PARTH RATHOD

CSP - 571 Fall 2021

A20458817

Recitation Answers

Chapter 4

Question 4)

a) Feature X is uniformly distributed on [0,1] and range of observation is within 10%. Hence available observations we can use for predictions is:

```
For X [0.55,0.65] = ((0.65 - 0.55) / (1-0)) * 100
= 10%
```

b) We are given with the set of observations with P= 2 features X1 and X2 that are also uniformly distributed over the same range such that (X1, X2) € [0,1] X [0, 1]

```
So, the fraction of available observations that we will use to make
prediction is given by:
= (10%) * (10%)
= 1%
```

c) We are given with the set of observations with P= 100 features and all of which are uniformly distributed over the range between 0 and 1.

```
So, the fraction of available observations that we will use to make
prediction is given by:
= (0.1)^100 * 100
= (10)^-98 %
```

d) From the above observations we can conclude that as the number of features increases, the percentage of observations that are used to predict KNN becomes very small. Hence, more features leads to fewer neighbors.

e)

```
For P= 1 => length = (0.10)
For P= 2 => length = (0.10)^1/2 = 0.316
For P= 100 => length = (0.10)^1/100 = 0.977
```

From the above, we can conclude that when we wish to make predictions for the test observations that contains on average 10% of the training observations, then if we have large number of features, it will be better to include all the features.

Question 6)

a)

Logistic Regression with multiple variables is given by:-

```
p(X) = e ^ (β0 + X1β1 + x2β2) / 1 + e ^ (β0 + X1β1 + X2β2)

β0 = -6, β1 = 0.05, β2 = 1, e = 2.71828

X1= 40 , X2 = 3.5
Substituting in Equation gives
p(X) = 37.755
```

b)

Logistic Regression with multiple variables is given by:-

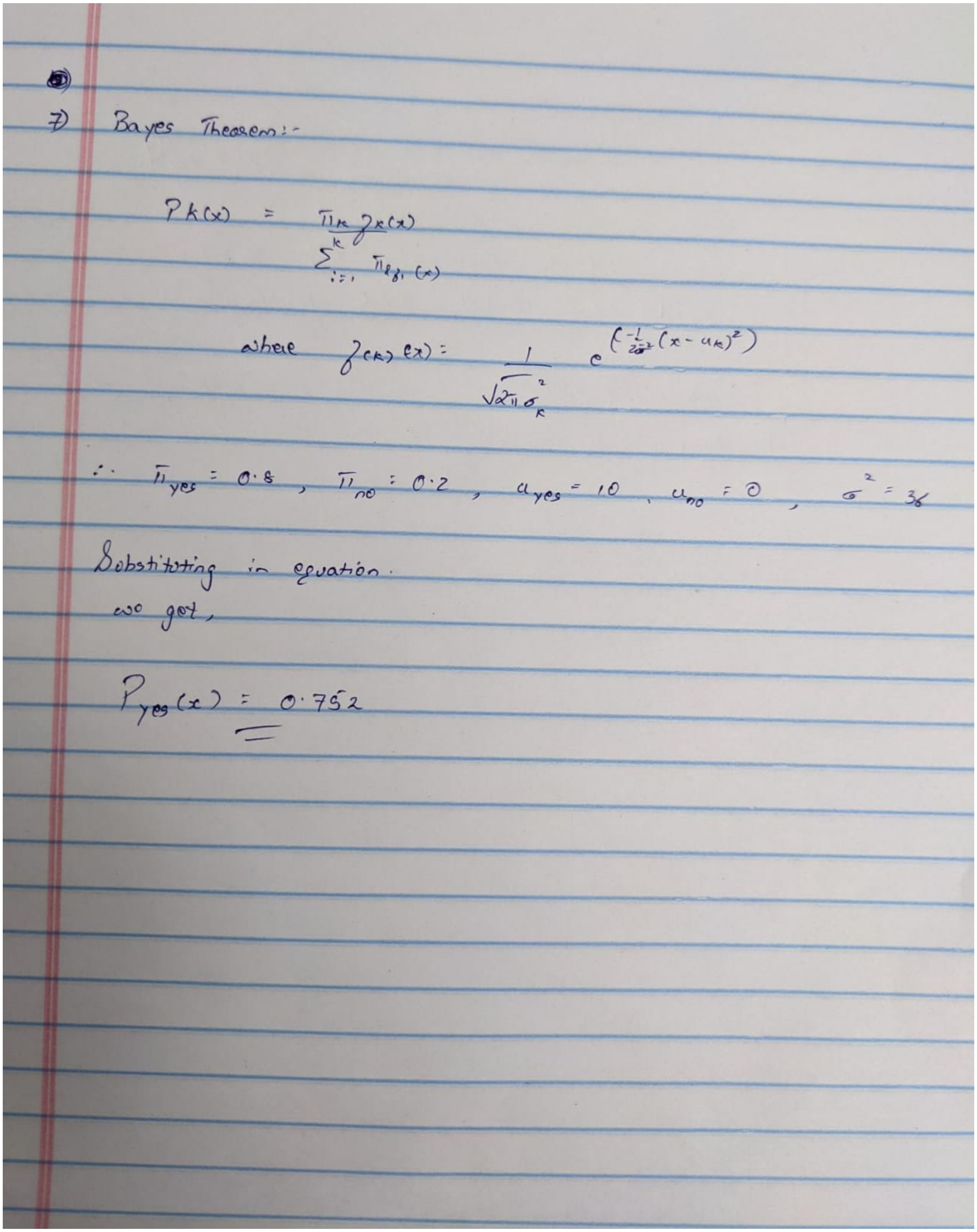
```
p(X) = e ^ (β0 + X1β1 + x2β2) / 1 + e ^ (β0 + X1β1 + X2β2)

β0 = -6, β1 = 0.05, β2 = 1, e = 2.71828

P(x) = 0.5 , X2 = 3.5

Substituting in equation gives
X1 = 50hrs
```

Question 7)



