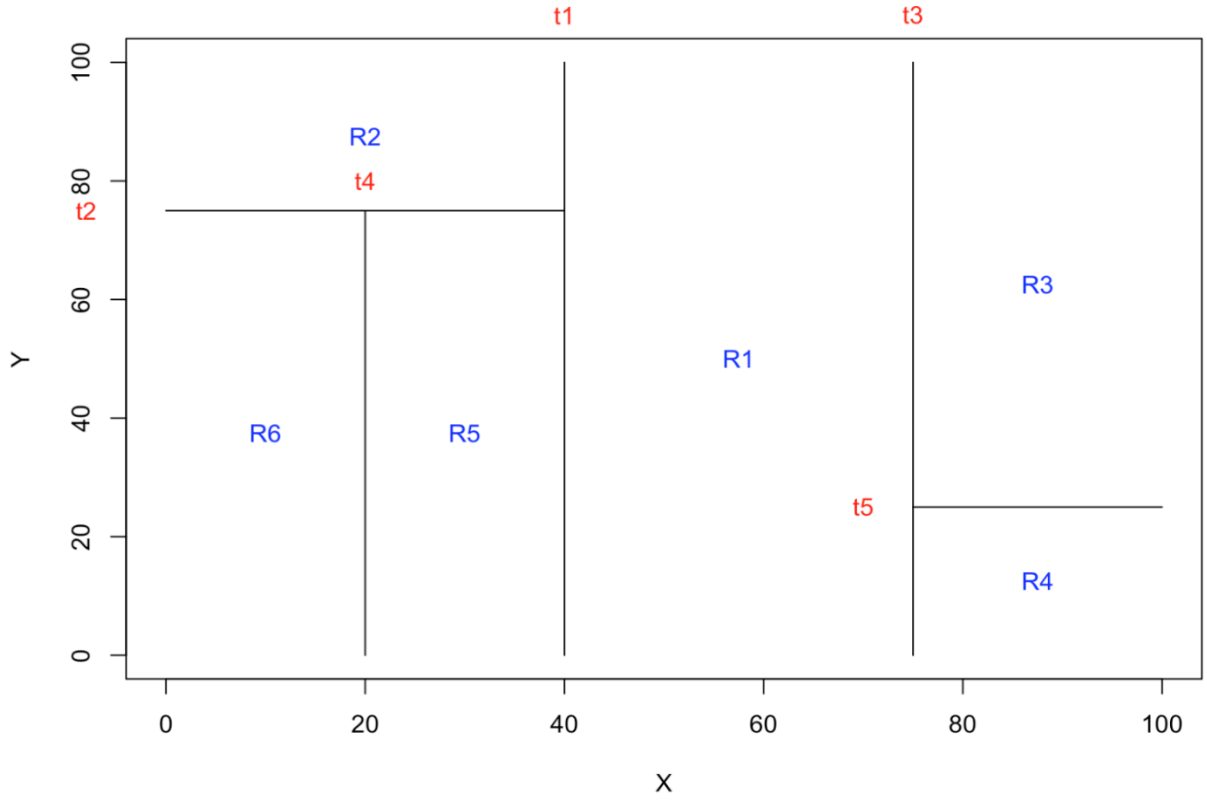
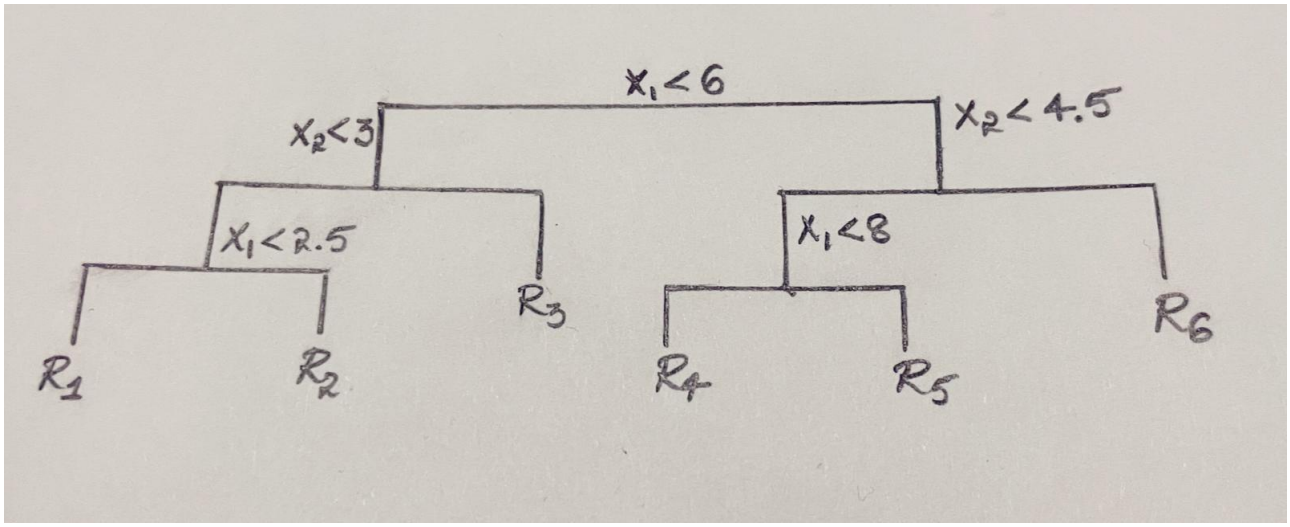
Shiva Sankar Modala | A20517528 | CSP 571 Spring 2023

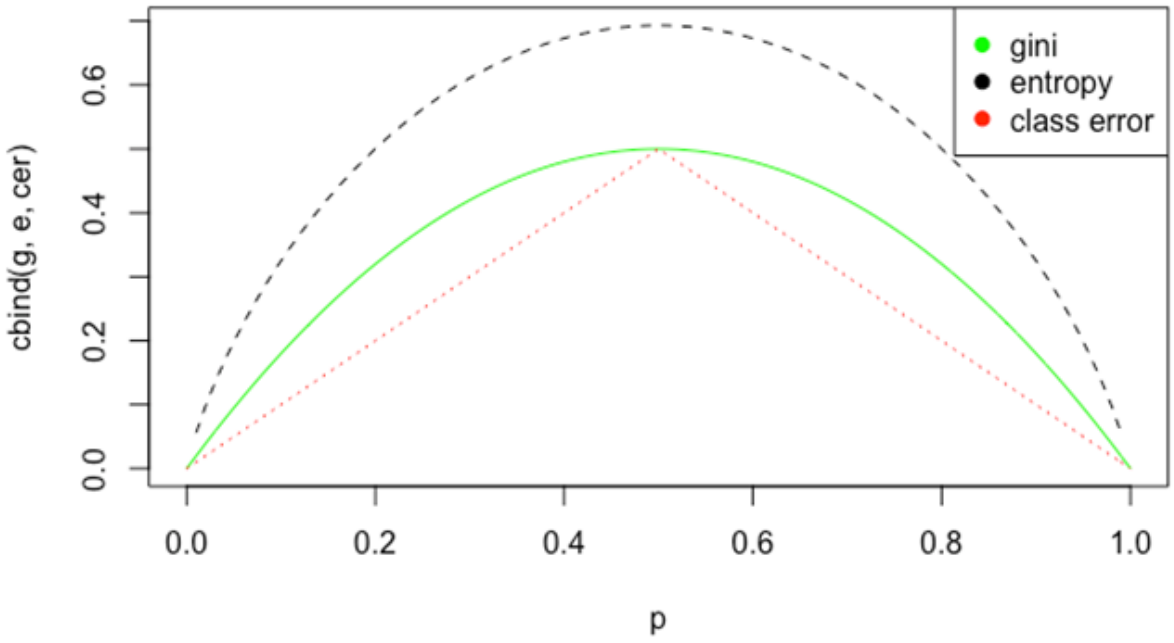
Data Preparation and Analysis Assignment 4

Recitation Exercises:
Chapter 8
Exercise 1 Answers:



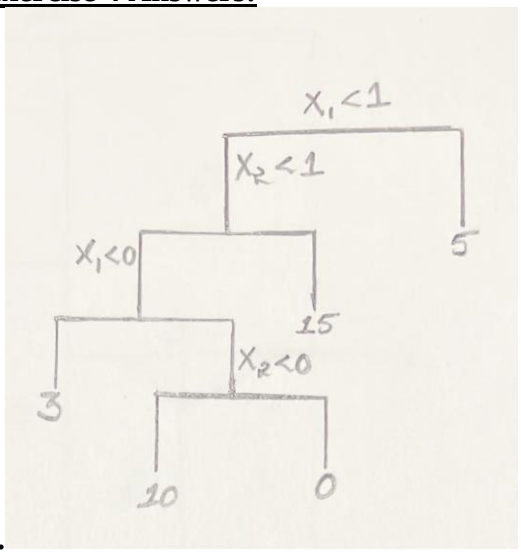


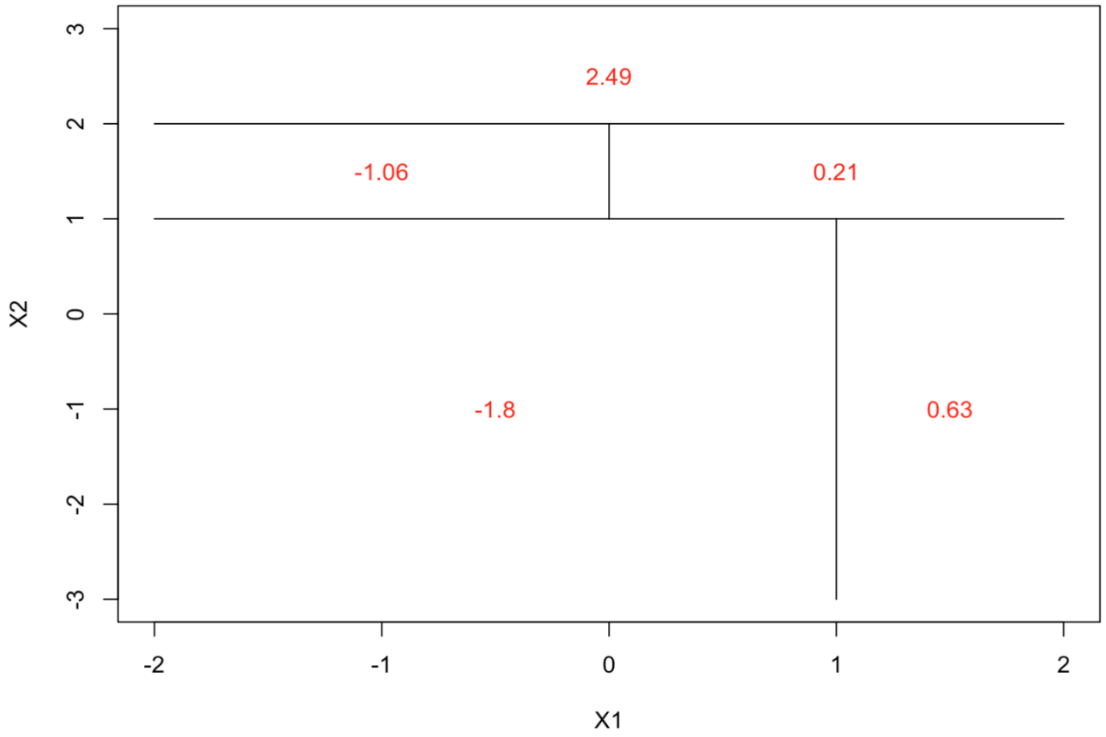
Exercise 3 Answers:



Exercise 4 Answers:

a.





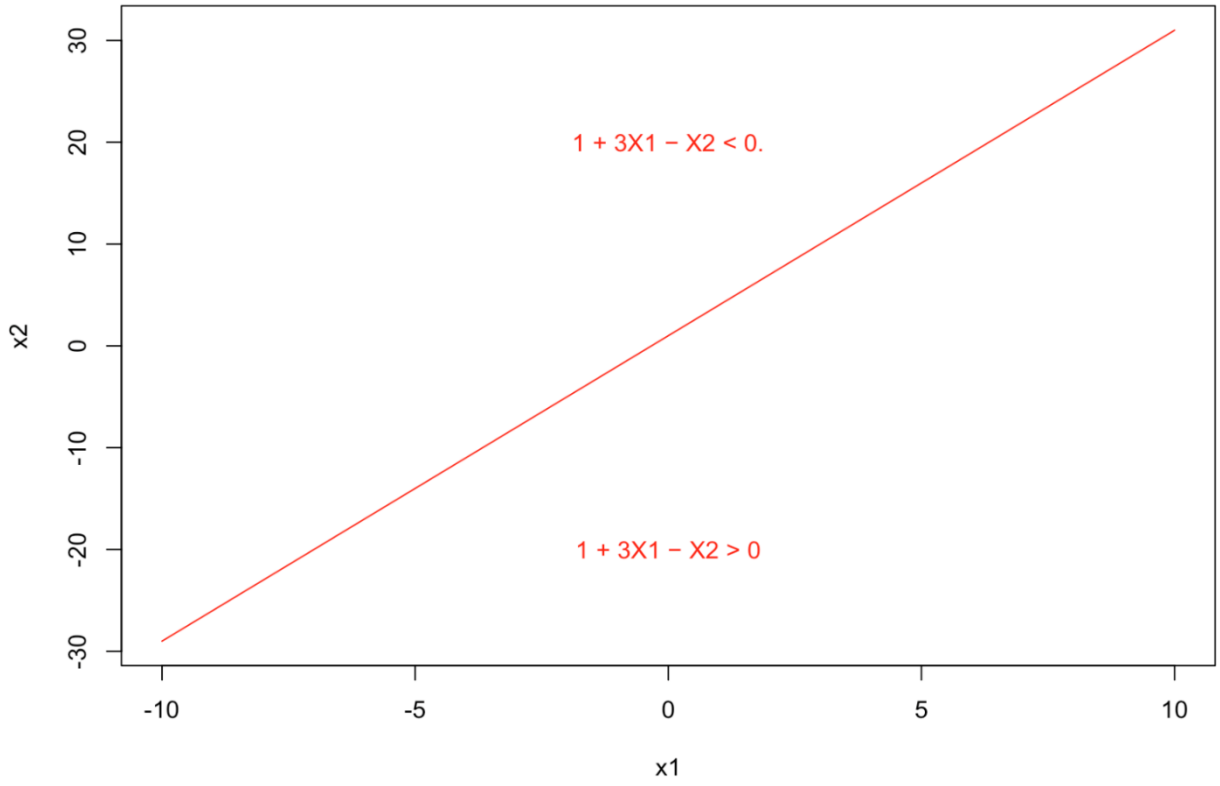
b.

Exercise 5 Answers:

According to the majority vote method, we classify X as Red because there are six instances of Red and four instances of Green in the ten forecasts. Because the average of the 10 probabilities is 0.45, we classify X as green according to the average probability approach.

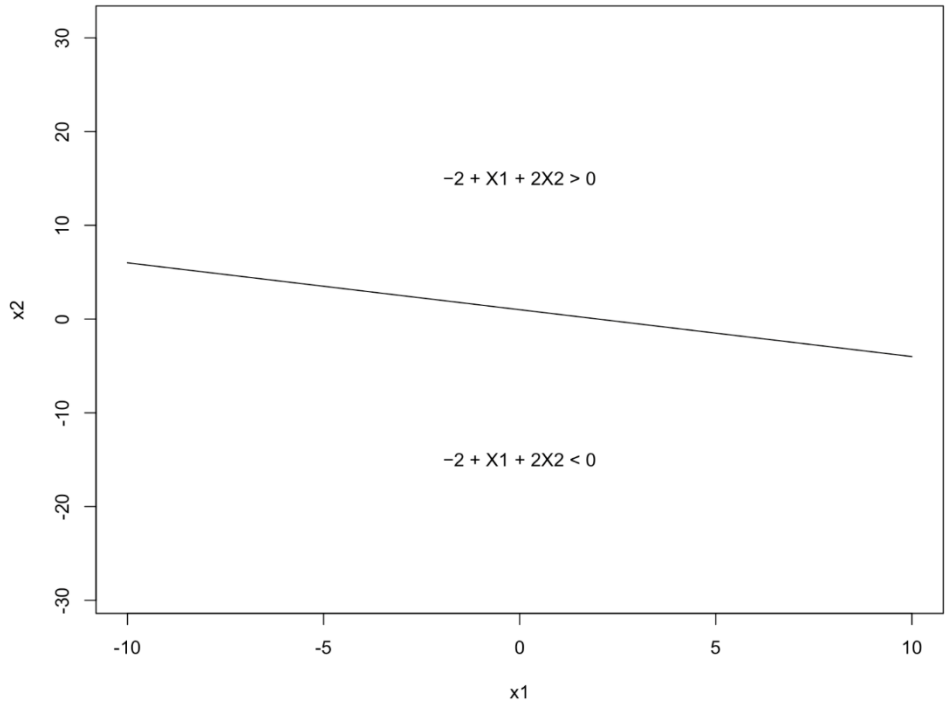
Chapter 9
Exercise 1 Answers:

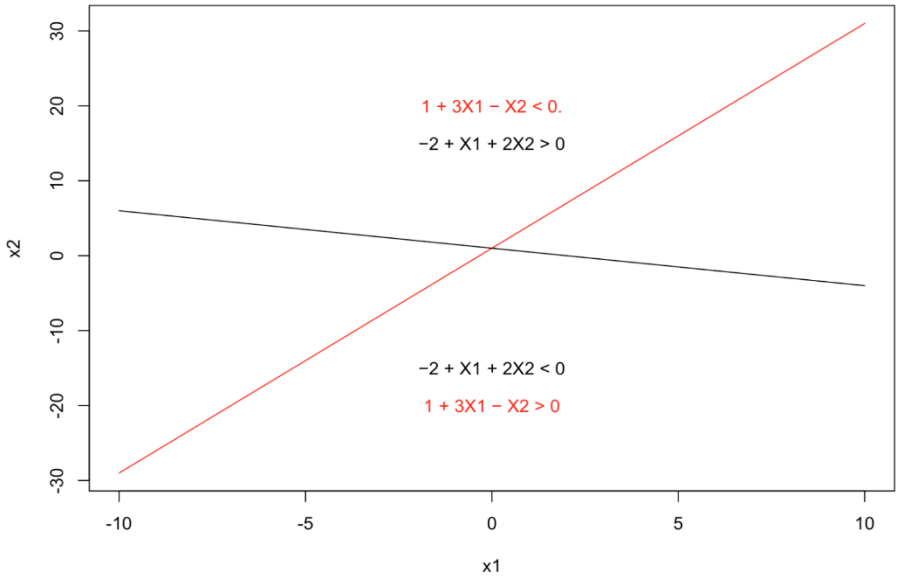
a.
Hyperplane for 1 + 3X1 − X2 = 0



b.

