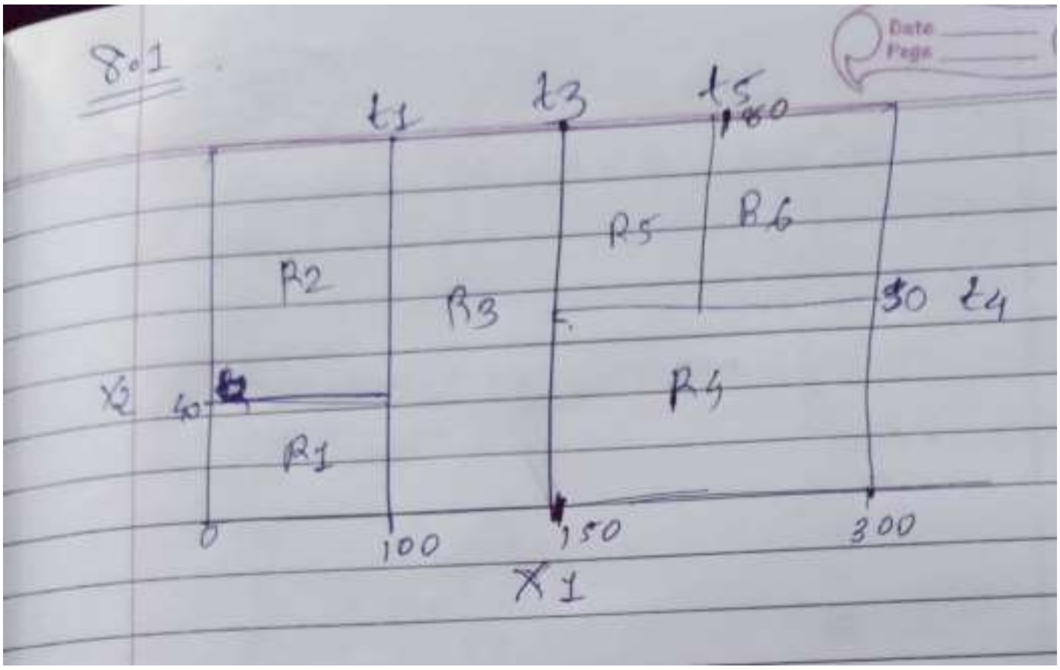
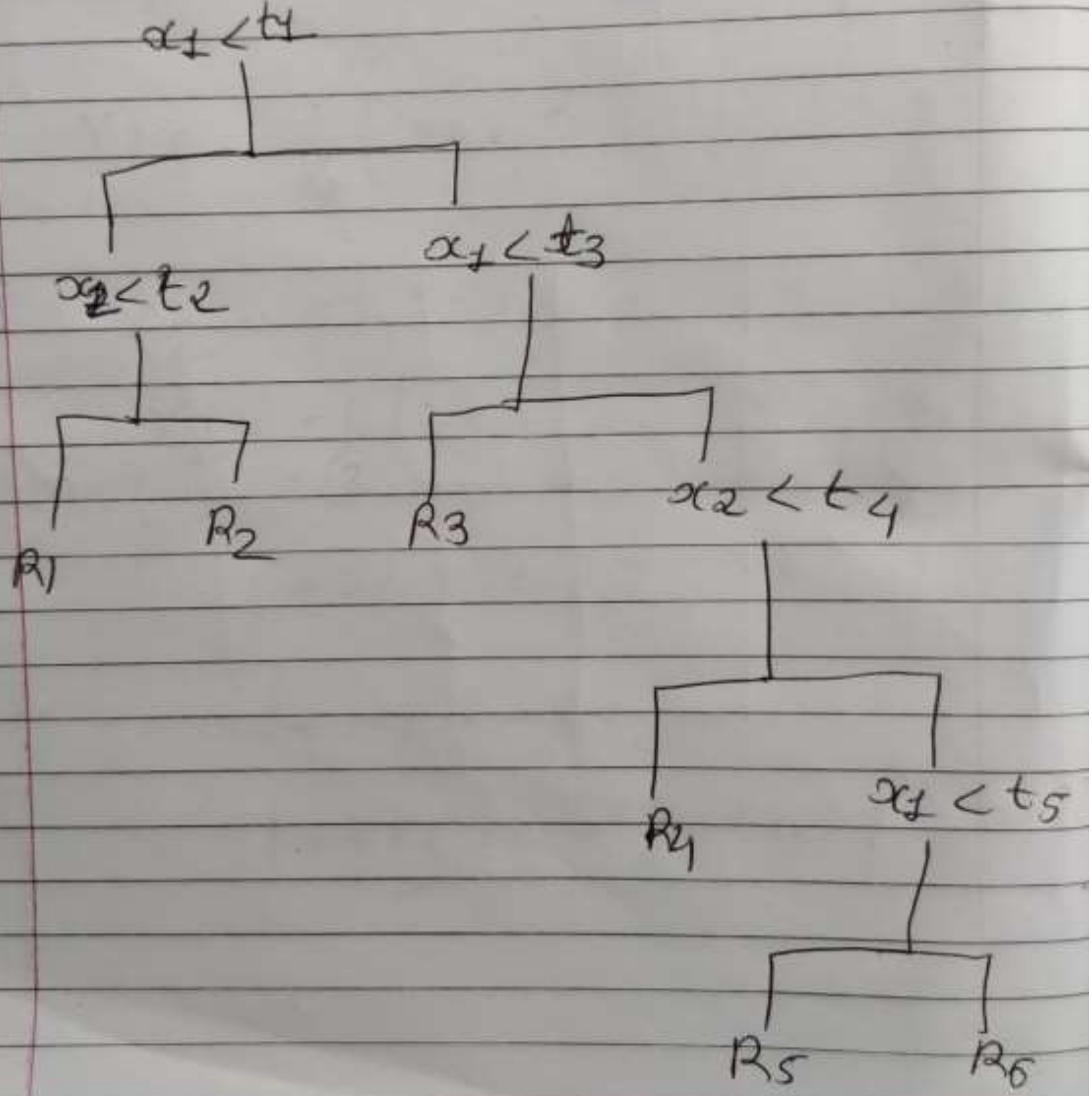
Chapter 8

Exercise 1:



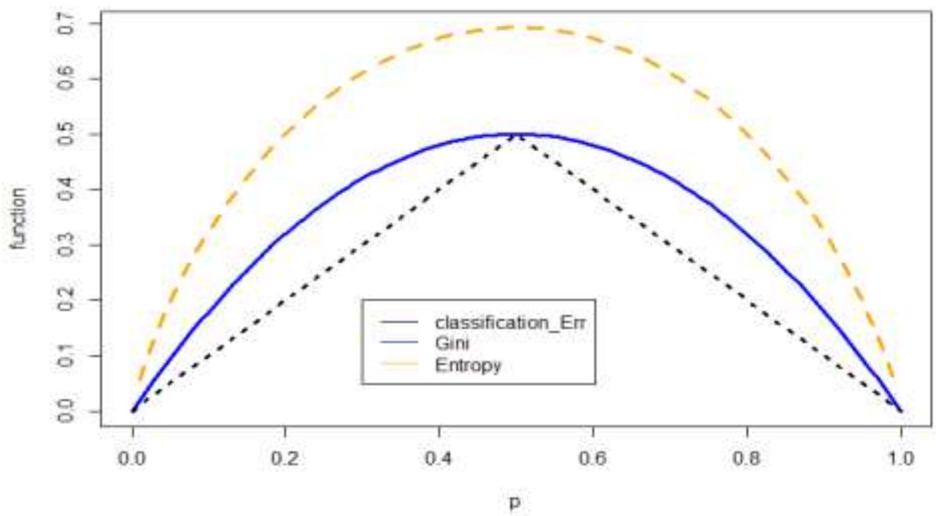
$t_1 = 100$ $t_3 = 150$ $t_5 = 180$

$t_2 = 40$ $t_4 = 50$



Exercise 3:

```
> p=seq(0,1,0.01)
> gini= p * (1-p) * 2
> entropy= -(p * log(p) + (1-p) * log(1-p))
> classification_error= 1 - pmax(p,1-p)
> matplot(p,cbind(gini,entropy,classification_error),type = "l",col = c("blue","orange","black"),xlab = "p",ylab = "function of
 'pm1",lwd = 3)
> |
```



