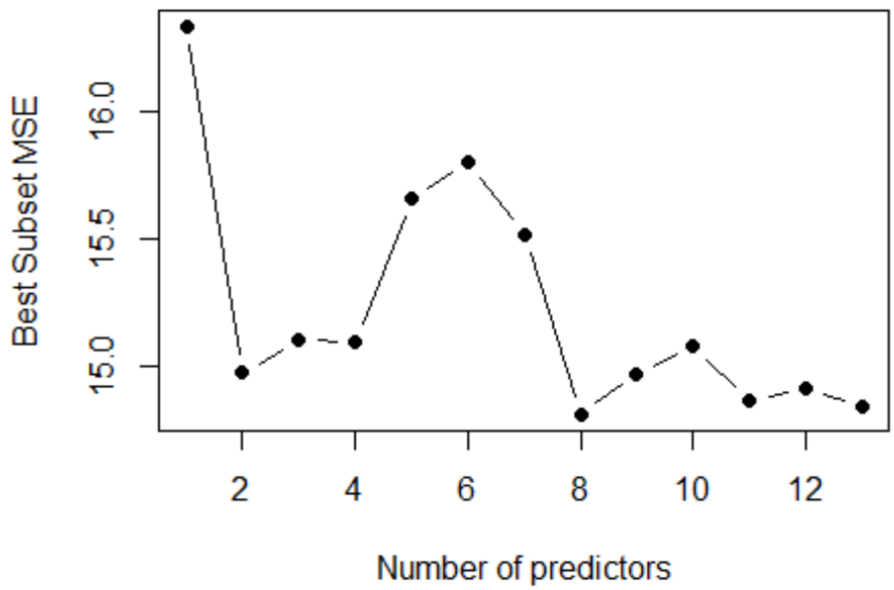
```
ols_bstn=lm(crim~.,data=bstn_train)
ols_pred_bstn=predict(ols_bstn,bstn_test)
ols_rmse=sqrt(mean((ols_pred_bstn-bstn_test$crim)^2))
print(paste("The error obtained with OLS : ",ols_rmse))

## [1] "The error obtained with OLS :  3.9517620206558"
```

2. Best Subsets

```
#Best Subset Model
library(leaps)
best_sb=regsubsets(crim~.,data=bstn_train,nvmax=13)
summary(best_sb)

## Subset selection object
## Call: regsubsets.formula(crim ~ ., data = bstn_train, nvmax = 13)
```

```
## 13 Variables  (and intercept)
##           Forced in Forced out
## zn            FALSE      FALSE
## indus         FALSE      FALSE
## chas          FALSE      FALSE
## nox           FALSE      FALSE
## rm            FALSE      FALSE
## age           FALSE      FALSE
## dis           FALSE      FALSE
## rad           FALSE      FALSE
## tax           FALSE      FALSE
## ptratio       FALSE      FALSE
## black         FALSE      FALSE
## lstat         FALSE      FALSE
## medv          FALSE      FALSE
## 1 subsets of each size up to 13
## Selection Algorithm: exhaustive
##             zn  indus chas nox rm  age dis rad tax ptratio black lstat medv
 ## 1  ( 1 )     """"  "" """"""""""*""""""                       ""  ""  ""
 ## 2  ( 1 )     """"  "" """"""""""*""""""                       ""  "*"  ""
 ## 3  ( 1 )     """"  " "  """"""""""*""""""                     " "   "*"  "*"
 ## 4  ( 1 )    "*"""  " "  """"""""*""*""""""                    ""  ""  "*"
 ## 5  ( 1 )    "*"""  " "  "*"""""""*""*""""""                   ""  ""  "*"
 ## 6  ( 1 )    "*"""  " "  "*"""""""*""*"""""*"                  " "   " "  "*"
 ## 7  ( 1 )    "*"""  " "  "*"""""""*""*"""""*"                  " "   "*"  "*"
 ## 8  ( 1 )    "*"""  " "  "*"""""""*""*"""""*"                  "*"   "*"  "*"
 ## 9  ( 1 )    "*"""  " "  "*" "*" " " "*" "*" " " "*"           "*"   "*"  "*"
## 10  ( 1 ) "*" " "   "*"  "*" "*" " " "*" "*" " " "*""*"   "*"   "*"
## 11  ( 1 ) "*" " "   "*"  "*" "*" " " "*" "*" "*" "*""*"   "*"   "*"
 ##  12  ( 1 ) "*"  " "   "*"  "*" "*" "*" "*" "*" "*" "*"     "*"   "*"  "*"
 ##  13  ( 1 ) "*"  "*"   "*"  "*" "*" "*" "*" "*" "*" "*"     "*"   "*"  "*"
```

```r
bstn_test_mat=model.matrix(crim~., data = bstn_test, nvmax = 13)
bstn_error=rep(NA, 13)
for (i in 1:13)
{
    coefi = coef(best_sb, id = i)
    pred = bstn_test_mat[, names(coefi)] %*% coefi
    bstn_error[i] <- mean((pred - bstn_test$crim)^2)
}
plot(bstn_error, xlab = "Number of predictors", ylab = "Best Subset MSE", pch
= 19, type = "b")
```




```
m_p=which.min(bstn_error)
print(paste("Minimum Test MSE is obtained with : ",m_p))

## [1] "Minimum Test MSE is obtained with :  8"

coef(best_sb,which.min(bstn_error))

##   (Intercept)            zn           nox           dis           rad
##  24.880365651   0.050529944 -16.557328860  -1.169721365   0.564539036
##       ptratio         black         lstat          medv
##  -0.418372125 -0.005609131 0.108277856 -0.227029910

best_sb_mse<-bstn_error[which.min(bstn_error)[1]]

print(paste("The error obtained with Best Subset Selection : ",sqrt(best_sb_m
se)))

## [1] "The error obtained with Best Subset Selection :  3.74782682891539"
```

Forward Selection

```
#forward selection

null_fit=lm(crim ~ 1, data = bstn_train)
full_fit=lm(crim ~ ., data = bstn_train)

fwd_model_bstn=step(null_fit, scope = list(lower = null_fit, upper = full_fit
),direction = "forward")
```

```
## Start:  AIC=1594.48
## crim ~ 1
##
##           Df Sum of Sq   RSS    AIC
## + rad      1   11504.2 20315 1437.6
## + tax      1   10340.1 21479 1457.4
## + lstat    1    6434.1 25385 1516.5
## + indus    1    5285.3 26534 1532.2
## + medv     1    5067.6 26751 1535.1
## + nox      1    4993.1 26826 1536.0
## + dis      1    4407.1 27412 1543.7
## + black    1    4012.1 27807 1548.8
## + age      1    3786.0 28033 1551.6
## + ptratio  1    2499.8 29319 1567.5
## + rm       1    1404.9 30414 1580.5
## + zn       1    1141.0 30678 1583.5
## + chas     1     199.6 31619 1594.2
## <none>                 31819 1594.5
##
## Step:  AIC=1437.63
## crim ~ rad
##
##           Df Sum of Sq   RSS    AIC
## + lstat    1    904.60 19410 1423.5
## + medv     1    840.63 19474 1424.7
## + black    1    246.99 20068 1435.3
## + dis      1    227.81 20087 1435.6
## + rm       1    226.35 20088 1435.7
## + age      1    210.02 20105 1436.0
## <none>                 20315 1437.6
## + chas     1     91.33 20223 1438.0
## + indus    1     53.71 20261 1438.7
## + nox      1     44.13 20271 1438.9
## + tax      1     36.13 20279 1439.0
## + zn       1      1.80 20313 1439.6
## + ptratio  1      0.08 20315 1439.6
##
## Step:  AIC=1423.51
## crim ~ rad + lstat
##
##           Df Sum of Sq   RSS    AIC
## + medv     1   146.161 19264 1422.8
## + black    1   132.246 19278 1423.1
## <none>                 19410 1423.5
## + zn       1    68.097 19342 1424.3
## + chas     1    58.949 19351 1424.4
## + nox      1    37.707 19372 1424.8
## + dis      1    29.069 19381 1425.0
## + ptratio  1    27.214 19383 1425.0
## + indus    1    22.476 19388 1425.1
```

```
## + rm         1    12.393 19398 1425.3
## + tax        1     1.979 19408 1425.5
## + age        1     1.011 19409 1425.5
##
## Step:  AIC=1422.83
## crim ~ rad + lstat + medv
##
##             Df Sum of Sq   RSS    AIC
## + ptratio   1   119.656 19144 1422.6
## + black     1   112.151 19152 1422.8
## <none>                  19264 1422.8
## + rm        1    99.055 19165 1423.0
## + zn        1    81.090 19183 1423.3
## + dis       1    74.482 19190 1423.5
## + nox       1    30.406 19234 1424.3
## + chas      1    30.390 19234 1424.3
## + indus     1    29.288 19235 1424.3
## + tax       1    14.189 19250 1424.6
## + age       1     7.973 19256 1424.7
##
## Step:  AIC=1422.62
## crim ~ rad + lstat + medv + ptratio
##
##           Df Sum of Sq   RSS    AIC
## <none>                19144 1422.6
## + black   1    93.536 19051 1422.9
## + rm      1    90.456 19054 1423.0
## + nox     1    85.108 19059 1423.0
## + dis     1    73.230 19071 1423.3
## + zn      1    44.264 19100 1423.8
## + chas    1    43.591 19101 1423.8
## + indus   1    27.804 19117 1424.1
## + tax     1    19.154 19125 1424.3
## + age     1     6.578 19138 1424.5
```

```
summary(fwd_model_bstn)
```

```
##
## Call:
## lm(formula = crim ~ rad + lstat + medv + ptratio, data = bstn_train)
##
## Residuals:
##    Min     1Q  Median     3Q    Max
## -9.018 -2.178  -0.540  0.847 75.215
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.06274    5.39400   1.124   0.2618
## rad          0.56631    0.05592  10.127   <2e-16 ***
## lstat        0.14535    0.08504   1.709   0.0883 .
```

```
## medv          -0.14611    0.07006  -2.086   0.0377 *
## ptratio       -0.34031    0.23042  -1.477   0.1406
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.406 on 349 degrees of freedom
## Multiple R-squared:  0.3983, Adjusted R-squared:  0.3914
## F-statistic: 57.76 on 4 and 349 DF,  p-value: < 2.2e-16
```

```
fwd_predict_bstn=predict(fwd_model_bstn,bstn_test)
fwd_mse_bstn=mean((fwd_predict_bstn-bstn_test$crim)^2)

print(paste("The error obtained with Forward Selection : ",sqrt(fwd_mse_bstn)
))
```

```
## [1] "The error obtained with Forward Selection :  3.86586777705972"
```

