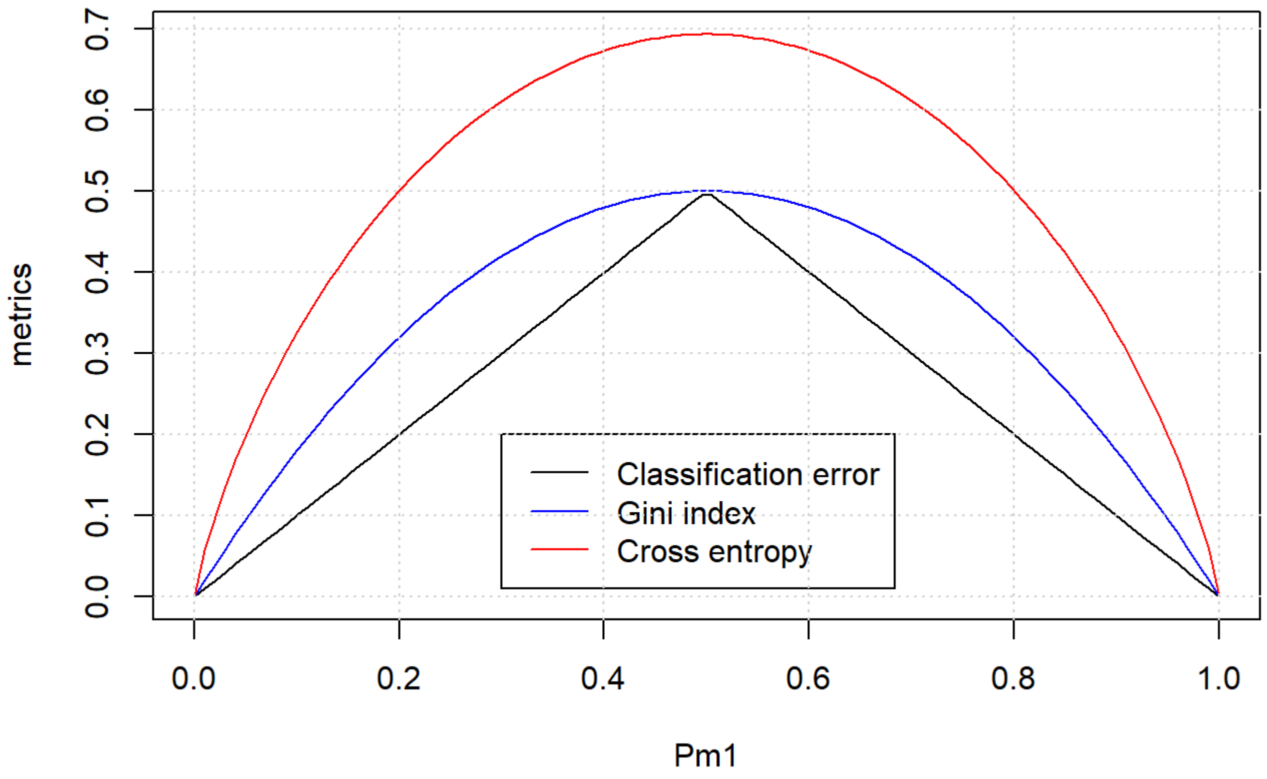
Chapter 8 Question 3

```
p1 = seq(0 + 1e-06, 1 - 1e-06, length.out = 100)
p2 = 1 - p1
class_error = 1 - apply(rbind(p1, p2), 2, max)
gini_index = p1 * (1 - p1) + p2 * (1 - p2)
cross_entropy = -(p1 * log(p1) + p2 * log(p2))
plot(p1, class_error, type = "l", col = "black", xlab = "Pm1", ylab = "Error
metrics", ylim = c(min(c(class_error, gini_index, cross_entropy)), max(class_error,
gini_index, cross_entropy)))
lines(p1, gini_index, col = "blue")
lines(p1, cross_entropy, col = "red")
legend(0.3, 0.2, c("Classification error", "Gini index", "Cross entropy"), col =
c("black", "blue", "red"), lty = c(1, 1))
grid()
```



Question 5

We have 2 classes Red and Green Different Approach to combine Results are:-

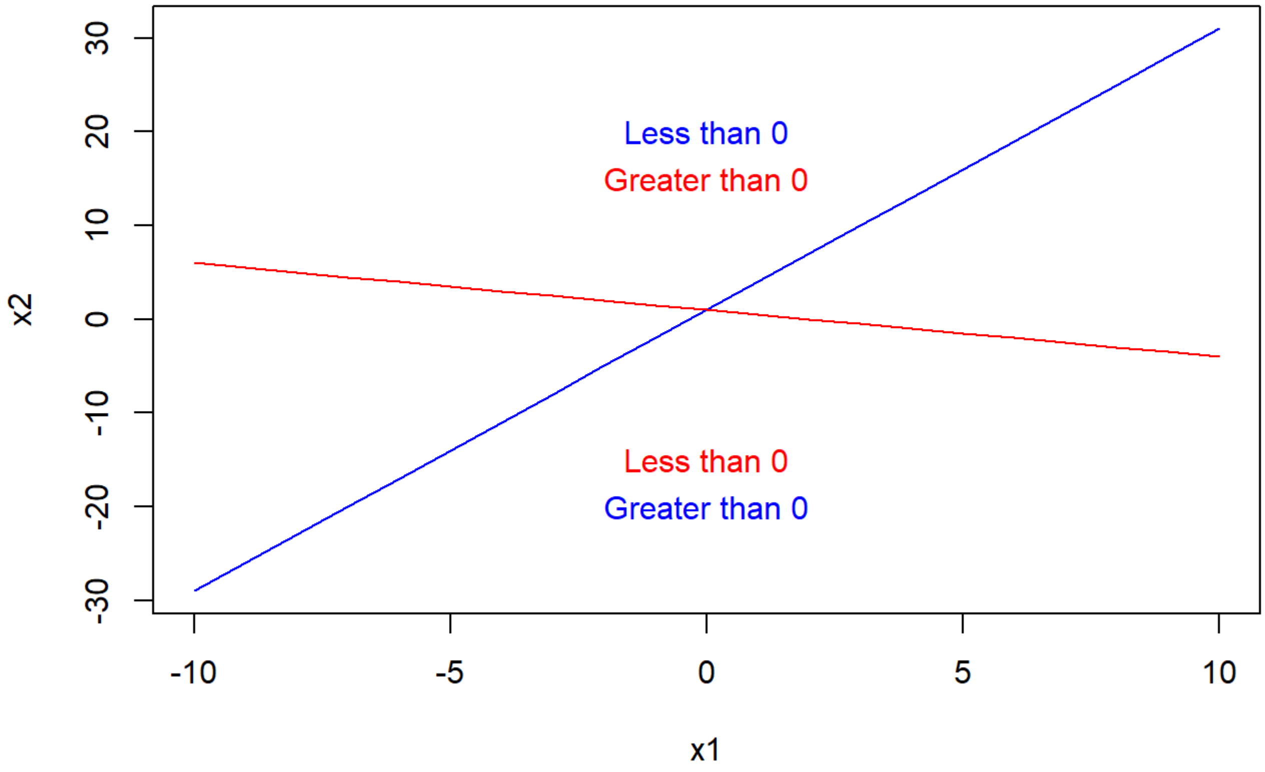
1. Majority vote Approach:- Out of 10 estimates we can see that 6 estimates have p>0.5 and 4 estimates with p<0.5, which suggest that majority of estimates classify X as Red.

2)Average Probability Approach:- Here we calculate average of all estimates and depends on the result if p>0.5, then class for X will be Red otherwise Green The average of 10 estimates comes out to be 0.45 Since p<0.5, Class for X will be Green.