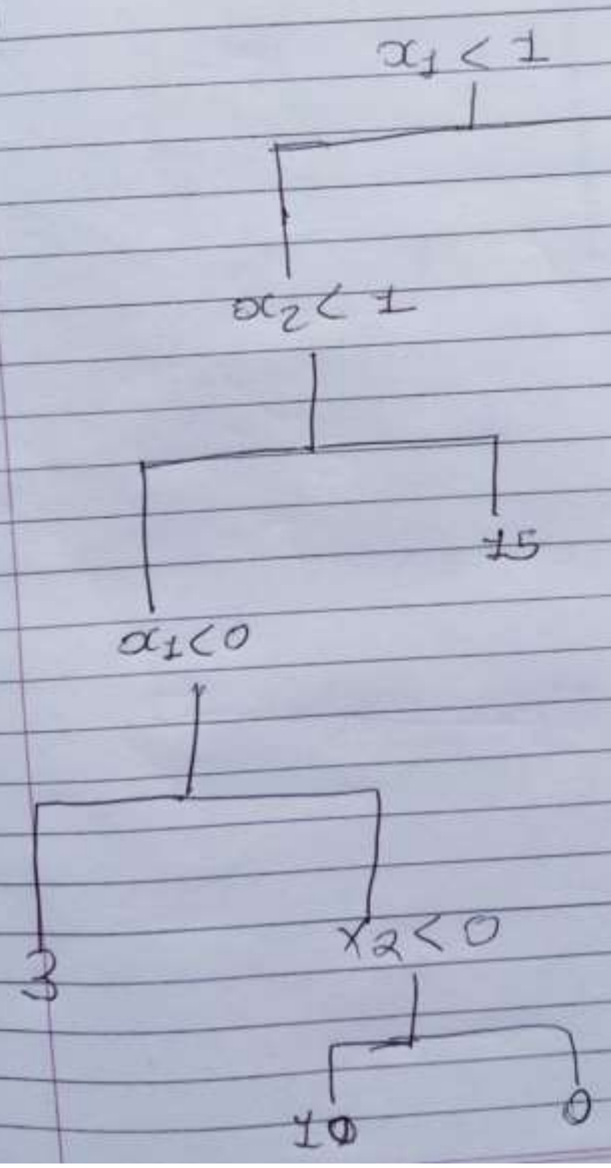
Exercise 4:

a) Sketch the tree corresponding to the partition of the predictor space illustrated in the left-hand panel of Figure 8.14. The numbers inside the boxes indicate the mean of Y within each region

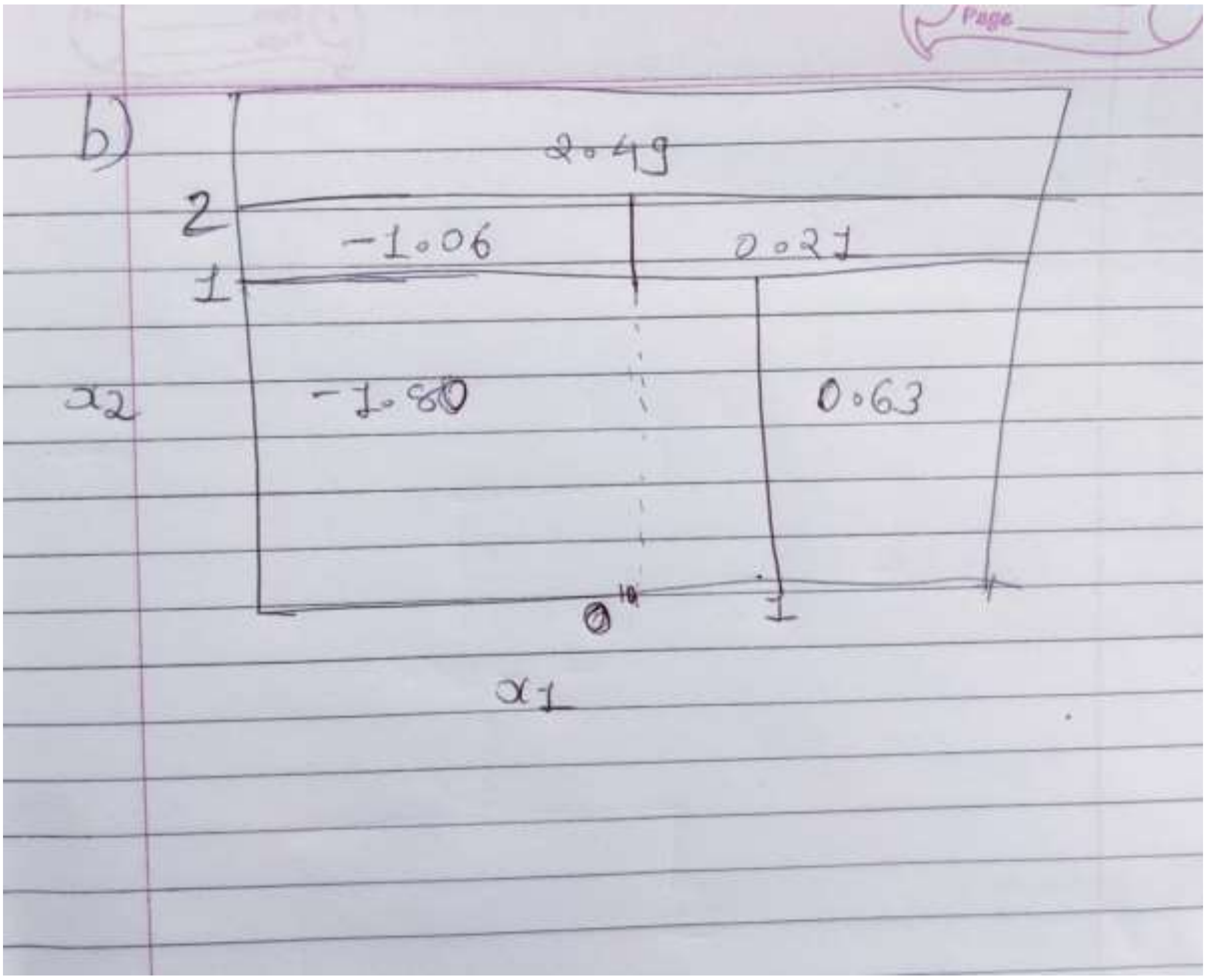
Chapter-8

Exercise-4

a)



b) Create a diagram similar to the left-hand panel of Figure 8.14, using the tree illustrated in the right-hand panel of the same figure. You should divide up the predictor space into the correct regions and indicate the mean for each region.



Exercise 5:

Suppose we produce ten bootstrapped samples from a data set containing red and green classes. We then apply a classification tree to each bootstrapped sample and, for a specific value of X, produce 10 estimates of P(Class is Red|X): 0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, and 0.75. There are two common ways to combine these results together into a single class prediction. One is the majority vote approach discussed in this chapter. The second approach is to classify based on the average probability. In this example, what is the final classification under each of these two approaches?

Solution:

The majority vote approach classifies X as red, from given 10 estimates 4 estimates have a probability less than 0.5 and 6 estimates have a probability > 0.5.

The average Probability Approach classifies X as green as an average of 10 estimates is less than 0.5

Avg = (0.1+0.15+0.2+0.2+0.55+0.6+0.6+0.65+0.7+0.75)/10

= 0.45

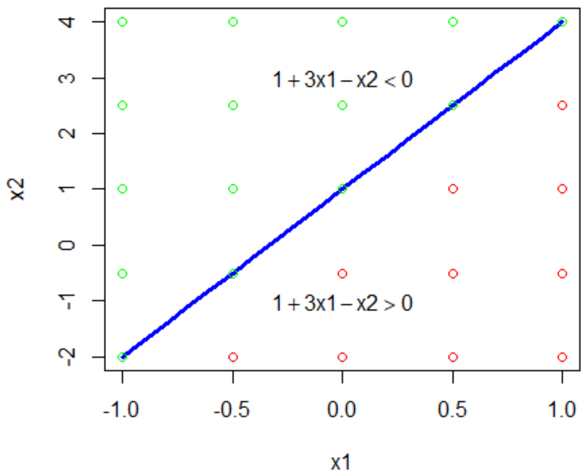
Since the Avg < 0.5, the Class for X will be Green.