<u>Backward Selection</u>

```
bwd_model_bstn=step(full_fit,direction = "backward")
```

```
## Start:  AIC=1424.5
## crim ~ zn + indus + chas + nox + rm + age + dis + rad + tax +
##     ptratio + black + lstat + medv
##
##            Df Sum of Sq   RSS    AIC
## - indus     1      0.87 18293 1422.5
## - age       1      3.41 18295 1422.6
## - tax       1     18.67 18311 1422.9
## - chas      1     30.16 18322 1423.1
## - rm        1     40.44 18332 1423.3
## - black     1     52.99 18345 1423.5
## - lstat     1     81.88 18374 1424.1
## <none>                  18292 1424.5
## - ptratio   1    134.11 18426 1425.1
## - zn        1    208.95 18501 1426.5
## - nox       1    242.80 18535 1427.2
## - dis       1    473.58 18765 1431.5
## - medv      1    527.11 18819 1432.6
## - rad       1   1246.55 19538 1445.8
##
## Step:  AIC=1422.52
## crim ~ zn + chas + nox + rm + age + dis + rad + tax + ptratio +
##     black + lstat + medv
##
##            Df Sum of Sq   RSS    AIC
## - age       1      3.58 18296 1420.6
## - tax       1     27.16 18320 1421.0
## - chas      1     31.96 18325 1421.1
## - rm        1     41.72 18334 1421.3
## - black     1     52.53 18345 1421.5
```

```
## - lstat     1     81.24 18374 1422.1
## <none>                  18293 1422.5
## - ptratio   1    140.31 18433 1423.2
## - zn        1    214.71 18507 1424.7
## - nox       1    278.76 18572 1425.9
## - dis       1    479.67 18772 1429.7
## - medv      1    528.91 18822 1430.6
## - rad       1   1373.60 19666 1446.2
##
## Step:  AIC=1420.59
## crim ~ zn + chas + nox + rm + dis + rad + tax + ptratio + black +
##     lstat + medv
##
##           Df Sum of Sq   RSS    AIC
## - tax      1     27.99 18324 1419.1
## - chas     1     31.38 18328 1419.2
## - rm       1     47.55 18344 1419.5
## - black    1     52.01 18348 1419.6
## - lstat    1     97.40 18394 1420.5
## <none>                 18296 1420.6
## - ptratio  1    138.07 18434 1421.2
## - zn       1    211.49 18508 1422.7
## - nox      1    282.74 18579 1424.0
## - medv     1    530.03 18826 1428.7
## - dis      1    560.62 18857 1429.3
## - rad      1   1371.71 19668 1444.2
##
## Step:  AIC=1419.13
## crim ~ zn + chas + nox + rm + dis + rad + ptratio + black + lstat +
##     medv
##
##           Df Sum of Sq   RSS    AIC
## - chas     1      27.0 18351 1417.7
## - rm       1      49.9 18374 1418.1
## - black    1      50.8 18375 1418.1
## <none>                 18324 1419.1
## - lstat    1     106.7 18431 1419.2
## - ptratio  1     145.8 18470 1419.9
## - zn       1     195.3 18520 1420.9
## - nox      1     337.5 18662 1423.6
## - medv     1     503.8 18828 1426.7
## - dis      1     537.2 18862 1427.4
## - rad      1    3427.3 21752 1477.8
##
## Step:  AIC=1417.65
## crim ~ zn + nox + rm + dis + rad + ptratio + black + lstat +
##     medv
##
##          Df Sum of Sq   RSS    AIC
## - rm      1      49.6 18401 1416.6
```

```
## - black      1      54.9 18406 1416.7
## <none>                   18351 1417.7
## - lstat      1     105.1 18456 1417.7
## - ptratio    1     137.5 18489 1418.3
## - zn         1     200.4 18552 1419.5
## - nox        1     353.1 18704 1422.4
## - dis        1     534.3 18886 1425.8
## - medv       1     537.1 18888 1425.9
## - rad        1    3428.3 21780 1476.3
##
## Step:  AIC=1416.6
## crim ~ zn + nox + dis + rad + ptratio + black + lstat + medv
##
##           Df Sum of Sq   RSS    AIC
## - black      1      72.3 18473 1416.0
## - lstat      1      74.4 18475 1416.0
## <none>                   18401 1416.6
## - ptratio    1     141.2 18542 1417.3
## - zn         1     207.5 18609 1418.6
## - nox        1     357.3 18758 1421.4
## - medv       1     490.9 18892 1423.9
## - dis        1     553.5 18954 1425.1
## - rad        1    3553.5 21954 1477.1
##
## Step:  AIC=1415.99
## crim ~ zn + nox + dis + rad + ptratio + lstat + medv
##
##           Df Sum of Sq   RSS    AIC
## - lstat      1      77.9 18551 1415.5
## <none>                   18473 1416.0
## - ptratio    1     158.5 18632 1417.0
## - zn         1     213.8 18687 1418.1
## - nox        1     364.6 18838 1420.9
## - medv       1     543.6 19017 1424.3
## - dis        1     576.9 19050 1424.9
## - rad        1    4227.8 22701 1487.0
##
## Step:  AIC=1415.48
## crim ~ zn + nox + dis + rad + ptratio + medv
##
##           Df Sum of Sq   RSS    AIC
## <none>                   18551 1415.5
## - ptratio    1     180.2 18731 1416.9
## - zn         1     223.8 18775 1417.7
## - nox        1     345.3 18897 1420.0
## - dis        1     728.8 19280 1427.1
## - medv       1    1402.5 19954 1439.3
## - rad        1    4427.8 22979 1489.2
```

```r
summary(bwd_model_bstn)
```

```
##
## Call:
## lm(formula = crim ~ zn + nox + dis + rad + ptratio + medv, data = bstn_tra
in)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.888 -2.451 -0.548  0.847 73.229
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  26.68034    7.96262   3.351 0.000895 ***
## zn            0.05242    0.02562   2.046 0.041527 *
## nox         -16.24233    6.39077  -2.542 0.011471 *
## dis          -1.29960    0.35198  -3.692 0.000258 ***
## rad           0.59781    0.06569   9.101  < 2e-16 ***
## ptratio      -0.46932    0.25563  -1.836 0.067223 .
## medv         -0.29429    0.05746  -5.122 5.03e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.312 on 347 degrees of freedom
## Multiple R-squared:  0.417,  Adjusted R-squared:  0.4069
## F-statistic: 41.36 on 6 and 347 DF,  p-value: < 2.2e-16
```

```r
bwd_predict_bstn=predict(bwd_model_bstn,bstn_test)
bwd_rmse_bstn=sqrt(mean((bwd_predict_bstn-bstn_test$crim)^2))
print(paste("The error obtained with Backward Selection : ",bwd_rmse_bstn))
```

```
## [1] "The error obtained with Backward Selection :  3.97492418190822"
```

Ridge on Boston

```r
train_mat=model.matrix(crim~.,data=bstn_train)
test_mat=model.matrix(crim~.,data=bstn_test)
ridge_model_bstn = cv.glmnet(train_mat,bstn_train$crim, alpha=0)


#1 se lmabda
lambda = ridge_model_bstn$lambda.1se
print(paste("The optimal value for lambda:",lambda))
```

```
## [1] "The optimal value for lambda: 104.364418540726"
```

```r
pred_ridge_bstn = predict(ridge_model_bstn, s=lambda, newx=test_mat)
ridge_bstn_rmse = sqrt(mean((bstn_test$crim - pred_ridge_bstn)^2))


print(paste("The test error in ridge regression:",ridge_bstn_rmse))
```

```
## [1] "The test error in ridge regression: 4.93259484179843"
```

Lasso on Boston

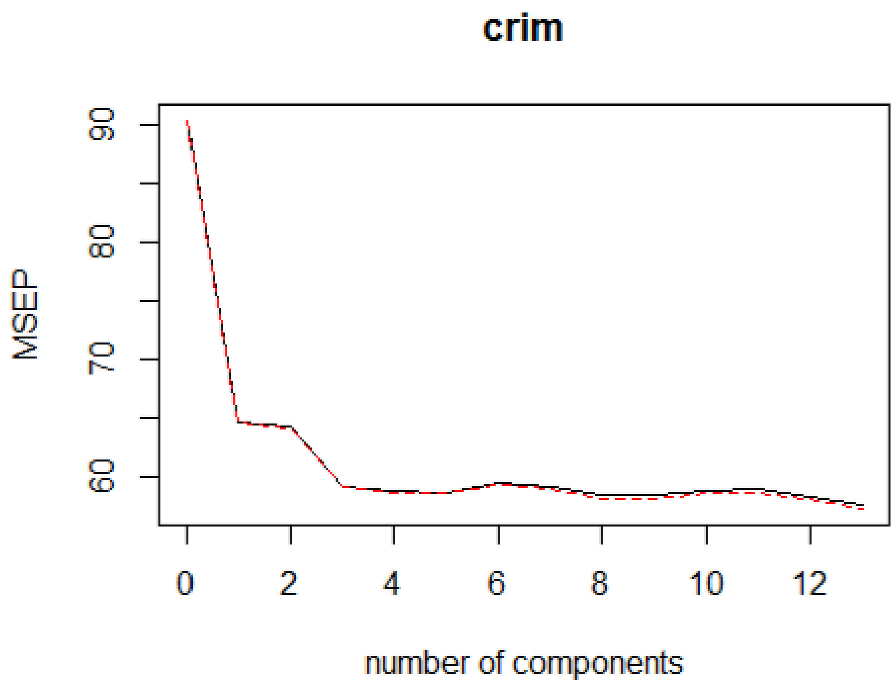
```
train_mat_bstn=model.matrix(crim~.,data=bstn_train)

test_mat_bstn=model.matrix(crim~.,data=bstn_test)

las_bstn=cv.glmnet(train_mat,bstn_train$crim, alpha=1)

#Chosing the optimal lambda value
lambda_las = las_bstn$lambda.1se

print(paste("The optimal value for lambda:",lambda_las))

## [1] "The optimal value for lambda: 3.92925334991307"

pred_lasso_bstn = predict(las_bstn, s=lambda, newx=test_mat)

rmse_lasso_bstn = sqrt(mean((bstn_test$crim - pred_lasso_bstn)^2))

print(paste("The test error in lasso regression:",rmse_lasso_bstn))

## [1] "The test error in lasso regression: 6.05950825285908"
```

PCR

```
library(pls)

pcr_model_bstn=pcr(crim~.,data=bstn_train,scale=TRUE,validation="CV")

summary(pcr_model_bstn)

## Data:X dimension: 354 13
##  Y dimension: 354 1
## Fit method: svdpc
## Number of components considered: 13
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##         (Intercept)  1 comps  2 comps  3 comps 4 comps   5 comps  6 comps
## CV            9.508    8.044    8.020    7.692   7.663     7.661    7.711
## adjCV         9.508    8.039    8.014    7.686   7.654     7.654    7.702
##         7 comps  8 comps   9 comps  10 comps 11 comps  12 comps  13 comps
## CV        7.692    7.644    7.641     7.669    7.678     7.639     7.589
## adjCV     7.681    7.625    7.626     7.652    7.661     7.618     7.567
##
## TRAINING: % variance explained
##       1  comps   2 comps   3 comps  4 comps  5 comps  6 comps   7 comps
## X       47.09     60.15     69.49    76.41   82.80     87.94     91.16
## crim    29.77     30.24     35.97    36.53   36.57     36.89     37.68
##       8  comps   9 comps  10 comps  11 comps 12 comps   13 comps
```

```
## X         93.40    95.47     97.17     98.54     99.60    100.00
## crim      39.37    39.44     39.65     39.78     41.36     42.51
```

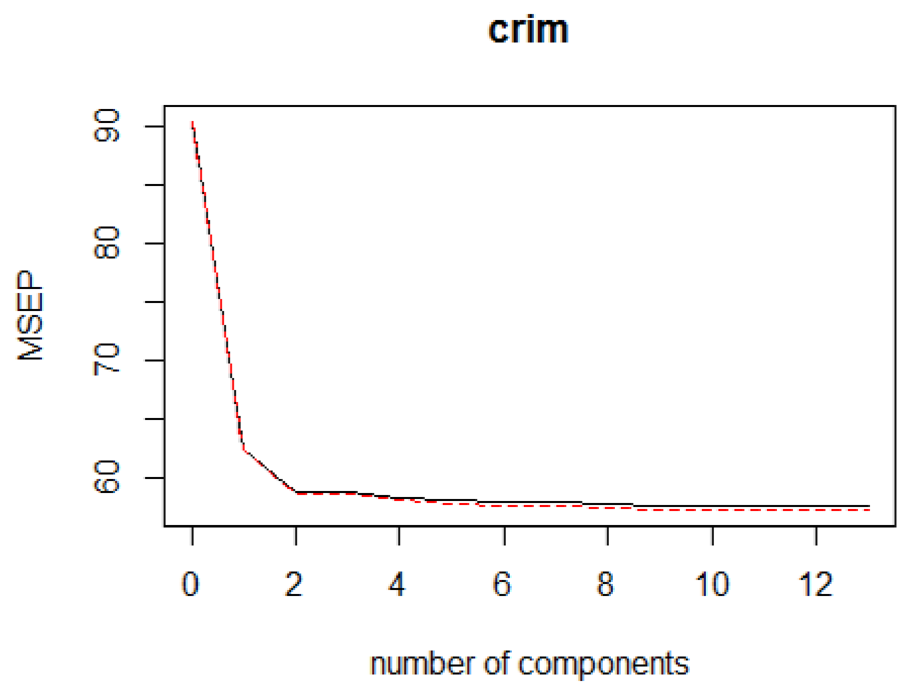
```
validationplot(pcr_model_bstn,val.type="MSEP")
```




```
bstn_pred_pcr=predict(pcr_model_bstn,bstn_test,ncomp=13)

pcr_rmse=sqrt(mean((bstn_test$crim-bstn_pred_pcr)^2))

pcr_rmse

## [1] 3.851762
```

PLS

```
library(pls)

pls_model_bstn=plsr(crim~.,data=bstn_train,scale=TRUE,validation="CV")

validationplot(pls_model_bstn,val.type="MSEP")
```




```
bstn_pred_plsr=predict(pls_model_bstn,bstn_test,ncomp=12)

pls_rmse_bstn=sqrt(mean((bstn_test$crim-bstn_pred_plsr)^2))
pls_rmse_bstn

## [1] 3.851758
```

Comparing RMSEs:

```
cat("PLS:",pls_rmse_bstn,"\nPCR:",pcr_rmse,"\nLasso:",rmse_lasso_bstn,"\nRidg
e:",ridge_bstn_rmse,"\nBackward Selection",bwd_rmse_bstn,"\nForward Selection
:",sqrt(fwd_mse_bstn),"\nBest Subset Selection",sqrt(best_sb_mse),"\nOLS:",ol
s_rmse)

## PLS: 3.851758
## PCR: 3.851762
## Lasso: 6.059508
## Ridge: 4.932595
## Backward Selection 3.974924
## Forward Selection: 3.865868
## Best Subset Selection 3.747827
## OLS: 3.951762
```

Findings:

- The training RMSE is the lowest for Best Subset selection as it contains all the variables in every iteration. This makes the process computationally heavy.