Hyperplane for −2 + X1 + 2X2 = 0

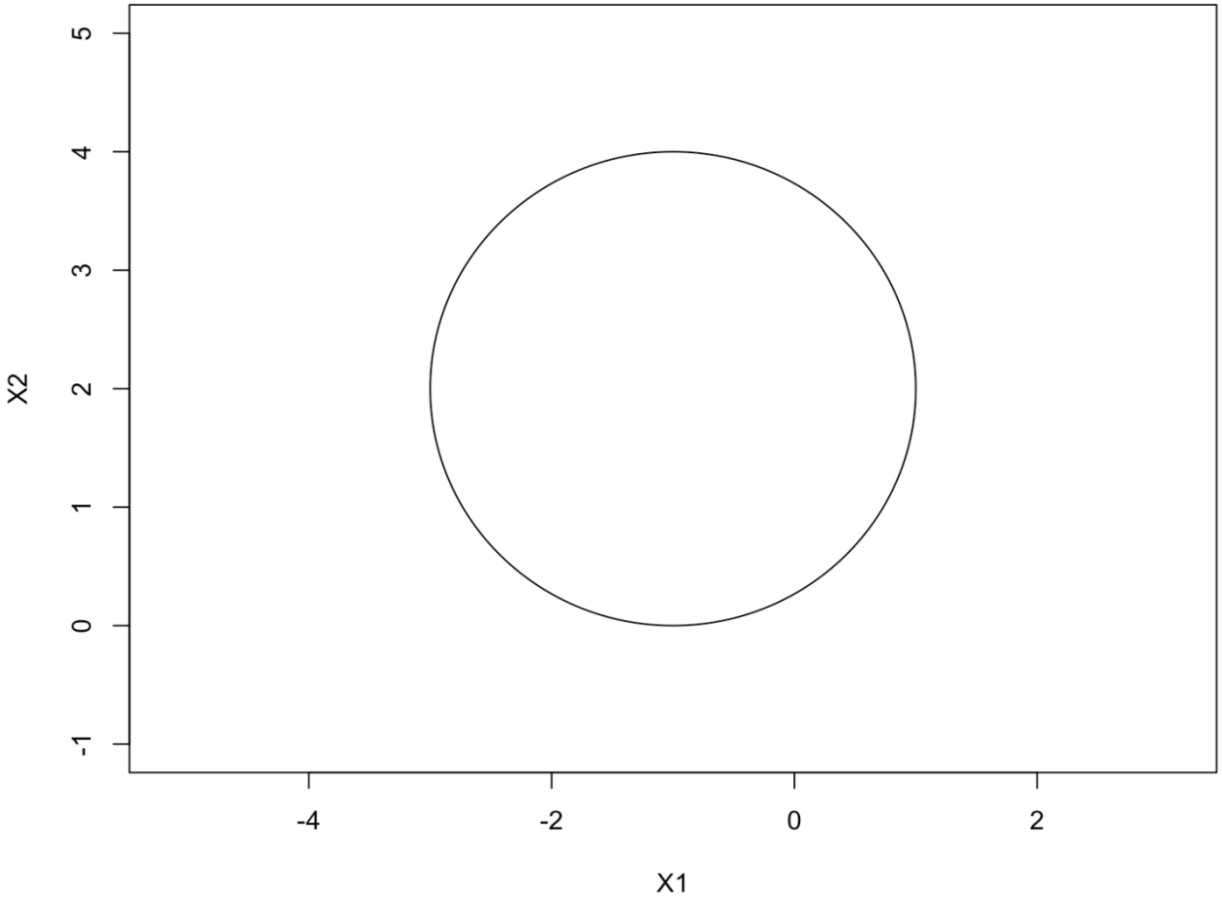




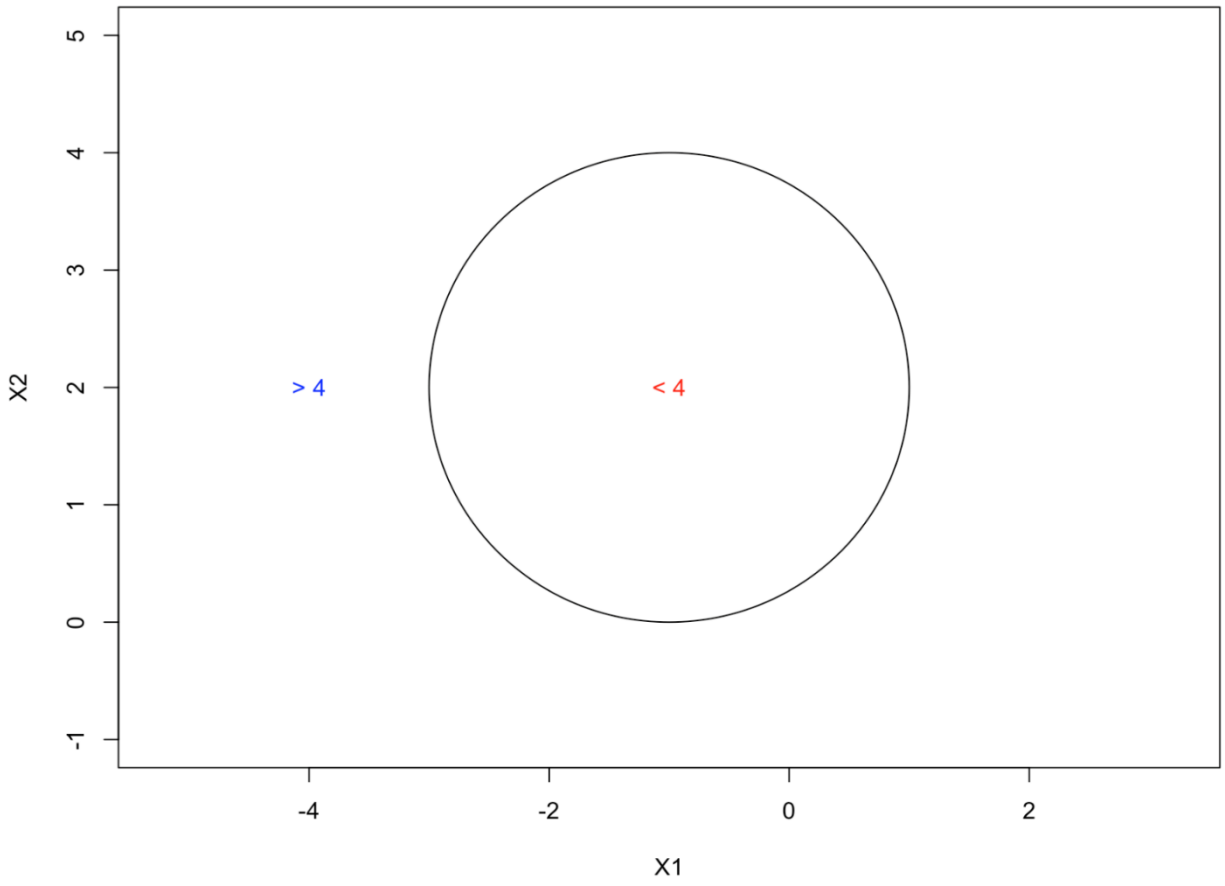
Exercise 2 Answers:

a.

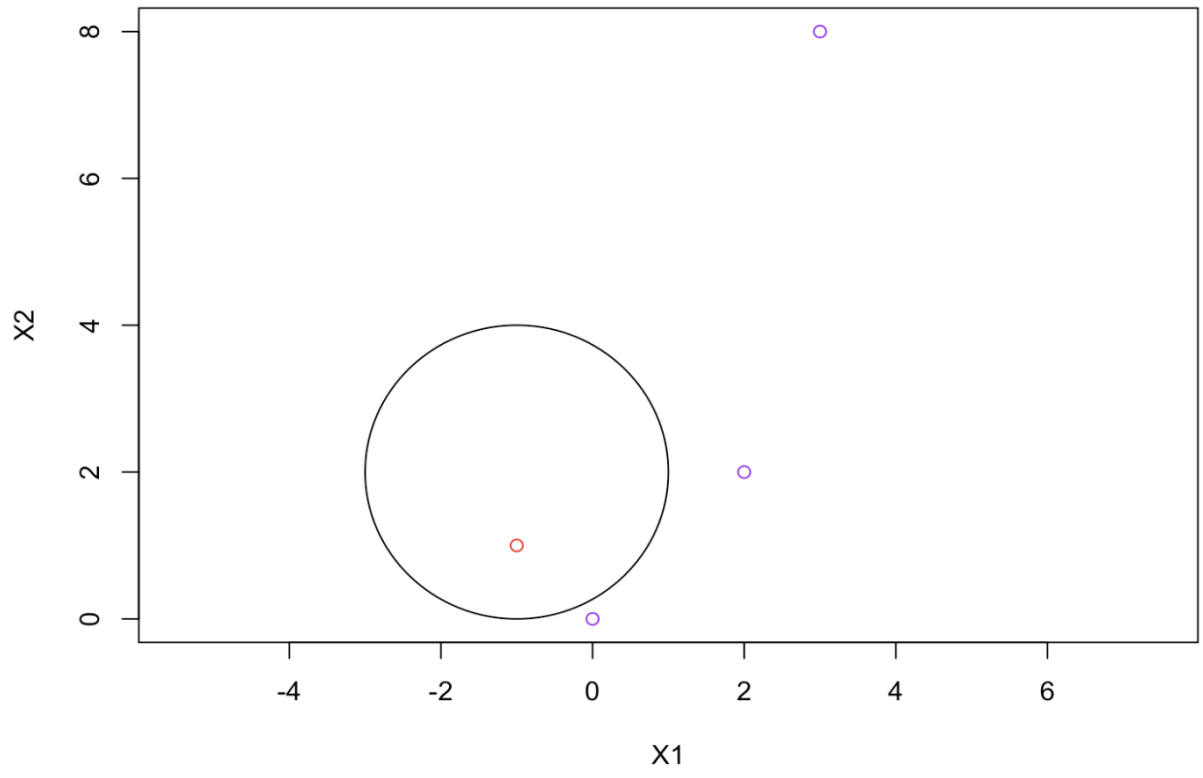
(1+X1)2+(2−X2)2=4 is a circle with radius 2 and center (-1, 2).



b.



c.



d.