Chapter 9

Question 1

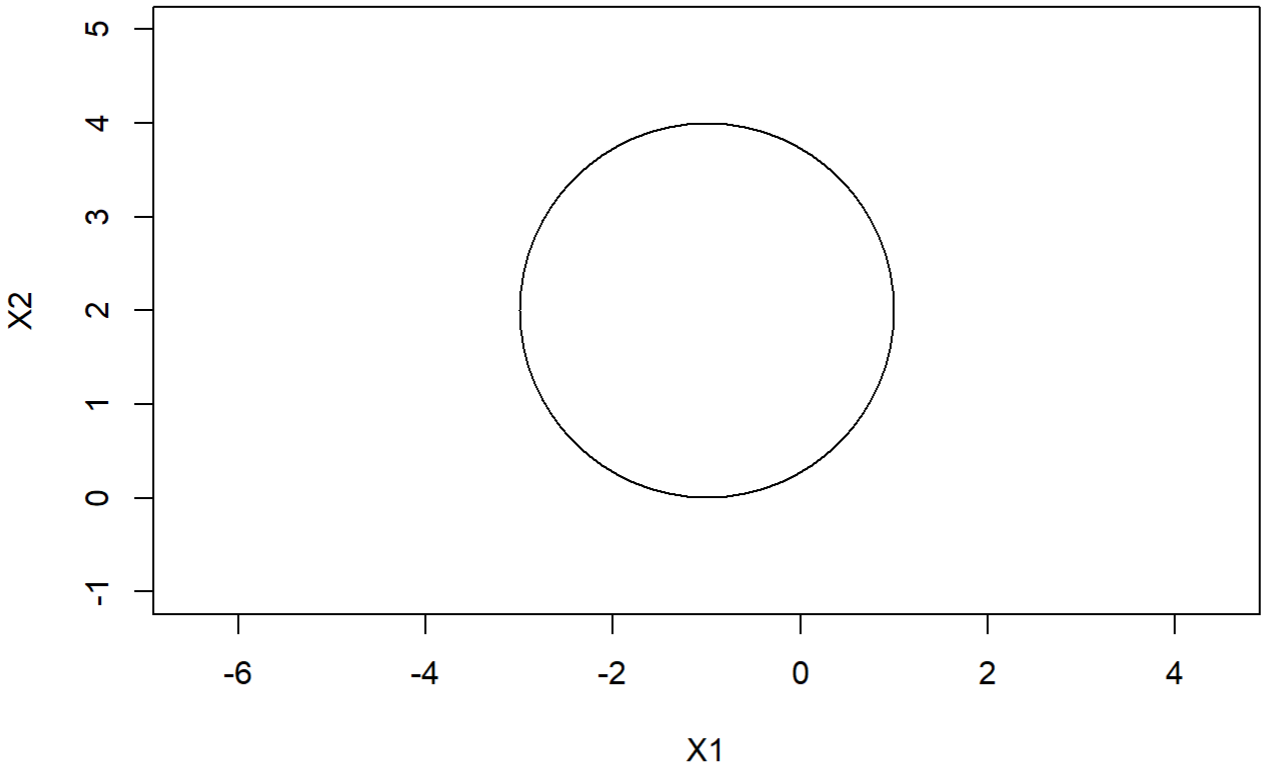
```
x1 <- -10:10
x2 <- 1 + 3 * x1
plot(x1, x2, type = "l", col = "blue")
text(c(0), c(-20), "Greater than 0", col = "blue")
text(c(0), c(20), "Less than 0", col = "blue")
lines(x1, 1 - x1/2, col = "red")
text(c(0), c(-15), "Less than 0", col = "red")
text(c(0), c(15), "Greater than 0", col = "red")
```



Question 2

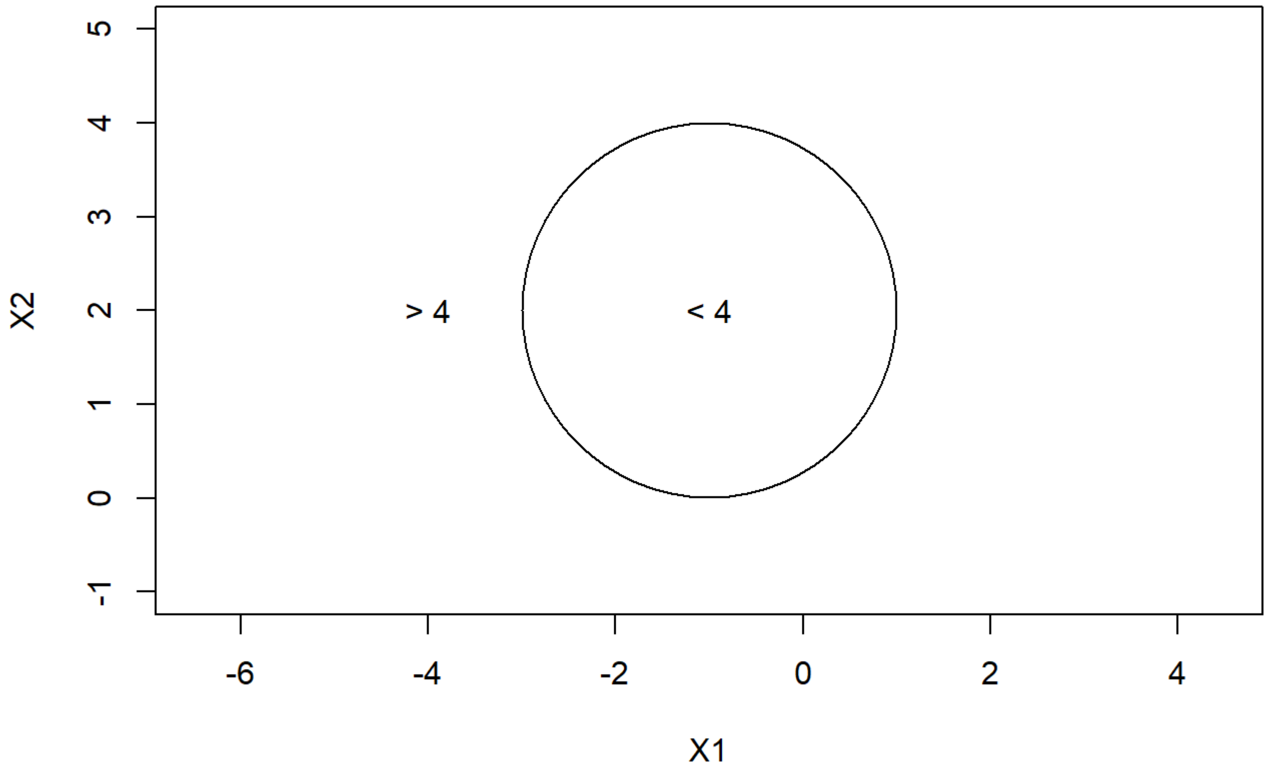
a.

```
plot(NA, NA, type = "n", xlim = c(-4, 2), ylim = c(-1, 5), asp = 1, xlab = "X1", ylab = "X2")
symbols(c(-1), c(2), circles = c(2), add = TRUE, inches = FALSE)
```



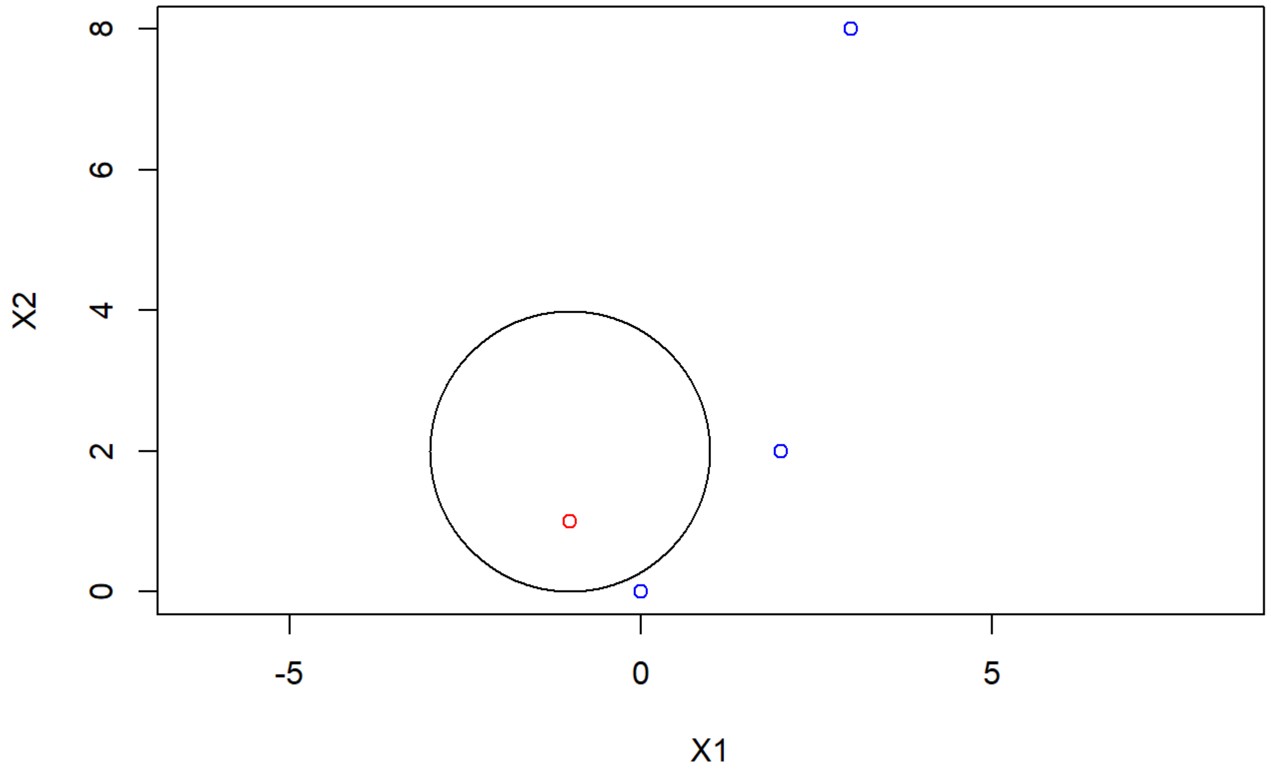
b.

```
plot(NA, NA, type = "n", xlim = c(-4, 2), ylim = c(-1, 5), asp = 1, xlab = "X1", ylab = "X2")
symbols(c(-1), c(2), circles = c(2), add = TRUE, inches = FALSE)
text(c(-1), c(2), "< 4")
text(c(-4), c(2), "> 4")
```



c.

```
plot(c(0, -1, 2, 3), c(0, 1, 2, 8), col = c("blue", "red", "blue", "blue"),
    type = "p", asp = 1, xlab = "X1", ylab = "X2")
symbols(c(-1), c(2), circles = c(2), add = TRUE, inches = FALSE)
```