- Amongst the other cross validated models, PLS and PCR provide almost equally accurate prediction capability

Q. Does your chosen model involve all of the features in the data set? Why or why not?

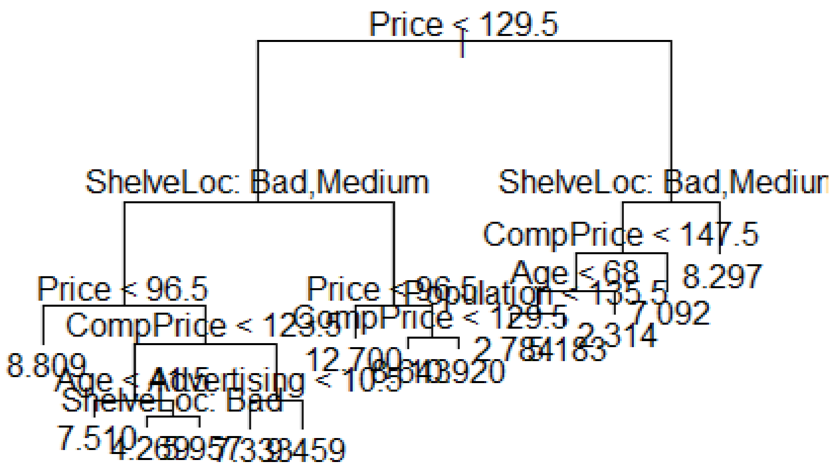
Ans- Yes, the chosen model involves all the features. PCR and PLS use all the coefficients and transform them to reduce the number of effective coefficients.

===

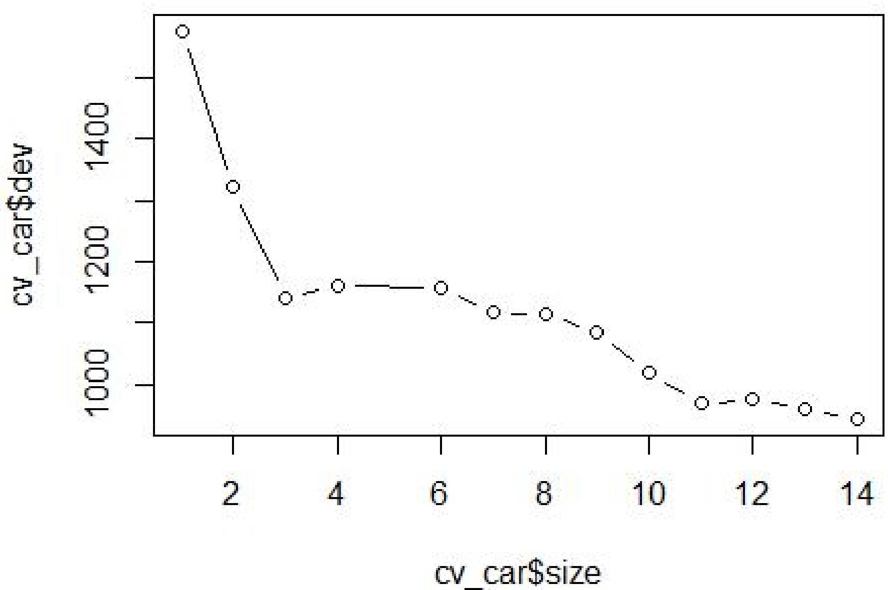
Chapter 8 : Question 8

a)Split the data set into a training set and a test set.
b) Fit a regression tree to the training set. Plot the tree, and interpret the results. What test MSE do you obtain?

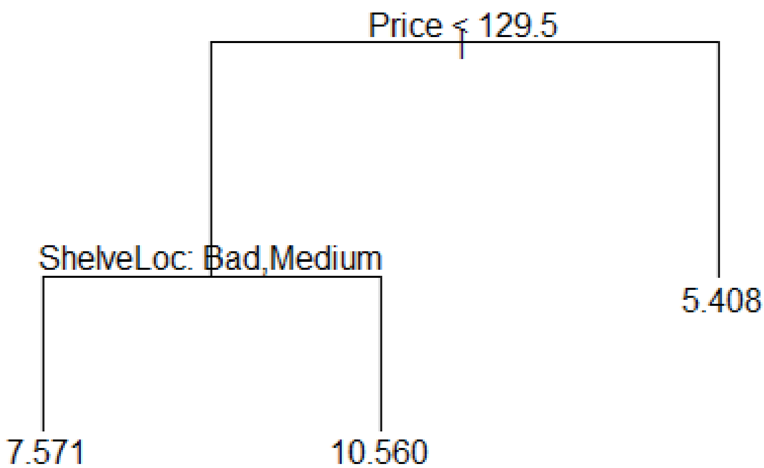
```
library(ISLR)
attach(Carseats)

#Divide into test and sample:
set.seed (2)
train_cr=sample(1: nrow(Carseats), 200)
car_train=Carseats[train_cr ,]
car_test=Carseats[-train_cr ,]

# Fit regression
tree library(tree)
tree_car =tree(Sales~. ,data=Carseats,subset=train_cr)

summary(tree_car)

##
## Regression tree:
## tree(formula = Sales ~ ., data = Carseats, subset = train_cr)
## Variables actually used in tree construction:
## [1] "Price"       "ShelveLoc"    "CompPrice"    "Age"          "Advertising"
## [6] "Population"
## Number of terminal nodes:  14
## Residual mean deviance:  2.602 = 484 / 186
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -4.71700 -1.08700 -0.01026  0.00000  1.11300  4.06600

plot(tree_car)
text(tree_car ,pretty =0)
```



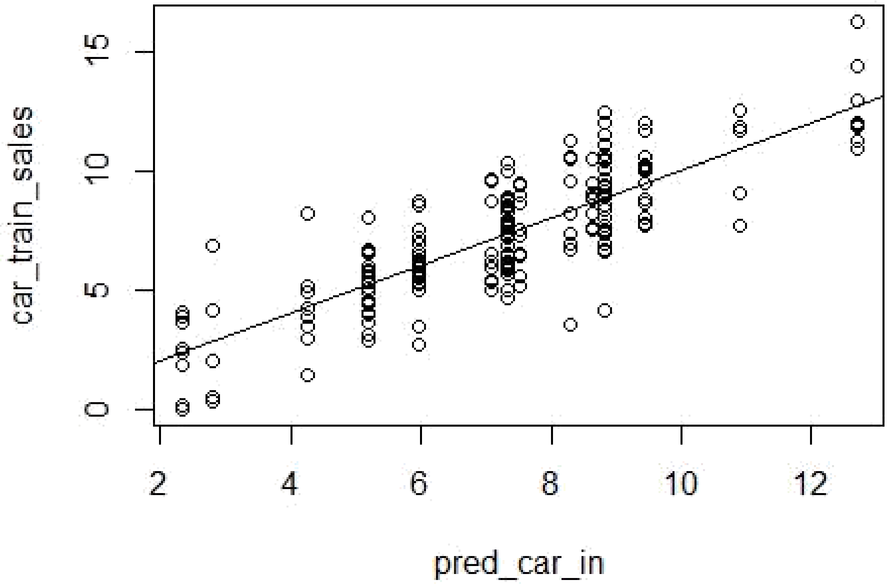

```r
# Check if pruning improves the results :
cv_car =cv.tree(tree_car )
plot(cv_car$size ,cv_car$dev ,type="b")
```



```r
#Predict for the test data set
#pred_tree=predict(tree_car,car_test,type="class")

prune_car=prune.tree(tree_car ,best =3)
plot(prune_car )
text(prune_car ,pretty =0)
```

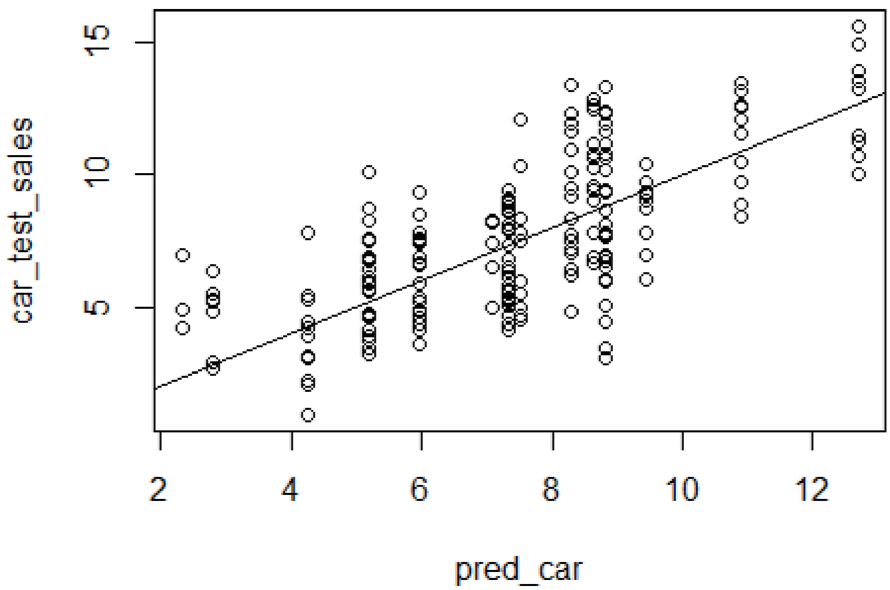



```r
#In-sample validation :
pred_car_in=predict(tree_car ,newdata
=car_train) car_train_sales=car_train[,"Sales"]
plot(pred_car_in ,car_train_sales)
abline (0,1)
```



c) Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test MSE?