The decision boundary is a sum of quadratic terms when expanded.

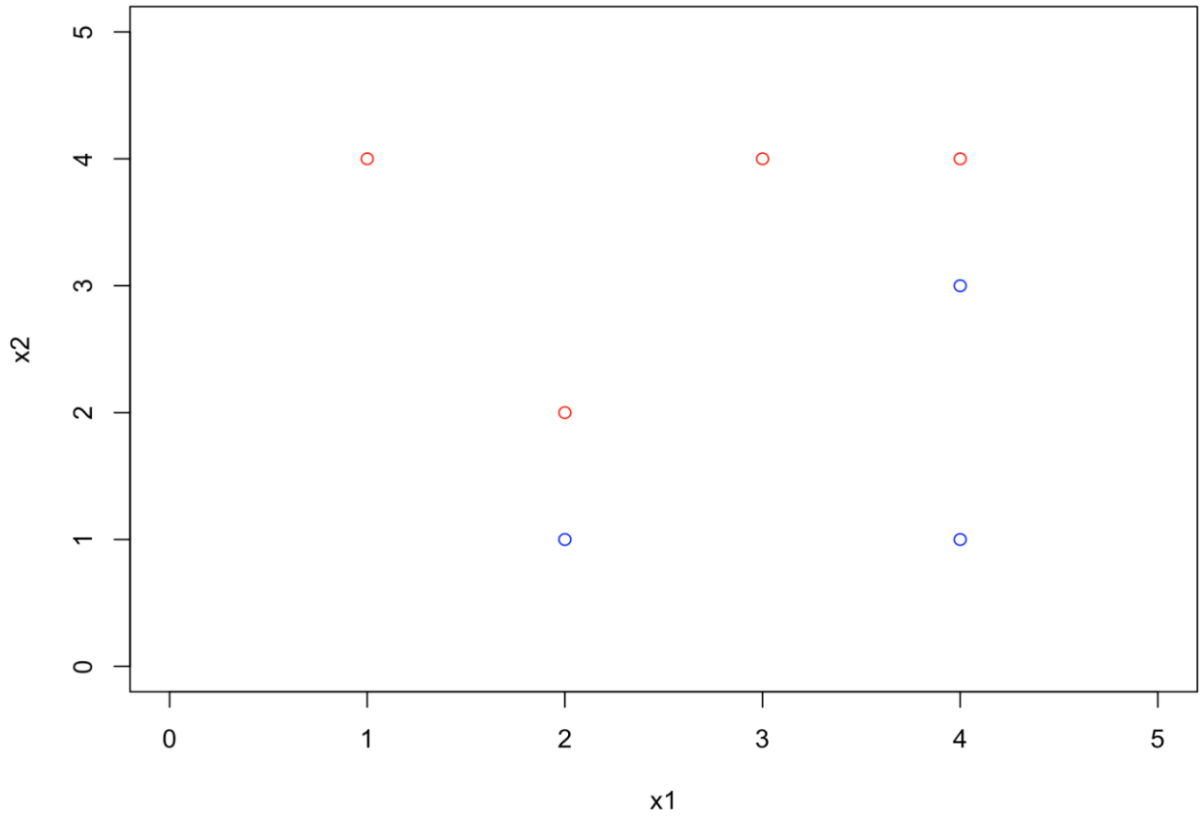
$$(1+X1)2+(2-X2)2>4$$

$$1+2X1+X21+4-4X2+X22>4$$

$$5+2X1-4X2+X21+X22>4$$

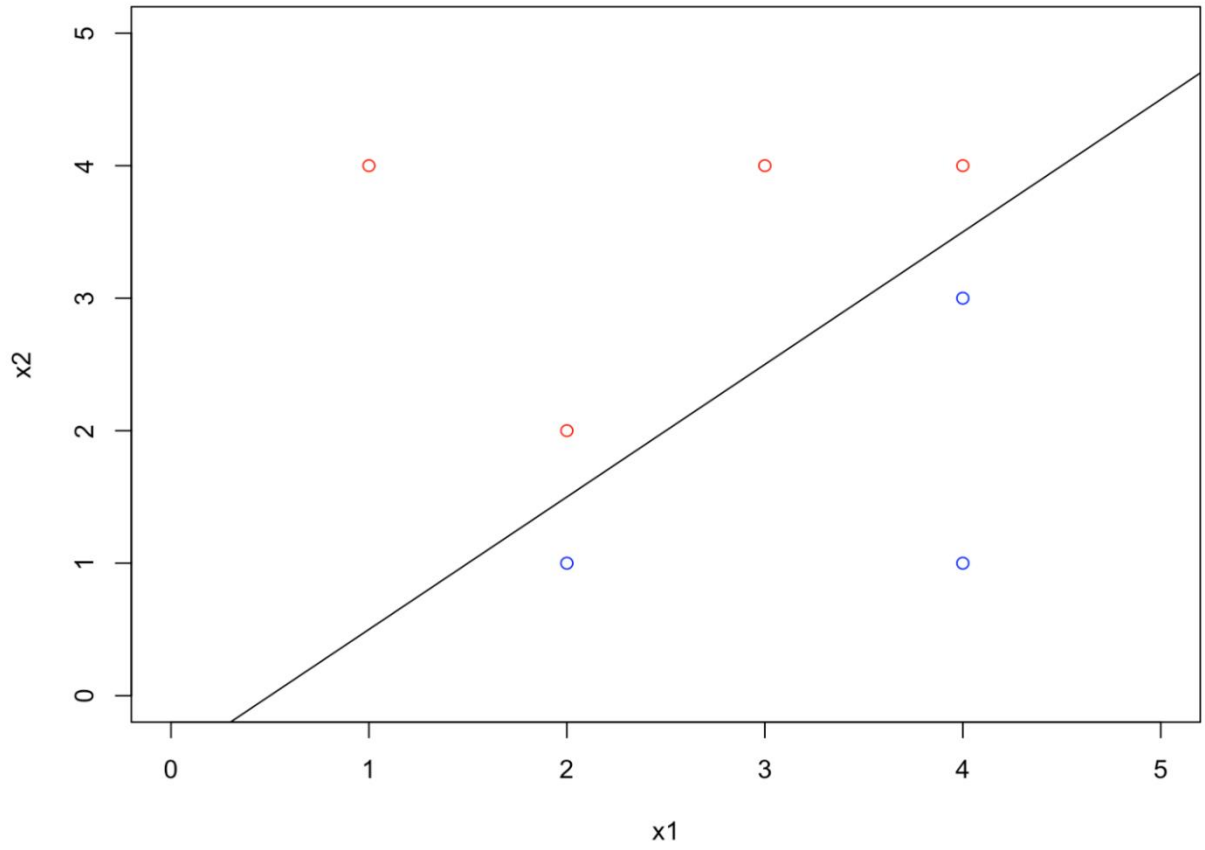
Exercise 3 Answers:

a.



b.

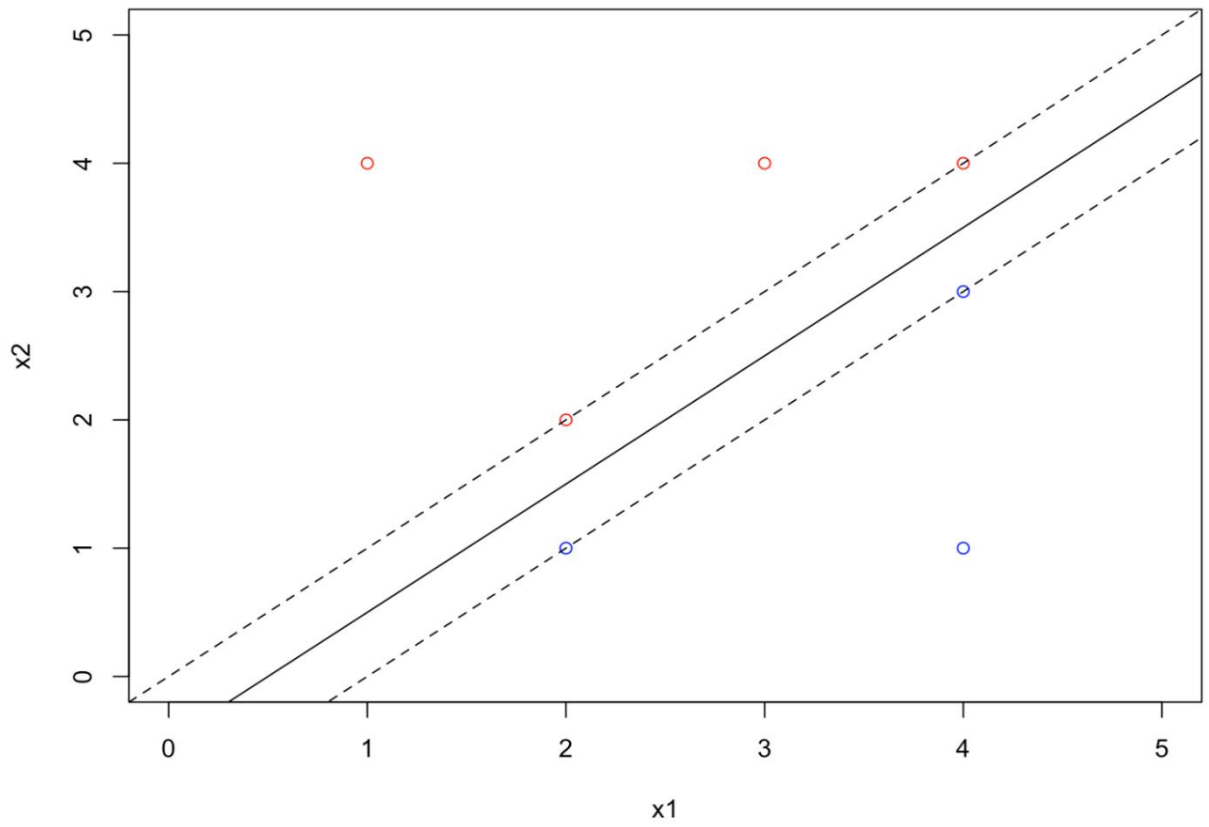
The best separating hyperplane must be between the observations (2,1) and (2,2) and between the observations (4,3) and (4,4), as indicated in the plot (4,4). Therefore, a line with the equation X1-X2-0.5=0 crosses through the points (2, 1.5) and (4, 3.5).



c.

