BSc Coursework 1

Salik Tariq / Student ID: 12516369

1) Statistical learning methods

For each of parts (a) through (d), indicate whether we would generally expect the performance of a flexible

statistical learning method to be better or worse than an inflexible method.

(a) The number of predictors p is extremely large, and the number of observations n is small.

Worse - If the number of predictors p is extremely large and the observations n is small then we will not have enough information about the effect and variation of each parameters considered, therefore flexible method will overfit the model due to small number of observations n.

(b) The sample size n is extremely large, and the number of predictors p is small.

Better - If the sample size n is extremely large, and the number of predictors p is small then we will have enough information about most predictors, so a flexible method would perform better.

(c) The relationship between the predictors and response is highly non-linear.

Better - a more flexible method will fit the data better due to the degree of freedom it provides, therefore felxible method would perform better.

(d) The standard deviation of the error terms, i.e. $\sigma = sd(\epsilon)$, is extremely high.

Worse - The flexible statistical learning method will incorporate the noise in error terms and thus will increase the SD of error and hence the model fitting would be poor in case of flexibel method.

2) Bayes' rule

Given a dataset including 20 samples (S__1, . . . , S__20) about the temperature (i.e. hot or cool) for playing golf

(i.e. yes or no), you are required to use the Bayes' rule to calculate the probability of playing golf according

to the temperature, i.e. P(Play Golf | Temperature).

P(Play Golf = Yes) = 10/20 = 0.5 P(Play Gold = No) = 10/20 = 0.5 P(Temperature = Hot) = 12/20 = 0.6 P(Temperature = Cool) = 8/20 = 0.4

P(Temperature = Hot | Play Golf = Yes) = 5/12 = 0.4166 P(Temperature = Hot | Play Golf = No) = 7/12 = 0.5834 P(Temperature = Cool | Play Golf = Yes) = 5/8 = 0.625 P(Temperature = Cool | Play Golf = No) = 3/8 = 0.375

P(Play Golf = Yes | Temperature = Hot) = (0.4166 * 0.5)/0.6 = 0.347 P(Play Golf = No | Temperature = Hot) = (0.5834 * 0.5)/0.6 = 0.486 P(Play Golf = Yes | Temperature = Cool) = (0.625 * 0.5)/0.4 = 0.78125 P(Play Golf = No | Temperature = Cool) = (0.375 * 0.5)/0.4 = 0.46875

3) Descriptive analysis

This exercise involves the Auto data set studied in the class.

```
#install.packages("ISLR")
library(ISLR) #this library contains Auto dataset
```

(a) Which of the predictors are quantitative, and which are qualitative?

```
str(Auto)
```

```
## 'data.frame':    392 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 14 161 141 54 223 241 1
```

Based on the output: Quantitative predictors are: mpg, cylinders, displacement, horsepower, weight, acceleration, year and origin. Qualitative predictors are: name

(b) What is the range of each quantitative predictor? You can answer this using the range() function.

```
range(Auto$mpg)
```

```
## [1]  9.0 46.6
```

Range for mpg is 9.0 to 46.6

```
range(Auto$cylinders)
```

```
## [1] 3 8
```

Range for cylinders is from 3 to 8

```
range(Auto$displacement)
```

```
## [1]  68 455
```

Range for displacement is from 68 to 455

```
range(Auto$horsepower)
```

```
## [1]  46 230
```

Range for horsepower is from 46 to 230

```
range(Auto$weight)
```

```
## [1] 1613 5140
```

Range for weight is from 1613 to 5140

```
range(Auto$acceleration)
```

```
## [1]  8.0 24.8
```

Range for acceleration is from 8.0 to 24.8

```
range(Auto$year)
```

```
## [1] 70 82
```

Range for year is from 70 to 82

```
range(Auto$origin)
```

```
## [1] 1 3
```

Range for origin is from 1 to 3 ### (c) What is the median and variance of each quantitative predictor?

```
mean(Auto$mpg)
```

```
## [1] 23.44592
```

```
var(Auto$mpg)
```

```
## [1] 60.91814
```

mean is 23.44592 and variance is 60.91814

```
mean(Auto$cylinders)
```

```
## [1] 5.471939
```

```
var(Auto$cylinders)
```

```
## [1] 2.909696
```

mean is 5.471939 and variance is 2.909696

```
mean(Auto$displacement)
```

```
## [1] 194.412
```

```
var(Auto$displacement)
```

```
## [1] 10950.37
```

mean is 194.412 and variance is 10950.37

```
mean(Auto$horsepower)
```

```
## [1] 104.4694
```

```
var(Auto$horsepower)
```

```
## [1] 1481.569
```

mean is 104.4694 and variance is 1481.569

```
mean(Auto$weight)
```

```
## [1] 2977.584
```

```
var(Auto$weight)
```

```
## [1] 721484.7
```

mean is 2977.584 and variance is 721484.7

```r
mean(Auto$acceleration)
```

```
## [1] 15.54133
```

```r
var(Auto$acceleration)
```

```
## [1] 7.611331
```

mean is 15.54133 and variance is 7.611331

```r
mean(Auto$year)
```

```
## [1] 75.97959
```

```r
var(Auto$year)
```

```
## [1] 13.56991
```

mean is 75.97959 and variance is 13.56991

```r
mean(Auto$origin)
```

```
## [1] 1.576531
```

```r
var(Auto$origin)
```

```
## [1] 0.6488595
```

mean is 1.576531 and variance is 0.6488595

(d) Now remove the 11th through 79th observations (inclusive) in the dataset. What is the range, median,

and variance of each predictor in the subset of the data that remains?

```r
Auto2 <- Auto[-c(11:79),] #removing rows 11th to 79th inclusive

apply(Auto2[,1:8], 2, range) #2 indicates that the operation is applied by column
```

```
##       mpg cylinders displacement horsepower weight acceleration year
## [1,] 11.0         3           68         46   1649          8.5   70
## [2,] 46.6         8          455        230   4997         24.8   82
##      origin
## [1,]      1
## [2,]      3
```

Range of the remaining data after removing rows 11th through 79th.

```r
apply(Auto2[,1:8], 2, median)
```

```
##          mpg    cylinders displacement   horsepower       weight
##         23.9          4.0        144.0         90.0       2789.0
## acceleration         year       origin
##         15.5         77.0          1.0
```

Median of the remaining data after removing rows 11th through 79th.

```r
apply(Auto2[,1:8], 2, var)
```

```
##          mpg    cylinders displacement   horsepower       weight
## 6.135039e+01 2.748938e+00 1.003229e+04 1.291977e+03 6.574970e+05
## acceleration         year       origin
```

```
## 7.296234e+00 1.004873e+01 6.803261e-01
```

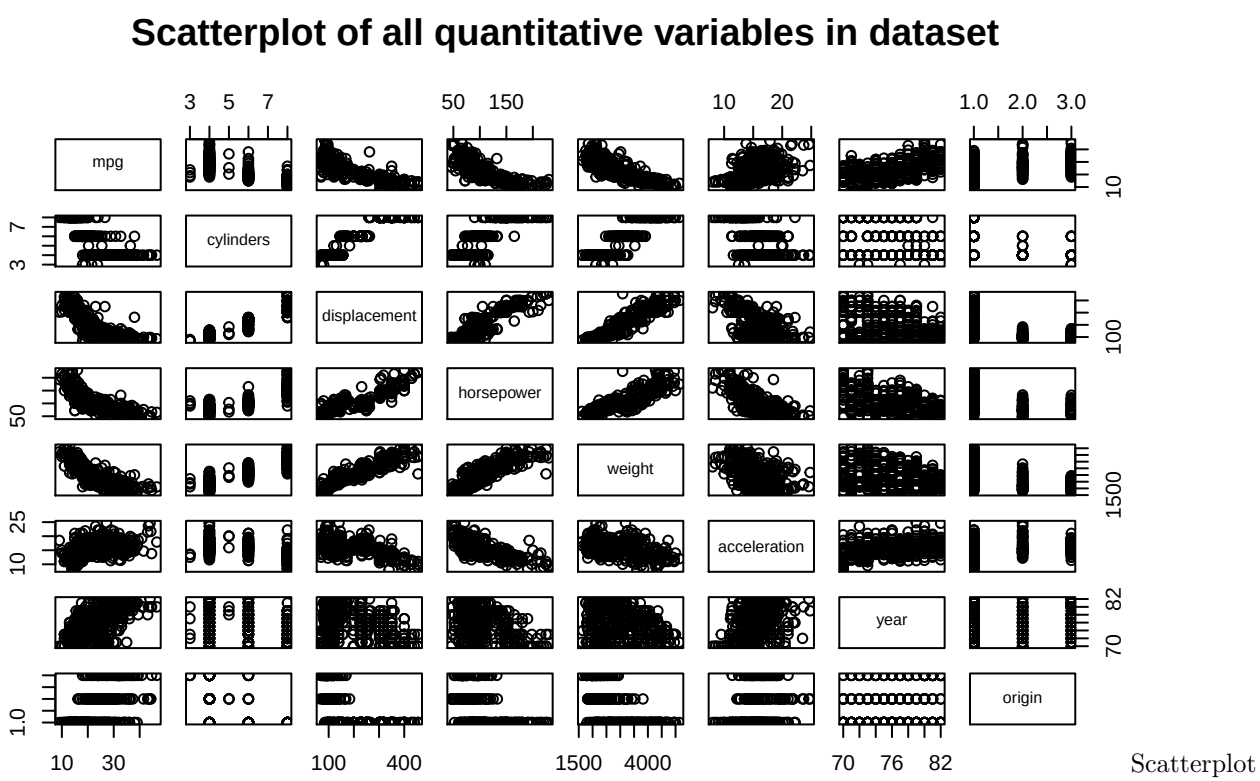
Variance of the remaining data after removing rows 11th through 79th.