Chapter 9

Exercise 1:

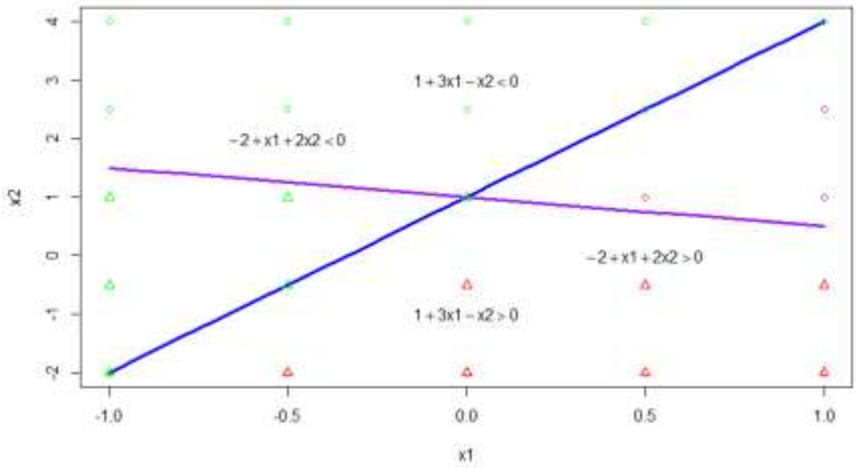
Sketch the hyperplane $1 + 3X1 - X2 = 0$. Indicate the set of points for which $1 + 3X1 - X2 > 0$, as well as the set of points for which $1 + 3X1 - X2 < 0$.

```
> x1= seq(-1,1,0.1)
> x2= 1+3 *x1
> plot(x1,x2,xlab="x1",ylab="x2",type="l",lwd=3,col="blue")
> text(0,3,expression(1+3*x1-x2<0),col="black")
> text(-0,-1,expression(1+3*x1-x2 > 0),col="black")
> for(i in seq(-1,1,length.out = 5)){
+     pts=data.frame(rep(i,5),seq(-2,4,length.out = 5))
+     points(pts,col=ifelse(1+3*pts[,1]-pts[,2]>0,'red','gree
n'),bg="grey")
+ }
>
```



B. On the same plot, sketch the hyperplane $-2 + X1 + 2X2 = 0$. Indicate the set of points for which $-2 + X1 + 2X2 > 0$, as well as the set of points for which $-2 + X1 + 2X2 < 0$

```
+ }
> lines(x1,1-x1/2,col="purple",lwd=3)
> text(-0.5,2,expression(-2 +x1+2*x2<0),col="black")
> text(0.5,0,expression(-2 +x1+2*x2>0),col="black")
> for(i in seq(-1,1,length.out = 5)){
+     pts=data.frame(rep(i,5),seq(-2,4,length.out = 5))
+     points(pts,col=ifelse(1+3*pts[,1]-pts[,2]>0,'red','green'),pch=ifelse(-2+pts[,1]+2*pts[,2]>0,1,2))
+ }
>
```

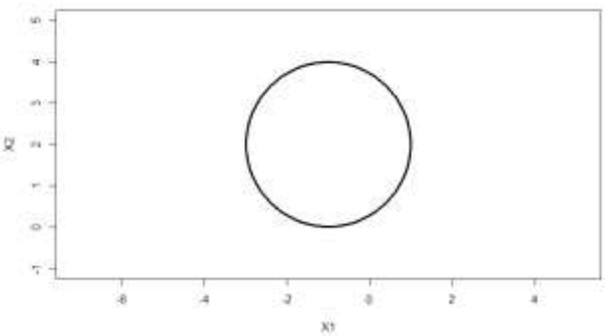


Exercise 2:

We have seen that in p = 2 dimensions, a linear decision boundary takes the form β0+β1X1+β2X2 = 0. We now investigate a non-linear decision boundary.

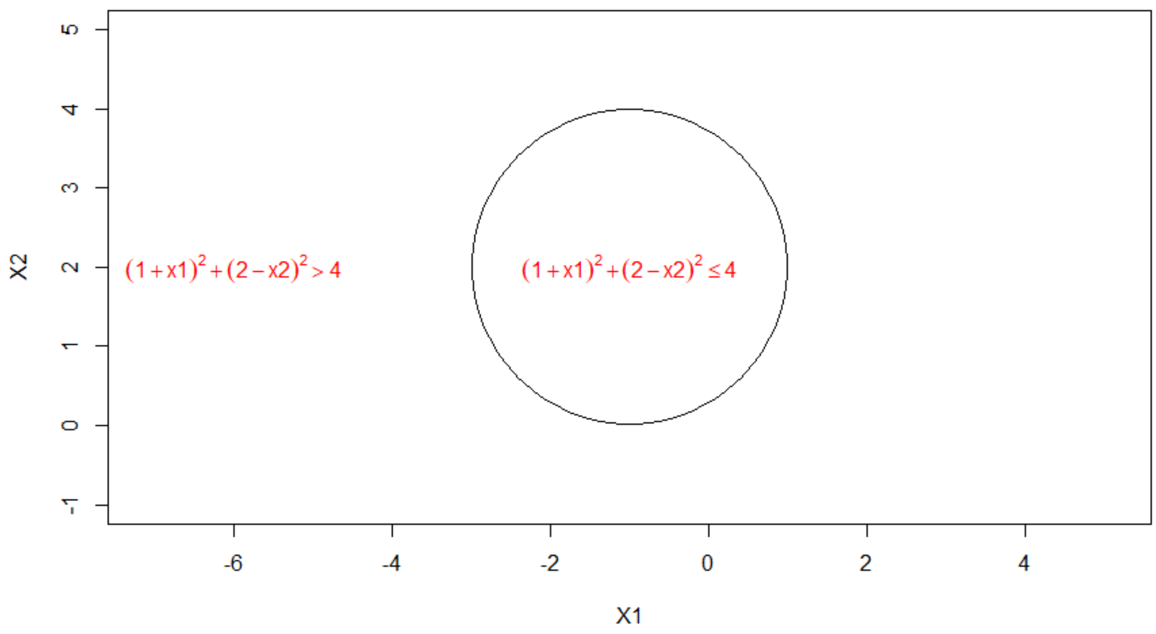
a) Sketch the curve (1 + X1) 2 + (2 − X2) 2 = 4.
The equation describes a circle centered at the point (-1,2)

```
> 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,lwd=3)
```



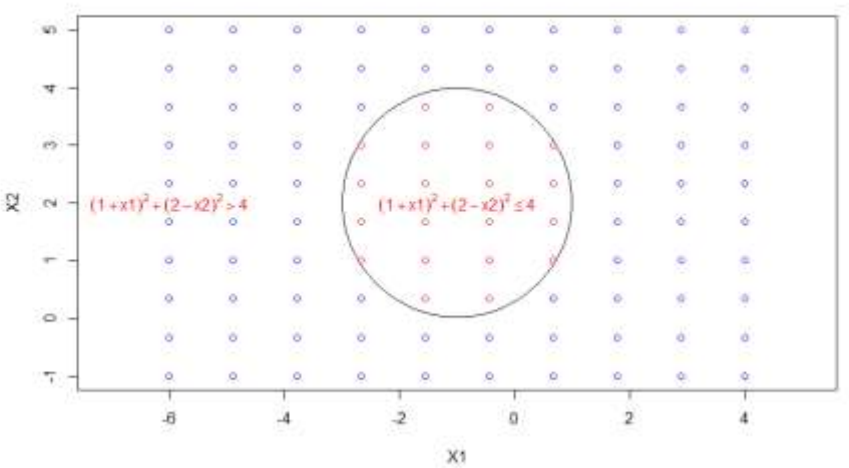
b) On your sketch, indicate the set of points for which (1 + X1) 2 + (2 − X2) 2 > 4, as well as the set of points for which (1 + X1) 2 + (2 − X2) 2 ≤ 4

```
> 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), expression((1+x1)^2 + (2-x2)^2<=4),col = "red")
> text(c(-6), c(2),expression((1+x1)^2 + (2-x2)^2 > 4),col = "red")
>
```



c) Suppose that a classifier assigns an observation to the blue class if (1 + X1) 2 + (2 − X2) 2 > 4, and to the red class otherwise. To what class is the observation (0, 0) classified? (−1, 1)? (2, 2)? (3, 8)?

```
> 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), expression((1+x1)^2 + (2-x2)^2<=4),col = "red")
> text(c(-6), c(2),expression((1+x1)^2 + (2-x2)^2 > 4),col = "red")
> for(i in seq(-6,4,length.out = 10)){
+     pts=data.frame(rep(i,10),seq(-1,5,length.out = 10))
+     points(pts,col=ifelse( (1 + pts[,1])^2 + (2-pts[,2])^2 > 4,'blue','red'))
+ }
> 
```



We can see from the graph that points (0,0), (2,2), and (3,8) belong to the class blue so classify them as blue, and (-1,1) classify as red.