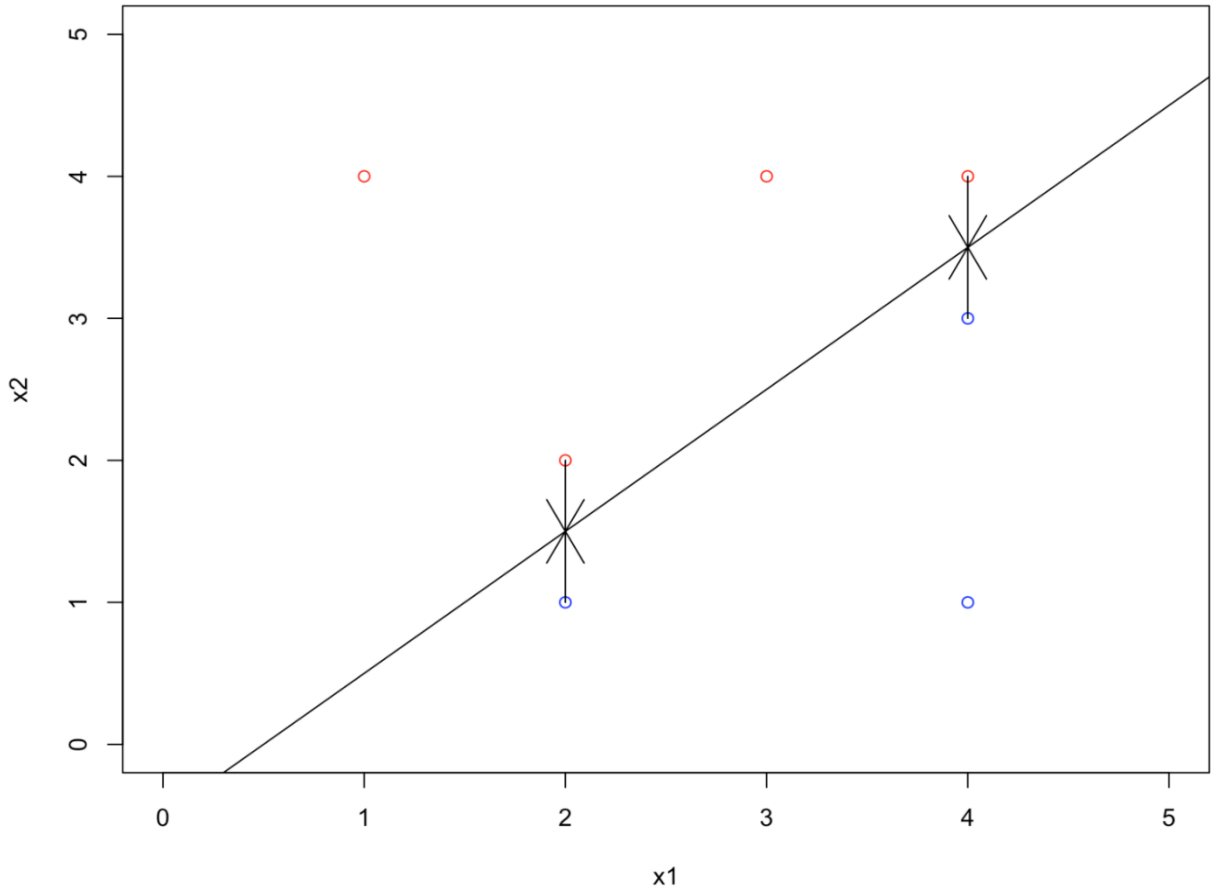
1. The ideal separating hyperplane from the above mentioned question contains the equation X2=X10.5, and close observation reveals that the maximal margin classifier's prediction will be Red when X2>X10.5X2X1+0.5>0 and Blue otherwise.
2. As a result, the maximal margin classifier will have the formula f(X)=X2X1+0.5.
3. Where a new observation X is classified as belonging to the Red class if f(X) > 0 and the Blue class if f(X) 0

d.



e.

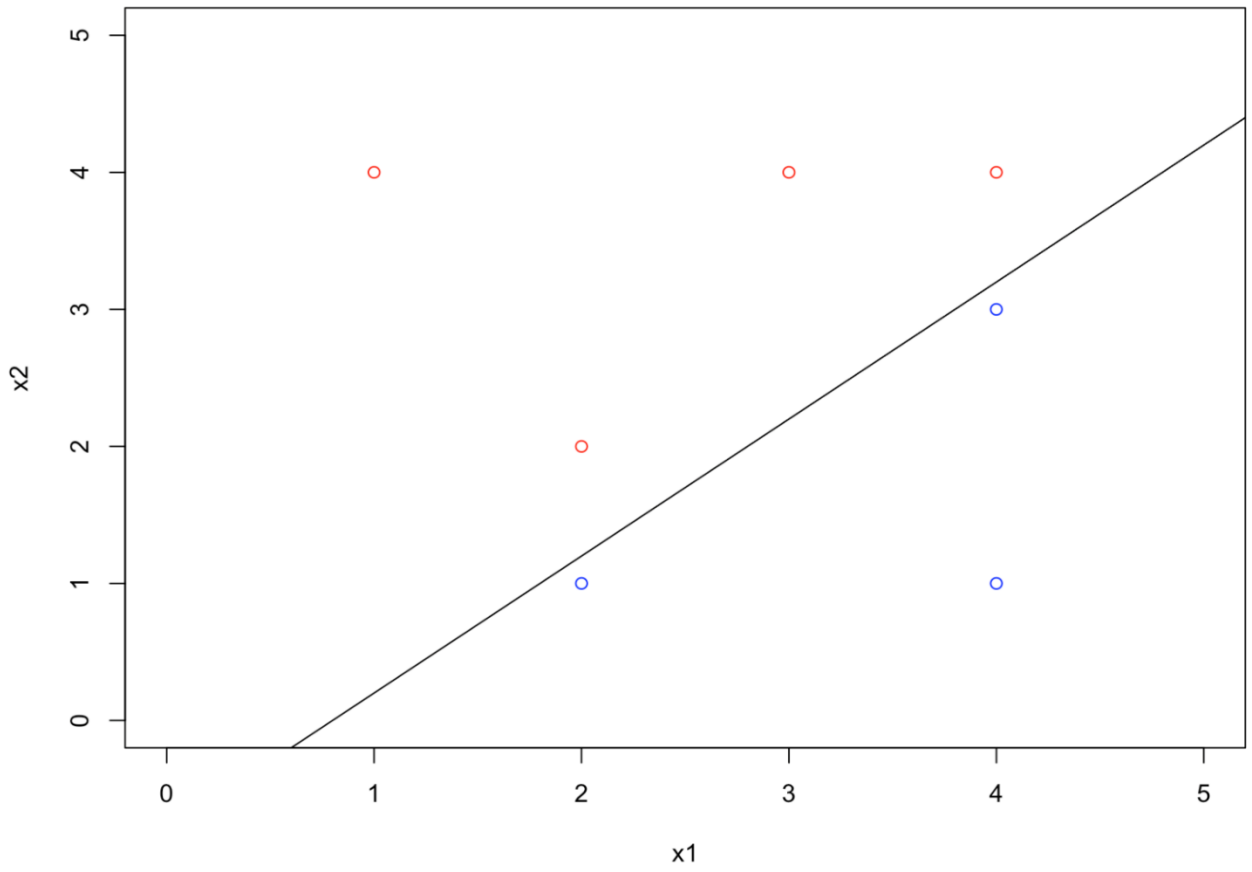
The points (2,1)(2,1), (2,2)(2,2), (4,3)(4,3), and (4,4) make up the support vectors (4,4)