(e) Using the full data set, investigate the predictors graphically, using scatterplots or other tools of your

choice. Create some plots highlighting the relationships among the predictors. Comment on your

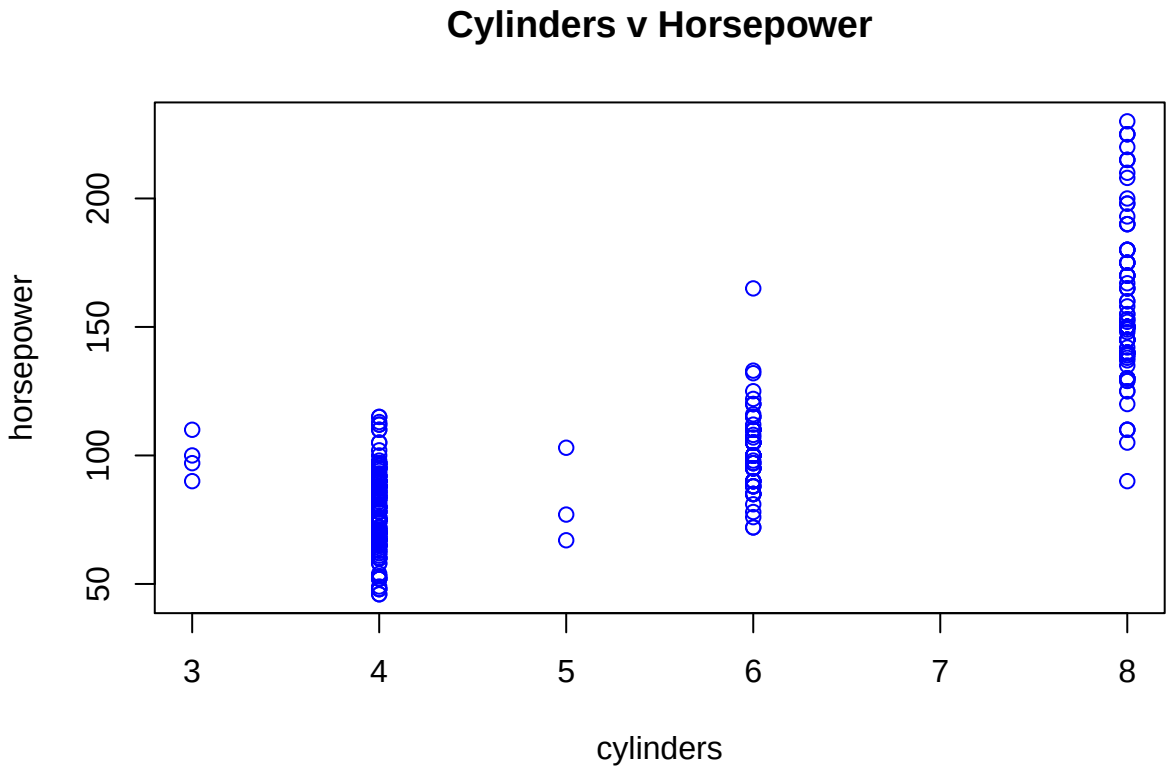
findings.

```
Auto3 <- Auto[, !sapply(Auto, is.factor)]
pairs(Auto3, main="Scatterplot of all quantitative variables in dataset")
```



Scatterplot of cylinders v horsepower

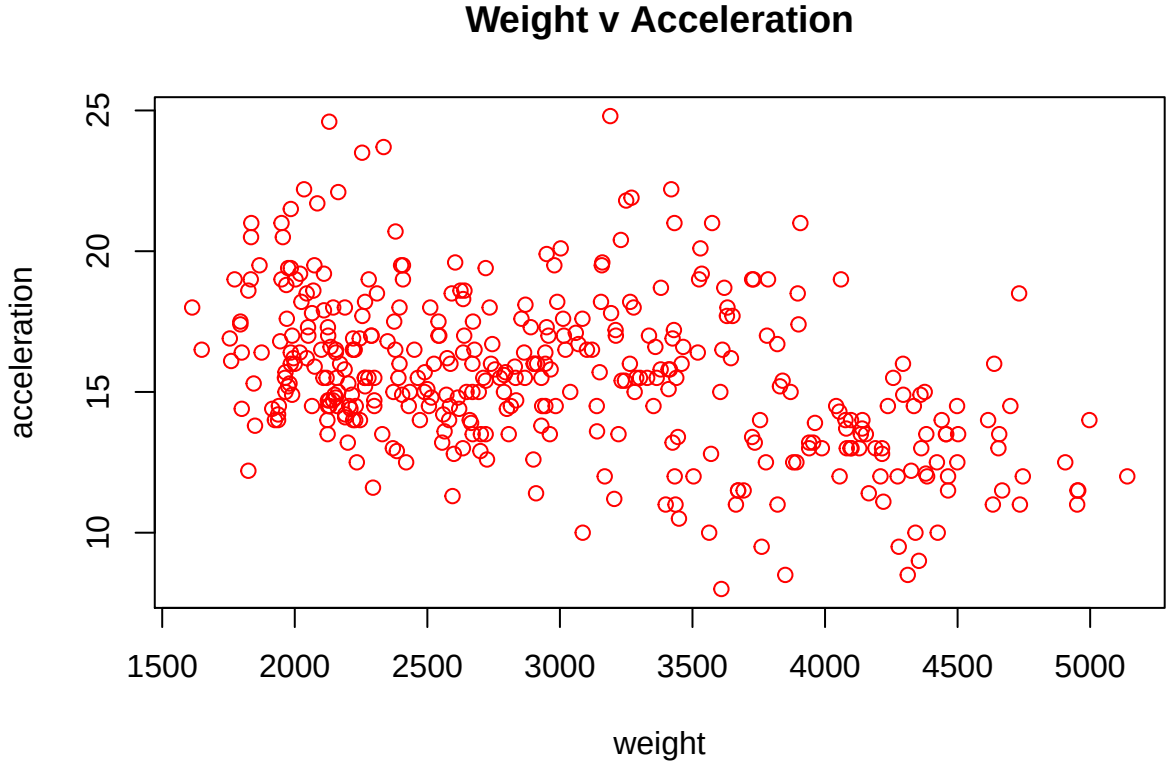
```
attach(Auto3) # to access variables of a Auto3 without calling the Auto3.
plot(cylinders, horsepower, col = "blue", main="Cylinders v Horsepower")
```



We can see that by increasing the number of cylinders, the horsepower increases, therefore horsepower is directly proportional to the number of cylinders.

Scatterplot of weight v acceleration

```
plot(weight, acceleration, col = "red", main="Weight v Acceleration")
```

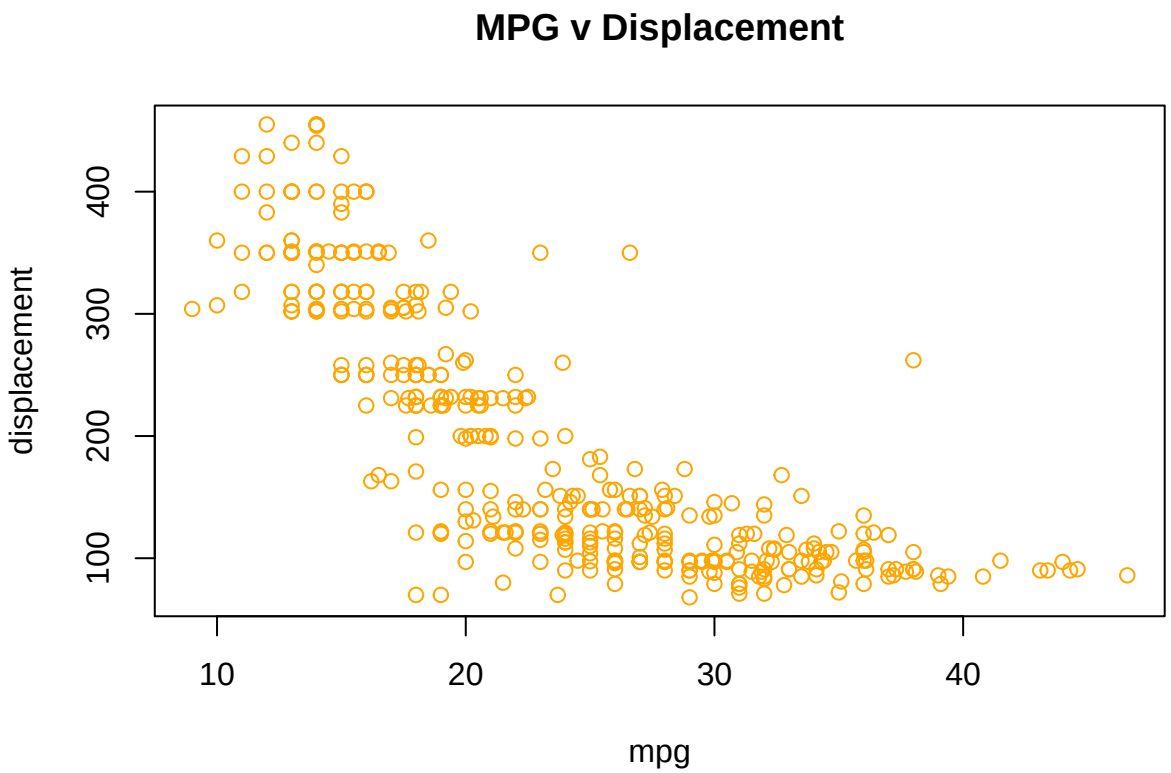


We can

see that by increasing the weight, the acceleration decreases, therefore weight is inversely proportional to the acceleration.