7) Bayes Theorem:-

$$P_k(x) = \frac{\pi_k f_k(x)}{\sum_{i=1}^{k} \pi_l f_l(x)}$$

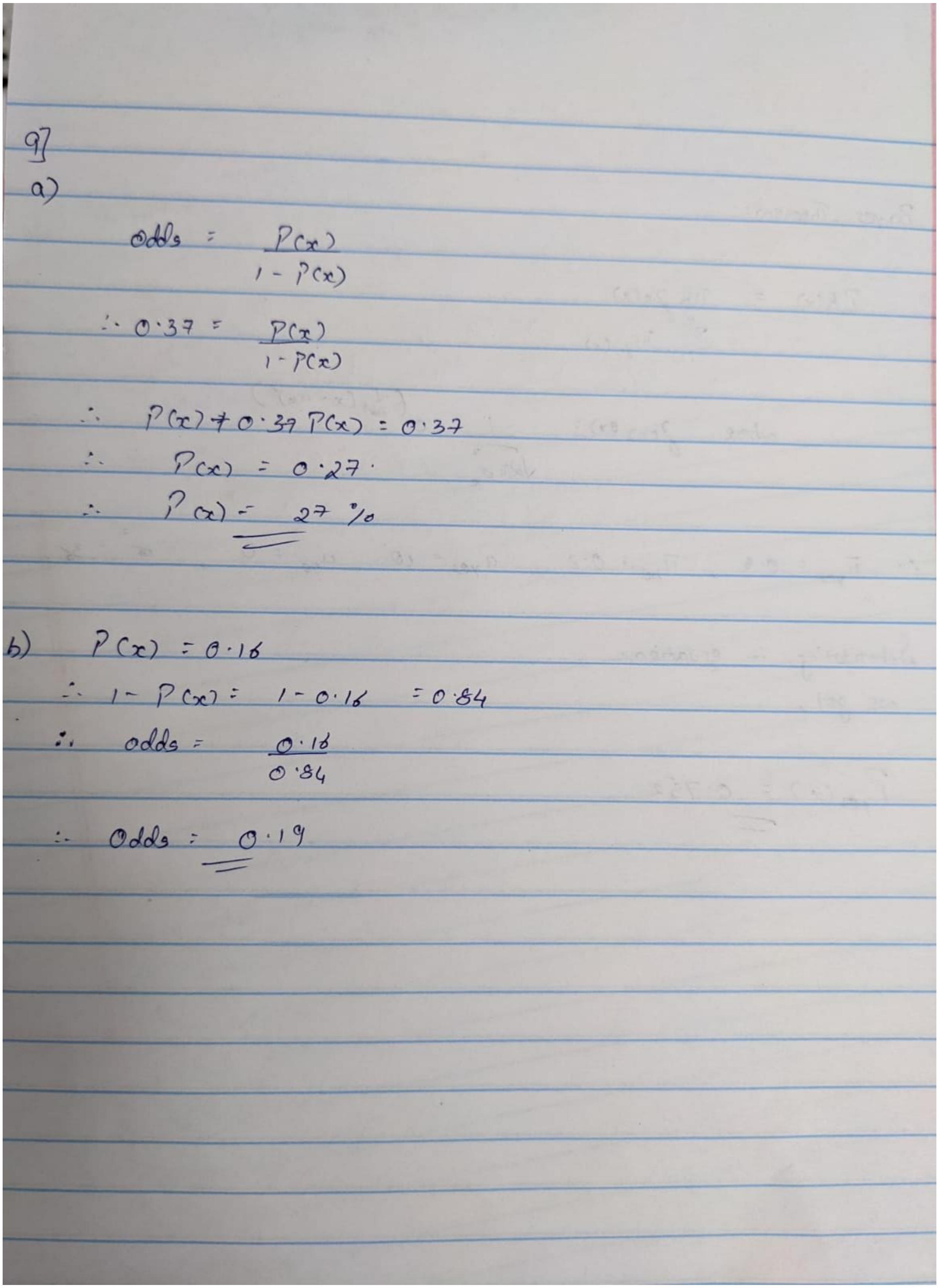
where $f_{(k)}(x) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{\left(-\frac{1}{2\sigma^2}(x-u_k)^2\right)}$

$\therefore$ $\pi_{yes} = 0.8$, $\pi_{no} = 0.2$, $u_{yes} = 10$, $u_{no} = 0$, $\sigma^2 = 36$

Sobstituting in equation.
we got,

$$\underline{P_{yes}(x) = 0.752}$$

Question 9)



9]

a)

$$\text{odds} = \frac{P(x)}{1 - P(x)}$$

$$\therefore 0\cdot37 = \frac{P(x)}{1 - P(x)}$$

$$\therefore P(x) + 0\cdot37\,P(x) = 0\cdot37$$

$$\therefore P(x) = 0\cdot27.$$

$$\therefore \underline{\underline{P(x) = 27\,\%}}$$

b) $P(x) = 0\cdot16$

$$\therefore 1 - P(x) = 1 - 0\cdot16 = 0\cdot84$$

$$\therefore \text{odds} = \frac{0\cdot16}{0\cdot84}$$

$$\therefore \text{Odds} = \underline{\underline{0\cdot19}}$$

Chapter 5

Question 2)

a)

```
Given: total number of observations = n
Since bootstrap allows sampling with replacement
Every observation in original sample is independent and has equal
probability to appear in each bootstrap observation
Probability that first bootstrap observation is the jth observation from
original sample is = 1/n
Probability that first bootstrap observation is not the jth observation
from original sample is = 1 – (1/n)
```

b)

```
Given: total number of observations = n
Since bootstrap allows sampling with replacement
Every observation in original sample has equal probability to appear in
each bootstrap observation
Probability that Second bootstrap observation is the jth observation from
original sample is =  1/n
Probability that Second bootstrap observation is not the jth observation
from original sample is = 1 – (1/n)
```

c)

```
p(jth observation is not in the first bootstrap sample) = 1 – ( 1/𝑛)
p(jth observation is not in the second bootstrap sample) = 1 – (1/n)
 :
p(jth observation is not in the nth bootstrap sample) = 1 –(1/n)

P(jth observation is not in the bootstrap sample) = (1 – (1/n)) * ( 1 -
(1/n)) * upto n times
                                                  = (1 -  (1/n)) ^ n
```

d)

```
From above answer we can replace n by 5 and get the answer
P(jth observation is in the bootstrap sample) = ( 1 - (1/5) ) ^ 5
                                              = 0.67
```