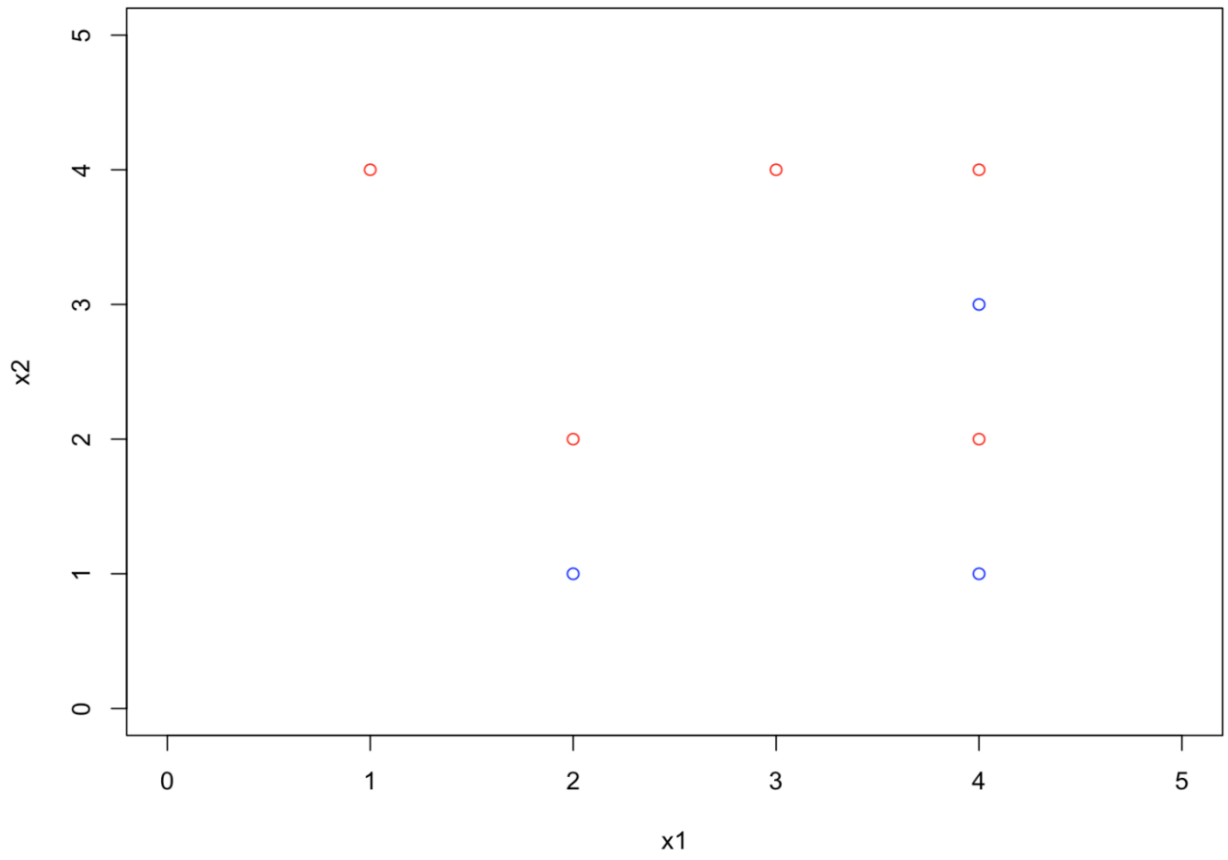
f.

The observation 7 is not a support vector, as we can see from the image above. The fact that X2X1+0.5=0 offers the best separation of observations with the largest margin of M=2/4 will remain unchanged regardless of how slightly this observation is moved in any direction (it will still be the mid-line of the "widest slab" we can fit between the classes). To begin altering the location of the maximal margin hyperplane in this situation, observation 7 would have to move inside the margin.

g.



h.



2. Practicum Problems

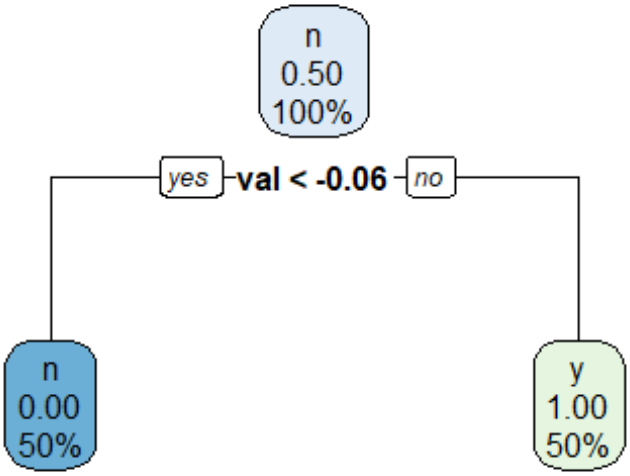
(I’m attaching the pdf markdown files I created from r script.)

rpart prune

Shiva Sankar Modala

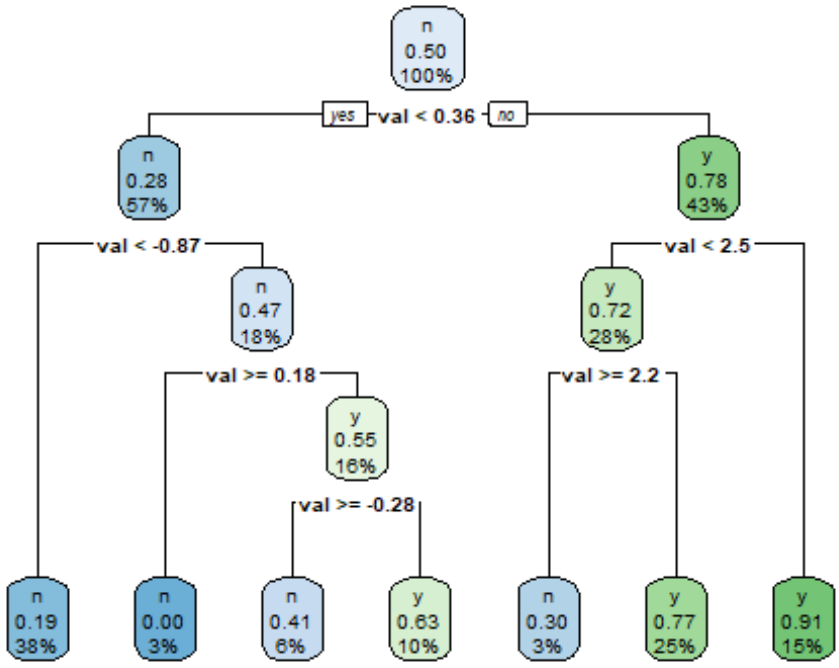
2023-03-28

```
# Loading the necessary packages for solving the question
library(rpart)
# Package to create the binary decision tree
library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 4.2.3

# Initialize a function for the gini index
gini <- function(m) {
  gini.index = 2 * m * (1 - m)
  return (gini.index)
}

# Initializing the function for the entropy value
entropy <- function(n) {
  entropy = (n * log(n) + (1 - n) * log(1 - n))
  return (entropy)
}

# set the seed value to 150
set.seed(150)

# Normal distribution for Mean as 5 and Standard Deviation as 2
x<-rnorm(n=150,mean=5,sd=2)

# Normal distribution for Mean as -5 and Standard Deviation as 2
y<-rnorm(n=150,mean=-5,sd=2)

# Make the dataframe based in the values from the Normal distribution
dt1 <- data.frame(val = x,label=rep("y",150))
dt2 <- data.frame(val = y,label=rep("n",150))

# Combine the two dataframes into one using rbind
dt <- rbind(dt1,dt2)

# Separating the label
dt$label <- as.factor(dt$label)

# Making use of rpart to place the text inside the tree
dtree <- rpart(label~val,dt,method="class")

# Now, plot the rpart of the decision tree
rpart.plot(dtree)
```




```
cat <- "The threshold value for the first split will be -0.06, as observed
from the above. Two leaf nodes and one root node make up the tree. Both
classes can be classified individually by a tree, demonstrating empirical
distribution."

# P is the probability of each node
p=c(.5, 0, 1)

# Calculating the gini vlues and entropy based on the above function
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 -  0.5, 0.0, 0.0
# The entropy values for above tree - 0.6931472, NaN, NaN

# set the seed value to 150
set.seed(150)

# Normal distribution for Mean as 1 and Standard Deviation as 2
x1<-rnorm(n=150,mean=1,sd=2)
```

```
# Normal distribution for Mean as -1 and Standard Deviation as 2
y1<-rnorm(n=150,mean=-1,sd=2)

# Make the dataframe based in the values from the Normal distribution
dt3 <- data.frame(val = x1,label=rep("y",150))
# Make the dataframe based in the values from the Normal distribution
dt4 <- data.frame(val = y1,label=rep("n",150))

# Combining the two dataframes into one using rbind
data <- rbind(dt3,dt4)
data$label <- as.factor(data$label)

# Making use of rpart to place the text inside the tree
dtree1 <- rpart(label~val,data,method="class")
# Now, plot the rpart of the decision tree
rpart.plot(dtree1)
```




```
cat <- "The threshold value for the first split, based on the tree above, is
0.36.
The tree comprises 13 nodes, one of which is the root node.
There are a total of 7 leaf nodes on the tree.
Large tree size indicates the node's presence of more distinct labels, which
led to the tree's size.
Consequently, this tree has higher label overlap in the nodes."

# P1 is the 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)
```

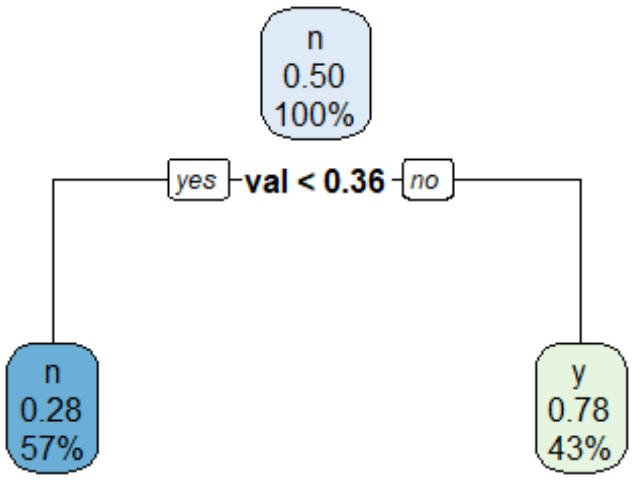
```
# Used the probability to get the gini and entropy
gini_values1=sapply(p1, gini)
gini_values1
```

```
##  [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
```

```
newtree <- prune.rpart(dtree1,cp=0.1)
rpart.plot(newtree)
```




```
cat <- " The threshold value for the first split will be 0.36. The tree has
one root node and 2 leaf nodes."
# P2 is the 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 =  0.5000, 0.3432, 0.4032
# The entropy values for above tree =  -0.6931472, -0.5269080, -0.5929533
```

winequality

Shiva Sankar Modala

2023-03-28

```
# Installing and loading the necessary packages
library(rpart)
#install.packages("rpart.plot")
# Package to create the binary decision tree
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.2.3
```