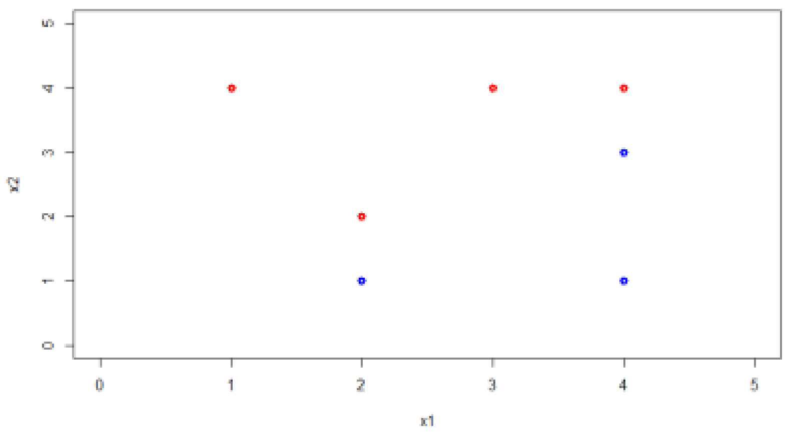
We can also predict class by the substitute value of coordinate (x, y) in a given equation and to check if the result is less or greater than 4.

d) Argue that while the decision boundary in (c) is not linear in terms of X1 and X2, it is linear in terms of X1, X1^1, X2, and X2^2
The decision boundary is simply the boundary described by f(X)=0
Means (1 + X1)^2 + (2 – X2)^2-4=0 ➔ x1^2+x2^2+2x1-4x2+1 =0 which is linear in term of x1,x1^2 ,x2 and x2^2.

Exercise 3:

We are given n = 7 observations in p = 2 dimensions. For each observation, there is an associated class label.

```
> 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),lwd=3)
> 
```

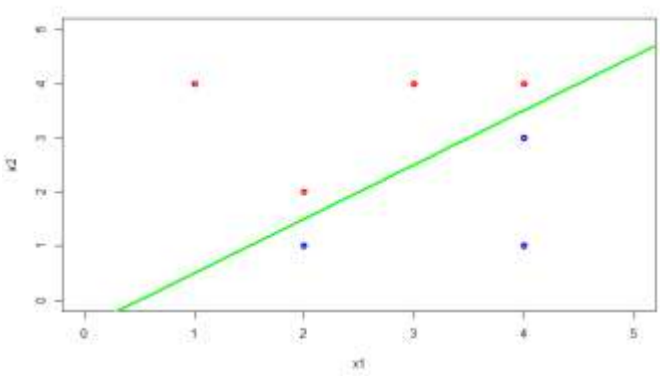


b) Sketch the optimal separating hyperplane, and provide the equation for this hyperplane (of the form (9.1)

We can see in the plot, the optimal separating hyperplane must be between observations (2,1) and (2,2), and between observations (4,3) and (4,4). So, it is a line that passes through the points (2,1.5) and (4,3.5) whose equation is

x1-x2-0.5=0

```
plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5),lwd=3)
abline(-0.5, 1,col="green",lwd=3)
```



c) Describe the classification rule for the maximal margin classifier. It should be something along the lines of "Classify to Red if β0 + β1X1 + β2X2 > 0, and classify to Blue otherwise." Provide the values for β0, β1, and β2.

Line Equation is x1-x2-0.5=0 ➔ x2 = x1 - 0.5

We can observe that the maximal margin classifier will be red if x2 > x1-0.5 else the classifier will be blue.

Maximum margin classifier f(x)= x2- x1 +0.5

f(x) > 0 ➔ observation class is red

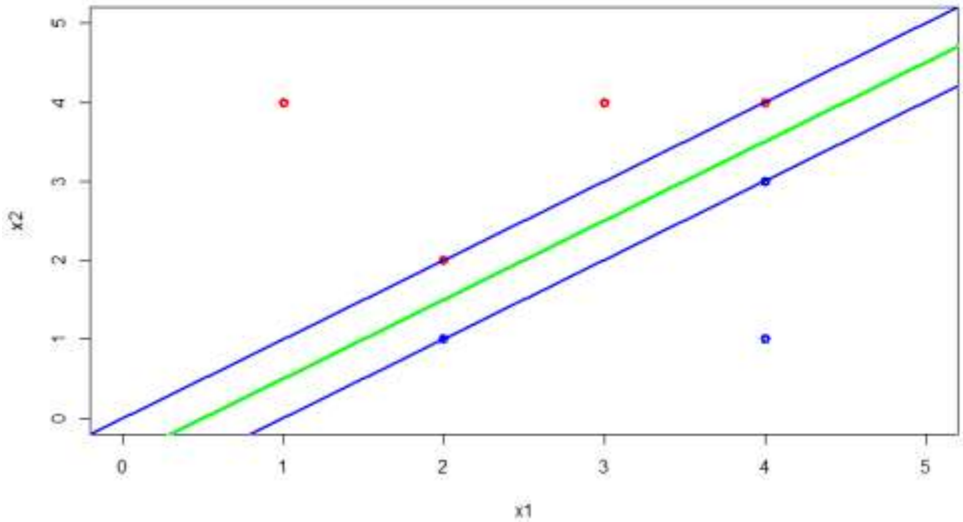
f(x) <=0 ➔ observation class is Blue.

We therefore have coefficients = (0.5, −1,1)

d) On your sketch, indicate the margin for the maximal margin hyperplane.

The solid line indicates the maximal margin hyperplane, and the margin is the distance from the solid green line to either of the blue lines

```
> plot(x1,x2,col=colors,xlim=c(0,5),ylim=c(0,5),lwd=3)
> abline(-0.5,1,col="green",lwd=3)
> abline(-1,1,col="blue",lwd=2)
> abline(0,1,col="blue",lwd=2)
>
```



e) Indicate the support vectors for the maximal margin classifier

The support vectors for the maximal margin classifier are (2,1), (2,2), (4,3), and (4,4)

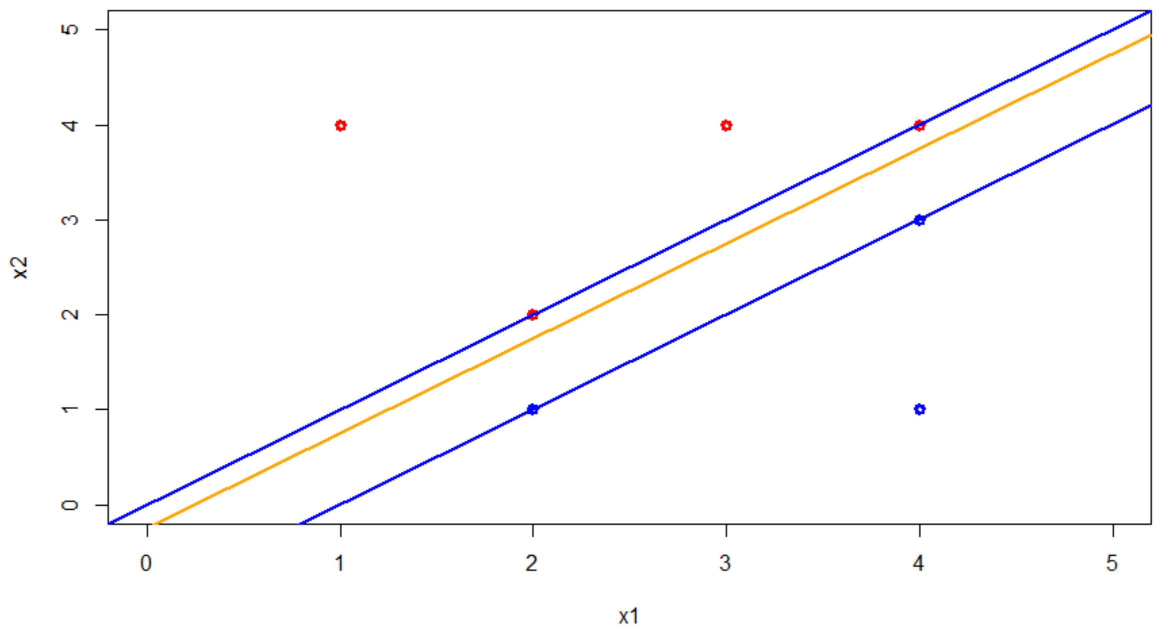
f) Argue that a slight movement of the seventh observation would not affect the maximal margin hyperplane.

From the above graph, we can see that the observation (4,1) is not a support vector. So, any movement of his observation in any direction would not affect the maximum margin hyperplane.