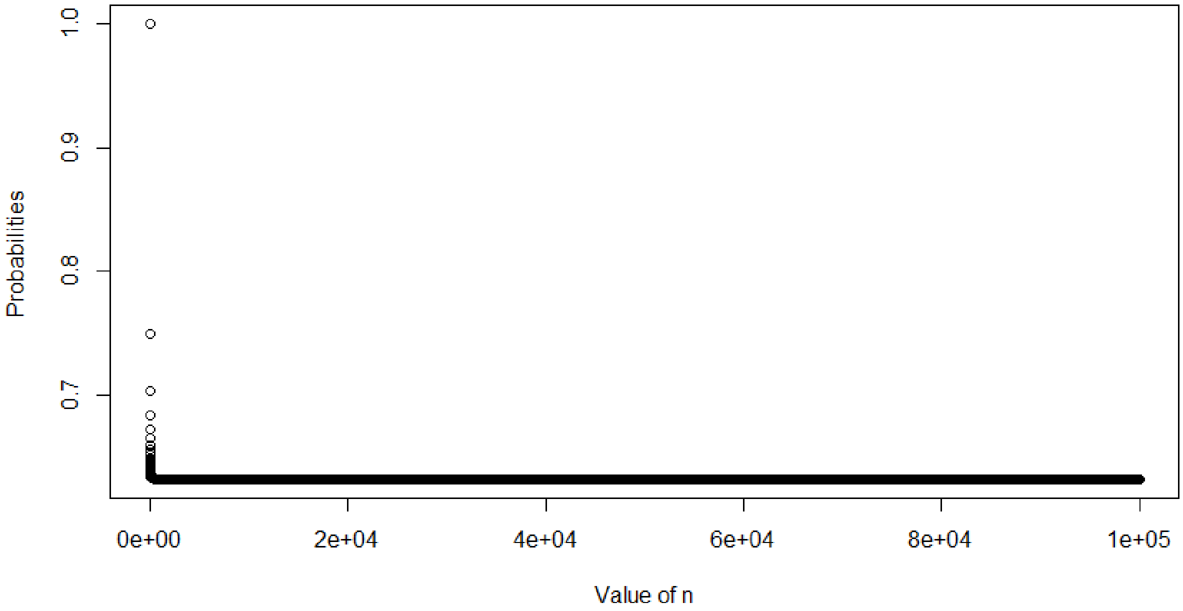
e)

```
P(jth observation is in the bootstrap sample) = 1 - (1 - (1/n))^n
                                              = 1 - (1 - (1/100))^100
                                              = 0.634
```

f)

```
P(jth observation is in the bootstrap sample) = 1 - (1 - (1/n))^n
                                              = 1 - (1 - (1/1000))^1000
                                              = 0.632
```

g) plot(1:100000, 1-(1-1/1:100000)^(1:100000), xlab= "Value of n",ylab ="Probabilities")



Probability quickly reaches to 0.62 and remains constant as n tends to infinity

h)

```
data <- rep(NA, 10000)
for (i in 1:10000)
{
  data[i] <- sum(sample(1:100, rep = TRUE) == 4) > 0
}
mean(data)

Answer:- 0.6289
```

Question 3)

a)

In k-fold cross validation,set of observations is randomly divided into k groups of approx equal size. The first fold is treated as validation set and remaining k-1 as where the fit method is used. The MSE is then computed. This process is repeated k times, each time a different group of observation is treated as validation set. The k-fold CV is computed by doing average of these values, CV (k) = 1/k * Summation(MSE (i) for i 1 to k)

b)

Validation set Approach:

Advantages:

This approach is conceptually simple.

This approach is easy to implement.

Disadvantages:

Validation estimate of test error rate can be highly variable, depends on which observations are included in training set and which observations are included in validation set.

In this approach, only observations which are included in training set are used to fit the model and model can perform worse if fewer.

LOOCV:

Advantages:

The LOOCV cross-validation method is a subset of k-fold cross-validation with k=n.

In this technique, each observation is only considered once in a validation set and (n-1) times in a training set.

This approach gives approximately unbiased estimates of the test error since each training set contain (n -1) observations.

Disadvantages:

It necessitates fitting the potentially computationally costly model n times, as opposed to k-fold cross-validation, which necessitates fitting the model just k times, where k < n.

This method has a greater variance than k-fold cross-validation (since we are averaging the outputs of n fitted models trained on a virtually similar set of observations, these outputs are highly correlated, and the mean of these outputs is higher).

#Problem 1

```
library(readr)
library(data.table)
library(corrplot)
```

```
## corrplot 0.90 loaded
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(ROCR)
```

```
abaloneURl = "https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data"
abaloneData = fread(abaloneURl, header = FALSE)
abaloneHeader = c("Sex","Length","Diameter","Height","Whole weight","Shucked weight","Viscera we
ight","Shell weight","Rings")
colnames(abaloneData) = abaloneHeader
#Display Summary:
summary(abaloneData)
```

```
##      Sex                Length         Diameter          Height
##  Length:4177        Min.   :0.075   Min.   :0.0550   Min.   :0.0000
##  Class :character   1st Qu.:0.450   1st Qu.:0.3500   1st Qu.:0.1150
##  Mode  :character   Median :0.545   Median :0.4250   Median :0.1400
##                     Mean   :0.524   Mean   :0.4079   Mean   :0.1395
##                     3rd Qu.:0.615   3rd Qu.:0.4800   3rd Qu.:0.1650
##                     Max.   :0.815   Max.   :0.6500   Max.   :1.1300
##   Whole weight    Shucked weight   Viscera weight    Shell weight
##  Min.   :0.0020   Min.   :0.0010   Min.   :0.0005   Min.   :0.0015
##  1st Qu.:0.4415   1st Qu.:0.1860   1st Qu.:0.0935   1st Qu.:0.1300
##  Median :0.7995   Median :0.3360   Median :0.1710   Median :0.2340
##  Mean   :0.8287   Mean   :0.3594   Mean   :0.1806   Mean   :0.2388
##  3rd Qu.:1.1530   3rd Qu.:0.5020   3rd Qu.:0.2530   3rd Qu.:0.3290
##  Max.   :2.8255   Max.   :1.4880   Max.   :0.7600   Max.   :1.0050
##      Rings
##  Min.   : 1.000
##  1st Qu.: 8.000
##  Median : 9.000
##  Mean   : 9.934
##  3rd Qu.:11.000
##  Max.   :29.000
```

```
#Returns the indices when the following condition is true in which()
condition = which(abaloneData$Sex!="I")
#All the Rows with Sex = I are removed
abaloneData2 = abaloneData[condition]
#check for all sex categories in new dataset:
unique(abaloneData2$Sex)
```

```
## [1] "M" "F"
```

```
#As the response variable need to be numeric, converting categorical Sex to a factor
abaloneData2$Sex = factor(abaloneData2$Sex)
#check for the change:
str(abaloneData2)
```

```
## Classes 'data.table' and 'data.frame':   2835 obs. of  9 variables:
##  $ Sex           : Factor w/ 2 levels "F","M": 2 2 1 2 1 1 2 1 1 2 ...
##  $ Length        : num  0.455 0.35 0.53 0.44 0.53 0.545 0.475 0.55 0.525 0.43 ...
##  $ Diameter      : num  0.365 0.265 0.42 0.365 0.415 0.425 0.37 0.44 0.38 0.35 ...
##  $ Height        : num  0.095 0.09 0.135 0.125 0.15 0.125 0.125 0.15 0.14 0.11 ...
##  $ Whole weight  : num  0.514 0.226 0.677 0.516 0.777 ...
##  $ Shucked weight: num  0.2245 0.0995 0.2565 0.2155 0.237 ...
##  $ Viscera weight: num  0.101 0.0485 0.1415 0.114 0.1415 ...
##  $ Shell weight  : num  0.15 0.07 0.21 0.155 0.33 0.26 0.165 0.32 0.21 0.135 ...
##  $ Rings         : int  15 7 9 10 20 16 9 19 14 10 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
#Creating 80-20 Training Testing Split, createDataPartition() returns the indices
trainIndex = createDataPartition(y = abaloneData2$Sex, p = 0.8, list = FALSE)
#Training data
trainData = abaloneData2[trainIndex,]
#Testing data (note the minus sign)
testData = abaloneData2[-trainIndex,]
```

```
#Predicting Sex using glm
model <- glm(Sex~.,family=binomial,data=trainData)
#Summary of our model
summary(model)
```

```
##
## Call:
## glm(formula = Sex ~ ., family = binomial, data = trainData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8422  -1.2014   0.8803   1.1195   1.4946
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        2.746856   0.517707   5.306 1.12e-07 ***
## Length            -1.404160   2.281488  -0.615  0.53825
## Diameter          -5.595478   2.738485  -2.043  0.04103 *
## Height            -1.457616   2.549472  -0.572  0.56750
## `Whole weight`     0.239246   0.845490   0.283  0.77720
## `Shucked weight`   2.877994   1.007477   2.857  0.00428 **
## `Viscera weight` -2.167277   1.420058  -1.526  0.12696
## `Shell weight`    -0.230110   1.320678  -0.174  0.86168
## Rings             -0.001951   0.017841  -0.109  0.91290
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3131.7  on 2268  degrees of freedom
## Residual deviance: 3072.8  on 2260  degrees of freedom
## AIC: 3090.8
##
## Number of Fisher Scoring iterations: 4
```

```
#Coefficients of our model
coef(model)
```

```
##      (Intercept)           Length         Diameter           Height
##      2.746855671     -1.404159621     -5.595477554     -1.457616395
##   `Whole weight` `Shucked weight` `Viscera weight`   `Shell weight`
##      0.239245893      2.877993519     -2.167277467     -0.230110287
##            Rings
##     -0.001951414
```

```
#Confidence Interval
confint(model)
```