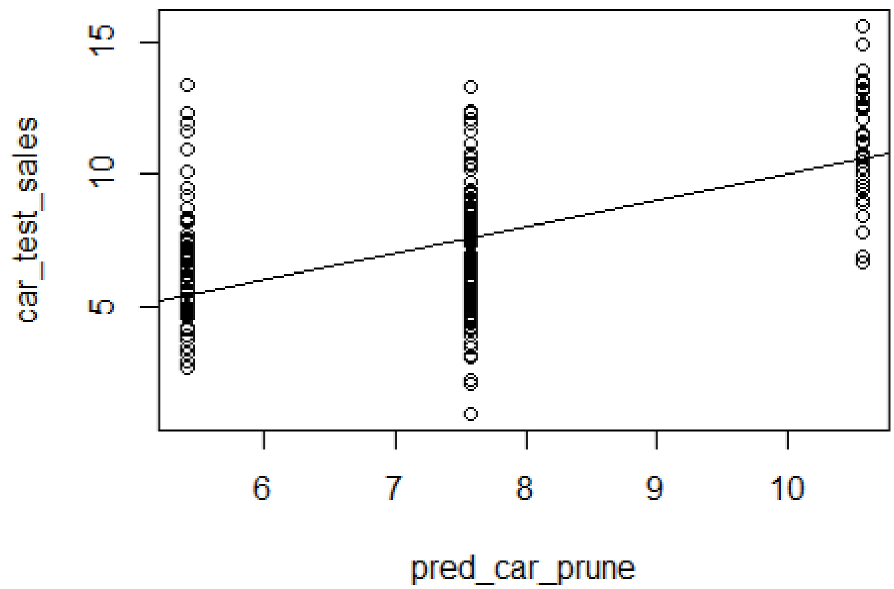
```
MSE_in= mean((pred_car_in -car_train_sales)^2)

# Cross -Validation :

pred_car=predict(tree_car ,newdata
=car_test) car_test_sales=car_test[,"Sales"]
plot(pred_car ,car_test_sales)
abline (0,1)
```




```
MSE_out= mean((pred_car -car_test_sales)^2)
MSE_out
```

```
## [1] 4.471569
```

```
#Check the MSE with pruned tree: cross-validation:

pred_car_prune=predict(prune_car ,newdata =car_test)
plot(pred_car_prune ,car_test_sales)
abline (0,1)
```



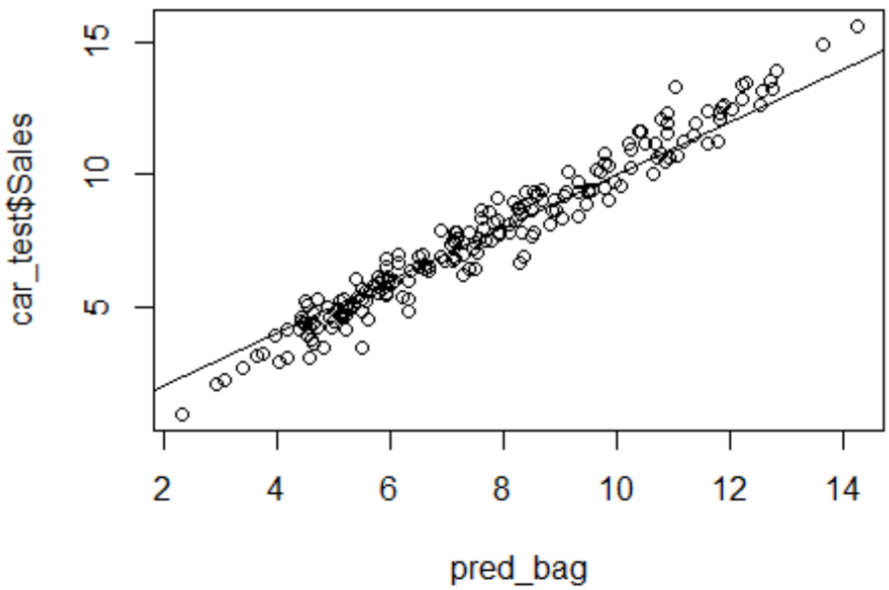

```
MSE_prune= mean((pred_car_prune -car_test_sales)^2)
MSE_prune

## [1] 6.555128

print(paste("In Sample MSE is ",MSE_in,"; Cross-validation MSE = ",MSE_out,"
and MSE with pruned tree = ",MSE_prune," . Hence pruning does not improve the
test MSE"))

## [1] "In Sample MSE is 2.41985071072909 ; Cross-validation MSE = 4.471569
and MSE with pruned tree = 6.55512800081223 . Hence pruning does not improve
the test MSE"
```

Findings:

- Tree without pruning has chosen 6 variables. Pruned tree has 2 tree.
- Since the elbow is obtained at size = 3. Post 3, there is slight increase in deviation and hence the pruned tree is trained for size=3
- Tree Explaination : The single tree, divides the decision right at the Price and goes on to check Shelvloc condition. The pruned tree stops right here without any further condition checking. In contrast, the single tree has 14 terminal nodes with further decision conditions on Compride, age, advertising and population.
- In Sample MSE is 2.41985071072909 ; Cross-validation MSE = 4.471569 and MSE with pruned tree = 6.55512800081223 . Hence pruning does not improve the test MSE. This is because the pruned tree just considers the levels

```
till which there is considerable decrease in entropy and not the
optimal number of levels for classification
```

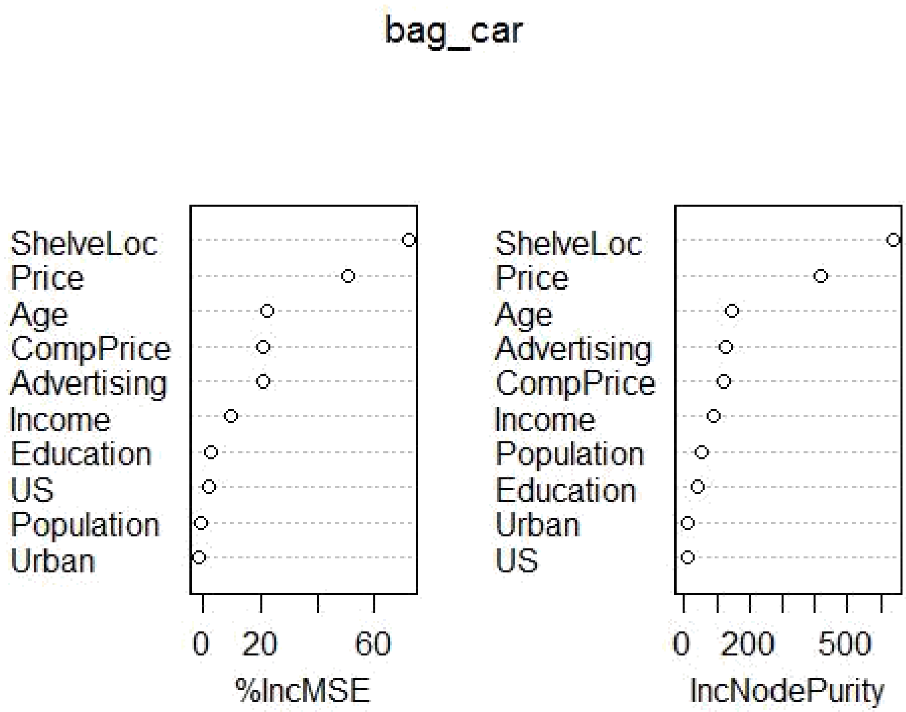
```
#Question 8 d)
# Bagging code library
(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

set.seed (11)
bag_car=randomForest(Sales~.,data=car_test, mtry=10, importance=TRUE) bag_car

##
## Call:
##  randomForest(formula = Sales ~ ., data = car_test, mtry = 10,          import
ance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##           Mean of squared residuals: 2.555682
##                     % Var explained: 69.63

pred_bag     =     predict(bag_car     ,newdata
=car_test)    MSE_bag=    mean((    pred_bag    -
car_test$Sales)^2) #MSE_bag

plot(pred_bag , car_test$Sales)
abline (0,1)
```




```
#Importance :
importance(bag_car)
```

```
##              %IncMSE IncNodePurity
## CompPrice    21.326597    117.770164
## Income        9.401547     89.061429
## Advertising 20.940926    127.826279
## Population   -1.022644     51.072364
## Price        50.872591    418.507629
## ShelveLoc    71.974481    640.876610
## Age          22.778683    143.562270
## Education     2.694575     37.789349
## Urban        -1.238814      8.219514
## US            1.774709      6.598833
```

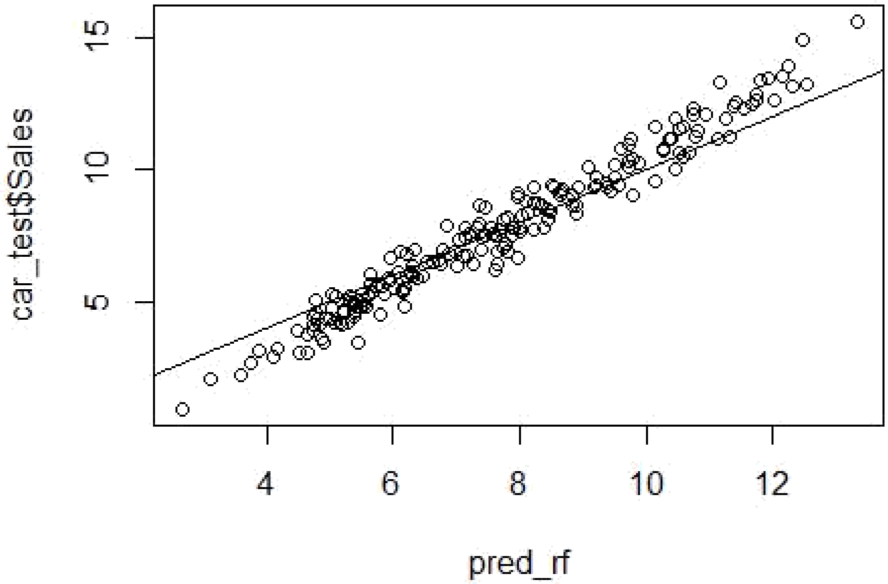
```
varImpPlot(bag_car)
```



Finding:

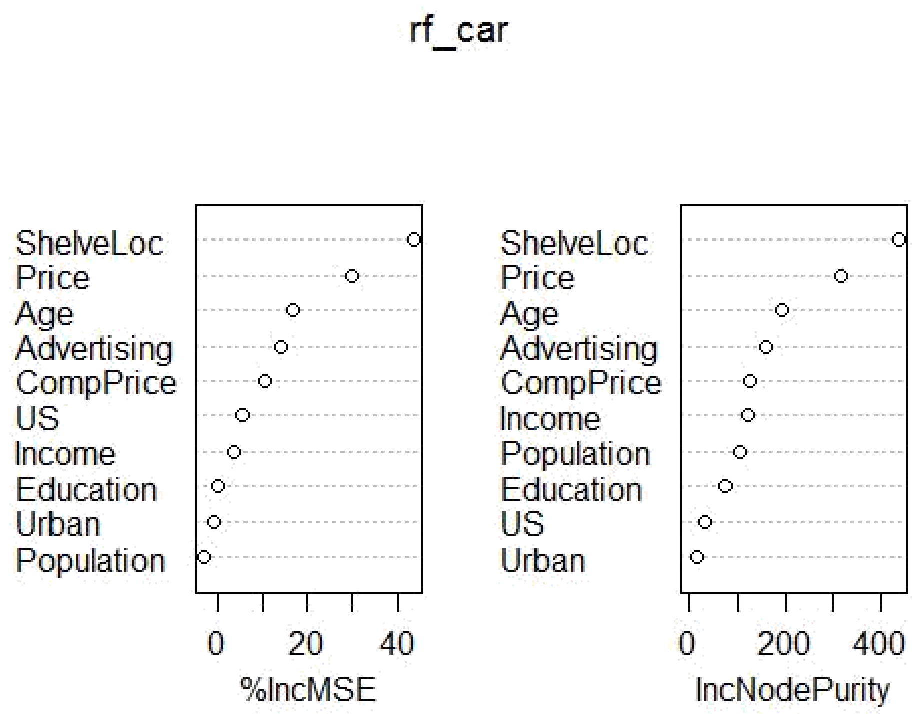
- With bagging, in sample MSE of 3.08 is obtained with 61.6% of the variability explained by the model and out of sample MSE drops to 3.02
- SheveLoc, Price, Age, CompPrice and Advertising are the most important variables.

```
#Question 8 e)
# Random Forest

set.seed (11)
rf_car=randomForest(Sales~.,data=car_test,                              importance=TRUE)
rf_car

##
## Call:
##  randomForest(formula = Sales ~ ., data = car_test, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 3.017619
##                     % Var explained: 64.15

pred_rf = predict(rf_car ,newdata =car_test)
plot(pred_rf , car_test$Sales)
abline (0,1)
```