The observation (4,1) would have to be inside the margin to start impacting the position of the maximal margin hyperplane.

g) Sketch a hyperplane that is not the optimal separating hyperplane, and provide the equation for this hyperplane

The hyperplane given by the line is **x1-x2-0.25=0**

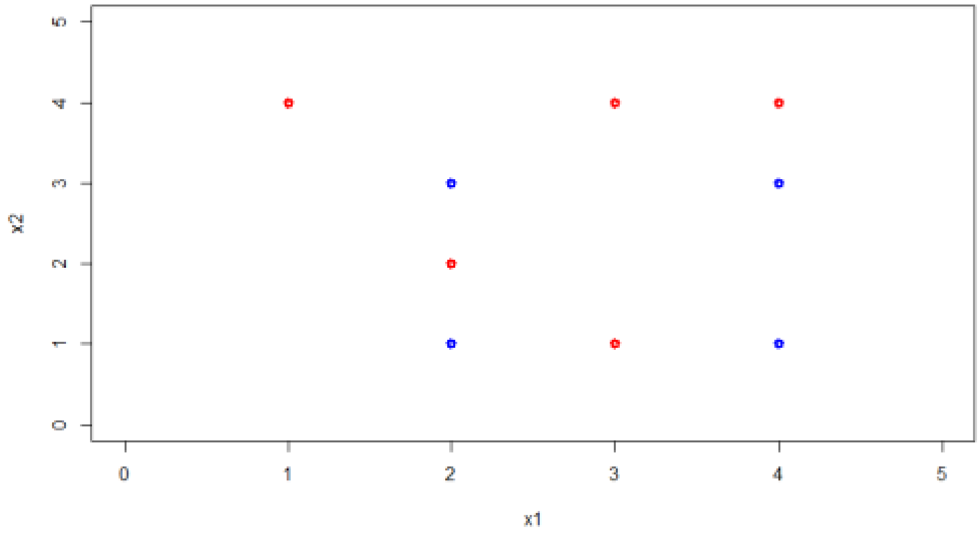


As we can see from the graph the line equation above separates all observations, but it is not optimal as its margin is smaller than the optimal margin.

(h) Draw an additional observation on the plot so that the two classes are no longer separable by a hyperplane.

If we add an observation of either red point (2,3) or blue point (3,1) to the plot, make the two classes no longer be separated by a hyperplane.

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



Problem 2.1

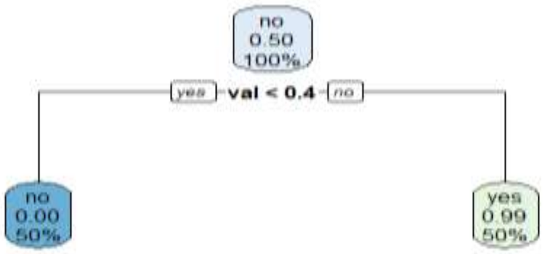
Simulate a binary classification dataset with a single feature via a mixture of normal distributions using R. The normal distribution parameters (using the function rnorm) should be (5,2) and (-5,2) for the pair of samples.

```
> library(rpart)
> library(rpart.plot)
> set.seed(200)
> d1=data.frame(val=rnorm(n=200,mean=5,sd=2),y=rep("yes",200))
> d2=data.frame(val=rnorm(n=200,mean=-5,sd=2),y=rep("no",200))
> data1=rbind(d1,d2)
> summary(data1)
      val                 y
 Min.   :-10.43362   Length:400
 1st Qu.: -5.06981   Class :character
 Median :  1.20257   Mode  :character
 Mean   :  0.03191
 3rd Qu.:  5.02870
 Max.   : 11.17596
```

Induce a binary decision tree (using rpart) and obtain the threshold value for the feature in the first split.

```
> data1$y=as.factor(data1$y)
> tree1=rpart(y~val,data1,method = "class")
> rpart.plot(tree1)
> printcp(tree1)

Classification tree:
rpart(formula = y ~ val, data = data1, method = "class")

Variables actually used in tree construction:
[1] val

Root node error: 200/400 = 0.5

n= 400

    CP nsplit rel error xerror      xstd
1 0.99      0      1.00  1.140 0.0495076
2 0.01      1      0.01  0.015 0.0086277
> rpart.plot(tree1,main="Binary Tree For Dataset1")
```

How does this value compare to the empirical distribution of the feature? How many nodes does this tree have?



The Threshold value for features in the first split is 0.4. Tree has three nodes, one root, and two nodes leaf nodes. The above tree can classify both classes separately which shows empirical distribution.

What is the entropy and Gini at each?

```
> gini=function(p){
+ GI=2 * p* (1-p)
+ return (GI)
+ }
> entropy=function(p){
+ Entropy=(p * log(p)+(1-p)*log(1-p))
+ return(Entropy)
+ }
>
```

```
> plot1=rpart.plot(tree1,extra = 106,fallen.leaves = TRUE,main="Binary Tree for Dataset1")
> plot2=rpart.plot(tree2,extra = 106,fallen.leaves = TRUE,main="Binary Tree for Dataset2")
```

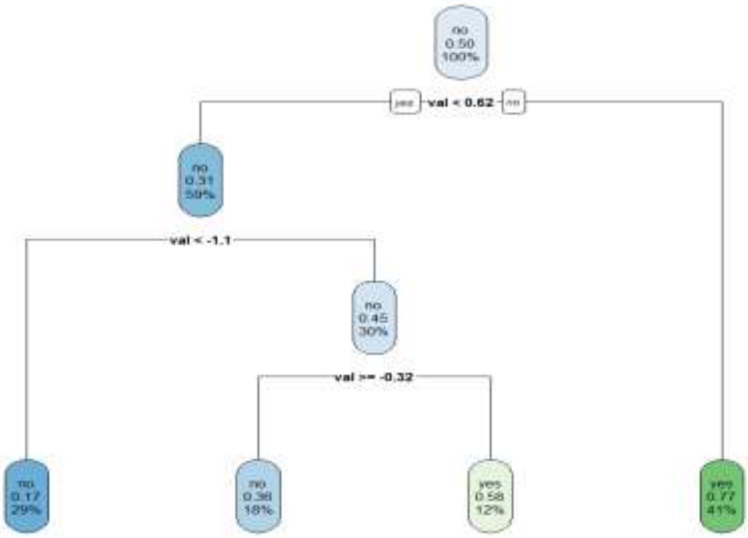
```
> gini_index1=sapply(plot1$obj$frame$yval2[,5],gini)
> gini_index1
[1] 0.50000000 0.00000000 0.01960592
> entropy1=sapply(plot1$obj$frame$yval2[,5],entropy)
> entropy1
[1] -0.69314718         NaN -0.05554608
>
```

Repeat with normal distributions of (1,2) and (-1,2)

```
> set.seed(200)
> d3=data.frame(val=rnorm(n=200,mean=1,sd=2),y=rep("yes",200))
> d4=data.frame(val=rnorm(n=200,mean=-1,sd=2),y=rep("no",200))
> data2=rbind(d3,d4)
> summary(data2)
      val                 y
 Min.   :-6.43362   Length:400
 1st Qu.:-1.38819   Class :character
 Median : 0.09315   Mode  :character
 Mean   : 0.03191
 3rd Qu.: 1.44028
 Max.   : 7.17596
```

Induce a binary decision tree (using rpart) and obtain the threshold value for the feature in the first split.

```
> data2$y=as.factor(data2$y)
> tree2=rpart(y~val,data2,method="class")
> printcp(tree2)

Classification tree:
rpart(formula = y ~ val, data = data2, method = "class")

Variables actually used in tree construction:
[1] val

Root node error: 200/400 = 0.5

n= 400

     CP nsplit rel error xerror     xstd
1 0.445      0     1.000   1.14 0.049508
2 0.020      1     0.555   0.64 0.046648
3 0.010      3     0.515   0.58 0.045376
> rpart.plot(tree2)
```



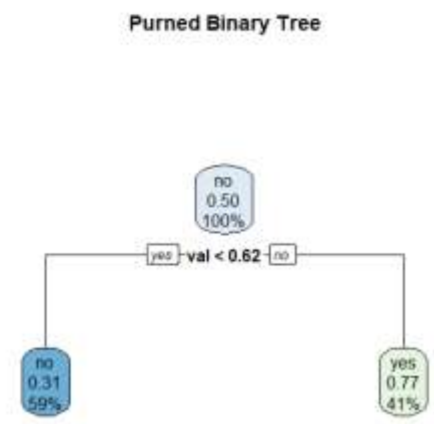
The threshold value for the first split is 0.62. The tree has total of 7 nodes, one root, and 4 leaf nodes. As normal distribution is the very near tree has many nodes with different labels in the node and has more overlapping of labels in nodes.