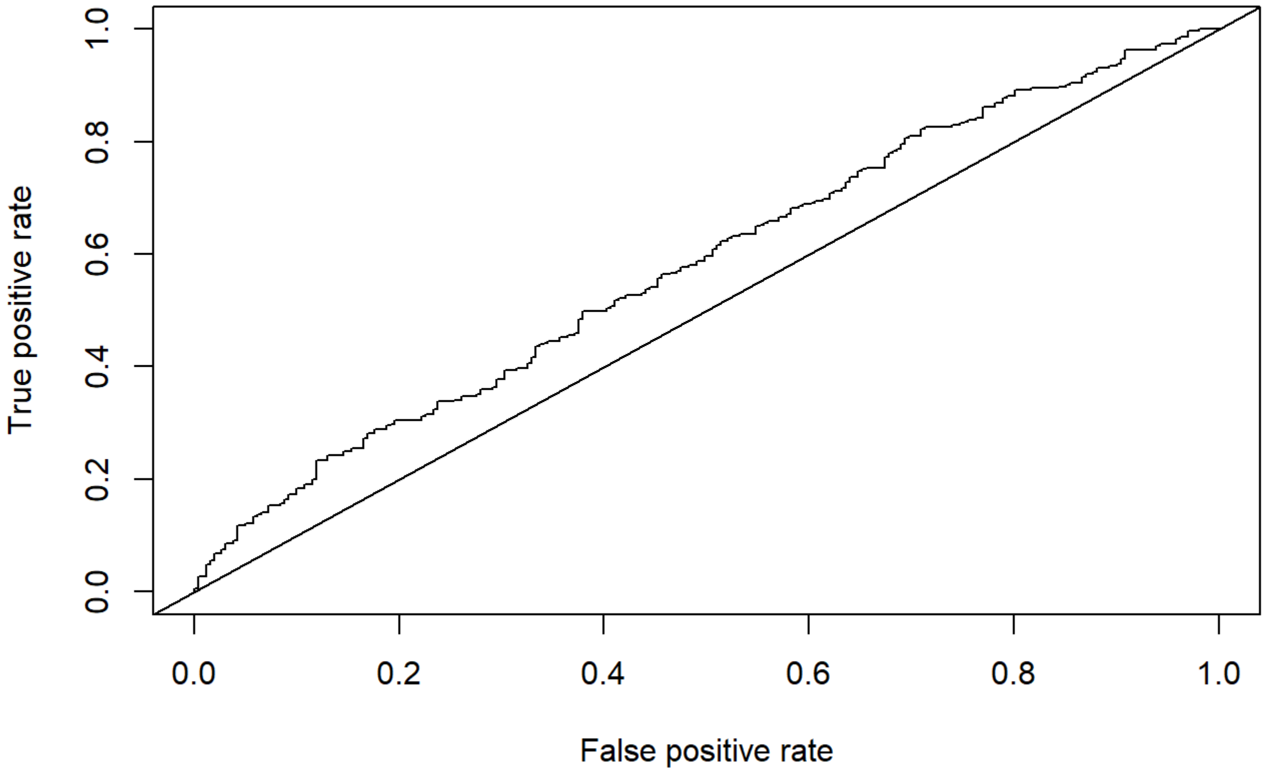
```
## Waiting for profiling to be done...
##                         2.5 %      97.5 %
## (Intercept)        1.74607346  3.77677275
## Length            -5.87792848  3.07173332
## Diameter         -10.98329493 -0.23982555
## Height            -6.46211360  3.60035582
## `Whole weight`    -1.41975707  1.90635573
## `Shucked weight`   0.90691967  4.86416179
## `Viscera weight`  -4.96292146  0.61144730
## `Shell weight`    -2.83273830  2.35616736
## Rings             -0.03694459  0.03303912
```