Scatterplot of mpg v displacement

```
plot(mpg, displacement, col = "orange", main="MPG v Displacement")
```



We can see that by increasing the mpg, the displacement decreases, therefore by increaseing the mpg(fuel efficiency), the displacement decreases.

(f) Suppose that we wish to predict gas mileage (mpg) on the basis of the other variables. Do your plots

suggest that any of the other variables might be useful in predicting mpg? Justify your answer.

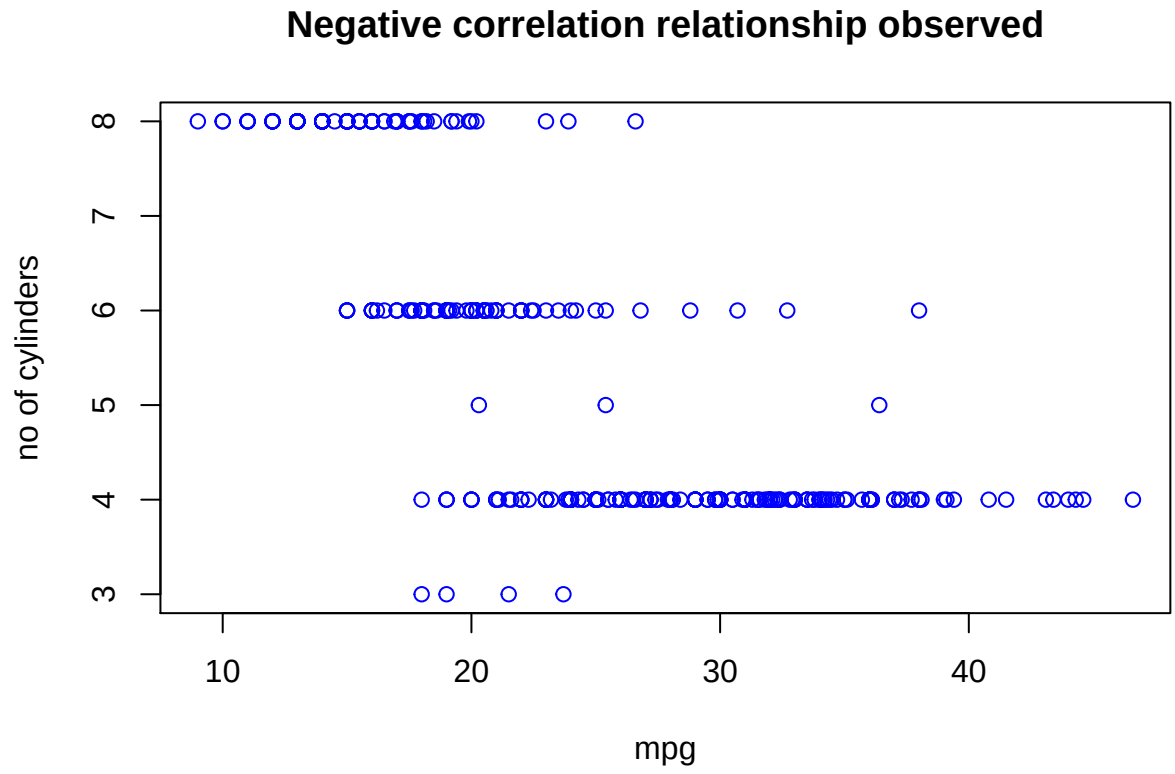
```
sapply(Auto3, function(x) cor(Auto$mpg, x))
```

```
##          mpg    cylinders displacement   horsepower       weight
##    1.0000000   -0.7776175   -0.8051269   -0.7784268   -0.8322442
## acceleration         year       origin
##    0.4233285    0.5805410    0.5652088
```

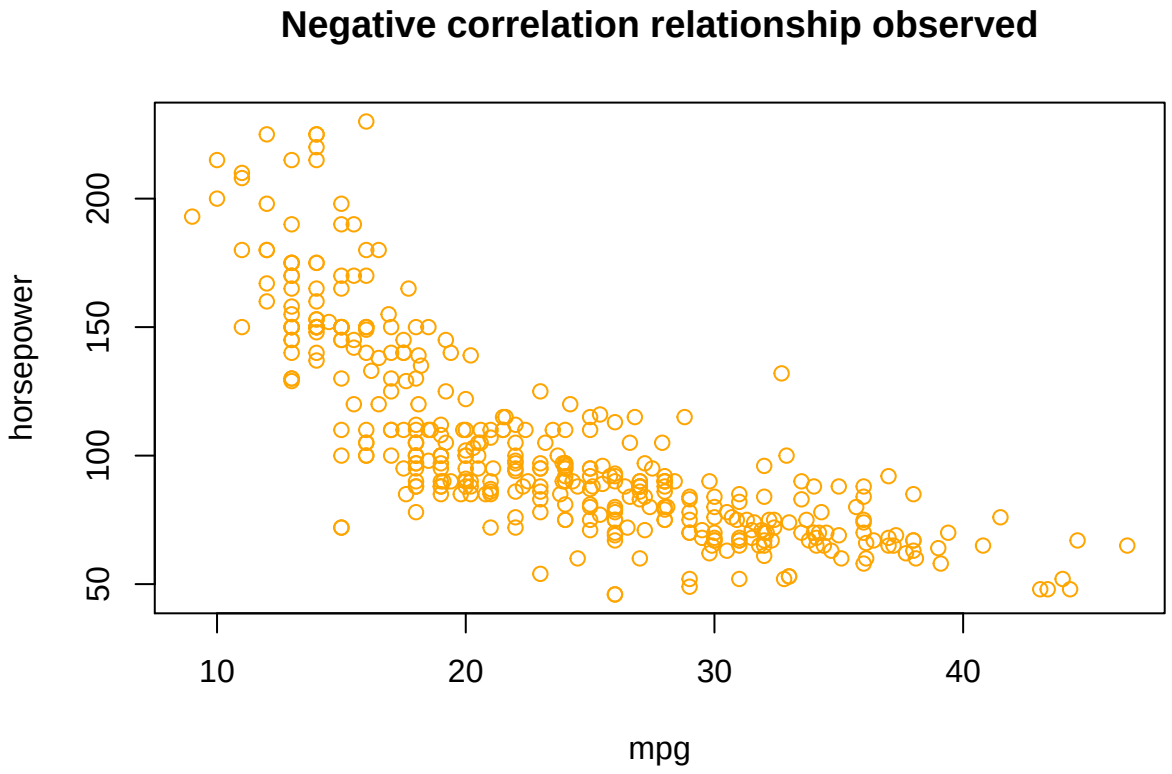
Looking at the above correlation data, we can see that cylinders, displacement, horsepower and weight have strong negative correlation relationship with mpg. Which means that any of these will need to be decreased in order to increase the gas milage. We have already observed the negative correlation relationship between mpg and displacement before in the scatterplot.

Plotting the respective plots:

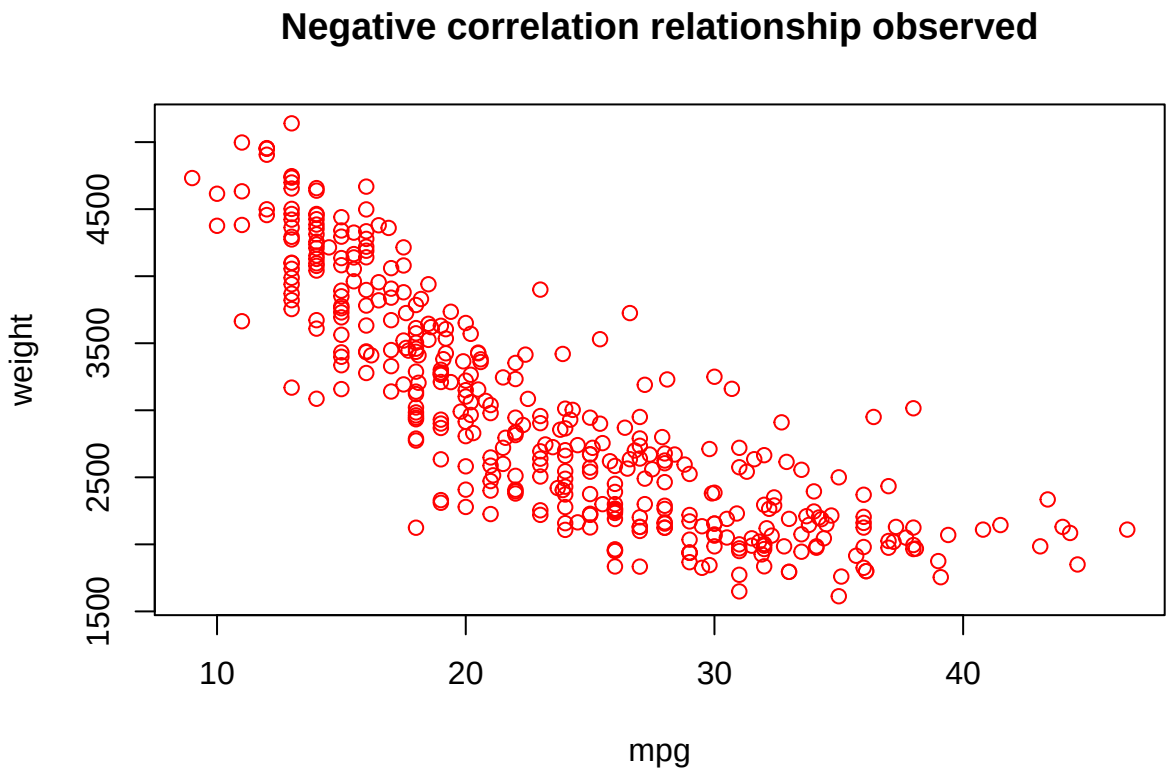
```
plot(mpg,cylinders, main="Negative correlation relationship observed", xlab="mpg", ylab="no of cylinders
```