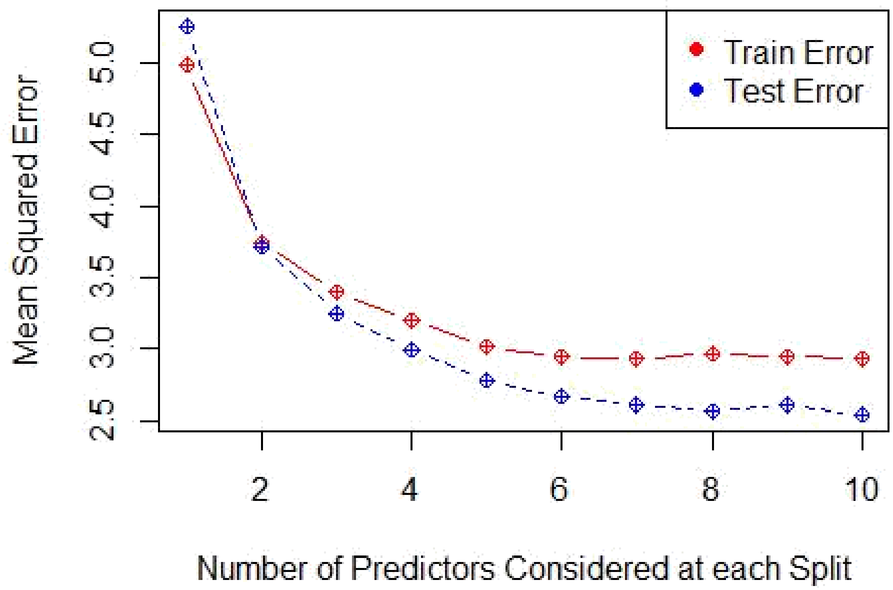
d) Use the bagging approach in order to analyze this data. What test MSE do you obtain? Use the importance() function to determine which variables are most important.

```
MSE_rf=mean(( pred_rf - car_test$Sales)^2)
MSE_rf

## [1] 0.6298638

#Importance :
importance(rf_car)

##                %IncMSE IncNodePurity
## CompPrice    10.3204054     126.67798
## Income        3.8874472     121.78778
## Advertising 14.0540453     161.59265
## Population   -2.7402971     104.74245
## Price        29.8009197     316.73983
## ShelveLoc    43.8285701     440.46815
## Age          16.9794139     192.04760
## Education     0.2229100      72.45231
## Urban        -0.8384124      15.06389
## US            5.4385267      32.51063

varImpPlot(rf_car)
```



e) Use random forests to analyze this data. What test MSE do you obtain? Use the importance() function to determine which variables are most important. Describe the effect of m, the number of variables considered at each split, on the error rate obtained.

```
# Effect of m :

rf_mse = list()
test_mse=list()


#mtry is no of Variables randomly chosen at each split
for(m in 1:10)
{
rf_car_iter=randomForest(Sales ~ . , data = car_train,mtry=m,ntree=500)
rf_mse[m] = rf_car_iter$mse[500]     #Stores the error of all Trees fitted

pred_rf<-predict(rf_car_iter,car_test)
test_mse[m]= with(car_test, mean( (Sales - pred_rf)^2))
}

matplot(1:m , cbind(rf_mse,test_mse), pch=10 , col=c("red","blue"),type="b",y
lab="Mean Squared Error",xlab="Number of Predictors Considered at each Split"
)
legend("topright",legend=c("Train Error","Test Error"),pch=19,col=c("red","bl
ue"))
```



Findings:

- The in sample MSE of 3.27 (with 59% variability explained) is obtained, which is still better than single tree MSE.However, randomforest MSE is higher than from that of obtained by bagging. Out of sample MSE of 3.16 is obtained with random forest model
- As m increases, MSE decreases. The rate of change of MSE decreases significantly post m=6
- SheveLoc, Price, Age, CompPrice and Advertising are the most important variables

==

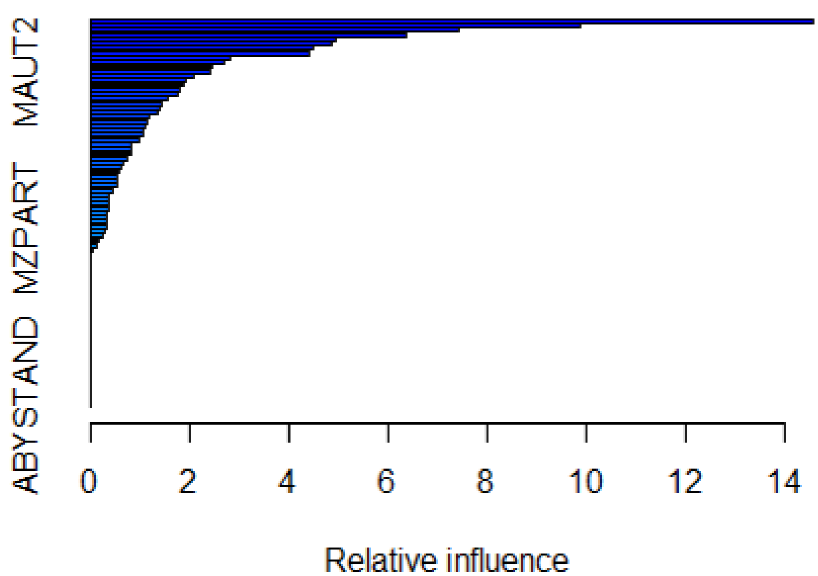
Chapter 8 : Question 11

a) Create a training set consisting of the first 1,000 observations, and a test set consisting of the remaining observations.

b) Fit a boosting model to the training set with Purchase as the response and the other variables as predictors. Use 1,000 trees, and a shrinkage value of 0.01. Which predictors appear to be the most important?

```
#Import the dataset
library(ISLR)
library(gbm)
library(InformationValue)
#ldetach(Caravan)
```

```
attach(Caravan)
#Data Set : Caravan
set.seed(11)
caravan <- Caravan
caravan$Purchase=as.numeric(ifelse(caravan$Purchase
=="Yes",1,0)) car_train1=caravan[1:1000,]
#car_train=subset(car_train, select = -c(PVRAAUT, AVRAAUT))
car_test1=caravan[1001:nrow(caravan),]
#car_test=subset(car_test, select = -c(PVRAAUT, AVRAAUT))


boost_car1 = gbm(Purchase~., data= car_train1, distribution = "bernoulli", n.
trees = 1000, shrinkage = 0.01)

## Warning in gbm.fit(x = x, y = y, offset = offset, distribution =
## distribution, : variable 50: PVRAAUT has no variation.

## Warning in gbm.fit(x = x, y = y, offset = offset, distribution =
## distribution, : variable 71: AVRAAUT has no variation.

summary(boost_car1)
```




```
##                   var    rel.inf
## PPERSAUT PPERSAUT 14.58283174
## MKOOPKLA MKOOPKLA  9.87705916
## MOPLHOOG MOPLHOOG  7.41838280
## MBERMIDD MBERMIDD  6.37552705
```

```
## PBRAND      PBRAND 4.96027724
## MINK3045 MINK3045 4.87910114
## MGODGE      MGODGE 4.47704761
## ABRAND      ABRAND 4.43276378
## MOSTYPE    MOSTYPE 2.81478144
## MSKA          MSKA 2.69812735
## MSKC          MSKC 2.45111416
## MAUT2        MAUT2 2.40889115
## MBERARBG MBERARBG  2.08673235
## PWAPART   PWAPART  1.93484921
## PBYSTAND PBYSTAND  1.87694576
## MAUT1        MAUT1 1.78866161
## MGODPR      MGODPR 1.77666285
## MINKGEM    MINKGEM 1.56051459
## MSKB1        MSKB1 1.42174319
## MRELGE      MRELGE 1.38718469
## MFWEKIND MFWEKIND  1.36232843
## MINKM30   MINKM30  1.19660950
## MGODOV     MGODOV  1.15894616
## MINK7512 MINK7512  1.11263053
## MBERHOOG MBERHOOG  1.05331863
## MRELOV      MRELOV 1.05145317
## MFGEKIND MFGEKIND  0.99069060
## MAUT0        MAUT0 0.84435821
## MGEMOMV    MGEMOMV 0.84119914
## MHKOOP      MHKOOP 0.80733338
## APERSAUT APERSAUT  0.72652051
## MGODRK      MGODRK 0.64892625
## MBERBOER MBERBOER  0.64006352
## MOPLLAAG MOPLLAAG  0.57410746
## MOPLMIDD MOPLMIDD  0.54634963
## MHHUUR     MHHUUR  0.54552412
## MZFONDS   MZFONDS  0.52058993
## MOSHOOFD MOSHOOFD  0.46281253
## PMOTSCO   PMOTSCO  0.39537232
## MINK123M MINK123M  0.37883319
## MSKB2       MSKB2  0.36155818
## MGEMLEEF MGEMLEEF  0.35730974
## MFALLEEN MFALLEEN  0.33207776
## MBERZELF MBERZELF  0.32808731
## MSKD          MSKD 0.32580552
## MZPART      MZPART 0.31777178
## MINK4575 MINK4575  0.30248329
## MBERARBO MBERARBO  0.25769188
## MRELSA      MRELSA 0.16753753
## MAANTHUI MAANTHUI 0.11840013
## PLEVEN      PLEVEN 0.06411078
## PWABEDR   PWABEDR  0.00000000
## PWALAND   PWALAND  0.00000000
## PBESAUT   PBESAUT  0.00000000
```

```
## PVRAAUT   PVRAAUT  0.00000000
## PAANHANG PAANHANG  0.00000000
## PTRACTOR PTRACTOR  0.00000000
## PWERKT       PWERKT 0.00000000
## PBROM         PBROM 0.00000000
## PPERSONG PPERSONG  0.00000000
## PGEZONG   PGEZONG  0.00000000
## PWAOREG   PWAOREG  0.00000000
## PZEILPL   PZEILPL  0.00000000
## PPLEZIER PPLEZIER  0.00000000
## PFIETS     PFIETS  0.00000000
## PINBOED   PINBOED  0.00000000
## AWAPART   AWAPART  0.00000000
## AWABEDR   AWABEDR  0.00000000
## AWALAND   AWALAND  0.00000000
## ABESAUT   ABESAUT  0.00000000
## AMOTSCO   AMOTSCO  0.00000000
## AVRAAUT   AVRAAUT  0.00000000
## AAANHANG AAANHANG  0.00000000
## ATRACTOR ATRACTOR  0.00000000
## AWERKT       AWERKT 0.00000000
## ABROM         ABROM 0.00000000
## ALEVEN       ALEVEN 0.00000000
## APERSONG APERSONG  0.00000000
## AGEZONG   AGEZONG  0.00000000
## AWAOREG   AWAOREG  0.00000000
## AZEILPL   AZEILPL  0.00000000
## APLEZIER APLEZIER  0.00000000
## AFIETS     AFIETS  0.00000000
## AINBOED   AINBOED  0.00000000
## ABYSTAND ABYSTAND  0.00000000
```

Finding:

- PPERSAUT ,MKOOPKLA and MOPLHOOG are observed to be the most important

```
#Boosting Predictions:

boost_pred_test = predict(boost_car1, car_test1,n.trees=1000,type="response",
shrinkage=0.01)
boost_mod_list=ifelse(boost_pred_test > 0.2 ,1,0)

#Create a confustion matrix with cutoff of 0.2 : this is wrong
#conf_mat_boost=confusionMatrix(car_test1$Purchase,boost_pred_test,threshold
=0.2)
table(car_test1$Purchase,boost_mod_list)

##    Confusion Matrix
```

```
##    Predicted
 ##        0    1
##   0 4408  125
##   1  258   31
```

```r
#conf_mat_boost

accuracy = (31+4408)/(125+258+4408+31)
```

Findings:

- Accuracy of Boosting : 92.05%
- Out of 289 customer predicted to make a purchase, 31 are actually making the purchase (10.7%)

c) Use the boosting model to predict the response on the test data. Predict that a person will make a purchase if the estimated probability of purchase is greater than 20 %.Form a confusion matrix. What fraction of the people predicted to make a purchase do in fact make one? How does this compare with the results obtained from applying KNN or logistic regression to this data set?

```r
#Logistic Regression
log_car=glm(Purchase~., data=car_train1,family=binomial)

summary(log_car)
```

```
##
## Call:
## glm(formula = Purchase ~ ., family = binomial, data = car_train1)
##
## Deviance Residuals:
 ##      Min       1Q   Median       3Q      Max
 ## -1.5422  -0.3307  -0.1710  -0.0766   3.3780
##
## Coefficients: (4 not defined because of singularities)
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   2.561e+02  4.912e+04   0.005   0.9958
## MOSTYPE      -1.814e-02  1.366e-01  -0.133   0.8944
## MAANTHUI      7.131e-02  4.419e-01   0.161   0.8718
## MGEMOMV      -9.298e-01  4.201e-01  -2.213   0.0269  *
## MGEMLEEF      4.187e-02  2.840e-01   0.147   0.8828
## MOSHOOFD      1.400e-01  6.127e-01   0.228   0.8193
## MGODRK       -7.230e-01  3.571e-01  -2.025   0.0429  *
## MGODPR       -3.201e-01  3.608e-01  -0.887   0.3750
## MGODOV       -6.499e-01  3.221e-01  -2.018   0.0436  *
## MGODGE       -2.773e-01  3.504e-01  -0.791   0.4287
## MRELGE        7.989e-01  4.116e-01   1.941   0.0522 .
## MRELSA        6.917e-01  3.877e-01   1.784   0.0744 .
```

```
## MRELOV        8.762e-01  4.205e-01   2.084  0.0372  *
## MFALLEEN     -3.702e-01  3.743e-01  -0.989  0.3227
## MFGEKIND     -2.986e-01  3.809e-01  -0.784  0.4331
## MFWEKIND     -5.345e-02  4.079e-01  -0.131  0.8957
## MOPLHOOG     -4.342e-01  3.924e-01  -1.106  0.2685
## MOPLMIDD     -6.936e-01  4.168e-01  -1.664  0.0961 .
## MOPLLAAG     -5.118e-01  4.168e-01  -1.228  0.2196
## MBERHOOG      1.080e-01  2.979e-01   0.363  0.7170
## MBERZELF     -2.737e-01  3.246e-01  -0.843  0.3991
## MBERBOER     -5.788e-01  3.757e-01  -1.541  0.1234
## MBERMIDD      4.193e-01  2.983e-01   1.406  0.1598
## MBERARBG      2.787e-01  2.847e-01   0.979  0.3276
## MBERARBO      3.511e-01  2.954e-01   1.188  0.2347
## MSKA          3.591e-01  2.859e-01   1.256  0.2092
## MSKB1         1.434e-01  2.839e-01   0.505  0.6134
## MSKB2         1.783e-01  2.516e-01   0.709  0.4785
## MSKC          1.093e-01  2.800e-01   0.390  0.6963
## MSKD         -3.869e-01  2.930e-01  -1.320  0.1867
## MHHUUR       -1.568e+01  3.329e+03  -0.005  0.9962
## MHKOOP       -1.561e+01  3.329e+03  -0.005  0.9963
## MAUT1         4.233e-01  4.097e-01   1.033  0.3015
## MAUT2         4.304e-01  3.733e-01   1.153  0.2489
## MAUT0         2.256e-01  3.742e-01   0.603  0.5466
## MZFONDS      -1.376e+01  4.325e+03  -0.003  0.9975
## MZPART       -1.377e+01  4.325e+03  -0.003  0.9975
## MINKM30       1.123e-01  2.957e-01   0.380  0.7041
## MINK3045      9.255e-02  2.820e-01   0.328  0.7428
## MINK4575      2.606e-01  2.932e-01   0.889  0.3740
## MINK7512      3.601e-01  3.083e-01   1.168  0.2427
## MINK123M     -1.407e-01  5.046e-01  -0.279  0.7803
## MINKGEM      -3.643e-01  2.777e-01  -1.312  0.1896
## MKOOPKLA      2.325e-01  1.340e-01   1.735  0.0828 .
## PWAPART       9.343e-01  9.813e-01   0.952  0.3411
## PWABEDR      -9.056e-01  4.221e+03   0.000  0.9998
## PWALAND      -1.752e+01  3.513e+03  -0.005  0.9960
## PPERSAUT      3.757e-01  1.473e-01   2.550  0.0108  *
## PBESAUT      -3.792e+01  1.332e+04  -0.003  0.9977
## PMOTSCO       2.230e-01  2.466e-01   0.904  0.3659
## PVRAAUT              NA         NA      NA      NA
## PAANHANG      1.573e+01  6.805e+03   0.002  0.9982
## PTRACTOR     -3.326e-01  1.953e+03   0.000  0.9999
## PWERKT        1.676e+01  6.051e+03   0.003  0.9978
## PBROM        -1.094e-01  1.334e+00  -0.082  0.9346
## PLEVEN        2.405e-01  7.074e-01   0.340  0.7339
## PPERSONG      1.269e+00  5.719e+03   0.000  0.9998
## PGEZONG       2.000e+01  3.768e+03   0.005  0.9958
## PWAOREG       1.452e+01  3.099e+03   0.005  0.9963
## PBRAND        3.369e-02  2.188e-01   0.154  0.8776
## PZEILPL       3.738e+01  1.205e+04   0.003  0.9975
## PPLEZIER      6.371e-01  6.421e-01   0.992  0.3211
```

```
## PFIETS        2.101e+01  5.469e+03   0.004  0.9969
## PINBOED       5.699e-01  3.915e+03   0.000  0.9999
## PBYSTAND      4.167e-01  9.550e-01   0.436  0.6626
## AWAPART      -1.475e+00  1.938e+00  -0.761  0.4468
## AWABEDR      -1.417e+01  8.880e+03  -0.002  0.9987
## AWALAND       5.416e+01  1.054e+04   0.005  0.9959
## APERSAUT     -6.572e-01  7.374e-01  -0.891  0.3728
## ABESAUT       1.730e+02  7.554e+04   0.002  0.9982
## AMOTSCO      -1.143e-01  5.805e-01  -0.197  0.8439
## AVRAAUT              NA         NA      NA      NA
## AAANHANG     -3.099e+01  1.361e+04  -0.002  0.9982
## ATRACTOR     -1.545e+01  6.899e+03  -0.002  0.9982
## AWERKT               NA         NA      NA      NA
## ABROM        -2.511e-01  4.265e+00  -0.059  0.9531
## ALEVEN       -8.600e-01  1.485e+00  -0.579  0.5625
## APERSONG     -1.879e+01  1.328e+04  -0.001  0.9989
## AGEZONG      -5.763e+01  1.130e+04  -0.005  0.9959
## AWAOREG      -3.190e+01  1.360e+04  -0.002  0.9981
## ABRAND        5.725e-01  6.981e-01   0.820  0.4122
## AZEILPL              NA         NA      NA      NA
## APLEZIER      1.083e+00  1.712e+00   0.633  0.5268
## AFIETS       -1.929e+01  5.469e+03  -0.004  0.9972
## AINBOED      -1.979e+01  8.686e+03  -0.002  0.9982
## ABYSTAND     -5.835e-03  3.395e+00  -0.002  0.9986
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 448.41  on 999  degrees of freedom
## Residual deviance: 320.64  on 918  degrees of freedom
## AIC: 484.64
##
## Number of Fisher Scoring iterations: 18
```

```
# predict for the entire data set
pred_cars1 = predict(log_car, car_test1, type="response")

#Decide on optimal cut-off
library(InformationValue)
#create confusion matrix:
log_pred_cars=ifelse(pred_cars1 > 0.2 ,1,0)

#conf_mat_car=confusionMatrix(car_test1$Purchase, pred_cars1, threshold = 0.2
)
#conf_mat_car

table(car_test1$Purchase, log_pred_cars)
```

```
##    Confusion Matrix:
```

```
     Predicted
 ##         0    1
##   0 4183  350
##   1  231   58
```

Findings:

- Accuracy with Logistic Regression = 87.95%
- Out of 289 people who have been predicted to make a purchase 58 actually made the purchase. (20%)
- Prediction accuracy is more with boosting (accuracy = 92.05%) compared to that of logistic regression.

==

Problem 1: Beauty Pays!

Using the data, estimate the effect of "beauty" into course ratings. Make sure to think about the potential many "other determinants". Describe your analysis and your conclusions.

```
#Import the data
beauty_data=read.csv('C:/Users/jayant
raisinghani/Documents/R/BeautyData.csv' )

#Check the effect of beauty on into course ratings along with other
deteminan ts

beau_eff=lm(CourseEvals~BeautyScore+female+lower+nonenglish+tenuretrack,data=
beauty_data)

summary(beau_eff)

##
## Call:
## lm(formula = CourseEvals ~ BeautyScore + female + lower + nonenglish +
##     tenuretrack, data = beauty_data)
##
## Residuals:
##      Min       1Q   Median      3Q      Max
## -1.31385 -0.30202  0.01011  0.29815  1.04929
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.06542    0.05145  79.020  < 2e-16 ***
## BeautyScore  0.30415    0.02543  11.959  < 2e-16 ***
## female      -0.33199    0.04075  -8.146 3.62e-15 ***
## lower       -0.34255    0.04282  -7.999 1.04e-14 ***
```

```
## nonenglish  -0.258080.08478  -3.044  0.00247 **
## tenuretrack -0.099450.04888  -2.035  0.04245 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4273 on 457 degrees of freedom
## Multiple R-squared:  0.3471, Adjusted R-squared:  0.3399
## F-statistic: 48.58 on 5 and 457 DF,  p-value: < 2.2e-16
```

```
#Check the effect of beauty on into course ratings along without other
detemi nants

beau_eff2=lm(CourseEvals~BeautyScore,data=beauty_data)

summary(beau_eff2)
```

```
##
## Call:
## lm(formula = CourseEvals ~ BeautyScore, data = beauty_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5936 -0.3346  0.0097  0.3702  1.2321
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.71340    0.02249 165.119  <2e-16 ***
## BeautyScore  0.27148    0.02837   9.569  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4809 on 461 degrees of freedom
## Multiple R-squared:  0.1657, Adjusted R-squared:  0.1639
## F-statistic: 91.57 on 1 and 461 DF,  p-value: < 2.2e-16
```

Findings:

- When other factors are considered, 1 unit increase in beauty score would have 0.304 units change in course scores
- The model has an standard error of 0.42 and the adjusted R-squared of 0.34. That means 34% of the variation in the course ratings is explained by beauty score and other determinant factors
- When checked for the impact of beauty score only, the standard error of the model is 0.4809 and adjusted R-squared = 0.1639. This indicates that the inclusion of other determinants is improving the predictibility of the model

Q.2 : What does "Disentangling whether this outcome represents productivity or discrimination is, as with the issue generally, probably impossible" mean?