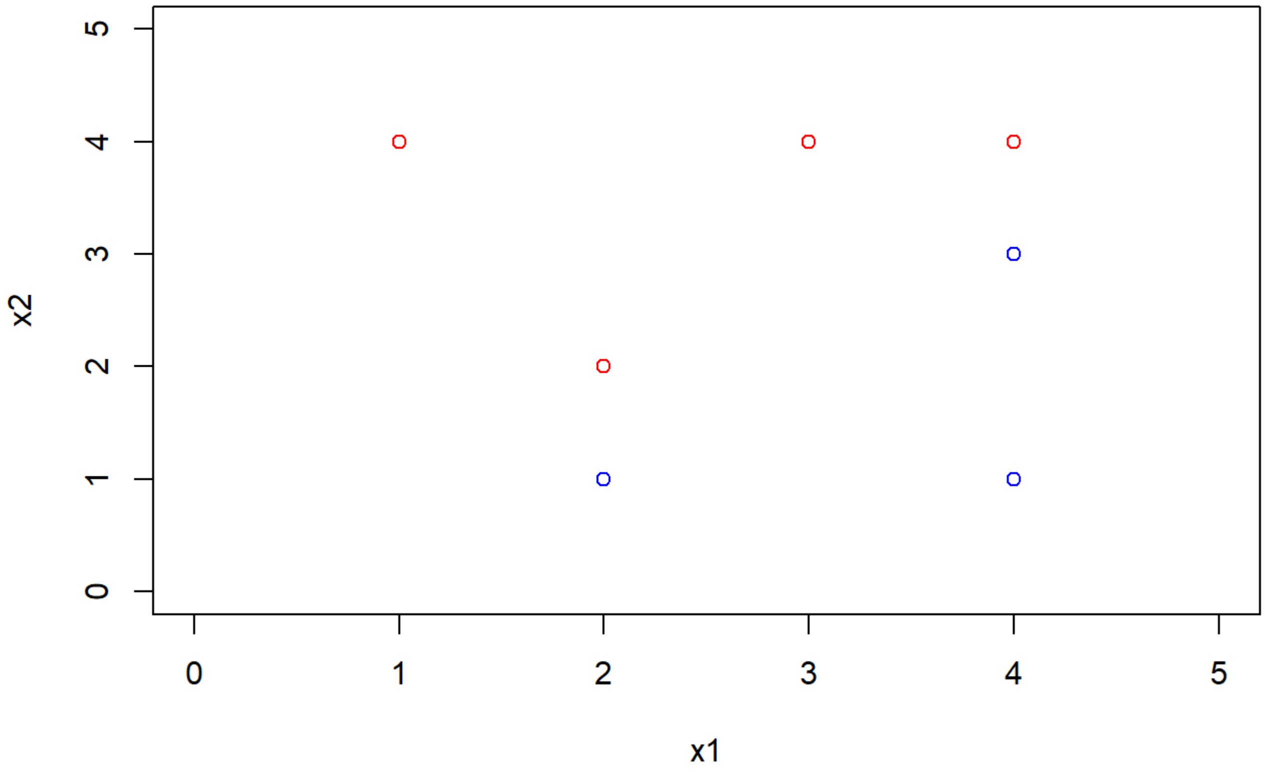


It sufficient to replace X1 and X2 by the coordinates of the points in the equation and to check if the result is less or greater than 4. For (0,0), we have 5>4 (blue class) For (−1,1), we have 1<4 (red class) For (2,2), we have 9>4 (blue class) For (3,8), we have 52>4 (blue class).

d. It is obvious when we expand the equation of decision boundary it becomes X1^2 + X2^2 + 2X1 - 4X2 + 1 = 0 which is linear.

Question 3 a)

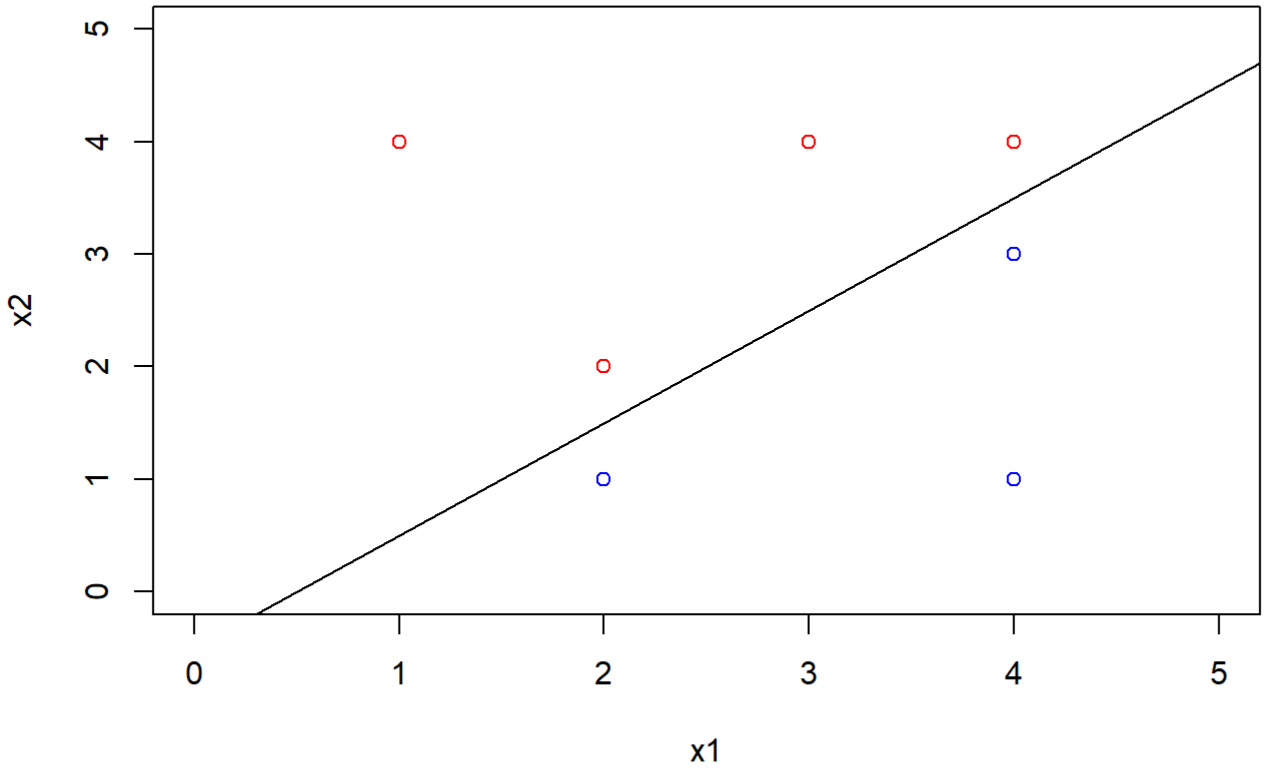
```
x1 = c(3, 2, 4, 1, 2, 4, 4)
x2 = c(4, 2, 4, 4, 1, 3, 1)
colors = c("red", "red", "red", "red", "blue", "blue", "blue")
plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
```



b. The optimal separating hyperplane has to be between the observations (2,1) and (2,2), and between the observations (4,3) and (4,4). So it is a line that passes through the points (2,1.5) and (4,3.5) with equation