From the above observations for confidence interval, we can see that, all the predictors, except the “Shucked Weight” contain 0 within their confidence interval range which is in line with the basic assumption of Null Hypothesis which says, No relationship between X and Y. The only attribute with a substantially low p-value is “Shucked Weight” with a p-value of 0.00105 and all the other attributes have a high p-value. Thus, we can conclude that, there is no relationship between the predictors and the response variable (except the “Shucked Weight”) and Null Hypothesis holds true for all the predictors except the “Shucked Weight”. Hence Shucked Weight is the only predictor which is relevant.

```
#Predict [By setting the parameter type='response', R will output probabilities in the form of P
(y=1|X)]
probs = predict(model, testData, type = "response")
#Using a 50% cut-off factor i.e probabilities > 0.5 are Males and rest are Females
resultSet = ifelse(probs > 0.5,"M","F")
resultSet2 = factor(resultSet)
#Creating a confusion matrix
confusionMatrix(resultSet2, testData$Sex)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   F   M
##          F 101  93
##          M 160 212
##                                          
##                Accuracy : 0.553          
##                  95% CI : (0.511, 0.5945)
##     No Information Rate : 0.5389         
##     P-Value [Acc > NIR] : 0.2638         
##                                          
##                   Kappa : 0.0836         
##                                          
##  Mcnemar's Test P-Value : 3.334e-05      
##                                          
##             Sensitivity : 0.3870         
##             Specificity : 0.6951         
##          Pos Pred Value : 0.5206         
##          Neg Pred Value : 0.5699         
##              Prevalence : 0.4611         
##          Detection Rate : 0.1784         
##    Detection Prevalence : 0.3428         
##       Balanced Accuracy : 0.5410         
##                                          
##        'Positive' Class : F              
## 
```

```
#Plotting the ROC Curve
roc.pred = prediction(probs,testData$Sex)
roc.perf = performance(roc.pred, measure = "tpr", x.measure = "fpr")
plot(roc.perf)
abline(0,1)
```



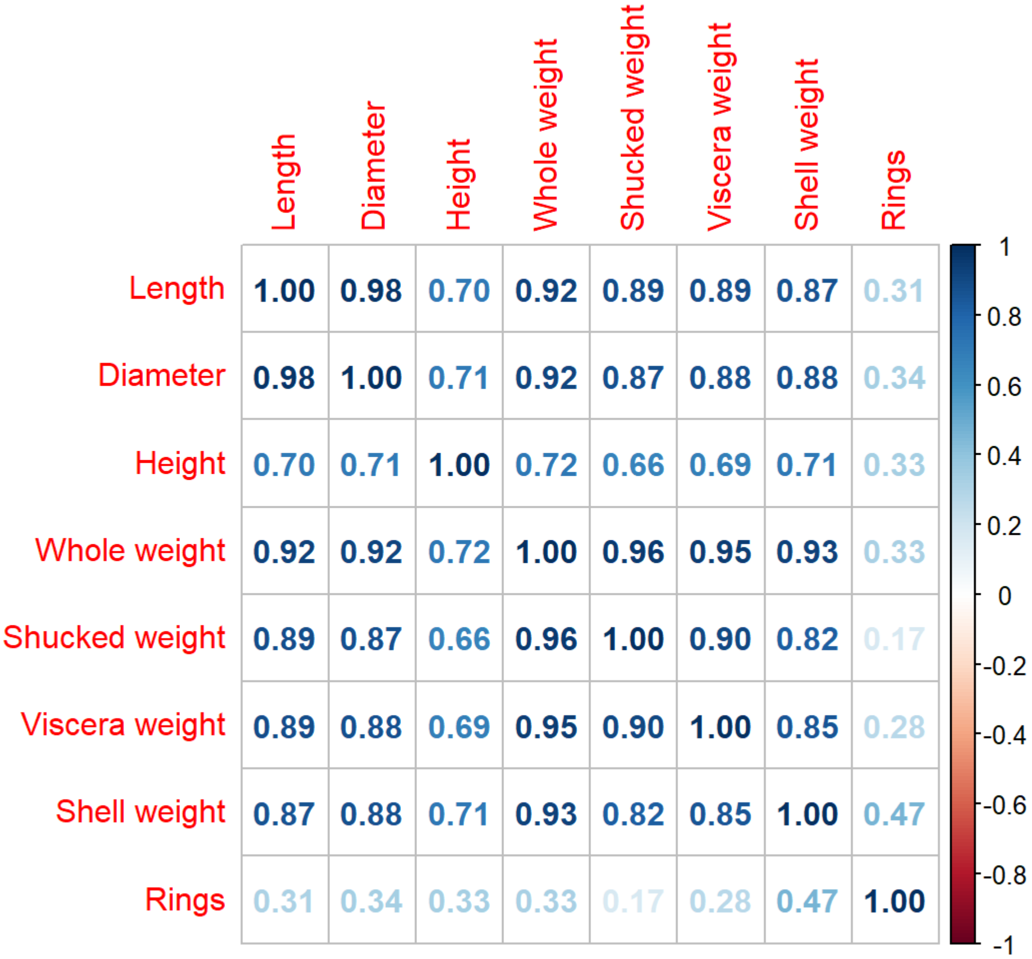
```
auc.perf = performance(roc.pred, measure = "auc")
cat("Area Under the Curve: ")
```

```
## Area Under the Curve:
```

```
auc.perf@y.values
```

```
## [[1]]
## [1] 0.5799259
```

```
#Plotting the correlations between the predictors
cm = cor(abaloneData2[,-1])
corrplot(cm, method = "number")
```



	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
Length	1.00	0.98	0.70	0.92	0.89	0.89	0.87	0.31
Diameter	0.98	1.00	0.71	0.92	0.87	0.88	0.88	0.34
Height	0.70	0.71	1.00	0.72	0.66	0.69	0.71	0.33
Whole weight	0.92	0.92	0.72	1.00	0.96	0.95	0.93	0.33
Shucked weight	0.89	0.87	0.66	0.96	1.00	0.90	0.82	0.17
Viscera weight	0.89	0.88	0.69	0.95	0.90	1.00	0.85	0.28
Shell weight	0.87	0.88	0.71	0.93	0.82	0.85	1.00	0.47
Rings	0.31	0.34	0.33	0.33	0.17	0.28	0.47	1.00

The above figure clearly shows that there is a positive linear connection for all factors. Only the Rings predictor has a weak uphill (positive) association, whereas the others have a high uphill (positive) relationship.