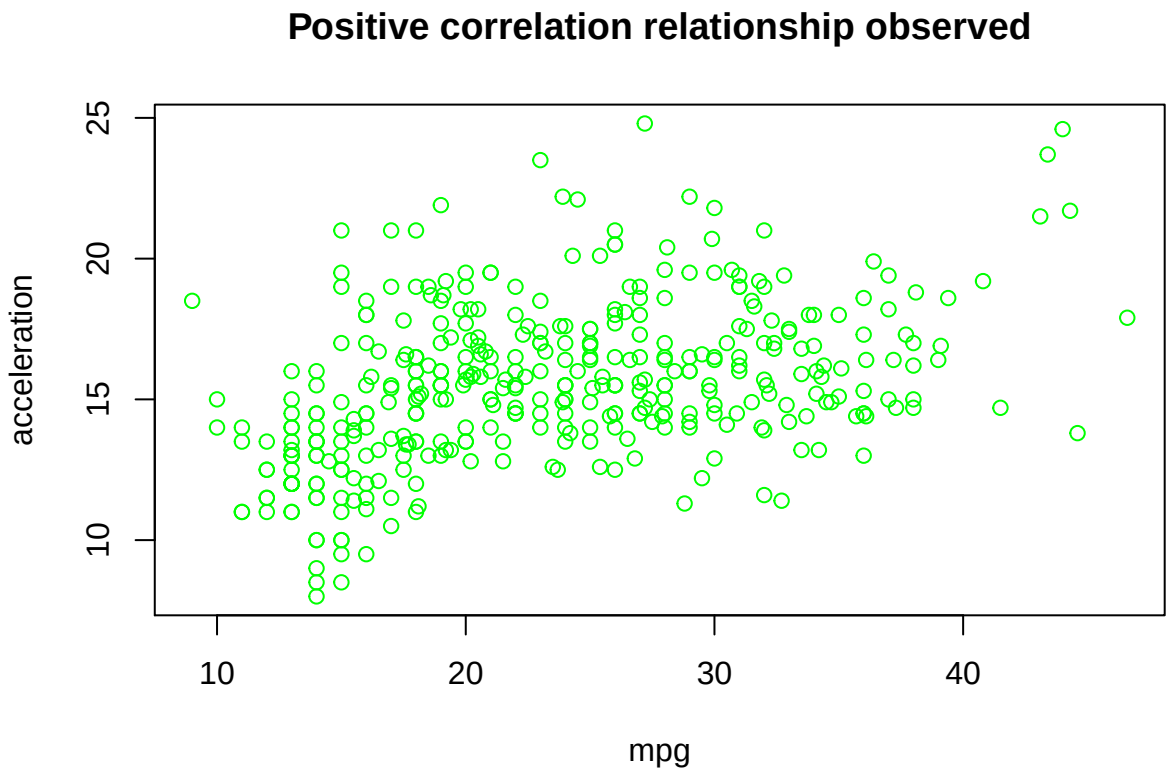

```
plot(mpg,horsepower, main="Negative correlation relationship observed", xlab="mpg", ylab="horsepower",
```




```
plot(mpg,weight, main="Negative correlation relationship observed", xlab="mpg", ylab="weight", col="red
```




```
plot(mpg,acceleration, main="Positive correlation relationship observed", xlab="mpg", ylab="acceleratio
```




```
detach(Auto3)
```

4) Linear regression

This question involves the use of simple linear regression on the Auto data set.

(a) Use the lm() function to perform a simple linear regression with mpg as the response and horsepower as

the predictor. Use the summary() function to print the results. Comment on the output. For example:

```
library(ISLR)
lm.fit <- lm(mpg ~ horsepower, data=Auto)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = mpg ~ horsepower, data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.5710  -3.2592  -0.3435   2.7630  16.9240
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 39.935861   0.717499   55.66   <2e-16 ***
## horsepower  -0.157845   0.006446  -24.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.906 on 390 degrees of freedom
## Multiple R-squared:  0.6059, Adjusted R-squared:  0.6049
## F-statistic: 599.7 on 1 and 390 DF,  p-value: < 2.2e-16
```

i. Is there a relationship between the predictor and the response?

The p-value for the coefficients is very close to zero, implying statistical significance which means that there is a relationship.

ii. How strong is the relationship between the predictor and the response?

The adjusted R square value indicated that 60.59% of variation in the response variable mpg is due to the horsepower.

iii. Is the relationship between the predictor and the response positive or negative?

The regression coefficient of horsepower is negative; therefore the relationship is negative.

iv. What is the predicted mpg associated with a horsepower of 89? What are the associated 99% confidence

and prediction intervals?

```
predict(lm.fit, data.frame(horsepower = c(89)))
```

```
##        1
## 25.88768
```

The predicted mpg associated with horsepower of 89 is 25.88768.

```
p_conf <- predict(lm.fit, data.frame(horsepower = c(89)), interval = "confidence", conf.level=0.99)
print(p_conf)
```

```
##        fit      lwr      upr
## 1 25.88768 25.36257 26.41279
```

The upper 99% confidence interval is 26.41279 The lower 99% confidence interval is 25.36257

```
p_pred <- predict(lm.fit, data.frame(horsepower = c(89)), interval = "prediction", pred.level=0.99)
print(p_pred)
```

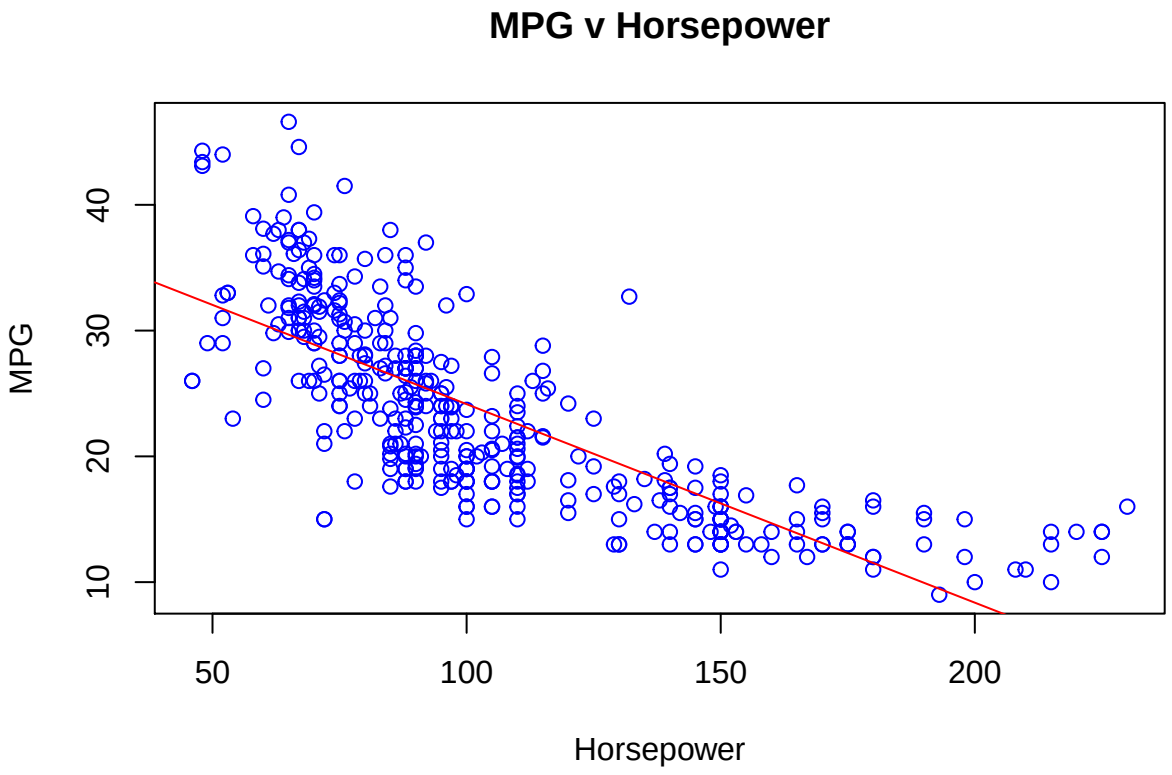
```
##        fit      lwr    upr
## 1 25.88768 16.22836 35.547
```

The upper 99% prediction interval is 35.547 The lower 99% prediction interval is 16.22836

(b) Plot the response and the predictor. Use the abline() function to display the least squares regression

line.