What are the entropy and Gini at each?

```
> gini_index2
[1] 0.5000000 0.4294896 0.2834393 0.4950000 0.4591837 0.4872000 0.3509353
> entropy2=sapply(plot2$obj$frame$yval2[,5],entropy)
> entropy2
[1] -0.6931472 -0.6208783 -0.4573738 -0.6881388 -0.6517566 -0.6802920 -0.5356184
> 
```

Prune this tree (using part.prune) with a complexity parameter of 0.1. Describe the resulting tree.

```
> tree3=prune.rpart(tree2,cp=0.1)
> printcp(tree3)

Classification tree:
rpart(formula = y ~ val, data = data2, method = "class")

Variables actually used in tree construction:
[1] val

Root node error: 200/400 = 0.5

n= 400

     CP nsplit rel error xerror     xstd
1 0.445      0     1.000   1.14 0.049508
2 0.100      1     0.555   0.64 0.046648
> plot3=rpart.plot(tree3,extra = 106,main="Purned Binary Tree")
> gini_index3=sapply(plot3$obj$frame$yval2[,5],gini)
> gini_index3
[1] 0.5000000 0.4294896 0.3509353
> entropy3=sapply(plot3$obj$frame$yval2[,5],entropy)
> entropy3
[1] -0.6931472 -0.6208783 -0.5356184
> 
```



The threshold value for the first split is 0.62. The tree has 3 nodes, one root node, and 2 leaf nodes. After the pruning process tree is much better than the previous one as this has only two leaf nodes with few overlapping labels.

Entropy and Gini value calculation:

Problem 2.2

Load the Wine Quality sample dataset from the UCI Machine Learning Repository (inequality-red.csv and winequality-white.csv) into R using a data frame.

```
> library(caret)
> library(randomForest)
> library(rpart)
> library(rpart.plot)
> white_winequality=read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv",header = TRUE,sep = ";",stringsAsFactors = TRUE)
> red_winequality=read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv",header = TRUE,sep = ";",stringsAsFactors = TRUE)
>
```

```
> str(white_winequality)
'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 ...
> summary(white_winequality)
 fixed.acidity    volatile.acidity  citric.acid     residual.sugar     chlorides       free.sulfur.dioxide total.sulfur.dioxide    density             pH
 Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600   Min.   :0.00900   Min.   :  2.00      Min.   :  9.0        Min.   :0.9871   Min.   :2.720
 1st Qu.: 6.300   1st Qu.:0.2100   1st Qu.:0.2700   1st Qu.: 1.700   1st Qu.:0.03600   1st Qu.: 23.00      1st Qu.:108.0        1st Qu.:0.9917   1st Qu.:3.090
 Median : 6.800   Median :0.2600   Median :0.3200   Median : 5.200   Median :0.04300   Median : 34.00      Median :134.0        Median :0.9937   Median :3.180
 Mean   : 6.855   Mean   :0.2782   Mean   :0.3342   Mean   : 6.391   Mean   :0.04577   Mean   : 35.31      Mean   :138.4        Mean   :0.9940   Mean   :3.188
 3rd Qu.: 7.300   3rd Qu.:0.3200   3rd Qu.:0.3900   3rd Qu.: 9.900   3rd Qu.:0.05000   3rd Qu.: 46.00      3rd Qu.:167.0        3rd Qu.:0.9961   3rd Qu.:3.280
 Max.   :14.200   Max.   :1.1000   Max.   :1.6600   Max.   :65.800   Max.   :0.34600   Max.   :289.00      Max.   :440.0        Max.   :1.0390   Max.   :3.820
   sulphates         alcohol         quality
 Min.   :0.2200   Min.   : 8.00   Min.   :3.000
 1st Qu.:0.4100   1st Qu.: 9.50   1st Qu.:5.000
 Median :0.4700   Median :10.40   Median :6.000
 Mean   :0.4898   Mean   :10.51   Mean   :5.878
 3rd Qu.:0.5500   3rd Qu.:11.40   3rd Qu.:6.000
 Max.   :1.0800   Max.   :14.20   Max.   :9.000
>
```

```
> str(red_winequality)
'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 ...
> summary(red_winequality)
 fixed.acidity   volatile.acidity  citric.acid    residual.sugar     chlorides       free.sulfur.dioxide total.sulfur.dioxide    density             pH
 Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900   Min.   :0.01200   Min.   : 1.00       Min.   :  6.00       Min.   :0.9901   Min.   :2.740
 1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900   1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00       1st Qu.:0.9956   1st Qu.:3.210
 Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200   Median :0.07900   Median :14.00       Median : 38.00       Median :0.9968   Median :3.310
 Mean   : 8.32   Mean   :0.5278   Mean   :0.271   Mean   : 2.539   Mean   :0.08747   Mean   :15.87       Mean   : 46.47       Mean   :0.9967   Mean   :3.311
 3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600   3rd Qu.:0.09000   3rd Qu.:21.00       3rd Qu.: 62.00       3rd Qu.:0.9978   3rd Qu.:3.400
 Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500   Max.   :0.61100   Max.   :72.00       Max.   :289.00       Max.   :1.0037   Max.   :4.010
   sulphates         alcohol         quality
 Min.   :0.3300   Min.   : 8.40   Min.   :3.000
 1st Qu.:0.5500   1st Qu.: 9.50   1st Qu.:5.000
 Median :0.6200   Median :10.20   Median :6.000
 Mean   :0.6581   Mean   :10.42   Mean   :5.636
 3rd Qu.:0.7300   3rd Qu.:11.10   3rd Qu.:6.000
 Max.   :2.0000   Max.   :14.90   Max.   :8.000
>
```

Create an 80/20 test-train split of each wine data frame

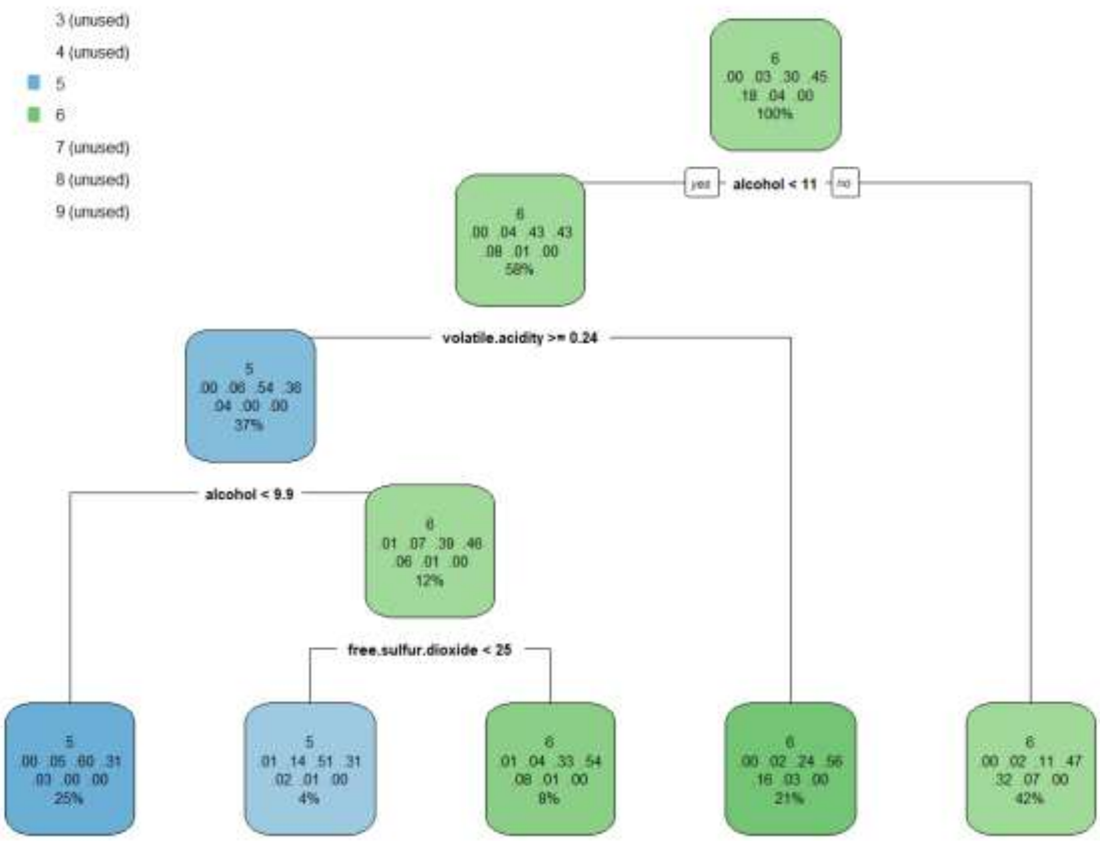
```
> white_winequality$quality=as.factor(white_winequality$quality)
> index=createDataPartition(white_winequality$quality,p=0.8,list=FALSE)
> train_white=white_winequality[index,]
> test_white=white_winequality[-index,]
> dim(train_white)
[1] 3920   12
> dim(test_white)
[1] 978  12
>
```

```
> red_winequality$quality=as.factor(red_winequality$quality)
> index1=createDataPartition(red_winequality$quality,p=0.8,list = FALSE)
> train_red=red_winequality[index1,]
> test_red=red_winequality[-index1,]
> dim(train_red)
[1] 1282   12
> dim(test_red)
[1] 317  12
>
```