$X1 - X2 - 0.5 = 0$

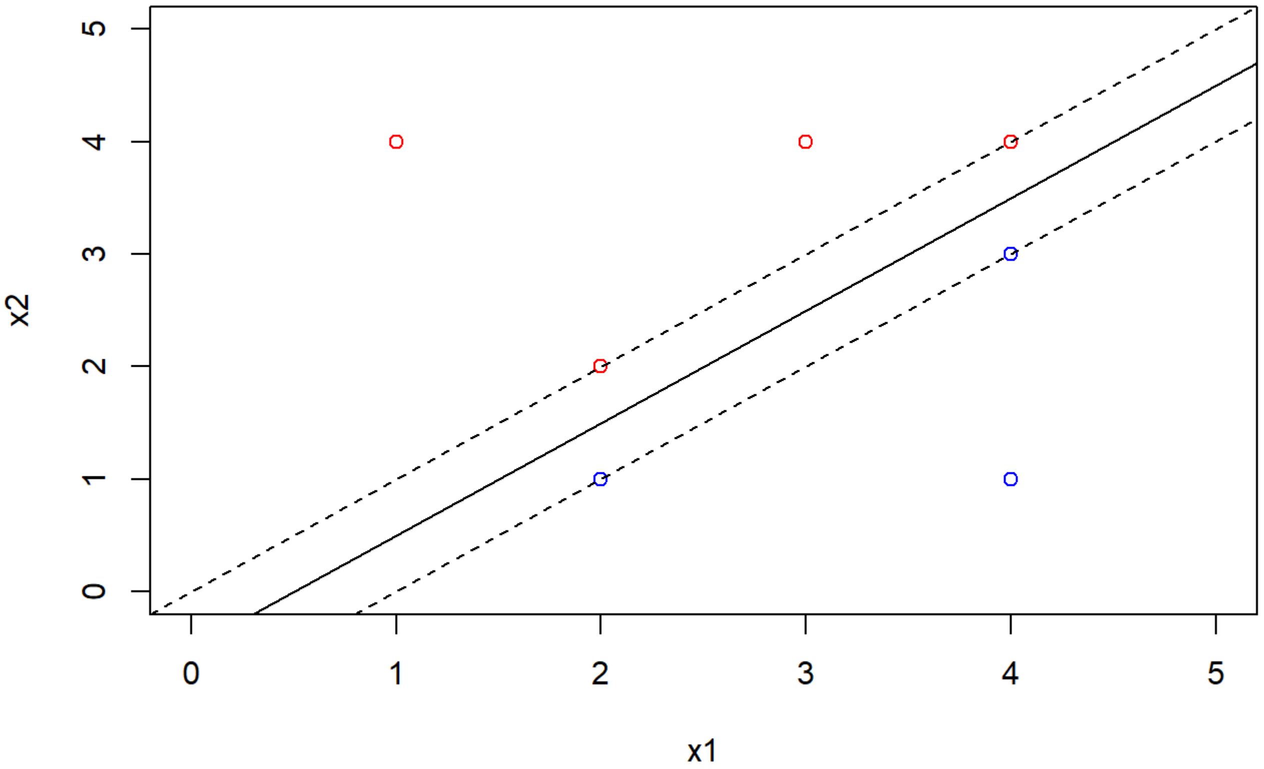
```
plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
abline(-0.5, 1)
```



c. The classification rule is Classify to Red if X1 − X2 − 0.5 < 0, and Classify to Blue otherwise.

d.

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



The

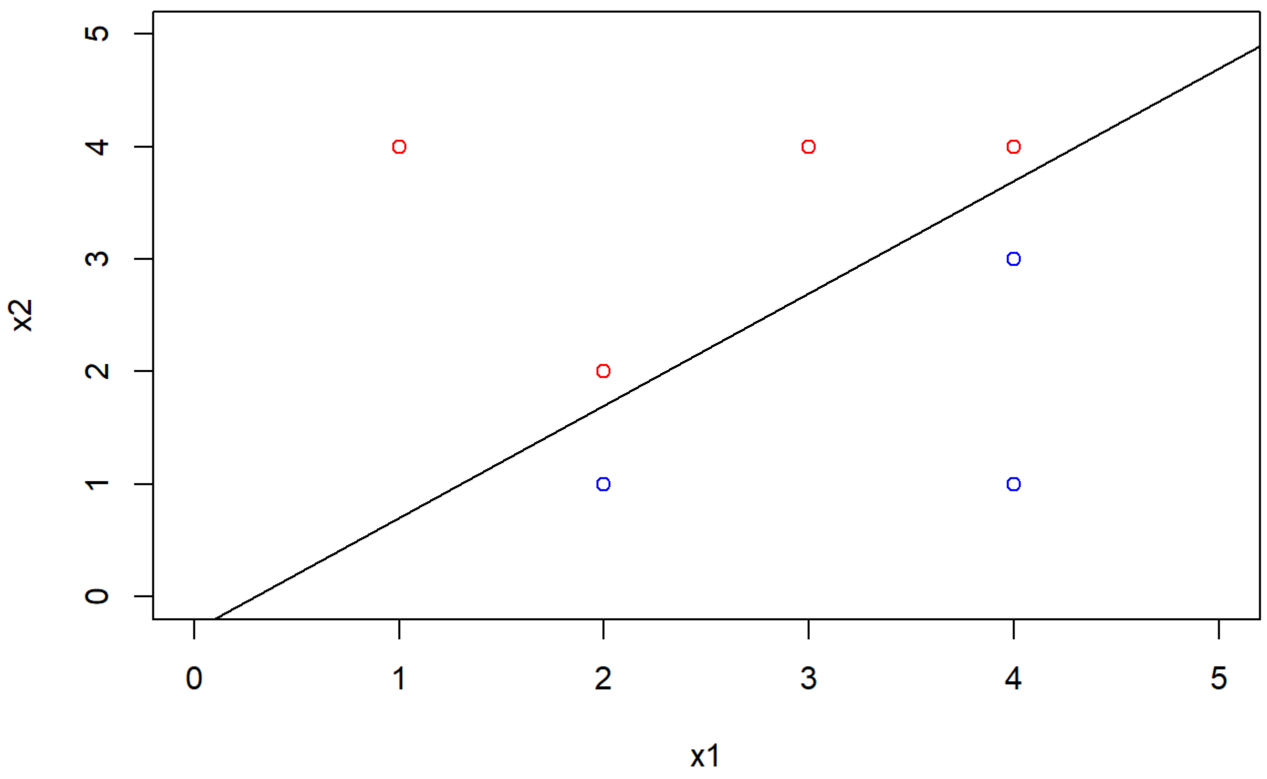
margin is here equal to 1/4

e. The support vectors are the points (2,1), (2,2), (4,3) and (4,4)

f. By examining the plot, it is clear that if we moved the observation (4,1), we would not change the maximal margin hyperplane as it is not a support vector.

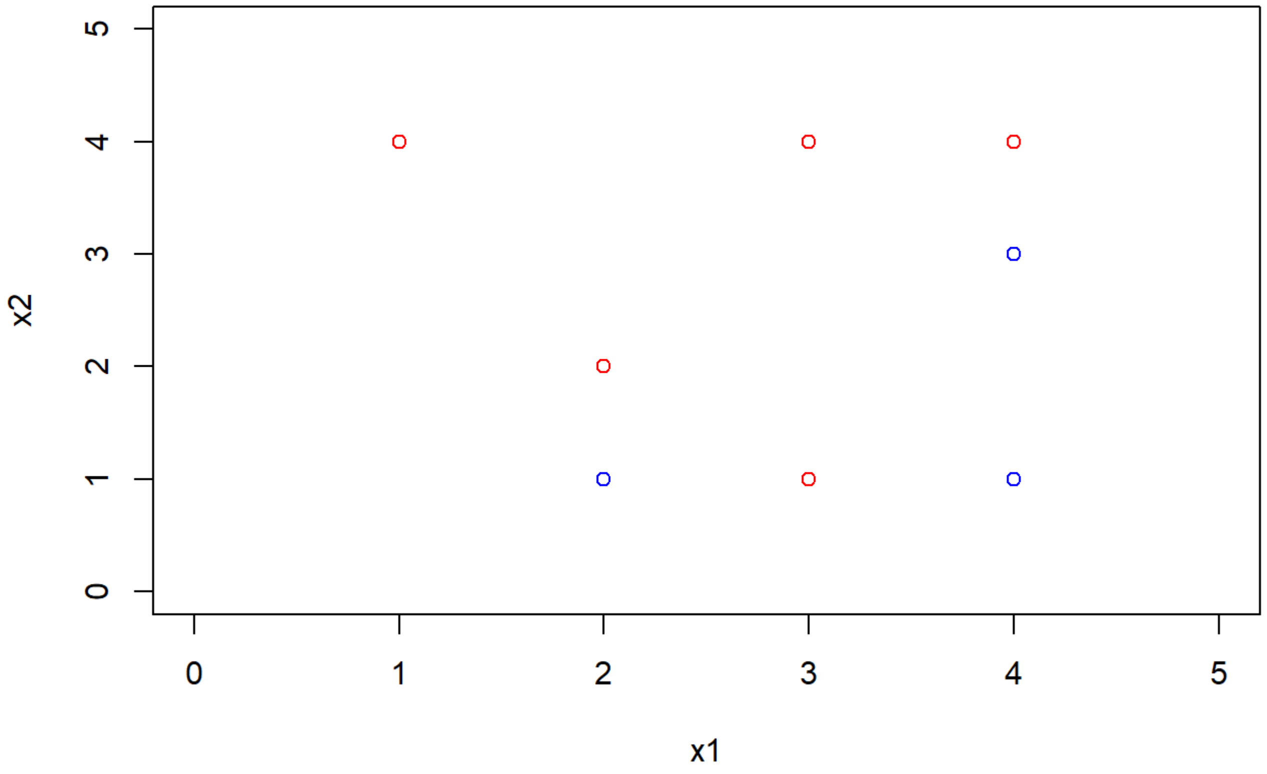
g. For example, the hyperplane which equation is X1−X2−0.3=0 is not the optimal separating hyperplane.

```
plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
abline(-0.3, 1)
```