Ans : -

- From the regression model, it is evident that there is a linear relationship between beauty score and course evaluation scores of the instructors. However, we can not infer that this relationship is due to productivity or distrimination. The correlation might not be the causation for productivity / discrimination. An instructor could be very confident and knowledgable, which in turn fetches him more salary than some other under-confident instructor. Hence, beauty being strong dirving factor for course evaluation can not be inferred as a person's productivity driver or a factor of discrimination

- Also, the R- square of 35% indicates that there are other factors which might affect the evalution score apart from beauty. These factors could be related to productivity of the instructor.Hence disentangling whether the outcome indicates productivity or discrimination is not possible according to the professor

===

Problem 2 : Midcity

```
#Import the data set;

mid_city=read.csv('MidCity.csv')

mid_city['old']=ifelse(mid_city$Nbhd < 3,1,0)

mid_city$Nbhd = as.factor(mid_city$Nbhd)
mid_city$Brick = as.factor(mid_city$Brick)
mid_city$old = as.factor(mid_city$old)


#Regress with y= price :
hm_price=lm(Price~Offers+SqFt+Bedrooms+Bathrooms+Brick+Nbhd,data=mid_city)

summary(hm_price)

##
## Call:
## lm(formula = Price ~ Offers + SqFt + Bedrooms + Bathrooms + Brick +
##     Nbhd, data = mid_city)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -27337.3  -6549.5-41.7   5803.4  27359.3
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2159.498   8877.810   0.243  0.80823
```

```
## Offers       -8267.488   1084.777 -7.621 6.47e-12 ***
## SqFt            52.994      5.734   9.242 1.10e-15 ***
## Bedrooms      4246.794   1597.911   2.658  0.00894 **
## Bathrooms     7883.278   2117.035   3.724  0.00030 ***
## BrickYes     17297.350   1981.616   8.729 1.78e-14 ***
## Nbhd2        -1560.579   2396.765  -0.651  0.51621
## Nbhd3        20681.037   3148.954   6.568 1.38e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10020 on 120 degrees of freedom
## Multiple R-squared:  0.8686, Adjusted R-squared:  0.861
## F-statistic: 113.3 on 7 and 120 DF,  p-value: < 2.2e-16
```

1. Is there a premium for brick houses everything else being equal?

Ans-

Coefficient of Brick is positive. This indicates that there is a premium for brick houses everything else being equal.

2. Is there a premium for houses in neighborhood 3?

Ans-

Yes. The Coefficients of neighborhood 3 with respect to neighborhood 1 is positive and it is greater than the coefficient of neighborhood 2. This indicates that there is a premium for houses in neighborhood 3.

3. Is there an extra premium for brick houses in neighborhood 3?

Interaction in brick and neighborhood 3

```
hm_price_inter=lm(Price~Offers+SqFt+Bedrooms+Bathrooms+Brick+Nbhd+Brick*Nbhd-
Home,data=mid_city)

summary(hm_price_inter)
```

```
##
## Call:
## lm(formula = Price ~ Offers + SqFt + Bedrooms + Bathrooms + Brick +
##     Nbhd + Brick * Nbhd - Home, data = mid_city)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -27225.1  -5219.0   -273.7   4297.4 27507.2
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      3695.511   8829.382   0.419  0.67631
## Offers          -8381.770   1068.248  -7.846 2.15e-12 ***
## SqFt               53.745      5.686   9.453 3.96e-16 ***
```

```
## Bedrooms         4777.216   1586.397   3.011  0.00318 **
## Bathrooms        6457.287   2160.867   2.988  0.00341 **
## BrickYes        12093.056   4082.168   2.962  0.00369 **
## Nbhd2           -1317.656   2679.849  -0.492  0.62385
## Nbhd3           16980.797   3437.529   4.940 2.60e-06 ***
## BrickYes:Nbhd2   2668.449   5068.893   0.526  0.59957
 ## BrickYes:Nbhd3 11933.197   5341.027   2.234  0.02735 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9847 on 118 degrees of freedom
## Multiple R-squared:  0.8752, Adjusted R-squared:  0.8657
## F-statistic: 91.94 on 9 and 118 DF,  p-value: < 2.2e-16
```

Ans-

The interaction of brick and neighborhood 3 is significant (with 95% confidence interval) and it's coefficient is positive (11933), indicating that the premium has to be paid for brick houses in neighborhood 3.

4. For the purposes of prediction could you combine the neighborhoods 1 and 2 into a single "older" neighborhood?

```
#Train and test :
set.seed(1)
train_prc=sample(1: nrow(mid_city), nrow(mid_city)*0.8)
prc_train=mid_city[train_prc ,] prc_test=mid_city[-
train_prc,]

price_reg2=lm(Price~Offers+SqFt+Bedrooms+Bathrooms+Brick+old,data=prc_train)

summary(price_reg2)

##
## Call:
## lm(formula = Price ~ Offers + SqFt + Bedrooms + Bathrooms + Brick +
##     old, data = prc_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -27246.3  -7908.8   -519.4   6919.8  26274.2
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  23231.43   10842.05   2.143  0.03469 *
## Offers       -8078.79    1156.22  -6.987 3.83e-10 ***
## SqFt            53.39       6.30   8.475  2.98e-13 ***
## Bedrooms      4031.36    1920.59   2.099  0.03847 *
## Bathrooms     8143.19    2623.06   3.104  0.00251 **
## BrickYes     15303.21    2320.09   6.596  2.38e-09 ***
## old1        -22834.16    2959.01  -7.717 1.18e-11 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10590 on 95 degrees of freedom
## Multiple R-squared:  0.8634, Adjusted R-squared:  0.8548
## F-statistic: 100.1 on 6 and 95 DF,  p-value: < 2.2e-16
```

```
## Prediction :
pred_beau=predict(price_reg2,prc_test)

pred_mse=sqrt(mean((pred_beau-prc_test$Price)^2))
pred_mse
```

```
## [1] 7730.464
```

Ans -

- The prediction error with combined neighborhoods, is 11,313 Old house prices are \$21,938 lesser than that of new houses when everything else is same.
- In sample error decreases as the neighbourhood 1 and 2 are combined as old neighborhood

===

Problem 3 : What Causes what ?

3.1 Why can't I just get data from a few different cities and run the regression of "Crime" on "Police" to understand how more cops in the streets affect crime? ("Crime" refers to some measure of crime rate and "Police" measures the number of cops in a city)

Ans-
There are 2 problems with taking crime data from a few cities and understanding how more cops in the street affect crime rate:

1. There could be other factors, which will not be considered while taking only crime rate and number of police data. The podcast talks about one such factor as number of victims on the road, when there was medium to high terror alert and the researchers found out no change in ridership in metros. Such underlying factors would be missed out while considering crime and police data solely.

2. The behavior does vary from a city to another city. For places like D.C. there might be a negative correlation of number of police to the crime rate. For some other cities, where crime rate is not so high, increasing the number of police might not make any difference. Hence, generalizing the effect for all the cities based on a small set of biased cities would be inappropriate

Question 3.2: How were the researchers from UPENN able to isolate this effect? Briefly describe their approach and discuss their result in the “Table 2” below

Ans -

Researchers from UPENN, tried to find out the examples where there is increase in number of police for the reasons unrelated to crime rate. They found out one such case of terrorism alert, as by law, the police force is increased in case of high terrorist alert. They found out that crime rate decreases on the days when there is a high terrorist alert in DC. Table 2 showcases negative coefficient of High Alert indicating, negative correlation with significance level of 5%.

The researchers also hypothesized about the number of victims available on the road on such days, when the crime rate was higher in the city. They found out that there was no dip in people’s ridership on such days. Indicating that the crime rate increases as the ridership for metro increases

Question 3.3 :

Why did they have to control for METRO ridership? What was that trying to capture?

Ans:

The researchers had the hypothesis that the number of tourists or people on street were less on the high terrorist alert days (as people might want to stay at home during the high alert to be safe) and hence they tried to check that hypothesis. They took METRO ridership as the measure of people on the street. They found out that there was no dip in ridership on such days and hence they were able to control for the metro ridership in the city

Question 3.4:

In the next page, I am showing you “Table 4” from the research paper. Just focus on the first column of the table. Can you describe the model being estimated here? What is the conclusion?

Ans:

The model considers the situation in district 1 vs in other districts. It checks the interaction between district 1 and high terrorist alert along with ridership.

Conclusion of the model :- In district 1 as the terrorist alert increases, the crime rate decreases. This is not significantly observed in other districts. However, ridership is still 5% significant while predicting crime rate.

===

Problem 4 : BART

Apply BART to caifornia data set . Does BART outperform RF or Boosting?

```r
bart_data=read.csv('CAhousing.csv')

bart_data$medianHouseValue = log(bart_data$medianHouseValue)
x=bart_data[,1:8]
y=bart_data$medianHouseValue

rows=nrow(bart_data)

samp = sample(1:rows,floor(.75*rows))

bart_train1=bart_data[samp,]
bart_test=bart_data[-samp,]


xtrain=x[samp,]; ytrain=y[samp] # training
data xtest=x[-samp,]; ytest=y[-samp]

#Divide into training and testing

library(BART)

## Loading required package: nlme

## Loading required package: nnet

## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

set.seed(99)
nd=200 # number of kept draws
burn=50 # number of burn in draws
bart_cal = wbart(x,y,nskip=burn,ndpost=nd)
```

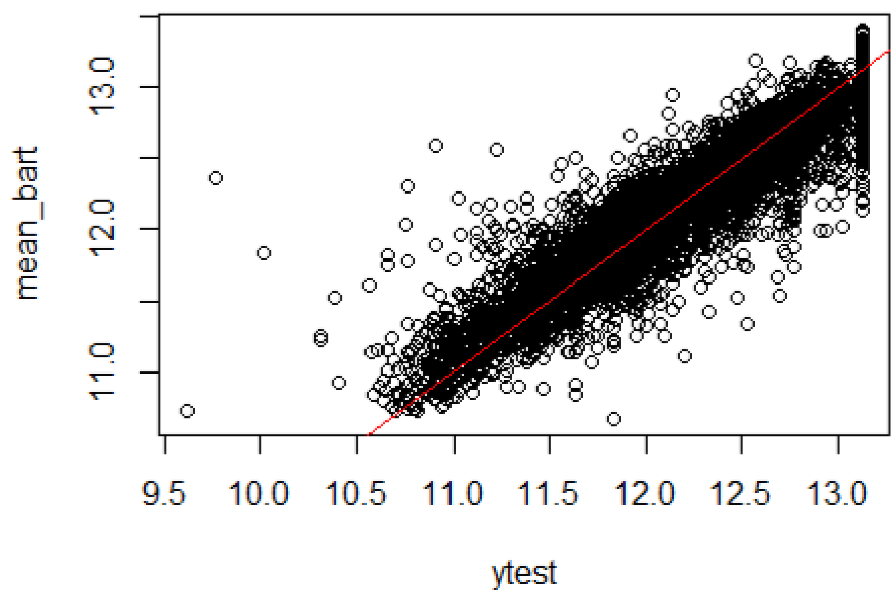
```
## *****Into main of wbart
## *****Data:
## data:n,p,np: 20640, 8, 0
## y1,yn: 0.937880, -0.684008
## x1,x[n*p]: -122.230000, 2.388600
## *****Number of Trees: 200
## *****Number of Cut Points: 100 ... 100
## *****burn and ndpost: 50, 200
## *****Prior:beta,alpha,tau,nu,lambda: 2.000000,0.950000,0.061989,3.000000,0
.022524
```

```
## *****sigma: 0.340045
## *****w (weights): 1.000000 ... 1.000000
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,8,0
## *****nkeeptrain,nkeeptest,nkeeptestme,nkeeptreedraws: 200,200,200,200
## *****printevery: 100
## *****skiptr,skipte,skipteme,skiptreedraws: 1,1,1,1
##
## MCMC
## done 0 (out of 250)
## done 100 (out of 250)
## done 200 (out of 250)
## time: 43s
## check counts
## trcnt,tecnt,temecnt,treedrawscnt: 200,0,0,200
```

```r
#
bart_train = wbart(xtrain,ytrain)
```

```
## *****Into main of wbart
## *****Data:
## data:n,p,np: 15480, 8, 0
## y1,yn: 0.681556, -0.330615
## x1,x[n*p]: -122.510000, 4.225000
## *****Number of Trees: 200
## *****Number of Cut Points: 100 ... 100
## *****burn and ndpost: 100, 1000
## *****Prior:beta,alpha,tau,nu,lambda: 2.000000,0.950000,0.061989,3.000000,0
.022337
## *****sigma: 0.338630
## *****w (weights): 1.000000 ... 1.000000
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,8,0
## *****nkeeptrain,nkeeptest,nkeeptestme,nkeeptreedraws: 1000,1000,1000,1000
## *****printevery: 100
## *****skiptr,skipte,skipteme,skiptreedraws: 1,1,1,1
##
## MCMC
## done 0 (out of 1100)
## done 100 (out of 1100)
## done 200 (out of 1100)
## done 300 (out of 1100)
## done 400 (out of 1100)
## done 500 (out of 1100)
## done 600 (out of 1100)
## done 700 (out of 1100)
## done 800 (out of 1100)
## done 900 (out of 1100)
## done 1000 (out of 1100)
## time: 171s
## check counts
## trcnt,tecnt,temecnt,treedrawscnt: 1000,0,0,1000
```

```
pred_bart = predict(bart_train,as.matrix(xtest))

## *****In main of C++ for bart prediction
## tc (threadcount): 1
## number of bart draws: 1000
## number of trees in bart sum: 200
## number of x columns: 8
## from x,np,p: 8, 5160
## ***using serial code

mse_bart=mean((pred_bart-ytest)^2)

mse_bart

## [1] 0.5962976

mean_bart = apply(pred_bart,2,mean)

plot(ytest,mean_bart)
abline(0,1,col=2)
```