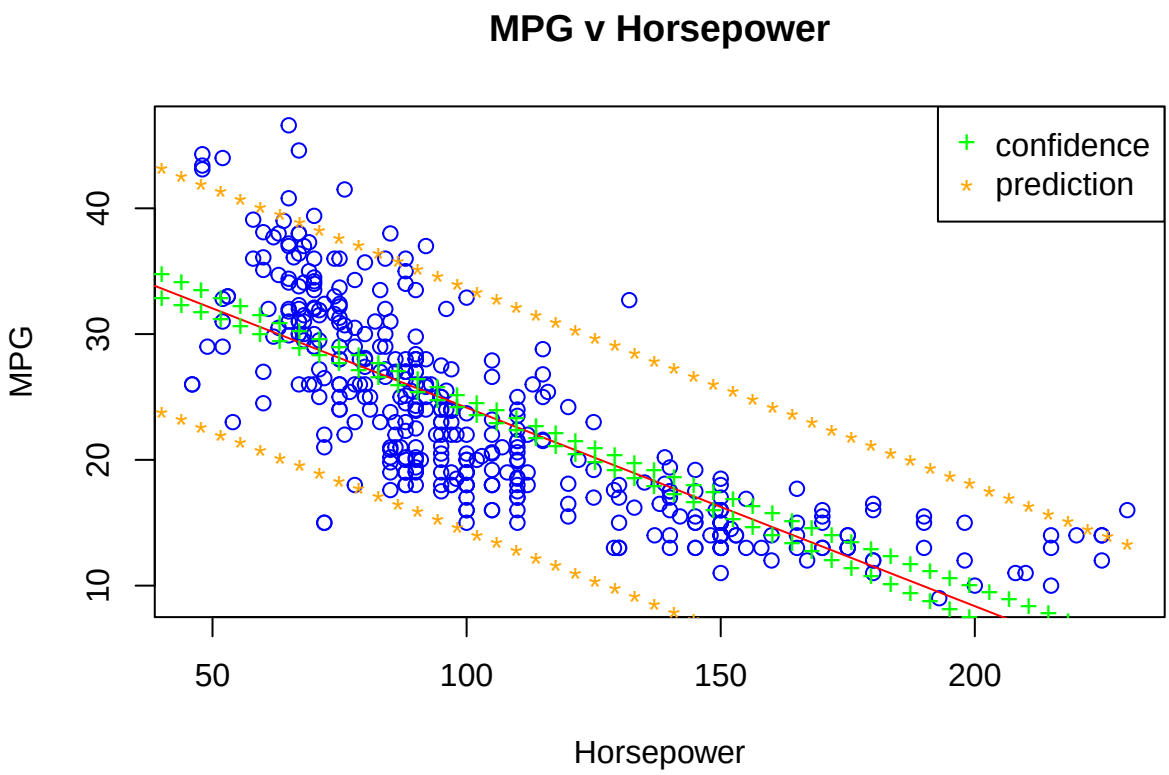
```
plot(Auto$mpg ~ Auto$horsepower, main="MPG v Horsepower", xlab="Horsepower", ylab="MPG", col="blue")
abline(coef = coef(lm.fit), col="red")
```



(c) Plot the 99% confidence interval and prediction interval in the same plot as (b) using different colours

and legends.

```
plot(Auto$mpg ~ Auto$horsepower, main="MPG v Horsepower", xlab="Horsepower", ylab="MPG", col="blue")
abline(coef = coef(lm.fit), col="red")
newHP <- data.frame(horsepower=seq(40,230,length=50)
                    )
p_conf <- predict(lm.fit, newHP,interval="confidence", conf.level=0.99)
p_pred <- predict(lm.fit, newHP,interval="prediction", conf.level=0.99)
lines(newHP$horsepower, p_conf[,"lwr"], col="green", type="b", pch="+")
lines(newHP$horsepower, p_conf[,"upr"], col="green", type="b", pch="+")
lines(newHP$horsepower, p_pred[,"upr"], col="orange", type="b", pch="*")
lines(newHP$horsepower, p_pred[,"lwr"], col="orange", type="b", pch="*")
legend("topright",
pch=c("+","*"),
col=c("green","orange"),
legend = c("confidence","prediction"))
```



5. Logistic regression

A recent study has shown that the accurate prediction of the office room occupancy leads to potential energy

savings of 30%. In this question, you are required to build logistic regression models by using different

environmental measurements as features, such as temperature, humidity, light, CO2 and humidity ratio, to

predict the office room occupancy. The provided training dataset consists of 2,000 samples, whilst the testing

dataset consists of 300 samples.

(a) Load the training and testing datasets from corresponding files, and display the statistics about different

features in the training dataset.

```
training_data <- read.table("Training_set for Q5.txt", header=T, sep=",")
testing_data <- read.table("Testing_set for Q5.txt", header = T, sep=",")
summary(training_data)
```

```
##   Temperature       Humidity         Light            CO2
##  Min.   :20.10   Min.   :18.96   Min.   :  0.0   Min.   : 426.0
##  1st Qu.:20.89   1st Qu.:21.82   1st Qu.:  0.0   1st Qu.: 448.0
##  Median :21.20   Median :25.00   Median :  0.0   Median : 485.5
##  Mean   :21.42   Mean   :24.22   Mean   :144.7   Mean   : 634.6
##  3rd Qu.:22.10   3rd Qu.:26.29   3rd Qu.:433.0   3rd Qu.: 845.8
##  Max.   :23.18   Max.   :28.50   Max.   :744.0   Max.   :1139.0
##  HumidityRatio        Occupancy
##  Min.   :0.002824   Min.   :0.0000
##  1st Qu.:0.003375   1st Qu.:0.0000
##  Median :0.003905   Median :0.0000
##  Mean   :0.003836   Mean   :0.2775
##  3rd Qu.:0.004343   3rd Qu.:1.0000
##  Max.   :0.004817   Max.   :1.0000
```

(b) Build a logistic regression model by only using the Temperature feature to predict the room occupancy.

Display the confusion matrix and the predictive accuracy obtained on the testing dataset.

```
glm.temp.fit <- glm(Occupancy ~ Temperature,data=training_data,family = binomial)
prediction1 <- predict(glm.temp.fit, testing_data, type="response")
CM <- table(prediction = prediction1, truth = testing_data$Occupancy)
print(CM)
```

```
##                     truth
## prediction            0  1
##   0.0166860166669195  1  0
##   0.0190677866046178  1  0
##   0.0217820005161907 16  0
##   0.0231373792048931  1  0
```

```
##   0.0236072149038195  1  0
##   0.0245749771845544  8  0
##   0.0277159338393461 44  0
##   0.0294294414041138  1  0
##   0.0300231190386382  2  0
##   0.0312454803480057  2  0
##   0.0325159405613673  0  1
##   0.0352082282784031 10  0
##   0.0391853388606521  0  1
##   0.0435914043825391  5  0
##   0.0551318820654362  3  0
##   0.060044444058581   1  0
##   0.0653644611043984  1  0
##   0.0711201739919605 25  0
##   0.079723900659004   2  0
##   0.0892684314063466  4  1
##   0.0961925639339331  1  0
##   0.11149185909913   10  0
##   0.122987490216672   1  0
##   0.135487662773234   5  2
##   0.150615598938448   1  0
##   0.16710615339361    3  0
##   0.176775304169381   0  1
##   0.186878402723771   2  0
##   0.208404754353376   6  0
##   0.218775568133968   2  0
##   0.236873774040424   1  0
##   0.252078018196289   1  0
##   0.267915557708563   1  0
##   0.276066849560409   1  0
##   0.288577255492577   2  0
##   0.30141854240886    4  2
##   0.325326766604515   1  0
##   0.333511140586483   0  1
##   0.350180763191296   3  4
##   0.364359362721513   0  1
##   0.40824222759278    8 15
##   0.424748654312608   0  1
##   0.430290242262035   0  1
##   0.441424631492178   0  1
##   0.458233790186732   0  3
##   0.475138560636598   1  4
##   0.521418934425174   1  0
##   0.536803370624606   1  2
##   0.552118059631115   1  0
##   0.577410513514276   1  0
##   0.597362689274151   1  2
##   0.636734476803695   1  0
##   0.649489387041494   1  2
##   0.663414442477107   1  0
##   0.703454714044345   1  2
##   0.756854760565199   1  0
##   0.799395492964894   1  1
##   0.818471289012515   1  0
```

```
##   0.824507871454219   1  0
##   0.836105039843346   3  0
##   0.843579377600696   2  0
##   0.85077368890104    1  0
##   0.884636804564408   0  1
##   0.890790752469595   1  0
##   0.903322439049645   0  1
##   0.914554058436245   1  0
##   0.931983241822535   1  1
##   0.946066893329512   1  0
##   0.956348135537873   1  0
##   0.975092051949003   0  1
##   0.976548659906771   0  1
##   0.979216654782467   1  0
##   0.982670387409565   0  1
##   0.983691229320534   4  2
##   0.984559278360855   2  0
##   0.985381811304723   1  0
##   0.98616114349668    3  0
##   0.986899479917871   1  0
##   0.98909072047539    1  0
##   0.989737307297615   1  0
##   0.99102928531551    2  0
##   0.991423054962259   1  0
##   0.992159903658743   2  0
##   0.992775088677437   1  0
##   0.993064463650796   1  0
##   0.993477082196438   1  0
##   0.993865304314124   3  0
##   0.994897439849437   1  0
##   0.99520153536028    5  1
##   0.995542730465826   1  0
##   0.996154338253147   1  2
##   0.996599615324175   1  0
##   0.996993490770172   0  1
```