h.

```
plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
points(c(3), c(1), col = c("red"))
```



When the red point (3,1) is added to the plot, the two classes are obviously not separable by a hyperplane anymore

Practicum Problems

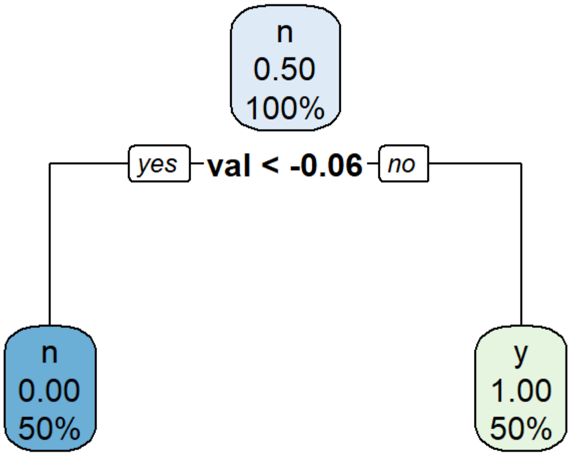
Question 1

```
library(rpart)
library(rpart.plot)
```

Function Definition

```
gini <- function(p)
{
  gini.index = 2 * p * (1 - p)
  return (gini.index)
}

entropy <- function(p)
{
  entropy = (p * log(p) + (1 - p) * log(1 - p))
  return (entropy)
}
```

```
set.seed(150)
a<-rnorm(n=150,mean=5,sd=2)
b<-rnorm(n=150,mean=-5,sd=2)
data1 <- data.frame(val = a,label=rep("y",150))
data2 <- data.frame(val = b,label=rep("n",150))
data <- rbind(data1,data2)
data$label <- as.factor(data$label)
d_tree <- rpart(label~val,data,method="class")
rpart.plot(d_tree)
```



From the above we can see that threshold value for the first split will be -0.06. The tree has one root node and two leaf nodes. Also, tree is able to classify both classes separately which clearly shows empirical distribution.

Calculating Gini and Entropy for Each Node: p=probability of each node

```
p=c(.5, 0, 1)

gini_values=sapply(p, gini)
gini_values
```

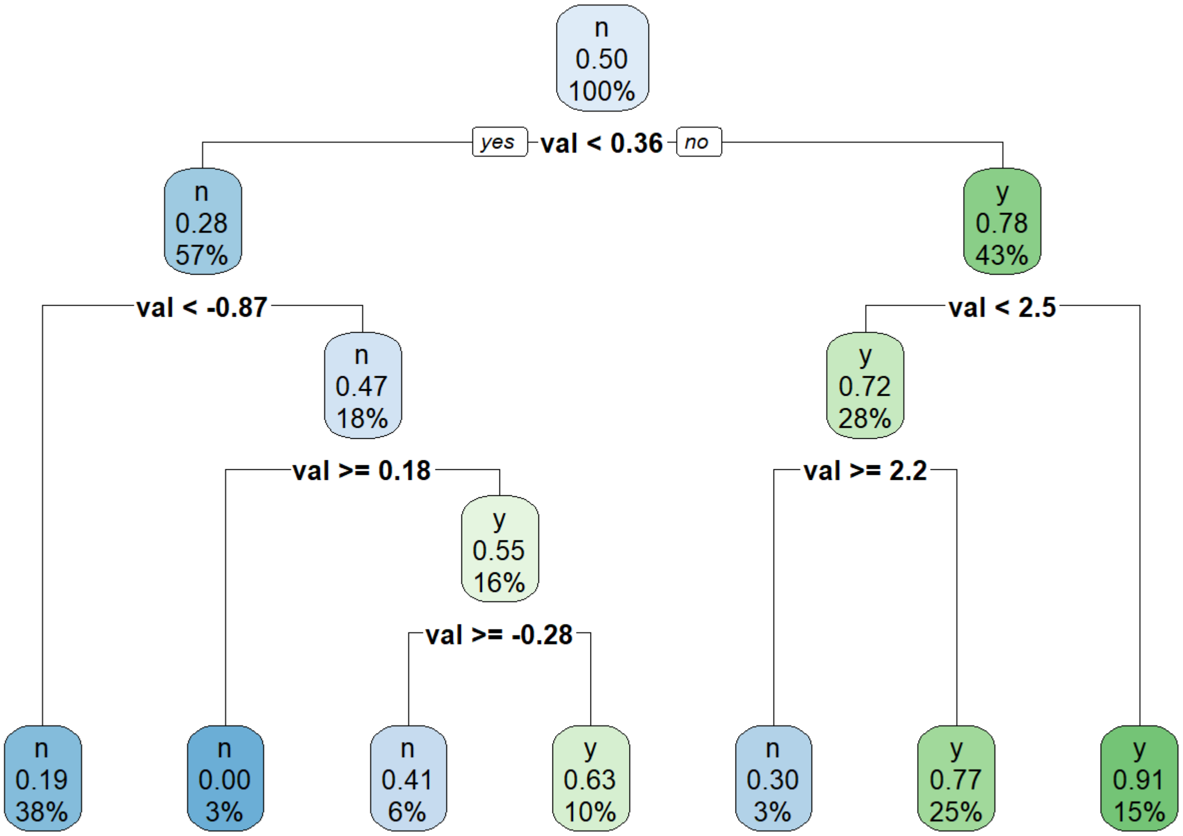
```
## [1] 0.5 0.0 0.0
```

```
entropy_values=sapply(p, entropy)
entropy_values
```

```
## [1] -0.6931472        NaN        NaN
```

The gini values for above tree will be 0.5, 0.0, 0.0 The entropy values for above tree will be -0.6931472, NaN, NaN

```
set.seed(150)
a1<-rnorm(n=150,mean=1,sd=2)
b1<-rnorm(n=150,mean=-1,sd=2)
data3 <- data.frame(val = a1,label=rep("y",150))
data4 <- data.frame(val = b1,label=rep("n",150))
dataa <- rbind(data3,data4)
dataa$label <- as.factor(dataa$label)
d_tree1 <- rpart(label~val,dataa,method="class")
rpart.plot(d_tree1)
```



From the above tree we can see that threshold value for the first split is 0.36. The tree has total of 13 nodes in which one of the nodes is root node and has total of 7 leaf nodes. Large tree size shows presence of more different labels in node, which resulted in a large tree. So, this tree has more overlapping of labels in nodes

Calculating Gini and Entropy for Each Node: p=probability of each node

```
p1=c(.5,0.22,0.72,0.28,0.53,0.45,0.09,0.23,0.70,0.37,0.59,1.0,0.81)
gini_values1=sapply(p1, gini)
gini_values1
```

```
##  [1] 0.5000 0.3432 0.4032 0.4032 0.4982 0.4950 0.1638 0.3542 0.4200 0.4662
## [11] 0.4838 0.0000 0.3078
```