#Problem 2

```r
library(data.table) #Data Import
library(e1071)      #Naive Bayes
```

```r
#Setting up the URL for data import
mushroomURL = "https://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/agaricus-lepiota.data"
mushroomData = fread(mushroomURL,header=FALSE)
#Adding the headers
mushroomHeader = c("Class","cap-shape","cap-surface","cap-color","bruises","odor","gill-attachment","gill-spacing","gill-size","gill-color","stalk-shape","stalk-root","stalk-surface-above-ring","stalk-surface-below-ring","stalk-color-above-ring","stalk-color-below-ring","veil-type","veil-color","ring-number","ring-type","spore-print-color","population","habitat")
colnames(mushroomData) = mushroomHeader
#Display Summary:
summary(mushroomData)
```

```
##     Class            cap-shape         cap-surface         cap-color        
##  Length:8124        Length:8124        Length:8124        Length:8124       
##  Class :character   Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character  
##    bruises              odor           gill-attachment    gill-spacing      
##  Length:8124        Length:8124        Length:8124        Length:8124       
##  Class :character   Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character  
##   gill-size          gill-color        stalk-shape         stalk-root       
##  Length:8124        Length:8124        Length:8124        Length:8124       
##  Class :character   Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character  
##  stalk-surface-above-ring stalk-surface-below-ring stalk-color-above-ring
##  Length:8124              Length:8124              Length:8124           
##  Class :character         Class :character         Class :character      
##  Mode  :character         Mode  :character         Mode  :character      
##  stalk-color-below-ring  veil-type          veil-color       
##  Length:8124            Length:8124        Length:8124       
##  Class :character       Class :character   Class :character  
##  Mode  :character       Mode  :character   Mode  :character  
##  ring-number         ring-type         spore-print-color   population       
##  Length:8124        Length:8124        Length:8124        Length:8124       
##  Class :character   Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character  
##    habitat         
##  Length:8124       
##  Class :character  
##  Mode  :character  
```

```
#Class Distribution
table(mushroomData$Class)
```

```
## 
##    e    p 
## 4208 3916
```

```
#Converting Class Attribute to a factor
mushroomData$Class = factor(mushroomData$Class)
#Dimensions
dim(mushroomData)
```

```
## [1] 8124   23
```

```
#Structure
str(mushroomData)
```

```
## Classes 'data.table' and 'data.frame':   8124 obs. of  23 variables:
##  $ Class                   : Factor w/ 2 levels "e","p": 2 1 1 2 1 1 1 1 2 1 ...
##  $ cap-shape               : chr  "x" "x" "b" "x" ...
##  $ cap-surface             : chr  "s" "s" "s" "y" ...
##  $ cap-color               : chr  "n" "y" "w" "w" ...
##  $ bruises                 : chr  "t" "t" "t" "t" ...
##  $ odor                    : chr  "p" "a" "l" "p" ...
##  $ gill-attachment         : chr  "f" "f" "f" "f" ...
##  $ gill-spacing            : chr  "c" "c" "c" "c" ...
##  $ gill-size               : chr  "n" "b" "b" "n" ...
##  $ gill-color              : chr  "k" "k" "n" "n" ...
##  $ stalk-shape             : chr  "e" "e" "e" "e" ...
##  $ stalk-root              : chr  "e" "c" "c" "e" ...
##  $ stalk-surface-above-ring: chr  "s" "s" "s" "s" ...
##  $ stalk-surface-below-ring: chr  "s" "s" "s" "s" ...
##  $ stalk-color-above-ring  : chr  "w" "w" "w" "w" ...
##  $ stalk-color-below-ring  : chr  "w" "w" "w" "w" ...
##  $ veil-type               : chr  "p" "p" "p" "p" ...
##  $ veil-color              : chr  "w" "w" "w" "w" ...
##  $ ring-number             : chr  "o" "o" "o" "o" ...
##  $ ring-type               : chr  "p" "p" "p" "p" ...
##  $ spore-print-color       : chr  "k" "n" "n" "k" ...
##  $ population              : chr  "s" "n" "n" "s" ...
##  $ habitat                 : chr  "u" "g" "m" "u" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
#Finding the number of missing values
cat("Number of missing values = ",sum(mushroomData=="?"))
```

```
## Number of missing values =  2480
```

```
#New dataset with removed missing values
mushroomData2 = mushroomData[mushroomData$`stalk-root`!="?"]
```

There is no damage in removing the observations with missing values because we have plenty after omitting the ones with missing values.

```
#Creating a split
trainSize = floor(0.80*nrow(mushroomData2))
trainIndex = sample(nrow(mushroomData2), size = trainSize)
trainData = mushroomData2[trainIndex,]
testData = mushroomData2[-trainIndex,]
```

```
#Naive Bayes
model = naiveBayes(trainData[,-1],trainData$Class)
#Prediction on Testing Data
testPred = predict(model,testData[,-1])
#Prediction on Training Data
trainPred = predict(model,trainData[,-1])
#Accuracy of Testing Model
cat("Accuracy of Testing Model: ",mean(testPred == testData$Class)*100,"%")
```

```
## Accuracy of Testing Model:  95.39415 %
```

```
#Accuracy of Training Model
cat("Accuracy of Training Model: ",mean(trainPred == trainData$Class)*100,"%")
```

```
## Accuracy of Training Model:  95.43743 %
```

```
#Confusion Matrix
table(testPred, testData$Class)
```

```
##
## testPred   e   p
##        e 691  47
##        p   5 386
```

From the above the confusion matrix we have the following: TP = 691, FP = 47, FN = 5 and TN = 386

#Problem 3

```
library(readr)
library(data.table)
library(caret)
```

```
hd_URL = "http://archive.ics.uci.edu/ml/machine-learning-databases/00243/yacht_hydrodynamics.dat
a"
hd_data = fread(hd_URL, header = FALSE)

hd_Header = c("longitudinalPos","prismaticCoef","LDR","BDR","LBR","froudeNo","Residuary")
colnames(hd_data) = hd_Header

#Direct use of attribute names in code
attach(hd_data)
```

```
#Creating 80-20 Training Testing Split, createDataPartition() returns the indices
trainIndex = createDataPartition(y = hd_data$Residuary , p = 0.8, list = FALSE)

#Training data
trainData = hd_data[trainIndex,]

#Testing data (note the minus sign)
testData = hd_data[-trainIndex,]
```

```
#Training fit for linear model
linearModel1 = lm(hd_data$Residuary~hd_data$longitudinalPos + hd_data$prismaticCoef + hd_data$LD
R + hd_data$BDR + hd_data$BDR + hd_data$LBR +hd_data$froudeNo, data = trainData)
```

```
# Function to compute MSE
MSE = function(yActual, yPred)
{
  return (mean((yActual - yPred)^2))
}
```

```
mse1 = MSE(hd_data$Residuary, linearModel1$fitted.values )

# Summarize the results
cat("Training MSE: ", mse1)
```

```
## Training MSE:  78.45015
```

```
cat("Training RMSE: ", sqrt(mse1))
```

```
## Training RMSE:  8.857209
```

```
cat("Training R-squared: ",summary(linearModel1)$r.sq)
```

```
## Training R-squared:  0.6575638
```

```
# Define training control
train.control =  trainControl(method = "boot", number = 1000)

# Train the model
linearModel2 = train(Residuary~., data = trainData, method = "lm", trControl = train.control)
```