####(c) Build a logistic regression model by only using the Humidity feature to predict the room occupancy. ####D isplay the confusion matrix and the predictive accuracy obtained on the testing dataset.

```
glm.humid.fit <- glm(Occupancy ~ Humidity,data=training_data,family = binomial)
prediction2 <- predict(glm.humid.fit, testing_data, type="response")
CM2 <- table(prediction = prediction2, truth = testing_data$Occupancy)
print(CM2)
```

```
##                     truth
## prediction           0 1
##   0.0497971148324407 1 0
##   0.0505391825664944 1 0
##   0.0571983464455488 1 0
##   0.0594794538866251 0 1
##   0.0606517175665758 1 0
##   0.062268574384415  1 0
##   0.0628164118046562 2 0
##   0.0639256002161111 1 0
##   0.0688460339530518 1 0
```

```
##   0.0699523474766306 1 0
##   0.0722144368252327 5 0
##   0.0741152129353879 1 0
##   0.0754078582877367 1 0
##   0.0765008463746861 4 0
##   0.0776083468587446 0 1
##   0.078730513721376  0 1
##   0.079486849220935  1 0
##   0.0798675014452775 1 0
##   0.0810194649788235 4 0
##   0.0824218080868895 1 0
##   0.0831312345940561 0 2
##   0.0854145158343613 1 0
##   0.0865157978674568 1 1
##   0.0904507862508742 1 0
##   0.0905362354953004 1 0
##   0.0972920354697503 1 0
##   0.0988069537809649 1 0
##   0.109250008872122  0 1
##   0.113253075164086  1 0
##   0.119333515507132  1 2
##   0.11949727542666   0 1
##   0.122645776519385  1 0
##   0.126724578455815  2 0
##   0.130211655841471  0 1
##   0.13571669655626   0 1
##   0.141164709830883  2 1
##   0.144597495158363  2 0
##   0.146274912956177  1 0
##   0.148099362587051  1 0
##   0.148624075370493  1 0
##   0.156132049055769  1 0
##   0.163023376399997  0 1
##   0.169279750095035  1 0
##   0.17640355445902   1 0
##   0.180974351951574  1 0
##   0.183294122506262  3 0
##   0.185401575683119  1 0
##   0.187527695893282  2 0
##   0.192319424390797  1 0
##   0.196711247347494  1 0
##   0.202181321869637  1 0
##   0.20268419007602   1 0
##   0.204704986328608  1 0
##   0.205720980318762  0 1
##   0.207764164243046  1 0
##   0.208791355445653  0 1
##   0.212937456722886  1 0
##   0.215734712305741  1 0
##   0.222484540462451  1 0
##   0.223563972104006  1 0
##   0.228742094595861  0 1
##   0.245354103928555  1 0
##   0.249413312985299  0 1
```

```
##   0.256475572218754  1 0
##   0.257367505504737  1 0
##   0.260855352744409  2 0
##   0.266397685750474  1 0
##   0.269350022396839  1 0
##   0.269758858536566  1 0
##   0.272837573382304  1 0
##   0.275627110827461  1 0
##   0.275938147843501  2 0
##   0.280944166994532  1 0
##   0.282204282944132  2 0
##   0.285369528817     0 1
##   0.291763534656874  1 0
##   0.294344594528983  2 0
##   0.29499193524024   1 1
##   0.300855036091059  1 0
##   0.303701561992459  1 0
##   0.306783959998483  1 0
##   0.314117977133525  0 1
##   0.3201885718693    2 0
##   0.327691559695301  3 0
##   0.330212498101483  1 0
##   0.334821041718842  1 0
##   0.335283651774886  1 0
##   0.340859469185051  1 0
##   0.341559592611002  1 0
##   0.3422604073756    1 0
##   0.356060992884869  1 0
##   0.356418136321038  1 0
##   0.359998302956322  1 0
##   0.363594114820398  2 0
##   0.366843444536228  1 0
##   0.367929267094679  1 0
##   0.370104925174937  1 0
##   0.370468050983953  1 0
##   0.373378302897707  1 0
##   0.387447609813943  1 0
##   0.391397308343666  4 0
##   0.403704996861527  1 0
##   0.40783547761678   4 1
##   0.419165209618731  1 0
##   0.422961438476983  2 0
##   0.429054194800425  1 0
##   0.429817343075121  1 0
##   0.436700274813211  1 0
##   0.452851108722236  2 0
##   0.453622954711144  1 0
##   0.46135307680057   1 0
##   0.467550765753585  1 0
##   0.468326221084514  2 0
##   0.484640384923089  1 0
##   0.500208693528721  0 1
##   0.507994583570392  1 0
##   0.529762438466031  0 1
```

```
##   0.539061789804554  1 0
##   0.545246688174049  1 0
##   0.561410982552565  1 0
##   0.561666603208809  1 0
##   0.56217774618744   2 0
##   0.567281744930295  0 1
##   0.569828468900025  2 0
##   0.575925306791703  1 0
##   0.576939255490688  1 0
##   0.577445987438019  2 0
##   0.5842705086755    1 0
##   0.596571915577616  1 0
##   0.606029711281802  1 1
##   0.616882003144778  1 0
##   0.620800209302521  3 0
##   0.621533139277346  1 0
##   0.636792119651673  1 1
##   0.64038627020552   0 1
##   0.643964739896271  1 0
##   0.664394208701935  1 0
##   0.665088335613838  1 0
##   0.668203045178458  0 1
##   0.677458475251898  2 2
##   0.678138671529322  1 0
##   0.69091680623757   1 2
##   0.698184998880772  2 0
##   0.705355164669639  2 0
##   0.711786618237443  1 0
##   0.714970826411601  1 0
##   0.718133786148093  5 0
##   0.722317723871743  2 0
##   0.724395296898996  1 0
##   0.72563720786596   1 0
##   0.727493534731769  2 0
##   0.729956381833133  2 0
##   0.736051789795083  1 0
##   0.738765755215499  1 0
##   0.742653945144687  1 0
##   0.759535157056146  1 0
##   0.765736870188729  0 1
##   0.769441059868594  0 1
##   0.781549297570747  1 0
##   0.787342217286751  1 0
##   0.796577946869192  4 0
##   0.797585480535818  1 0
##   0.799089790533547  1 0
##   0.80157834397882   1 0
##   0.804043632065453  1 0
##   0.806485684780553  2 0
##   0.810025406266754  1 0
##   0.811776592010452  1 0
##   0.816489453745676  1 0
##   0.826087594278857  2 0
##   0.828313590075525  0 1
```

```
##   0.829050552268413  1 0
##   0.831973297828458  1 0
##   0.842441173069422  1 1
##   0.843266244880052  1 0
##   0.845720361292211  1 0
##   0.847875342530755  1 0
##   0.848543689676606  1 0
##   0.850534262985602  1 0
##   0.851324433901723  1 0
##   0.852894430131568  0 1
##   0.858660001772675  1 0
##   0.861285373030741  1 0
##   0.862397950015924  1 0
##   0.862767138889779  0 1
##   0.865931979806032  1 0
##   0.867670052645205  1 0
##   0.869977314782547  1 0
##   0.877979359075294  1 0
##   0.879857606086274  2 0
##   0.885973925077972  0 1
##   0.889389601704114  1 0
##   0.892416683270802  1 0
##   0.892715350819141  1 0
##   0.894492074606131  0 1
##   0.897108486063481  0 1
##   0.897967777350592  1 0
##   0.898537098942     1 0
##   0.903896655988309  0 1
##   0.904076878090122  1 2
##   0.908829875174681  0 1
##   0.911377946328628  0 1
##   0.911503647222147  0 1
##   0.913861552342532  0 1
##   0.917821905934202  0 2
##   0.921390496852685  0 1
##   0.925069081665572  1 0
##   0.925785561504464  0 1
##   0.928829449910804  1 0
##   0.943724426592383  1 0
##   0.943999481856489  0 1
##   0.944545802464988  0 1
##   0.951012459283776  1 0
##   0.953925686127664  0 1
##   0.956241183575526  1 1
##   0.964295181364838  0 1
##   0.968345141767243  1 0
##   0.969378650971912  1 0
##   0.979300823178631  0 1
##   0.980585928457372  0 1
##   0.983525079320839  1 0
##   0.983792090554751  1 0
##   0.983891118629584  1 0
##   0.991611300861858  1 0
##   0.994790760438958  1 0
```

```
##   0.997146457573425  2 0
```

(d) Build a logistic regression model by using all features to predict the room occupancy. Display the