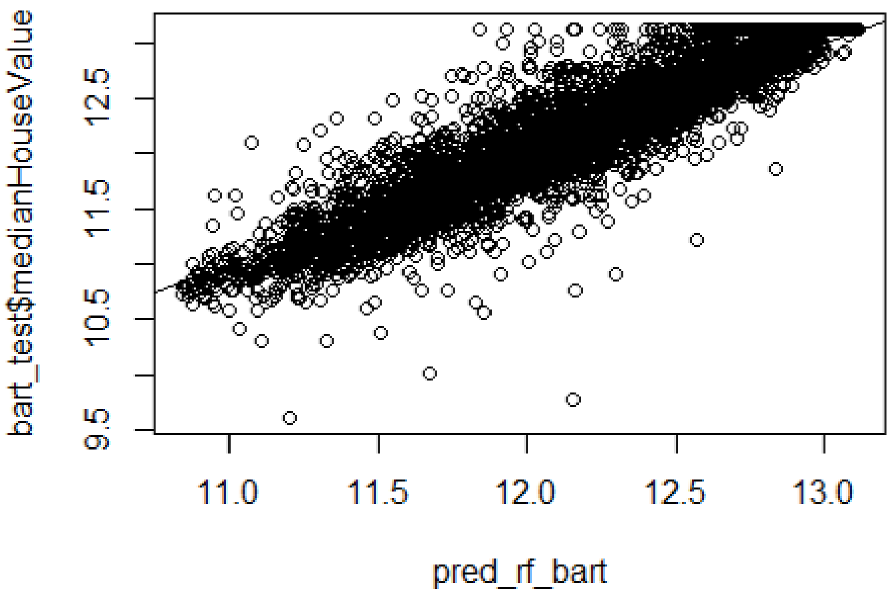



```
#BART compared to RF and Boosting


set.seed (11)
library(randomForest)
```

```
rf_bart=randomForest(medianHouseValue~.,data=bart_train1, importance=TRUE) rf_bart
```

```
##
## Call:
##  randomForest(formula = medianHouseValue ~ ., data = bart_train1,          imp
ortance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 0.06029681
##                     % Var explained: 81.33
```

```
pred_rf_bart = predict(rf_bart ,newdata =bart_test)
plot(pred_rf_bart , bart_test$medianHouseValue)
abline (0,1)
```




```
MSE_rf_bart=mean(( pred_rf_bart - bart_test$medianHouseValue)^2)
```

Finding :

- The RMSE with BART is 0.58 and that of with random forest is 0.53 . This indicates that BART is not outperforming random forest model. Standard error is slightly lower for random forest and hence the accuracy is better than BART

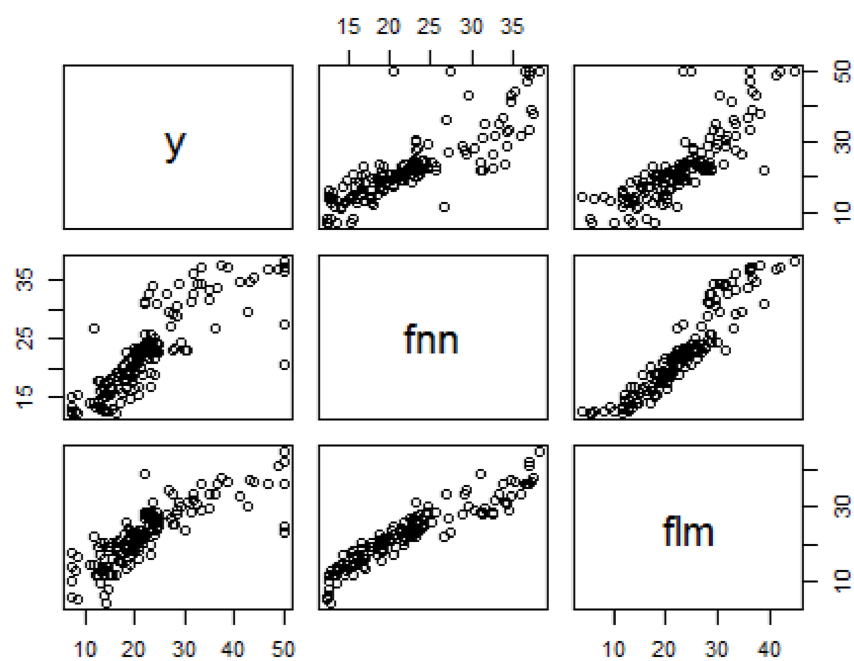
==

Problem 5 : Neural Network

```
library(MASS)  # library for Boston data set
###standardize the x's
minv = rep(0,3)
maxv = rep(0, 3)
bstn = Boston

##Train and test data
samp1 = sample(1:nrow(bstn),floor(0.70*nrow(bstn)))

# standardization of the
data for(i in 1:3)
{
minv[i]  =  min(Boston[[i]])
maxv[i] = max(Boston[[i]])
bstn[[i]] = (Boston[[i]]-minv[i])/(maxv[i]-minv[i])
}

train_bstn=bstn[samp1,]
test_bstn=bstn[-samp1,]

### nn library
library(nnet)

###fit nn with just one x=food
set.seed(111)
bstn_nn = nnet(medv~.,train_bstn,size=3,decay=.1,linout=T)

## # weights:  46
## initial  value 207901.794096
## iter  10 value 29304.185678
## iter  20 value 23206.317300
## iter  30 value 20059.142591
## iter  40 value 15994.745450
## iter  50 value 14698.697535
## iter  60 value 12774.760369
## iter  70 value 11117.362168
## iter  80 value 9666.678498
## iter  90 value 8373.423593
## iter 100 value 7177.340985
## final  value 7177.340985
## stopped after 100 iterations

summary(bstn_nn)

## a 13-3-1 network with 46 weights
## options were - linear output units  decay=0.1
##   b->h1  i1->h1  i2->h1  i3->h1  i4->h1  i5->h1  i6->h1  i7->h1  i8->h1
```

```
##    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.03   0.00
##  i9->h1 i10->h1 i11->h1 i12->h1 i13->h1
##    0.00    0.01    0.01    0.28    0.00
##   b->h2  i1->h2  i2->h2  i3->h2  i4->h2  i5->h2  i6->h2  i7->h2 i8->h2
##   -0.31   -7.44   -0.78    0.46    0.86   -4.61    0.75    0.00  -0.06
##  i9->h2 i10->h2 i11->h2 i12->h2 i13->h2
##    0.05    0.00   -0.13    0.00   -0.07
##   b->h3  i1->h3  i2->h3  i3->h3  i4->h3  i5->h3  i6->h3  i7->h3 i8->h3
##   -2.96   -1.07    2.88   14.71   11.14  -13.87    0.44    0.78 -14.87
##  i9->h3 i10->h3 i11->h3 i12->h3 i13->h3
##   -7.58    0.14   -9.63    0.88  -23.13
##   b->o  h1->o  h2->o  h3->o
##   5.45   6.46  20.28   7.35
```

```
###get fits, print summary, and plot fit
fznn = predict(bstn_nn,test_bstn)

zlm = lm(medv~.,train_bstn)
fzlm = predict(zlm,test_bstn)
temp = data.frame(y=test_bstn$medv,fnn=fznn,flm=fzlm)
pairs(temp)
```




```
print("Correlation matrix for linear and NN with y")

## [1] "Correlation matrix for linear and NN with y"

print(cor(temp))
```

```
##              y       fnn       flm
## y   1.0000000 0.8314786 0.8209151
## fnn 0.8314786 1.0000000 0.9335533
## flm 0.8209151 0.9335533 1.0000000

rmse_nn=sqrt(mean(fznn-test_bstn$medv)^2)
print(paste("RMSE when size=3 and decay=0.1:",rmse_nn))

## [1] "RMSE when size=3 and decay=0.1: 0.237389792390011"

################################################
## Size and Decay

### try four different fits

set.seed(14)
znn1 = nnet(medv~.,train_bstn,size=3,decay=.5,linout=T)

## # weights:  46
## initial  value 212870.500333
## iter  10 value 31466.534586
## iter  20 value 23066.321247
## iter  30 value 22243.195656
## iter  40 value 21787.840473
## iter  50 value 16182.315496
## iter  60 value 14429.770039
## iter  70 value 13431.429924
## iter  80 value 12376.135196
## iter  90 value 11107.427717
## iter 100 value 10241.220195
## final  value 10241.220195
## stopped after 100 iterations

fznn1 = predict(znn1,test_bstn)
rmse_nn1=sqrt(mean(fznn1-test_bstn$medv)^2)
print(paste("RMSE when size=3 and decay=0.5: ",rmse_nn1))

## [1] "RMSE when size=3 and decay=0.5: 0.652871439655054"

znn2 = nnet(medv~.,train_bstn,size=3,decay=.00001,linout=T)

## # weights:  46
## initial  value 227921.200785
## final  value 29269.075276
## converged

fznn2 = predict(znn2,test_bstn)
rmse_nn2=sqrt(mean(fznn2-test_bstn$medv)^2)
print(paste("RMSE when size=3 and decay=0.00001:",rmse_nn2))

## [1] "RMSE when size=3 and decay=0.00001: 0.167260608803289"
```

```r
znn3 = nnet(medv~.,train_bstn,size=50,decay=.5,linout=T)
```

```
## # weights:  751
## initial  value 228249.434762
## iter  10 value 22962.521224
## iter  20 value 22057.985074
## iter  30 value 21127.198220
## iter  40 value 19471.165796
## iter  50 value 14013.039379
## iter  60 value 12129.262918
## iter  70 value 9832.775234
## iter  80 value 8129.469924
## iter  90 value 7737.992199
## iter 100 value 7491.822877
## final  value 7491.822877
## stopped after 100 iterations
```

```r
fznn3 = predict(znn3,test_bstn)
rmse_nn3=sqrt(mean(fznn3-test_bstn$medv)^2)
print(paste("RMSE when size=50 and decay=0.5:",rmse_nn3))
```

```
## [1] "RMSE when size=50 and decay=0.5: 0.190946335231213"
```

```r
znn4 = nnet(medv~.,train_bstn,size=50,decay=.00001,linout=T)
```

```
## # weights:  751
## initial  value 185747.774239
## iter  10 value 24232.945973
## iter  20 value 20758.975024
## iter  30 value 20285.679790
## iter  40 value 19556.055372
## iter  50 value 19429.651520
## iter  60 value 13580.095240
## iter  70 value 9535.212948
## iter  80 value 8875.128220
## iter  90 value 8709.222701
## iter 100 value 8629.425440
## final  value 8629.425440
## stopped after 100 iterations
```

```r
fznn4 = predict(znn4,test_bstn)
rmse_nn4=sqrt(mean(fznn4-test_bstn$medv)^2)
print(paste("RMSE when size=50 and decay=0.00001:",rmse_nn4))
```

```
## [1] "RMSE when size=50 and decay=0.00001:
```

0.126733413410111" RMSE Comparison:

RMSE when size=3 and decay=0.1: 0.237389792390011

RMSE when size=3 and decay=0.5: 0.652871439655054

RMSE when size=3 and decay=0.00001: 0.167260608803289

```
RMSE when size=50 and decay=0.5: 0.190946335231213

RMSE when size=50 and decay=0.00001: 0.126733413410111
```

Findings:

- As the size of neural network increases, the in test prediction reduces.
- As the decay factor decreases, the error in test prediction increases