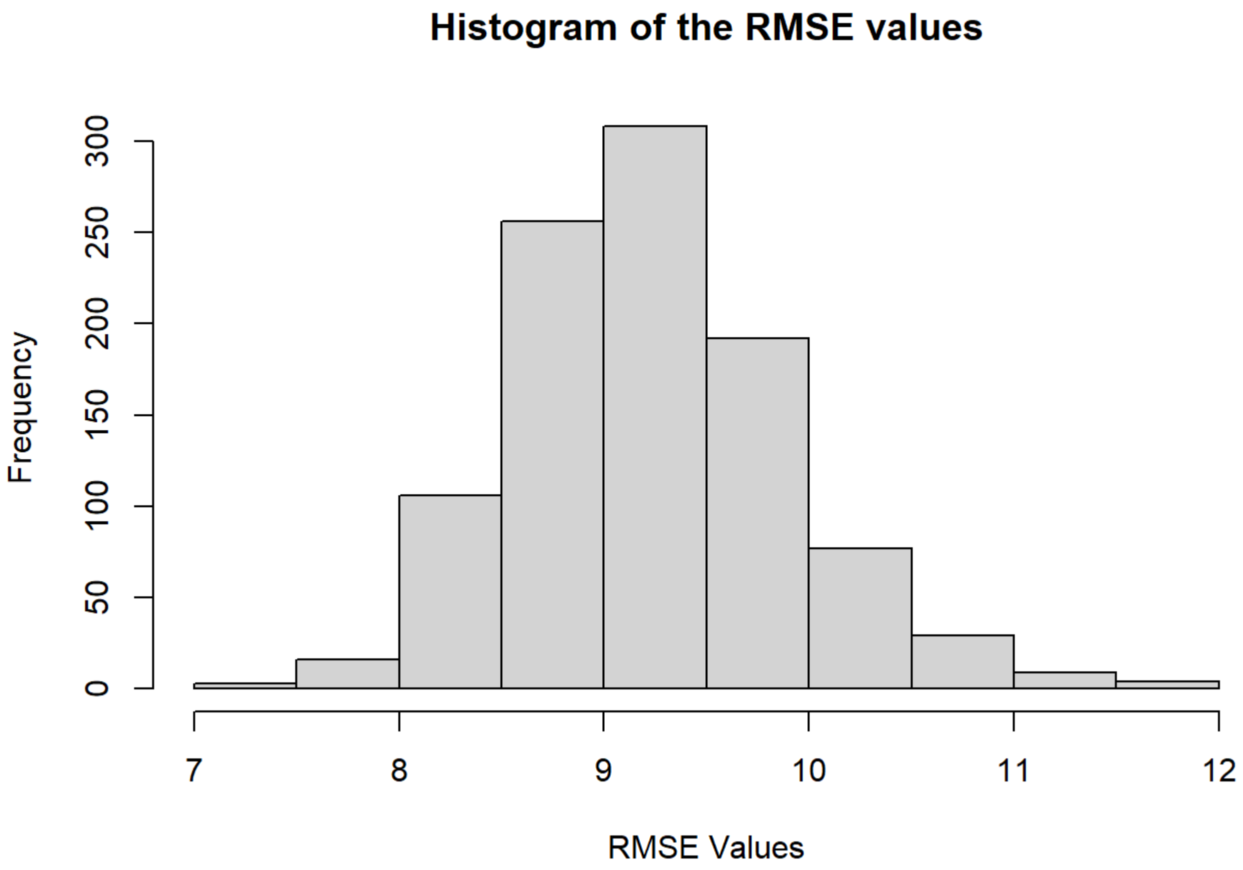
```
# 5 Point Summary for resulting RMSE for each resample
summary(linearModel2$resample$RMSE)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   7.400   8.802   9.196   9.245   9.619  11.801
```

```
# 5 Point Summary for resulting R-Squared for each resample
summary(linearModel2$resample$Rsquared)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.5322  0.6239  0.6431  0.6425  0.6617  0.7235
```

```
# Histogram of RMSE values
hist(linearModel2$resample$RMSE, xlab = "RMSE Values", main = "Histogram of the RMSE values")
```



```
# Calculating the MSE from RMSE
mse2 = mean(linearModel2$resample$RMSE)^2

# Summarize the results
cat("Training Mean MSE (Bootstrap): ", mse2)
```

```
## Training Mean MSE (Bootstrap):  85.46133
```

```
cat("Training Mean RMSE (Bootstrap): ", mean(linearModel2$resample$RMSE))
```

```
## Training Mean RMSE (Bootstrap):  9.24453
```

```
cat("Training Mean R-squared (Bootstrap): ",mean(linearModel2$resample$Rsquared))
```

```
## Training Mean R-squared (Bootstrap):  0.6424873
```

```
predVals_boot = predict(linearModel2,testData)
mse3 = MSE(testData$Residuary, predVals_boot)
```

```
RSS = function (yActual, yPred)
{
  return (sum((yActual - yPred)^2))
}

TSS = function (yActual)
{
  return (sum((yActual - mean(yActual))^2))
}
```

```
rss = RSS(testData$Residuary, predVals_boot)
tss = TSS(testData$Residuary)

# Summarize the results
cat("Testing MSE (Bootstrap): ", mse3)
```

```
## Testing MSE (Bootstrap):  80.36479
```

```
cat("Testing RMSE (Bootstrap): ", sqrt(mse3))
```

```
## Testing RMSE (Bootstrap):  8.964641
```

```
cat("Testing Mean R-squared (Bootstrap): ",1 - (rss/tss))
```

```
## Testing Mean R-squared (Bootstrap):  0.6435649
```

On the test set, there is no difference in performance between the original and bootstrap models.