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

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

```
library(caret)
```

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

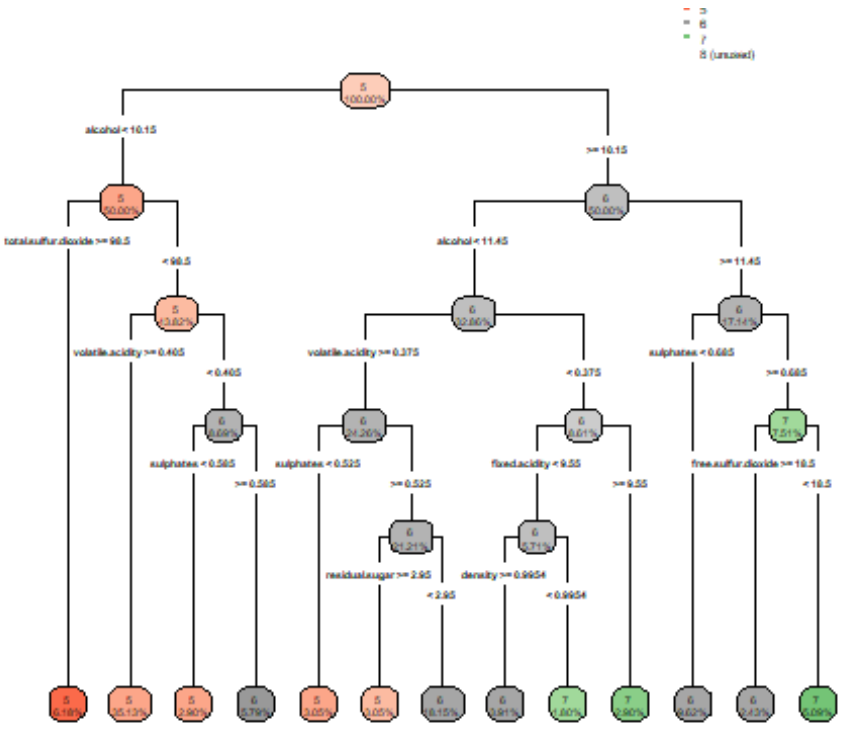
```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

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

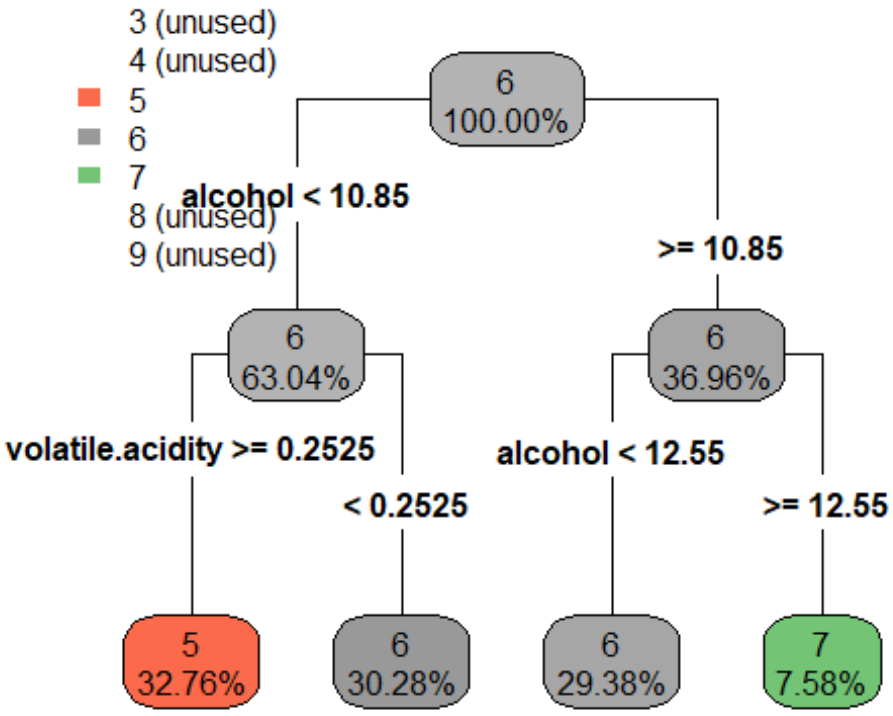
```
# Loading the Wine Quality sample dataset from the UCI Machine Learning
Repository
url_red = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-
quality/winequality-red.csv"
url_white = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-
quality/winequality-white.csv"
# Preparing the table
RedWine <- read.table(file=url_red, header=TRUE,
sep=";",stringsAsFactors=TRUE)
WhiteWine <- read.table(file=url_white, header=TRUE,
sep=";",stringsAsFactors=TRUE)

#redwine
set.seed(1)

# Create an 80/20 test-train split of each wine dataframe
index <- createDataPartition(RedWine$quality,p=0.2,list=FALSE)
```

```
# Separating the data based on the test and train data.
test_red <-RedWine[index,]
train_red <-RedWine[-index,]

train_red$quality <- factor(train_red$quality)
test_red$quality <- factor(test_red$quality)

# Use the rpart package to induce a decision tree of both the red and white
wines
rpart_tree_red = rpart(quality~., data = train_red)
# targeting the quality output variable
rpart_predict_red <- predict(rpart_tree_red, test_red, type = "class")

# Visualizing the tree using the rpart.plot library
rpart.plot(rpart_tree_red, digits = 4, fallen.leaves = TRUE, type = 4, extra
= 100)
```



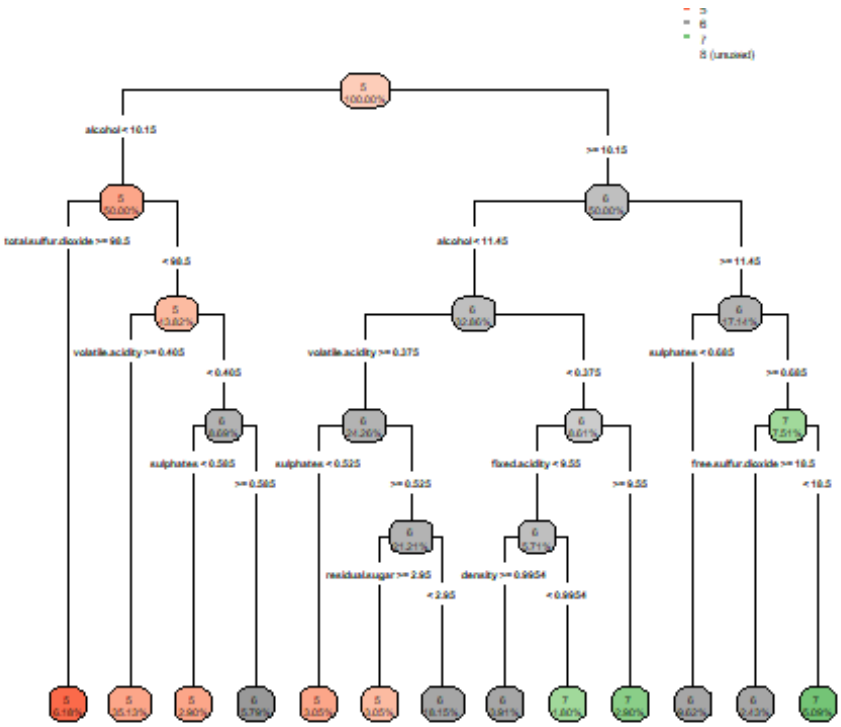
```
table(rpart_predict_red)

## rpart_predict_red
##   3   4   5   6   7   8
##   0   0 158 134  29   0

# Using the caret package confusionMatrix method to determine the decision
tree accuracy on the test set
decision_tree_red_cm<-confusionMatrix(data = rpart_predict_red, reference =
test_red$quality)
```

```
#First split was done at “alcohol < 11” for White wine dataset
#First split was done at “alcohol < 9.5” for Red wine dataset
#Sulphates was taken into consideration in Red Wine Dataset. On the other
hand its absent in White Wine Dataset.
#Total Sulfur Dioxide was taken into consideration in Red Wine Dataset and
its absent in White Wine Dataset.
#Free Sulfur Dioxide was taken into consideration in White Wine Dataset and
its absent in Red Wine Dataset.