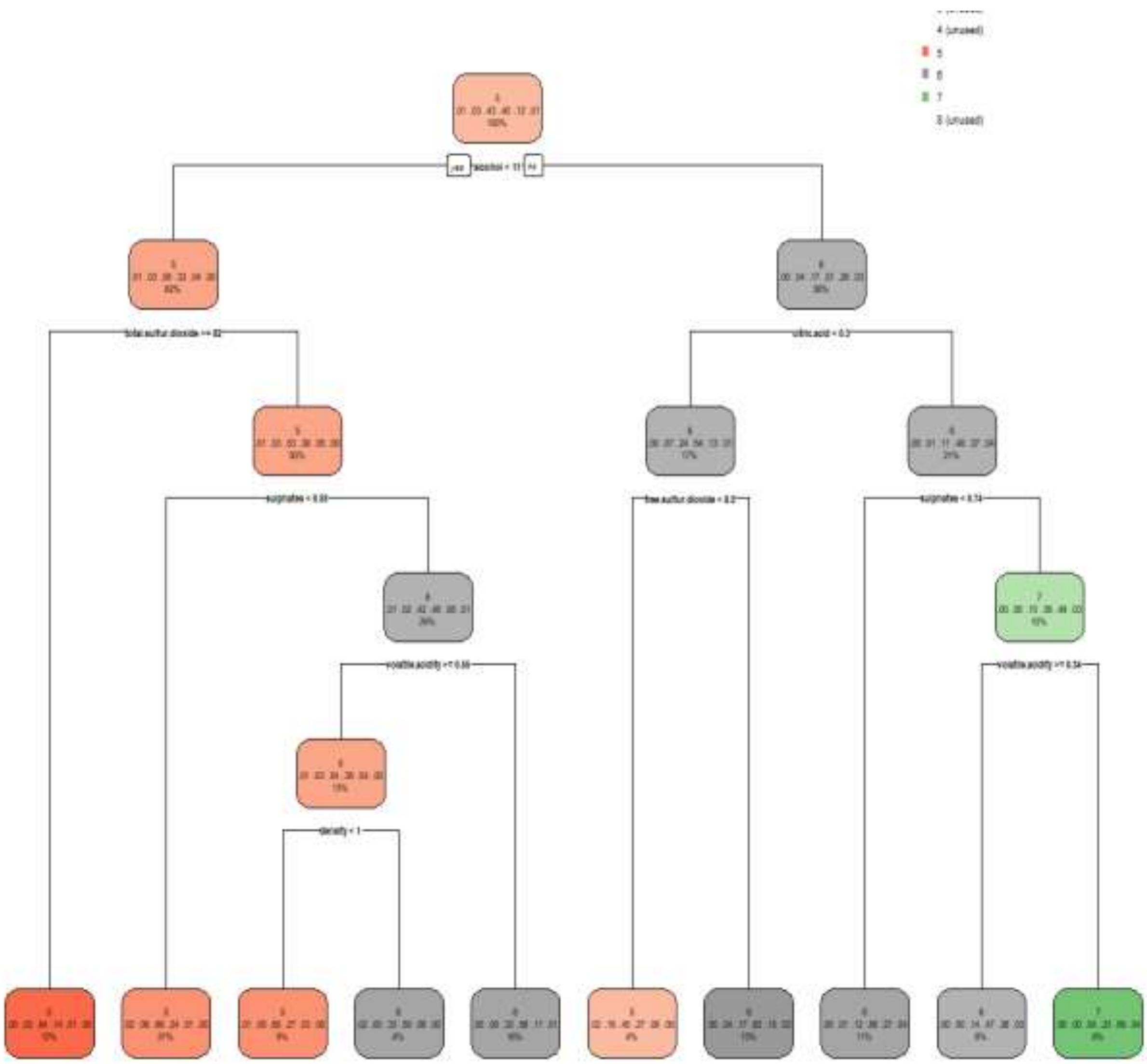
Decision Tree Model

Use the rpart package to induce a decision tree of both the red and white wines, targeting the quality output variable and visualize the tree using the part. Plot library

```
> tree_white=rpart(quality~.,data = train_white)
> rpart.plot(tree_white)
```



```
> tree_red=rpart(quality~.,data = train_red)
> rpart.plot(tree_red)
>
```



use the caret package confusion Matrix method to determine the decision tree accuracy on the test set.

```
> tree_predict_white=predict(tree_white,test_white,type = "class")
> confusionMatrix(tree_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  12 167  90   9   0   0
         6   3  20 124 349 167  35   1
         7   0   0   0   0   0   0   0
         8   0   0   0   0   0   0   0
         9   0   0   0   0   0   0   0

Overall Statistics

               Accuracy : 0.5276
                 95% CI : (0.4958, 0.5593)
    No Information Rate : 0.4489
    P-Value [Acc > NIR] : 4.729e-07

                  Kappa : 0.2051

 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   0.5739   0.7950     0.00  0.00000
Specificity           1.00000  1.00000   0.8370   0.3506     1.00  1.00000
Pos Pred Value            NaN      NaN   0.5986   0.4993      NaN      NaN
Neg Pred Value        0.99591  0.96728   0.8226   0.6774     0.82  0.96421
Prevalence            0.00409  0.03272   0.2975   0.4489     0.18  0.03579
Detection Rate        0.00000  0.00000   0.1708   0.3569     0.00  0.00000
Detection Prevalence  0.00000  0.00000   0.2853   0.7147     0.00  0.00000
Balanced Accuracy     0.50000  0.50000   0.7054   0.5728     0.50  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
> 
```

```
> tree_predict_red=predict(tree_red,test_red,type = "class")
> confusionMatrix(tree_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  6 92 30  1  0
         6  1  4 43 89 31  2
         7  0  0  1  8  7  1
         8  0  0  0  0  0  0

Overall Statistics

               Accuracy : 0.5931
                 95% CI : (0.5367, 0.6476)
    No Information Rate : 0.429
    P-Value [Acc > NIR] : 3.186e-09

                  Kappa : 0.3247

 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   0.6765   0.7008  0.17949 0.000000
Specificity          1.000000  1.00000   0.7901   0.5737  0.96403 1.000000
Pos Pred Value            NaN      NaN   0.7077   0.5235  0.41176      NaN
Neg Pred Value       0.993691  0.96845   0.7647   0.7415  0.89333 0.990536
Prevalence           0.006309  0.03155   0.4290   0.4006  0.12303 0.009464
Detection Rate       0.000000  0.00000   0.2902   0.2808  0.02208 0.000000
Detection Prevalence 0.000000  0.00000   0.4101   0.5363  0.05363 0.000000
Balanced Accuracy    0.500000  0.50000   0.7333   0.6372  0.57176 0.500000
>
```

Compare the decision trees for red and white wine - what differences in terms of tree structure and variables of interest can be noted?

```
> table(tree_predict_white)
tree_predict_white
  3   4   5   6   7   8   9
  0   0 279 699   0   0   0
> table(tree_predict_red)
tree_predict_red
  3   4   5   6   7   8
  0   0 130 170  17   0
>
```

```
> varImp(tree_white)
                        Overall
alcohol              205.048111
chlorides             88.026582
citric.acid           23.183313
density              114.509589
free.sulfur.dioxide   42.231833
pH                    12.091773
residual.sugar         7.215579
total.sulfur.dioxide  57.891128
volatile.acidity     152.888927
fixed.acidity          0.000000
sulphates              0.000000
> varImp(tree_red)
                        Overall
alcohol              109.388331
chlorides              5.276332
citric.acid           21.855882
density               38.366207
fixed.acidity         34.513261
pH                     4.443578
residual.sugar         3.356979
sulphates             76.659617
total.sulfur.dioxide  56.906256
volatile.acidity      87.244008
free.sulfur.dioxide    0.000000
>
```

Use the Random Forest package to repeat the fit with a random forest tree model, and compare the resulting test accuracy against the original single tree model.

```
> rand_forest_white=randomForest(quality~.,data = train_white)
> rand_forest_predictw=predict(object = rand_forest_white,newdata = test_white)
> cm_rand_forest_white=confusionMatrix(data = rand_forest_predictw,reference = test_white$quality)
> rand_forest_red=randomForest(quality~.,data = train_red)
> rand_forest_predictr=predict(object = rand_forest_red,newdata = test_red)
> cm_rand_forest_red=confusionMatrix(data = rand_forest_predictr,reference = test_red$quality)
```

Original Tree Model Accuracy

```
> cm_red=confusionMatrix(tree_predict_red,test_red$quality)
> cm_red$overall["Accuracy"]
 Accuracy
0.5930599
> cm_white=confusionMatrix(tree_predict_white,test_white$quality)
> cm_white$overall["Accuracy"]
 Accuracy
0.5276074
>
```

Rainforest Model Accuracy

```
> cm_rand_forest_red$overall["Accuracy"]
 Accuracy
0.6940063
> cm_rand_forest_white$overall["Accuracy"]
 Accuracy
0.6779141
>
```

Random Forest performed better as it returned an accuracy of 69.4% for the red Wine Dataset Random Forest returned an accuracy of 67.7% for the white Wine Dataset. The Accuracy increased from 59% to 69% in Random Forest Classifier in the red Wine Dataset and increased from 52% to 67% in Random Forest Classifier in the white Wine Dataset.