#Problem 4

```
library(readr)
library(data.table)
library(caret)
```

```
gcd_URL = "https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.data-
numeric"
gc_data = fread(gcd_URL, header = FALSE)
```

```
#As the response variable need to be numeric, converting categorical V25 to a factor
gc_data$V25 = factor(gc_data$V25)
```

```
#Creating 80-20 Training Testing Split, createDataPartition() returns the indices
trainIndex = createDataPartition(y = gc_data$V25 , p = 0.8, list = FALSE)

#Training data
trainData = gc_data[trainIndex,]

#Testing data (note the minus sign)
testData = gc_data[-trainIndex,]
```

```
# Creating model for y = V25 using glm
logisticModel1 = glm(V25~.,family=binomial,data=trainData)
```

```
actualVals = trainData$V25

# Using a 50% cut-off factor i.e probabilities > 0.5 are 2 and rest are 1
fittedVals = ifelse(logisticModel1$fitted.values > 0.5,2,1)
fittedVals = factor(fittedVals)
```

```
# Confusion matrix
cm = confusionMatrix(fittedVals, trainData$V25)
cm
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   1   2
##          1 505 112
##          2  55 128
## 
##                Accuracy : 0.7912
##                  95% CI : (0.7614, 0.8189)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 3.653e-09
## 
##                   Kappa : 0.4668
## 
##  Mcnemar's Test P-Value : 1.468e-05
## 
##             Sensitivity : 0.9018
##             Specificity : 0.5333
##          Pos Pred Value : 0.8185
##          Neg Pred Value : 0.6995
##              Prevalence : 0.7000
##          Detection Rate : 0.6312
##    Detection Prevalence : 0.7712
##       Balanced Accuracy : 0.7176
## 
##        'Positive' Class : 1
## 
```

```
# Summarize the results
cat("Training Precision: ", cm$byClass[5] * 100, "%")
```

```
## Training Precision:  81.84765 %
```

```
cat("Training Recall: ", cm$byClass[6] * 100, "%")
```

```
## Training Recall:  90.17857 %
```

```
cat("Training F1-Score: ", cm$byClass[7] * 100, "%")
```

```
## Training F1-Score:  85.81138 %
```

```
# Predict [By setting the parameter type='response', R will output probabilities in the form of
 P(y=1|X)]
probs = predict(logisticModel1, testData, type = "response")

#Using a 50% cut-off factor i.e probabilities > 0.5 are Males and rest are Females
fittedVals_test = ifelse(probs > 0.5,2,1)
fittedVals_test = factor(fittedVals_test)

# Confusion matrix
cm_test = confusionMatrix(fittedVals_test, testData$V25)
cm_test
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   1   2
##          1 122  29
##          2  18  31
##                                          
##                Accuracy : 0.765          
##                  95% CI : (0.7, 0.8219)  
##     No Information Rate : 0.7            
##     P-Value [Acc > NIR] : 0.02493        
##                                          
##                   Kappa : 0.4095         
##                                          
##  Mcnemar's Test P-Value : 0.14466        
##                                          
##             Sensitivity : 0.8714         
##             Specificity : 0.5167         
##          Pos Pred Value : 0.8079         
##          Neg Pred Value : 0.6327         
##              Prevalence : 0.7000         
##          Detection Rate : 0.6100         
##    Detection Prevalence : 0.7550         
##       Balanced Accuracy : 0.6940         
##                                          
##        'Positive' Class : 1              
## 
```

```
# Summarize the results
cat("Testing Precision: ", cm_test$byClass[5] * 100, "%")
```

```
## Testing Precision:  80.7947 %
```

```
cat("Testing Recall: ", cm_test$byClass[6] * 100, "%")
```

```
## Testing Recall:  87.14286 %
```

```
cat("Testing F1-Score: ", cm_test$byClass[7] * 100, "%")
```

```
## Testing F1-Score:  83.8488 %
```

#—————————Cross-Validation————————————

```
# Define training control
train.control =  trainControl(method = "cv", number = 10)

# Train the model
logisticModel2 = train(V25~., data = trainData, method = "glm", family = "binomial", trControl =
train.control)
```

```
fittedVals_cv = ifelse(logisticModel2$finalModel$fitted.values > 0.5,2,1)
fittedVals_cv = factor(fittedVals_cv)

# Confusion matrix
cm_cv = confusionMatrix(fittedVals_cv, trainData$V25)
cm_cv
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   1   2
##          1 505 112
##          2  55 128
##                                           
##                Accuracy : 0.7912          
##                  95% CI : (0.7614, 0.8189)
##     No Information Rate : 0.7             
##     P-Value [Acc > NIR] : 3.653e-09       
##                                           
##                   Kappa : 0.4668          
##                                           
##  Mcnemar's Test P-Value : 1.468e-05       
##                                           
##             Sensitivity : 0.9018          
##             Specificity : 0.5333          
##          Pos Pred Value : 0.8185          
##          Neg Pred Value : 0.6995          
##              Prevalence : 0.7000          
##          Detection Rate : 0.6312          
##    Detection Prevalence : 0.7712          
##       Balanced Accuracy : 0.7176          
##                                           
##        'Positive' Class : 1               
## 
```

```
# Summarize the results
cat("Training Precision with 10-fold CV: ", cm_cv$byClass[5] * 100, "%")
```

```
## Training Precision with 10-fold CV:  81.84765 %
```

```
cat("Training Recall with 10-fold CV: ", cm_cv$byClass[6] * 100, "%")
```

```
## Training Recall with 10-fold CV:  90.17857 %
```

```
cat("Training F1-Score with 10-fold CV: ", cm_cv$byClass[7] * 100, "%")
```

```
## Training F1-Score with 10-fold CV:  85.81138 %
```

```
# Predict [By setting the parameter type='response', R will output probabilities in the form of
 P(y=1|X)]
probs_cv = predict(logisticModel2, testData, type = "prob")

#Using a 50% cut-off factor i.e probabilities > 0.5 are Males and rest are Females
fittedVals_cv_test = ifelse(probs > 0.5,2,1)
fittedVals_cv_test = factor(fittedVals_test)

# Confusion matrix
cm_cv_test = confusionMatrix(fittedVals_test, testData$V25)
cm_cv_test
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   1   2
##          1 122  29
##          2  18  31
##                                           
##                Accuracy : 0.765           
##                  95% CI : (0.7, 0.8219)   
##     No Information Rate : 0.7             
##     P-Value [Acc > NIR] : 0.02493         
##                                           
##                   Kappa : 0.4095          
##                                           
##  Mcnemar's Test P-Value : 0.14466         
##                                           
##             Sensitivity : 0.8714          
##             Specificity : 0.5167          
##          Pos Pred Value : 0.8079          
##          Neg Pred Value : 0.6327          
##              Prevalence : 0.7000          
##          Detection Rate : 0.6100          
##    Detection Prevalence : 0.7550          
##       Balanced Accuracy : 0.6940          
##                                           
##        'Positive' Class : 1               
## 
```

```
# Summarize the results
cat("Testing Precision: ", cm_cv_test$byClass[5] * 100, "%")
```

```
## Testing Precision:  80.7947 %
```

```
cat("Testing Recall: ", cm_cv_test$byClass[6] * 100, "%")
```

```
## Testing Recall:  87.14286 %
```

```
cat("Testing F1-Score: ", cm_cv_test$byClass[7] * 100, "%")
```

```
## Testing F1-Score:  83.8488 %
```

On the test set, there is no difference in performance between the original and bootstrap models.