confusion matrix and the predictive accuracy obtained on the testing dataset.

```
glm.all.fit <- glm(Occupancy ~ Temperature+Humidity+Light+CO2+HumidityRatio,data=training_data,family =
prediction3 <- predict(glm.all.fit, testing_data, type="response")
CM3 <- table(prediction = prediction3, truth = testing_data$Occupancy)
print(CM3)
```

```
##                       truth
## prediction             0 1
##   0.000282989654197081 0 1
##   0.000408132463507101 1 0
##   0.000414908476108393 1 0
##   0.000503837057632348 1 0
##   0.000509392035104103 1 0
##   0.000628245610047359 1 0
##   0.000638939636781844 1 0
##   0.000641196547309012 1 0
##   0.000677269065775944 1 0
##   0.000695394016364476 1 0
##   0.000747106802711544 1 0
##   0.000798356489454428 1 0
##   0.000894098165879229 1 0
##   0.00100910912443515  1 0
##   0.00103007699417286  1 0
##   0.00116817663551909  1 0
##   0.00121622039557811  1 0
##   0.0012400912659854   1 0
##   0.00154714307150047  1 0
##   0.001556950098269    1 0
##   0.00161096891538289  0 1
##   0.00165227356422208  1 0
##   0.00168573246704206  1 0
##   0.00189133570481751  1 0
##   0.00201826211252427  1 0
##   0.00221648839032982  1 0
##   0.00236651250496649  1 0
##   0.00277532713994852  1 0
##   0.00350502287500795  1 0
##   0.00413806716874582  1 0
##   0.00483088625519318  1 0
##   0.00555260313220431  1 0
##   0.00719846167533543  1 0
##   0.00778704259677026  1 0
##   0.00787039819019585  1 0
##   0.00787408902774255  1 0
##   0.00842340871784338  1 0
##   0.00863285487915723  1 0
##   0.00900106512895055  1 0
##   0.0095551045963924   0 1
```

```
##   0.00976439587219282  1 0
##   0.0105707709810737   1 0
##   0.0106317442358456   1 0
##   0.0107224377307497   1 0
##   0.0129461318896063   1 0
##   0.0137538441960398   1 0
##   0.0149072628403434   1 0
##   0.0149869825654668   1 0
##   0.016540334407421    1 0
##   0.0169863648229573   1 0
##   0.0174625560704426   1 0
##   0.0179073810438886   1 0
##   0.0180210198392396   0 1
##   0.0185326002458289   1 0
##   0.0187775966716826   1 0
##   0.0189611681986007   1 0
##   0.0195955625438477   1 0
##   0.0201101328467171   1 0
##   0.0202786868092239   1 0
##   0.0202911545032466   1 0
##   0.0204331764379698   1 0
##   0.0204524655495146   1 0
##   0.0204671719404841   1 0
##   0.0205442656057057   1 0
##   0.0206449124346188   1 0
##   0.0211234774930978   1 0
##   0.0212438185179816   1 0
##   0.0215240705011809   1 0
##   0.0216060398759844   1 0
##   0.0216633541978201   1 0
##   0.0216997579061578   1 0
##   0.0219352682921384   1 0
##   0.0219536591059776   1 0
##   0.0220569535504941   1 0
##   0.0221401469634537   1 0
##   0.0225758685907736   1 0
##   0.0227493991704405   1 0
##   0.0227846856058337   1 0
##   0.023109769845511    1 0
##   0.0234253636261203   1 0
##   0.0234450592833118   1 0
##   0.023552875483171    1 0
##   0.0236851657610963   1 0
##   0.0241786829940952   1 0
##   0.0244257975660704   1 0
##   0.0245228171559379   1 0
##   0.0253650813332674   1 0
##   0.0255397592276525   1 0
##   0.0256090973863414   1 0
##   0.0265177790237405   1 0
##   0.0265192848082703   1 0
##   0.0267315472328586   1 0
##   0.0267359638832564   1 0
##   0.0268163623925254   1 0
```

```
##   0.0270995137887304   1 0
##   0.0274077265214519   1 0
##   0.0274717707922052   1 0
##   0.0275139539615094   1 0
##   0.0281577811789977   1 0
##   0.0284142819124935   1 0
##   0.0291112307076082   1 0
##   0.0293716080290813   1 0
##   0.0296980889880182   1 0
##   0.0305402751485184   1 0
##   0.030616074022339    1 0
##   0.0307949146448957   1 0
##   0.0310688212474631   1 0
##   0.0311224960627943   1 0
##   0.0317274166680576   1 0
##   0.0319423959422095   1 0
##   0.032007738854547    1 0
##   0.0323621215965984   1 0
##   0.0324994431798682   1 0
##   0.0327078914403393   1 0
##   0.0329837048787834   1 0
##   0.0330233138353506   1 0
##   0.0331902190452831   1 0
##   0.0341148101755208   1 0
##   0.0345559467225412   1 0
##   0.0347174779183941   1 0
##   0.034879060179234    1 0
##   0.0352526410668777   1 0
##   0.0355915938510541   1 0
##   0.0357134520249373   1 0
##   0.0363313743140816   1 0
##   0.0363388182343019   1 0
##   0.036790638054533    1 0
##   0.0371323293357945   1 0
##   0.0378023376763132   1 0
##   0.0381322724726417   1 0
##   0.0388067631520905   1 0
##   0.0389020088081645   1 0
##   0.0393416351815121   1 0
##   0.0395338657244252   1 0
##   0.0396729242986573   1 0
##   0.0398488145156508   1 0
##   0.0407187871459007   1 0
##   0.0409783880947679   1 0
##   0.0410670548649531   1 0
##   0.0433868309951944   1 0
##   0.0438707900140793   1 0
##   0.0446640493377263   1 0
##   0.0448927624587792   1 0
##   0.0453562417539408   1 0
##   0.0470905828064844   1 0
##   0.0471549090895886   1 0
##   0.0477628407511819   1 0
##   0.0477803392961848   1 0
```

```
##   0.0485627355369559   1 0
##   0.0488633683447205   1 0
##   0.0489577098528911   1 0
##   0.0502016837254262   1 0
##   0.0503854784369314   1 0
##   0.0506267594791506   1 0
##   0.0507170122594981   1 0
##   0.051490402296037    1 0
##   0.052003437073132    1 0
##   0.052067341828677    1 0
##   0.0525096716406328   1 0
##   0.0535957207919295   1 0
##   0.0539098460936397   1 0
##   0.0544962795893776   1 0
##   0.0567169975255818   1 0
##   0.057116933677605    1 0
##   0.0574349870386058   1 0
##   0.0576779821057341   1 0
##   0.0594224185026175   1 0
##   0.0600182615660812   1 0
##   0.060379229602143    1 0
##   0.0605849199783086   0 1
##   0.0609302317480996   1 0
##   0.0620176372470898   1 0
##   0.0686884650593405   1 0
##   0.0701338467786725   1 0
##   0.0701472273986822   1 0
##   0.0701852300602422   1 0
##   0.070430784917389    1 0
##   0.072178500013107    1 0
##   0.0734248520895457   1 0
##   0.0753907137956943   1 0
##   0.0758380841753035   1 0
##   0.0771358554060739   1 0
##   0.0788393025586879   1 0
##   0.0800087975156363   1 0
##   0.0816608567584635   0 1
##   0.270611770348033    1 0
##   0.283288020261926    0 1
##   0.312405625721404    0 1
##   0.385985441572571    0 1
##   0.424644663163522    1 0
##   0.457135643714612    1 0
##   0.461250665378868    0 1
##   0.462151918863939    1 0
##   0.4729546665936      0 1
##   0.476367515453657    1 0
##   0.483832058934532    0 1
##   0.498111162107835    0 1
##   0.500622866656367    1 0
##   0.524518739024375    1 0
##   0.533487194651887    1 0
##   0.546318188542079    1 0
##   0.555186516945972    1 0
```

```
##   0.559580452785232    0 1
##   0.571214086685538    1 0
##   0.58232989294768     1 0
##   0.629970341258082    1 0
##   0.633568265700814    1 0
##   0.636521218263185    0 1
##   0.639491226969673    0 1
##   0.643604860302438    1 0
##   0.649295603672355    1 0
##   0.656115700767532    1 0
##   0.684462104719962    1 0
##   0.690043973555757    1 0
##   0.699664249331654    0 1
##   0.707640269300927    0 1
##   0.729179873433972    0 1
##   0.738373234575141    1 0
##   0.739267169753574    0 1
##   0.757514966763524    0 1
##   0.765967796661318    0 1
##   0.780919563695929    1 0
##   0.800411925549111    0 1
##   0.813262676663361    0 1
##   0.814457287648349    1 0
##   0.829374010310157    0 1
##   0.835113490643854    0 1
##   0.849011311018785    1 0
##   0.867156306542878    0 1
##   0.873250206948696    0 1
##   0.880572250059538    0 1
##   0.88128203094769     0 1
##   0.892174779298716    0 1
##   0.900817302612631    1 0
##   0.91530233584122     1 0
##   0.917484324971477    0 1
##   0.925435718600217    0 1
##   0.926943910553738    0 1
##   0.927181685756908    1 0
##   0.930219806653545    0 1
##   0.930870344723229    1 0
##   0.931022758611173    0 1
##   0.933842051734074    0 1
##   0.93411811759881     0 1
##   0.934342379194796    0 1
##   0.935550262366666    0 1
##   0.935976281867267    1 0
##   0.936608037928727    0 1
##   0.938608053187779    0 1
##   0.93875294739864     1 0
##   0.939025900151872    0 1
##   0.939517595217415    0 1
##   0.94055834542692     0 1
##   0.941492706261442    0 1
##   0.941501390781931    1 0
##   0.943582742600318    1 0
```

```
##   0.945714263816444    0 1
##   0.947112211920847    1 0
##   0.950652410001657    1 0
##   0.951148229772317    1 0
##   0.95180435561137     0 1
##   0.952803688613685    0 1
##   0.954164591323048    1 0
##   0.961330836471797    1 0
##   0.966702127563952    1 0
##   0.967914088846175    0 1
##   0.97061023708985     0 1
##   0.971228532222785    1 0
##   0.976201287048276    0 1
##   0.978094460447056    0 1
##   0.978124486858749    1 0
##   0.978332599131111    1 0
##   0.978343325550224    0 1
##   0.979061989973089    0 1
##   0.979438940786177    1 0
##   0.979863263877865    1 0
##   0.979885058447416    1 0
##   0.980258290289372    1 0
##   0.981333725976814    1 0
##   0.981664683068704    0 1
##   0.981729854457493    1 0
##   0.982583152883989    1 0
##   0.983986769412702    1 0
##   0.984051387759906    1 0
##   0.984816438789984    1 0
##   0.985196327666187    1 0
##   0.986263465009129    1 0
##   0.987364968695895    1 0
##   0.988628990919362    0 1
##   0.988725001012061    0 1
##   0.991108081761879    1 0
##   0.991922146187556    0 1
##   0.991981578422674    1 0
##   0.99201989456455     0 1
##   0.99406909015386     1 0
##   0.994086530950478    1 0
##   0.995805416280446    1 0
##   0.995888514601699    1 0
##   0.996415502307123    1 0
##   0.997550702644144    1 0
```

(e) Compare the predictive performance of three different models by drawing ROC curves and calculating

the AUROC values. Discuss the comparison results.