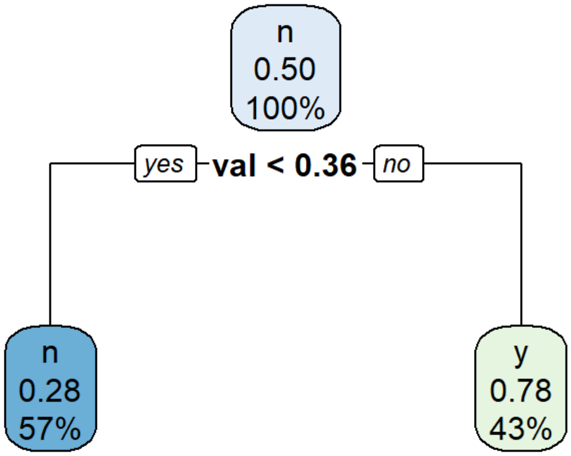
```
entropy_values1=sapply(p1, entropy)
entropy_values1
```

```
##  [1] -0.6931472 -0.5269080 -0.5929533 -0.5929533 -0.6913461 -0.6881388
##  [7] -0.3025378 -0.5392763 -0.6108643 -0.6589557 -0.6768585        NaN
## [13] -0.4862230
```

The gini values for above tree will be 0.5000, 0.3432, 0.4032, 0.4032, 0.4982, 0.4950, 0.1638, 0.3542, 0.4200, 0.4662,0.4838, 0.0000, 0.3078

The entropy values for above tree will be -0.6931472, -0.5269080, -0.5929533, -0.5929533, -0.6913461, -0.6881388, - 0.3025378, -0.5392763, -0.6108643, -0.6589557, -0.6768585, NaN, -0.4862230

```
new_tree <- prune.rpart(d_tree1,cp=0.1)
rpart.plot(new_tree)
```



From the above tree we can see that threshold value for the first split will be 1.5. The tree has one root node and 2 leaf nodes. Also, this pruned tree is much better than the previous as this has only two leaf nodes with less overlapping labels.

Calculating Gini and Entropy for Each Node: p=probability of each node

```
p2=c(.5,0.22,0.72)
gini_values2=sapply(p2, gini)
gini_values2
```

```
## [1] 0.5000 0.3432 0.4032
```

```
entropy_values2=sapply(p2, entropy)
entropy_values2
```

```
## [1] -0.6931472 -0.5269080 -0.5929533
```

The gini values for above tree will be 0.5000, 0.3432, 0.4032 The entropy values for above tree will be -0.6931472, -0.5269080, -0.5929533

Problem 2

Import Libraries

```
library(caret)
```

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

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

```
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
```

```
library(rpart)
library(rpart.plot)
```

```
white.url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality
-white.csv"
white.raw <- read.csv(white.url, header = TRUE, sep = ";")
white <- white.raw
str(white)
```

```
## 'data.frame':    4898 obs. of  12 variables:
##  $ fixed.acidity       : num  7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
##  $ volatile.acidity    : num  0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22 ...
##  $ citric.acid         : num  0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43 ...
##  $ residual.sugar      : num  20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
##  $ chlorides           : num  0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.049 0.044 ...
##  $ free.sulfur.dioxide : num  45 14 30 47 47 30 30 45 14 28 ...
##  $ total.sulfur.dioxide: num  170 132 97 186 186 97 136 170 132 129 ...
##  $ density             : num  1.001 0.994 0.995 0.996 0.996 ...
##  $ pH                  : num  3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
##  $ sulphates           : num  0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45 ...
##  $ alcohol             : num  8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
##  $ quality             : int  6 6 6 6 6 6 6 6 6 6 ...
```

```
red.url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-r
ed.csv"
red.raw <- read.csv(red.url, header = TRUE, sep = ";")
red <- red.raw
str(red)
```

```
## 'data.frame':    1599 obs. of  12 variables:
##  $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
##  $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
##  $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
##  $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
##  $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ...
##  $ free.sulfur.dioxide : num  11 25 15 17 11 13 15 15 9 17 ...
##  $ total.sulfur.dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
##  $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
##  $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
##  $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
##  $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
##  $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
```

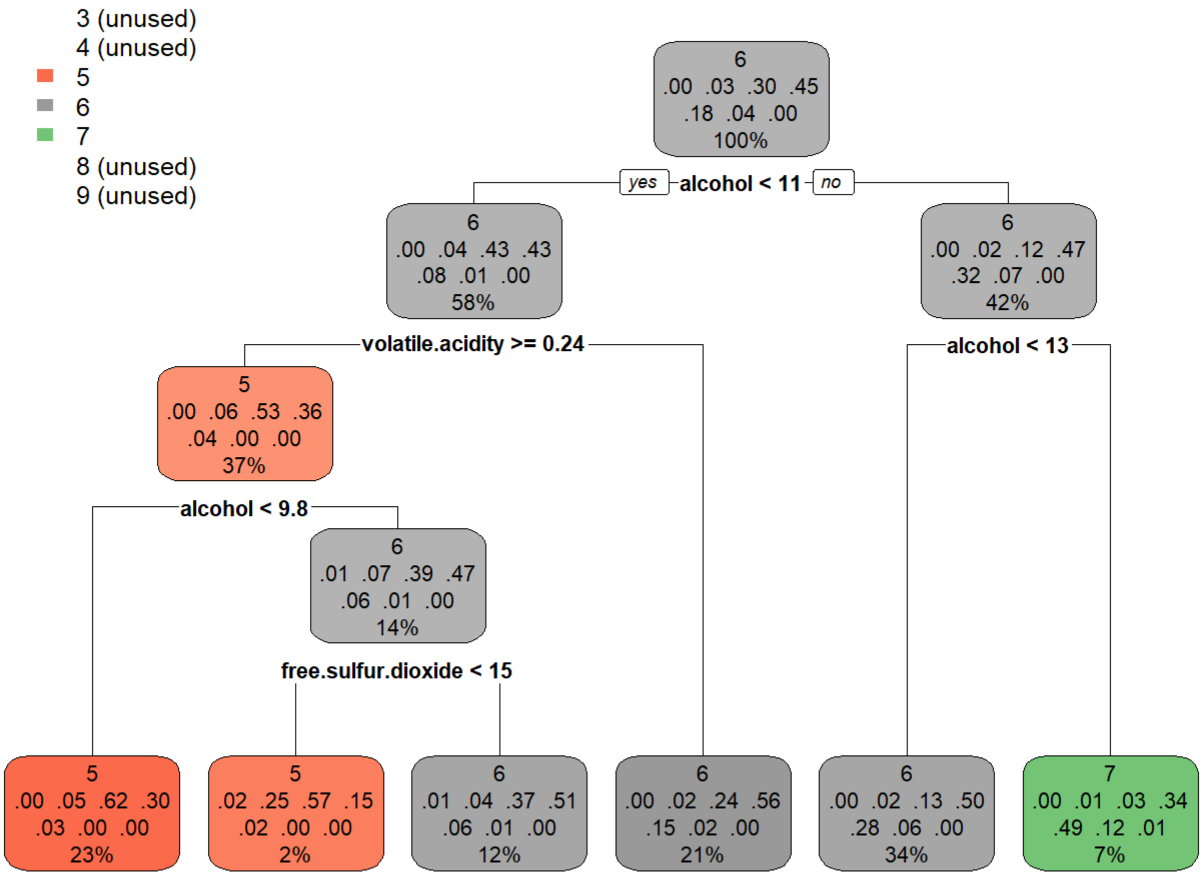
The output tells us that there are 4898 samples and 12 variables in WHITE WINE Dataset. The output tells us that there are 1599 samples and 12 variables in RED WINE Dataset.