Problem 2.3

Load the SMS Spam Collection sample dataset from the UCI Machine Learning Repository (smsspamcollection.zip) into R using a data frame

```
> sms_data <- tempfile()
>
> download.file("https://archive.ics.uci.edu/ml/machine-learning-databases/00228/smsspamcollection.zip",sms_data)
```

```
> smsdata <- read.table(unz(sms_data, "SMSSpamCollection"),header = FALSE,sep="\t",stringsAsFactors = FALSE)
```

```
> unlink(sms_data)
> colnames(smsdata) <- c("type","text")
>
> head(smsdata)
  type
1  ham
2  ham
3 spam
4  ham
5  ham
6  ham

                                                                                                    text
1                                                                                                                         Go until jurong point,
 crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
2
                                                                                   Ok lar... Joking wif u oni...
3                                                                                    Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 200
5. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
4
                                                             U dun say so early hor... U c already then say...
5 Nah I dont think he goes to usf, he lives around here though\nspam\tFreeMsg Hey there darling its been 3 weeks now and no wo
rd back! Id like some fun you up for it still? Tb ok! XxX std chgs to send, £1.50 to rcv
6
           Even my brother is not like to speak with me. They treat me like aids patent.
> str(smsdata)
'data.frame':   1779 obs. of  2 variables:
 $ type: chr  "ham" "ham" "spam" "ham" ...
 $ text: chr  "Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wa
t..." "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..." ...
>
```

Use the tm package to create a Corpus of documents (Hint: Construct the corpus using a Vector Source of the text column)

```
> smsdata$type <- as.factor(smsdata$type)
> str(smsdata)
'data.frame':   1779 obs. of  2 variables:
 $ type: Factor w/ 2 levels "ham","spam": 1 1 2 1 1 1 1 2 2 1 ...
 $ text: chr  "Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wa
t..." "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..." ...
> summary(smsdata)
   type          text
 ham :1541   Length:1779
 spam: 238   Class :character
             Mode  :character
>
```

```
> library(tm)
Loading required package: NLP
Warning message:
package 'tm' was built under R version 4.2.2
> smsCorpus <- VCorpus(VectorSource(smsdata$text))
> print(smsCorpus)
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 1779
>
```

Inspect used to see a summary of a specific message

```
> inspect(head(smsCorpus,2))
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 2

[[1]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 111

[[2]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 29

> |
```

as. character() used to view the desired message

```
> as.character(head(smsCorpus,1))
[1] "list(list(content = \"Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got
 amore wat...\", meta = list(author = character(0), datetimestamp = list(sec = 29.1702029705048, min = 36, hour = 5, mday = 1
4, mon = 10, year = 122, wday = 1, yday = 317, isdst = 0), description = character(0), heading = character(0), id = \"1\", lan
guage = \"en\", origin = character(0))))"
[2] "list()"


[3] "list()"
```

Apply the following transformations from the tm package to the corpus in order to prepare the data:

a) Convert lowercase b) Remove stop words c) Strip whitespace d) Remove punctuation.

```
> smsdataT <- tm_map(smsCorpus,content_transformer(tolower))
>
> as.character(smsdataT[[1]])
[1] "go until jurong point, crazy.. available only in bugis n great world la e buffet... cine there got amore wat..."
> smsdataT <- tm_map(smsdataT,removewords,stopwords())
> as.character(smsdataT[[1]])
[1] "go  jurong point, crazy.. available   bugis n great world la e buffet... cine  got amore wat..."
> smsdataT <- tm_map(smsdataT,stripwhitespace)
> as.character(smsdataT[[1]])
[1] "go jurong point, crazy.. available bugis n great world la e buffet... cine got amore wat..."
> smsdataT <- tm_map(smsdataT,removePunctuation)
> as.character(smsdataT[[1]])
[1] "go jurong point crazy available bugis n great world la e buffet cine got amore wat"
> |
```

Use findFreqTerms to construct features from words occurring more than 10 times and proceed to split the data into a training and test set - for each creates a DocumentTermMatrix.

Create DocumentTermMatrix

Splitting into Train Test Split in 80% Training and 20% Testing

```
> smsDTM <- DocumentTermMatrix(smsdataT)
> smsFreqterm <- findFreqTerms(smsDTM,lowfreq=10)
```

```
> head(smsFreqterm)
[1] "£100"  "£1000" "£150"  "£2000" "£250"  "£350"
```

```
> smsfeatures.df <- as.data.frame(data.matrix(smsDTM),stringsAsFactors=FALSE)
> smsfeatures.df <- smsfeatures.df[,smsFreqterm]
> smsdataT.df <- cbind("Type" = smsdata$type, smsfeatures.df)
```

```
> library(caret)
> splitIndex <- createDataPartition(smsdataT.df$Type,p=0.8,list=FALSE)
>
> sms_train=data.matrix((smsdataT.df[splitIndex,-c(1)]))
> sms_test=data.matrix((smsdataT.df[-splitIndex,-c(1)]))
> sms_train_label=smsdataT.df[splitIndex,c(1)]
> sms_test_label=smsdataT.df[-splitIndex,c(1)]
> makeboolean=function(x){
+ x=ifelse(x>0,1,0)
+ }
> sms.train=apply(sms_train,2,makeboolean)
> sms.test=apply(sms_test,2,makeboolean)
>
```

fit an SVM using the e1071 package.

```
> sms_clsssifier=svm(sms.train,sms_train_label)
> summary(sms_clsssifier)

Call:
svm.default(x = sms.train, y = sms_train_label)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  665

 ( 181 484 )


Number of Classes:  2

Levels:
 ham spam
```

calculate test and train predictions, create confusion matrices

```
> sms_train_pred=predict(sms_clsssifier,sms.train)
> sms_test_pred=predict(sms_clsssifier,sms.test)
> sms_train_cm=confusionMatrix(sms_train_pred,sms_train_label)
> sms_test_cm=confusionMatrix(sms_test_pred,sms_test_label)
```

Training and Test Confusion Matrix

```
> sms_train_cm
Confusion Matrix and Statistics

          Reference
Prediction  ham spam
      ham  1233   21
      spam    0  170

               Accuracy : 0.9853
                 95% CI : (0.9775, 0.9908)
    No Information Rate : 0.8659
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9334

 Mcnemar's Test P-Value : 1.275e-05

            Sensitivity : 1.0000
            Specificity : 0.8901
         Pos Pred Value : 0.9833
         Neg Pred Value : 1.0000
             Prevalence : 0.8659
         Detection Rate : 0.8659
   Detection Prevalence : 0.8806
      Balanced Accuracy : 0.9450

       'Positive' Class : ham

> |
```

```
> sms_test_cm
Confusion Matrix and Statistics

          Reference
Prediction ham spam
      ham  308   21
      spam   0   26

               Accuracy : 0.9408
                 95% CI : (0.911, 0.963)
    No Information Rate : 0.8676
    P-Value [Acc > NIR] : 5.803e-06

                  Kappa : 0.6824

 Mcnemar's Test P-Value : 1.275e-05

            Sensitivity : 1.0000
            Specificity : 0.5532
         Pos Pred Value : 0.9362
         Neg Pred Value : 1.0000
             Prevalence : 0.8676
         Detection Rate : 0.8676
   Detection Prevalence : 0.9268
      Balanced Accuracy : 0.7766

       'Positive' Class : ham

>
```

Report your training and test set accuracy

```
> sms_train_cm$overall["Accuracy"]
 Accuracy
0.9852528
> sms_test_cm$overall["Accuracy"]
 Accuracy
0.9408451
>
```