ROC Curve and AUROC values for prediction based on Temperature

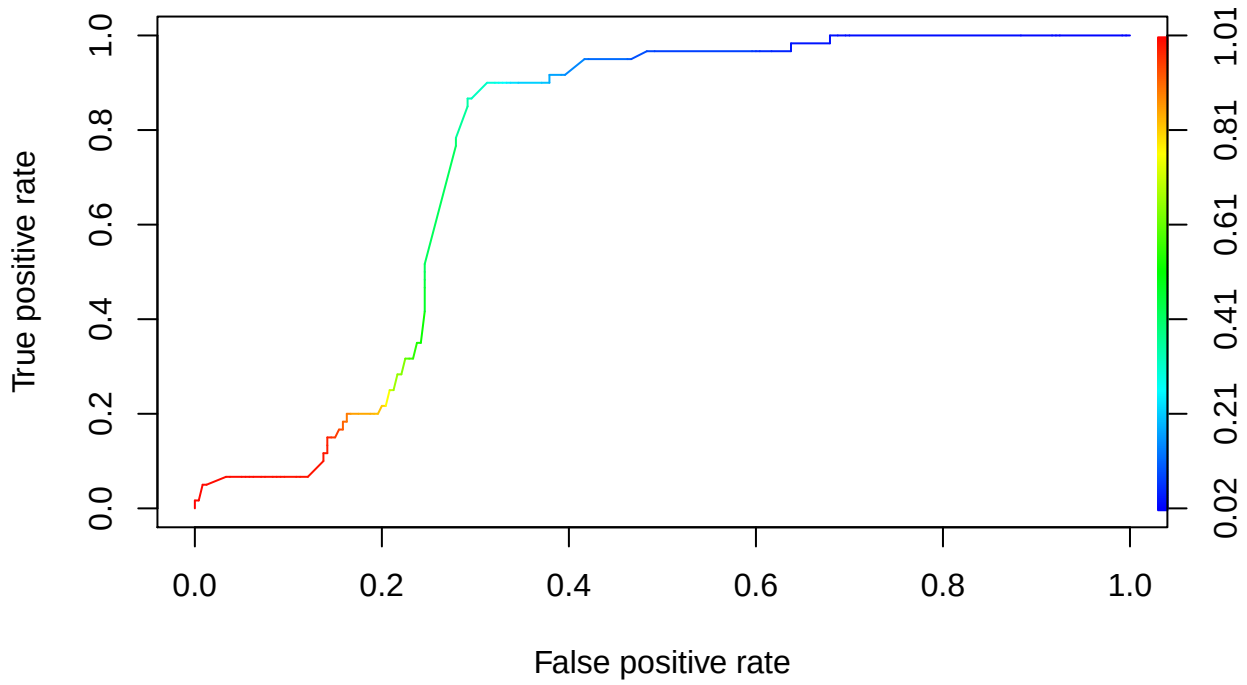
```
#install.packages("ROCR")
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
pred <- prediction(prediction1, testing_data$Occupancy)
perf <- performance(pred, measure="tpr", x.measure = "fpr")
plot(perf,colorize=TRUE)
```

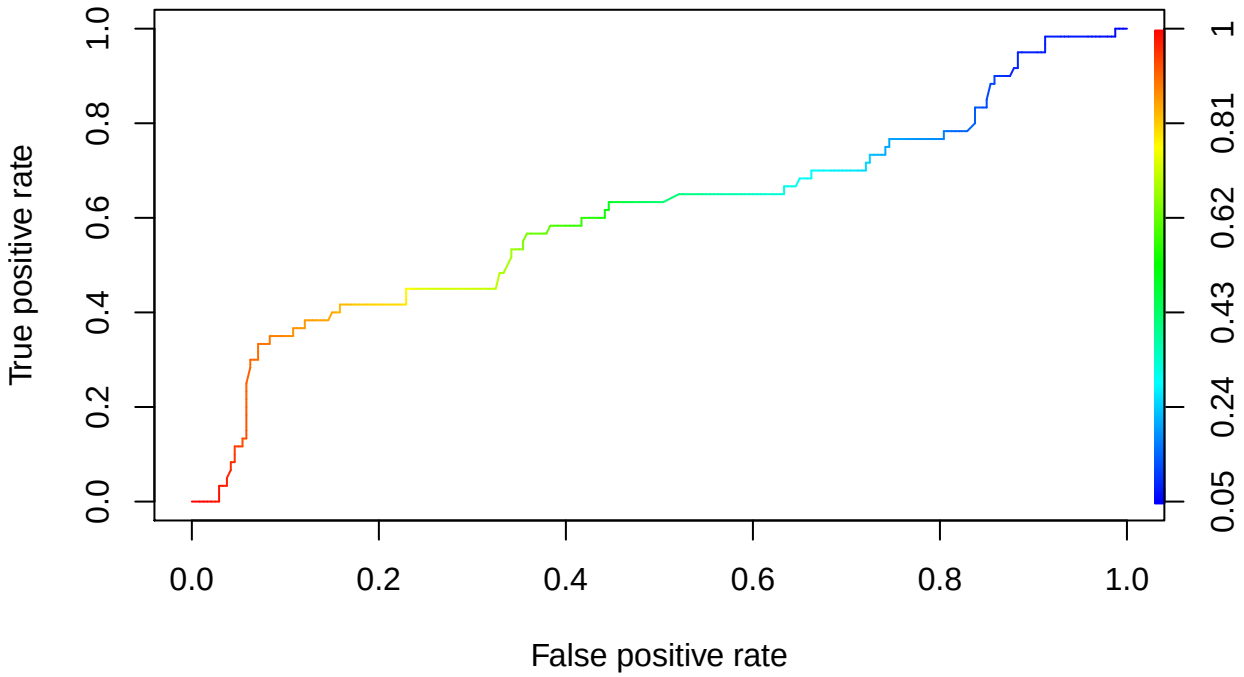



```
auroc <- performance(pred, measure="auc")
auroc_value <- auroc@y.values[[1]]
auroc_value
```

```
## [1] 0.7527431
```

ROC Curve and AUROC values for prediction based on Humidity

```
pred2 <- prediction(prediction2, testing_data$Occupancy)
perf2 <- performance(pred2, measure="tpr", x.measure = "fpr")
plot(perf2,colorize=TRUE)
```

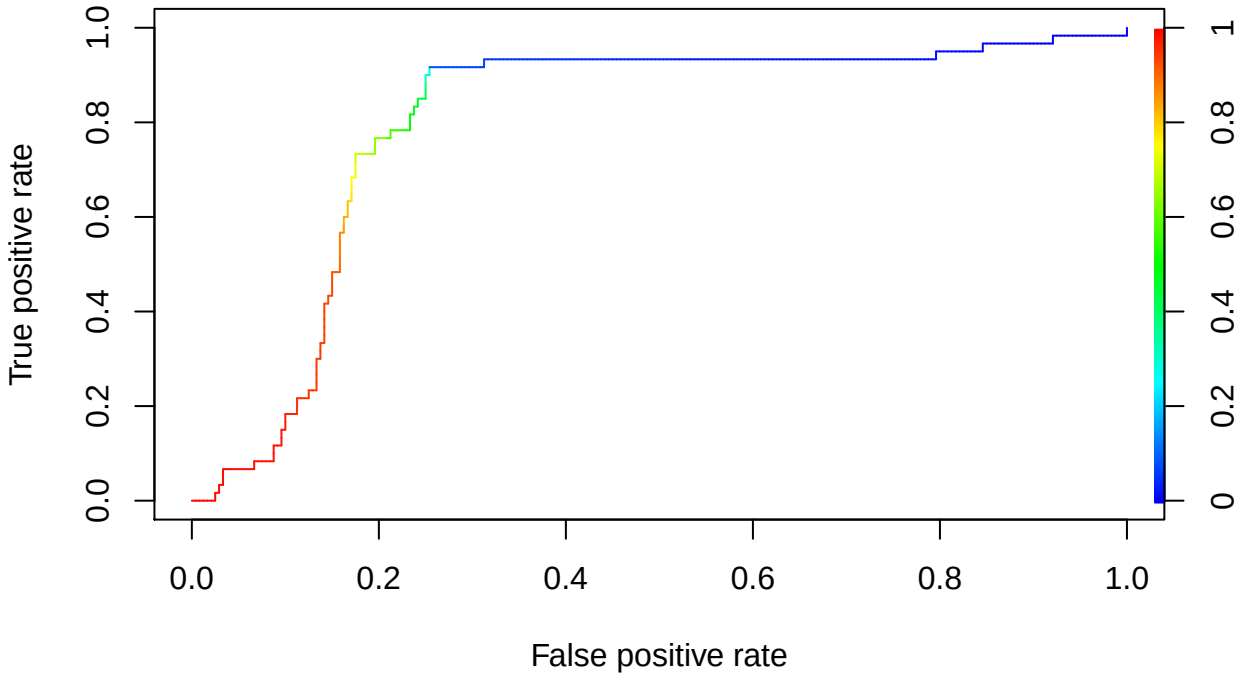



```
auroc2 <- performance(pred2, measure="auc")
auroc_value2 <- auroc2@y.values[[1]]
auroc_value2
```

```
## [1] 0.6038194
```

ROC Curve and AUROC values for prediction based on all available variables

```
pred3 <- prediction(prediction3, testing_data$Occupancy)
perf3 <- performance(pred3, measure="tpr", x.measure = "fpr")
plot(perf3,colorize=TRUE)
```



```
auroc3 <- performance(pred3, measure="auc")
auroc_value3 <- auroc3@y.values[[1]]
auroc_value3
```

```
## [1] 0.7977778
```

From the values of AUROC and ROC curves, we can see that the predictive performance of logistic regression model to predict the occupancy is best when all features are assessed; therefore the AUROC value of the last model is largest at 0.797778 and the area under the ROC curve is largest for that model hence it is the best predictive model amongst the 3 we observed here.