Train Test Split

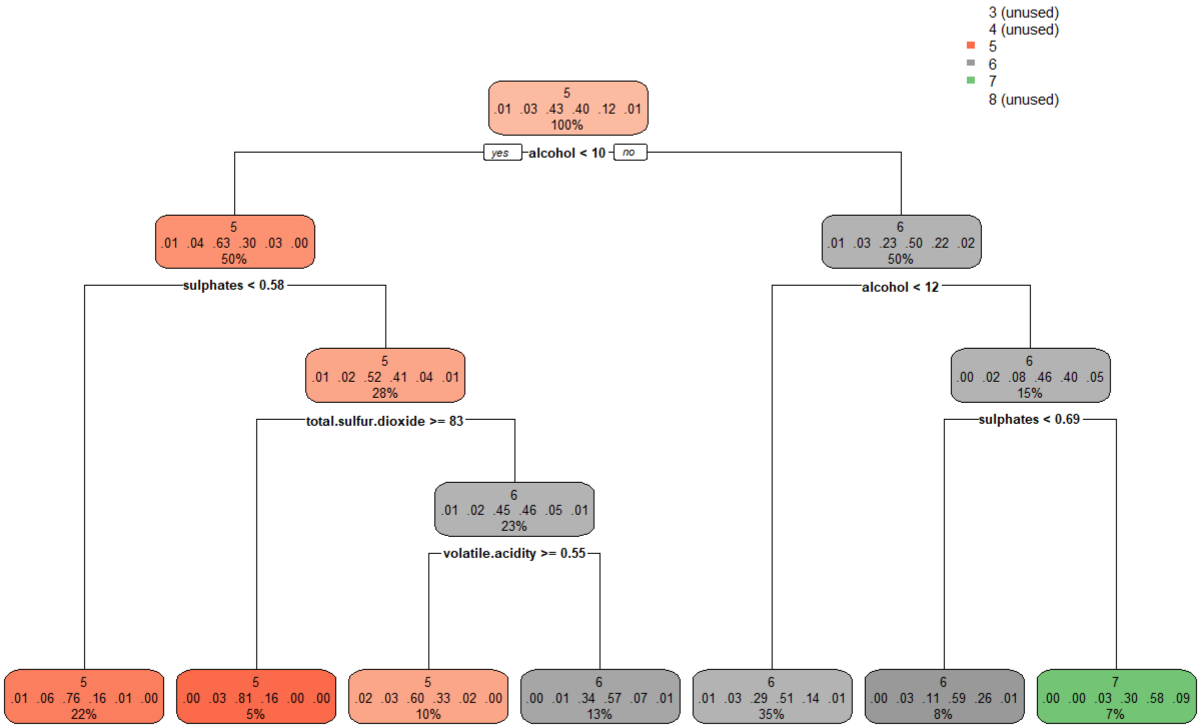
```
white$quality <- as.factor(white$quality)
inTrain <- createDataPartition(white$quality, p = 0.8, list = F)
train.white <- white[inTrain,]
test.white <- white[-inTrain,]

red$quality <- as.factor(red$quality)
inTrainr <- createDataPartition(red$quality, p = 0.8, list = F)
train.red <- red[inTrainr,]
test.red <- red[-inTrainr,]
```

Decision Tree Model

```
dt.train.white <- rpart(quality~., data=train.white )
rpart.plot(dt.train.white)
```



```
dt.train.red <- rpart(quality~., data=train.red )
rpart.plot(dt.train.red)
```



Displaying Confusion Matrix

```
print("Confusion matrix for White Wine Dataset")
```

```
## [1] "Confusion matrix for White Wine Dataset"
```

```
dt.predict.white <- predict(dt.train.white, test.white, type = 'class')
confusionMatrix(dt.predict.white, test.white$quality)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   3   4   5   6   7   8   9
##          3   0   0   0   0   0   0   0
##          4   0   0   0   0   0   0   0
##          5   1  16 148  86   3   1   0
##          6   3  16 142 331 139  23   1
##          7   0   0   1  22  34  11   0
##          8   0   0   0   0   0   0   0
##          9   0   0   0   0   0   0   0
## 
## Overall Statistics
##                                           
##                Accuracy : 0.5245          
##                  95% CI : (0.4927, 0.5562)
##     No Information Rate : 0.4489          
##     P-Value [Acc > NIR] : 1.235e-06       
##                                           
##                   Kappa : 0.2196          
##                                           
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity           0.00000  0.00000   0.5086   0.7540  0.19318  0.00000
## Specificity           1.00000  1.00000   0.8443   0.3989  0.95761  1.00000
## Pos Pred Value            NaN      NaN   0.5804   0.5053  0.50000      NaN
## Neg Pred Value        0.99591  0.96728   0.8022   0.6656  0.84396  0.96421
## Prevalence            0.00409  0.03272   0.2975   0.4489  0.17996  0.03579
## Detection Rate        0.00000  0.00000   0.1513   0.3384  0.03476  0.00000
## Detection Prevalence  0.00000  0.00000   0.2607   0.6697  0.06953  0.00000
## Balanced Accuracy     0.50000  0.50000   0.6764   0.5764  0.57539  0.50000
##                      Class: 9
## Sensitivity          0.000000
## Specificity          1.000000
## Pos Pred Value            NaN
## Neg Pred Value       0.998978
## Prevalence           0.001022
## Detection Rate       0.000000
## Detection Prevalence 0.000000
## Balanced Accuracy    0.500000
```

```
print("-------------------------------------------------------------------------")
```

```
## [1] "-------------------------------------------------------------------------"
```

```
print("-------------------------------------------------------------------------")
```

```
## [1] "-------------------------------------------------------------------------"
```

```
print("-------------------------------------------------------------------------")
```

```
## [1] "-------------------------------------------------------------------------"
```

```
print("Confusion matrix for Red Wine Dataset")
```

```
## [1] "Confusion matrix for Red Wine Dataset"
```

```
dt.predict.red <- predict(dt.train.red, test.red, type = 'class')
confusionMatrix(dt.predict.red, test.red$quality)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction  3  4  5  6  7  8
##          3  0  0  0  0  0  0
##          4  0  0  0  0  0  0
##          5  1  4 84 32  0  0
##          6  1  6 52 85 26  0
##          7  0  0  0 10 13  3
##          8  0  0  0  0  0  0
## 
## Overall Statistics
##                                           
##                Accuracy : 0.5741          
##                  95% CI : (0.5176, 0.6292)
##     No Information Rate : 0.429           
##     P-Value [Acc > NIR] : 1.451e-07       
##                                           
##                   Kappa : 0.3033          
##                                           
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity          0.000000  0.00000   0.6176   0.6693  0.33333 0.000000
## Specificity          1.000000  1.00000   0.7956   0.5526  0.95324 1.000000
## Pos Pred Value            NaN      NaN   0.6942   0.5000  0.50000      NaN
## Neg Pred Value       0.993691  0.96845   0.7347   0.7143  0.91065 0.990536
## Prevalence           0.006309  0.03155   0.4290   0.4006  0.12303 0.009464
## Detection Rate       0.000000  0.00000   0.2650   0.2681  0.04101 0.000000
## Detection Prevalence 0.000000  0.00000   0.3817   0.5363  0.08202 0.000000
## Balanced Accuracy    0.500000  0.50000   0.7066   0.6110  0.64329 0.500000
```

Decision Tree returned an accuracy of 52.5% (+-2) for White Wine Dataset Decision Tree returned an accuracy of 53.9% (+-2) for Red Wine Dataset

For White Wine Dataset the first split was done at “alcohol < 11” whereas In Red Wine Dataset the first split was done at “alcohol < 10”

Sulphates was taken into consideration in Red Wine Dataset whereas its absent in White Wine Dataset.

Total Sulfur Dioxide was taken into consideration in Red Wine Dataset whereas its absent in White Wine Dataset.

Free Sulfur Dioxide was taken into consideration in White Wine Dataset whereas its absent in Red Wine Dataset.

Random Forest Model

```
rf.train.white <- train(quality ~ ., data = train.white, method = "rf",preProcess = c("center",
"scale"))
```

```
## Warning: model fit failed for Resample22: mtry= 2 Error in randomForest.default(x, y, mtry =
min(param$mtry, ncol(x)), ...) :
##   Can't have empty classes in y.
```

```
## Warning: model fit failed for Resample22: mtry= 6 Error in randomForest.default(x, y, mtry =
min(param$mtry, ncol(x)), ...) :
##   Can't have empty classes in y.
```

```
## Warning: model fit failed for Resample22: mtry=11 Error in randomForest.default(x, y, mtry =
min(param$mtry, ncol(x)), ...) :
##   Can't have empty classes in y.
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
rf.train.red <- train(quality ~ ., data = train.red, method = "rf",preProcess = c("center", "sca
le"))
```