#white wine
set.seed(1)
index <- createDataPartition(WhiteWine$quality,p=0.3,list=FALSE)
test_white <-WhiteWine[index,]
train_white <-WhiteWine[-index,]

train_white$quality <- factor(train_white$quality)
test_white$quality <- factor(test_white$quality)
rpart_tree_white = rpart(quality~., data = train_white)
rpart_predict_white <- predict(rpart_tree_white, test_white, type = "class")
rpart.plot(rpart_tree_white, digits = 4, fallen.leaves = TRUE, type = 4,
extra = 100)
```

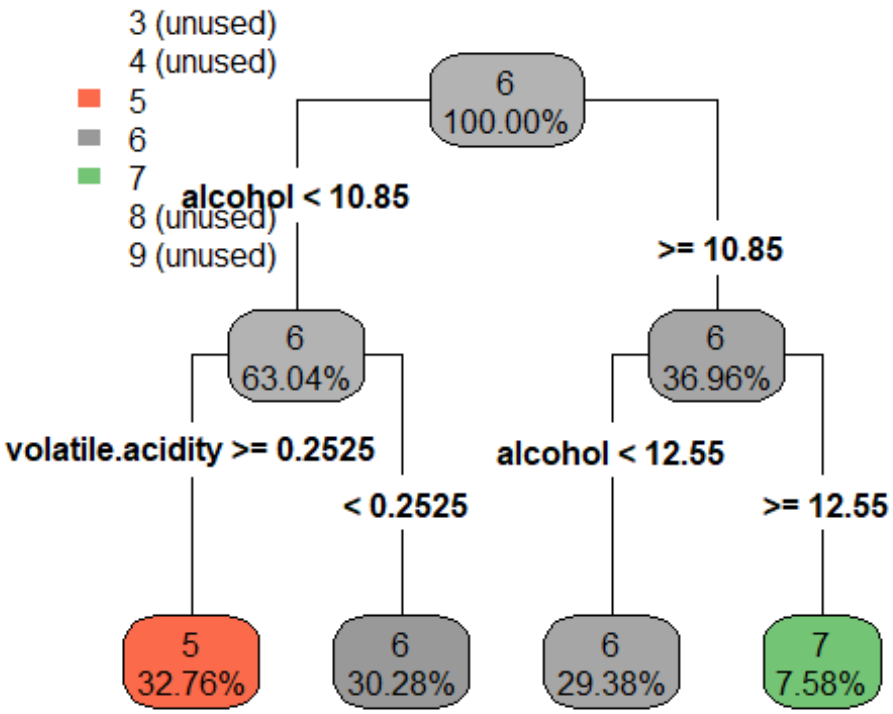



```
table(rpart_predict_white)

## rpart_predict_white
##   3   4   5   6   7   8   9
##   0   0 487 888  95   0   0
```

```
# Using the caret package confusionMatrix method to determine the decision
tree accuracy on the test set
decision_tree_white_cm<-confusionMatrix(rpart_predict_white,
test_white$quality)
# Using the rpart package to induce a decision tree of both the red and white
wines
rpart.plot(rpart_tree_red, digits = 4, fallen.leaves = TRUE, type = 4, extra
= 100)
```



```
# Using the rpart package to induce a decision tree of both the red and white
wines
rpart.plot(rpart_tree_white, digits = 4, fallen.leaves = TRUE, type = 4,
extra = 100)
```




```
varImp(rpart_tree_red)

##                         Overall
## alcohol                99.52523
## chlorides               4.26621
## citric.acid            25.23650
## density                43.89424
## fixed.acidity          46.27422
## free.sulfur.dioxide    12.96934
## pH                     17.60606
## residual.sugar         20.44688
## sulphates             104.79208
## total.sulfur.dioxide   76.46514
## volatile.acidity      103.15831

varImp(rpart_tree_white)

##                         Overall
## alcohol               187.18188
## chlorides              86.82763
## citric.acid            17.63781
## density               100.60980
## free.sulfur.dioxide    26.76760
## total.sulfur.dioxide   42.57525
## volatile.acidity      133.60637
## fixed.acidity           0.00000
## residual.sugar          0.00000
```

```
## pH                    0.00000
## sulphates             0.00000

#randomforest
random_forest_red <- randomForest(quality~., data = train_red)
randomforestred_predict <- predict(object = random_forest_red, newdata =
test_red)
randomforest_red_cm<-confusionMatrix(data = randomforestred_predict,
reference = test_red$quality)

random_forest_white <- randomForest(quality~., data = train_white)

randomforestwhite_predict <- predict(object = random_forest_white, newdata =
test_white)
randomforest_white_cm<-confusionMatrix(data = randomforestwhite_predict,
reference = test_white$quality)

#Comparision
print("Comparision of accuracy between red wine decision tree vs
randomforest: for the Red Wine Decision Tree")

## [1] "Comparision of accuracy between red wine decision tree vs
randomforest: for the Red Wine Decision Tree"

decision_tree_red_cm$overall["Accuracy"]

##  Accuracy
## 0.6105919

print("Red Wine Random Forest")

## [1] "Red Wine Random Forest"

randomforest_red_cm$overall["Accuracy"]

##  Accuracy
## 0.7102804

print("Comparision of accuracy between white wine decision tree vs
randomforest: White Wine Decision Tree")

## [1] "Comparision of accuracy between white wine decision tree vs
randomforest: White Wine Decision Tree"

decision_tree_white_cm$overall["Accuracy"]

##  Accuracy
## 0.4986395

print("White Wine Random Forest")

## [1] "White Wine Random Forest"

randomforest_white_cm$overall["Accuracy"]
```

```
## Accuracy
## 0.670068

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

# 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
```

SMS Spam Collection

Shiva Sankar Modala

2023-03-28

```
library(readxl)

## Warning: package 'readxl' was built under R version 4.2.3

library(tm)

## Warning: package 'tm' was built under R version 4.2.3

## Loading required package: NLP

#install.packages("SnowballC")
library(SnowballC)
library(e1071)

## Warning: package 'e1071' was built under R version 4.2.3

library(caret)

## Warning: package 'caret' was built under R version 4.2.3

## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:NLP':
##
##     annotate

## Loading required package: lattice

# Load the SMS Spam Collection sample dataset
SpamData = read.csv("C:/Users/shiva/OneDrive/Desktop/dpa
Assignments/Assignment
4/smsspamcollection/SMSSpamCollection",sep="\t",header=FALSE,quote="",strings
AsFactors=FALSE)
colnames(SpamData) <- c("Class", "Messages")
smsCorpus <- Corpus(VectorSource(SpamData$Messages))

# Use the tm package to create a Corpus of documents
cleaningSpamData <- function(data){
  data <- tm_map(data, tolower)   # a) Convert lowercase
  data <- tm_map(data, removeWords,stopwords("english"))  # b) Remove
stopwords,
  data <- tm_map(data,stripWhitespace)    #  c) Strip whitespace,
```

```
  data <- tm_map(data, removePunctuation) #  d) Remove punctuation
}
transformedData <- cleaningSpamData(smsCorpus)

## Warning in tm_map.SimpleCorpus(data, tolower): transformation drops
documents

## Warning in tm_map.SimpleCorpus(data, removeWords, stopwords("english")):
## transformation drops documents

## Warning in tm_map.SimpleCorpus(data, stripWhitespace): transformation
drops
## documents

## Warning in tm_map.SimpleCorpus(data, removePunctuation): transformation
drops
## documents

# Building Document Term Matrix
dataDtm <- DocumentTermMatrix(transformedData)

# Use findFreqTerms tocontruct features from words occuring more than 10
times
df_new <- findFreqTerms(dataDtm, lowfreq = 10)
sparse <- removeSparseTerms(dataDtm, 0.99)
sparse

## <<DocumentTermMatrix (documents: 5574, terms: 117)>>
## Non-/sparse entries: 14050/638108
## Sparsity           : 98%
## Maximal term length: 9
## Weighting          : term frequency (tf)

smsSparse <- as.data.frame(data.matrix((sparse)))

smsSparse$class <- SpamData$Class
smsSparse$class <- as.factor(smsSparse$class)

# proceed to split the data into a training and test set - for each create a
DocumentTermMatrix
set.seed(12345)
index <- createDataPartition(smsSparse$class, p = 0.8, list= FALSE)
trainSms <- smsSparse[index,]
testSms <- smsSparse[-index,]

#  convert the DocumentTermMatrix train/test matrices to a Boolean
representation
#  fit a SVM using the e1071 package
modelSvm <- svm(class~., data = trainSms, scale = FALSE, kernel ="linear",
type = "C")
predictTrain <- predict(modelSvm, trainSms)
predictLinear <- predict(modelSvm, testSms)
```

```
accuracyTrain <- confusionMatrix(as.factor(predictTrain), as.factor(trainSms$
                                                                    class))
accuracyTest <-
confusionMatrix(as.factor(predictLinear),as.factor(testSms$class))

# Report your training and test set accuracy.
cat("\n Accuracy Train: ")

##
##  Accuracy Train:

accuracyTrain

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  ham spam
##       ham  3835  126
##       spam   27  472
##
##                Accuracy : 0.9657
##                  95% CI : (0.9599, 0.9708)
##     No Information Rate : 0.8659
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8412
##
##  Mcnemar's Test P-Value : 2.322e-15
##
##             Sensitivity : 0.9930
##             Specificity : 0.7893
##          Pos Pred Value : 0.9682
##          Neg Pred Value : 0.9459
##              Prevalence : 0.8659
##          Detection Rate : 0.8599
##    Detection Prevalence : 0.8881
##       Balanced Accuracy : 0.8912
##
##        'Positive' Class : ham
##

cat("\n Accuracy Test: ")

##
##  Accuracy Test:

accuracyTest

## Confusion Matrix and Statistics
##
##           Reference
## Prediction ham spam
```

```
##       ham  954   39
##       spam  11  110
##
##                Accuracy : 0.9551
##                  95% CI : (0.9413, 0.9665)
##     No Information Rate : 0.8662
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7896
##
##  Mcnemar's Test P-Value : 0.0001343
##
##             Sensitivity : 0.9886
##             Specificity : 0.7383
##          Pos Pred Value : 0.9607
##          Neg Pred Value : 0.9091
##              Prevalence : 0.8662
##          Detection Rate : 0.8564
##    Detection Prevalence : 0.8914
##       Balanced Accuracy : 0.8634
##
##        'Positive' Class : ham
##
```