Displaying the Confusion Matrix

```
print("Confusion matrix for White Wine Dataset")
```

```
## [1] "Confusion matrix for White Wine Dataset"
```

```
rf.predict.white <- predict(rf.train.white, test.white)
confusionMatrix(rf.predict.white, test.white$quality)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   3   4   5   6   7   8   9
##          3   0   0   0   0   0   0   0
##          4   0   9   2   1   0   0   0
##          5   1  14 206  64   1   1   0
##          6   3   9  82 353  71   8   0
##          7   0   0   1  21 102   9   1
##          8   0   0   0   0   2  17   0
##          9   0   0   0   0   0   0   0
## 
## Overall Statistics
## 
##                Accuracy : 0.7025
##                  95% CI : (0.6727, 0.731)
##     No Information Rate : 0.4489
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.5391
## 
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity           0.00000 0.281250   0.7079   0.8041   0.5795  0.48571
## Specificity           1.00000 0.996829   0.8821   0.6790   0.9601  0.99788
## Pos Pred Value            NaN 0.750000   0.7178   0.6711   0.7612  0.89474
## Neg Pred Value        0.99591 0.976190   0.8770   0.8097   0.9123  0.98123
## Prevalence            0.00409 0.032720   0.2975   0.4489   0.1800  0.03579
## Detection Rate        0.00000 0.009202   0.2106   0.3609   0.1043  0.01738
## Detection Prevalence  0.00000 0.012270   0.2935   0.5378   0.1370  0.01943
## Balanced Accuracy     0.50000 0.639039   0.7950   0.7416   0.7698  0.74180
##                      Class: 9
## Sensitivity          0.000000
## Specificity          1.000000
## Pos Pred Value            NaN
## Neg Pred Value       0.998978
## Prevalence           0.001022
## Detection Rate       0.000000
## Detection Prevalence 0.000000
## Balanced Accuracy    0.500000
```

```
print("-------------------------------------------------------------------------")
```

```
## [1] "-------------------------------------------------------------------------"
```

```
print("-------------------------------------------------------------------------")
```

```
## [1] "-------------------------------------------------------------------------"
```

```
print("-------------------------------------------------------------------------")
```

```
## [1] "-------------------------------------------------------------------------"
```

```
print("Confusion matrix for Red Wine Dataset")
```

```
## [1] "Confusion matrix for Red Wine Dataset"
```

```
rf.predict.red <- predict(rf.train.red, test.red)
confusionMatrix(rf.predict.red, test.red$quality)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   3   4   5   6   7   8
##          3   0   0   0   0   0   0
##          4   1   0   0   0   0   0
##          5   0   5 112  32   1   0
##          6   1   5  24  87  22   2
##          7   0   0   0   8  16   1
##          8   0   0   0   0   0   0
## 
## Overall Statistics
## 
##                Accuracy : 0.6782          
##                  95% CI : (0.6237, 0.7294)
##     No Information Rate : 0.429           
##     P-Value [Acc > NIR] : < 2.2e-16       
## 
##                   Kappa : 0.4716          
## 
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity          0.000000 0.000000   0.8235   0.6850  0.41026 0.000000
## Specificity          1.000000 0.996743   0.7901   0.7158  0.96763 1.000000
## Pos Pred Value            NaN 0.000000   0.7467   0.6170  0.64000      NaN
## Neg Pred Value       0.993691 0.968354   0.8563   0.7727  0.92123 0.990536
## Prevalence           0.006309 0.031546   0.4290   0.4006  0.12303 0.009464
## Detection Rate       0.000000 0.000000   0.3533   0.2744  0.05047 0.000000
## Detection Prevalence 0.000000 0.003155   0.4732   0.4448  0.07886 0.000000
## Balanced Accuracy    0.500000 0.498371   0.8068   0.7004  0.68894 0.500000
```

Random Forest returned an accuracy of 69.4% (+-2) for White Wine Dataset Random Forest returned an accuracy of 71.9% (+-2) for Red Wine Dataset

The Accuracy increased from 52% to 69% in Random Forest Classifier in White Wine Dataset The Accuracy increased from 53% to 71% in Random Forest Classifier in Red Wine Dataset

Problem 3

Importing Libraries

```
library(readxl)
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(SnowballC)
library(e1071)
```

```
smsData <- read_excel("D:\\Temp\\sms_spam.xlsx")
str(smsData)
```

```
## tibble [5,571 x 2] (S3: tbl_df/tbl/data.frame)
##  $ ham
: chr [1:5571] "ham" "spam" "ham" "ham" ...
##  $ Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine t
here got amore wat...: chr [1:5571] "Ok lar... Joking wif u oni..." "Free entry in 2 a wkly comp
to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question("| __truncate
d__ "U dun say so early hor... U c already then say..." "Nah I don't think he goes to usf, he li
ves around here though" ...
```

```
names(smsData)[1] = "type"
names(smsData)[2] = "text"
```

```
smsData$type <- factor(smsData$type)
str(smsData)
```

```
## tibble [5,571 x 2] (S3: tbl_df/tbl/data.frame)
##  $ type: Factor w/ 2 levels "ham","spam": 1 2 1 1 2 1 1 2 2 1 ...
##  $ text: chr [1:5571] "Ok lar... Joking wif u oni..." "Free entry in 2 a wkly comp to win FA
Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question("| __truncated__ "U dun
say so early hor... U c already then say..." "Nah I don't think he goes to usf, he lives around
here though" ...
```

```
table(smsData$type)
```

```
##
##  ham spam
## 4824  747
```

```
smsDataCorpus <- VCorpus(VectorSource(smsData$text))
print(smsDataCorpus)
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 5571
```

Stemming

```
smsData_corpus_clean <- tm_map(smsDataCorpus,stemDocument)
```

Convert to LowerCase

```
smsData_corpus_clean <- tm_map(smsData_corpus_clean,content_transformer(tolower))
as.character(smsData_corpus_clean[[1]])
```

```
## [1] "ok lar... joke wif u oni..."
```

Remove Stop Words

```
smsData_corpus_clean <- tm_map(smsData_corpus_clean,removeWords,stopwords())
```

Remove Punctuation

```
smsData_corpus_clean <- tm_map(smsData_corpus_clean,removePunctuation)
```

Remove WhietSpace

```
smsData_corpus_clean <- tm_map(smsData_corpus_clean,stripWhitespace)
```

```
as.character(smsData_corpus_clean[[1]])
```

```
## [1] "ok lar joke wif u oni"
```

Creating Document Term Matrix

```
sms_dtm <- DocumentTermMatrix(smsData_corpus_clean)
sms_dtm
```

```
## <<DocumentTermMatrix (documents: 5571, terms: 8435)>>
## Non-/sparse entries: 45123/46946262
## Sparsity           : 100%
## Maximal term length: 51
## Weighting          : term frequency (tf)
```

Splitting into Train Test Split in 75% Training and 25% into Testing

```
sms_dtm_train <- sms_dtm[1:4169,]
sms_dtm_test <- sms_dtm[4170:5571,]


sms_train_labels <- smsData[1:4169,]$type
sms_test_labels <- smsData[4170:5571,]$type
```

Using FindFrequencyTerms

```
frequent_terms <- findFreqTerms(sms_dtm_train,10)

sms_dtm_freq_train <- sms_dtm_train[,frequent_terms]
sms_dtm_freq_test <- sms_dtm_test[,frequent_terms]
```

Function to Convert into Booleans

```
convert_counts <- function(x){
  x <- ifelse(x > 0,1,0)
}
```

```
sms_train <- apply(sms_dtm_freq_train,MARGIN = 2,convert_counts)
sms_test <- apply(sms_dtm_freq_test,MARGIN = 2,convert_counts)
```

Building the Model

```
sms_classifier <- naiveBayes(sms_train,sms_train_labels)

#Predicting using the Model
sms_train_pred <- predict(sms_classifier,sms_train)
sms_test_pred <- predict(sms_classifier,sms_test)
```

Displaying the Accuracy

```
print("Train Set Accuracy with Confusion Matrix:")
```

```
## [1] "Train Set Accuracy with Confusion Matrix:"
```

```
confusionMatrix(sms_train_pred, sms_train_labels)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction  ham spam
##       ham    25    1
##       spam 3579  564
##                                           
##                Accuracy : 0.1413          
##                  95% CI : (0.1308, 0.1522)
##     No Information Rate : 0.8645          
##     P-Value [Acc > NIR] : 1               
##                                           
##                   Kappa : 0.0014          
##                                           
##  Mcnemar's Test P-Value : <2e-16          
##                                           
##             Sensitivity : 0.006937        
##             Specificity : 0.998230        
##          Pos Pred Value : 0.961538        
##          Neg Pred Value : 0.136133        
##              Prevalence : 0.864476        
##          Detection Rate : 0.005997        
##    Detection Prevalence : 0.006237        
##       Balanced Accuracy : 0.502583        
##                                           
##        'Positive' Class : ham             
## 
```

```
print("Test Set Accuracy with Confusion Matrix:")
```

```
## [1] "Test Set Accuracy with Confusion Matrix:"
```

```
confusionMatrix(sms_test_pred, sms_test_labels)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction  ham spam
##       ham     9    0
##       spam 1211  182
##                                           
##                Accuracy : 0.1362          
##                  95% CI : (0.1187, 0.1553)
##     No Information Rate : 0.8702          
##     P-Value [Acc > NIR] : 1               
##                                           
##                   Kappa : 0.0019          
##                                           
##  Mcnemar's Test P-Value : <2e-16          
##                                           
##             Sensitivity : 0.007377        
##             Specificity : 1.000000        
##          Pos Pred Value : 1.000000        
##          Neg Pred Value : 0.130653        
##              Prevalence : 0.870185        
##          Detection Rate : 0.006419        
##    Detection Prevalence : 0.006419        
##       Balanced Accuracy : 0.503689        
##                                           
##        'Positive' Class : ham             
## 
```

Training Set Accuracy is 14& Test Set Accuracy is 13%