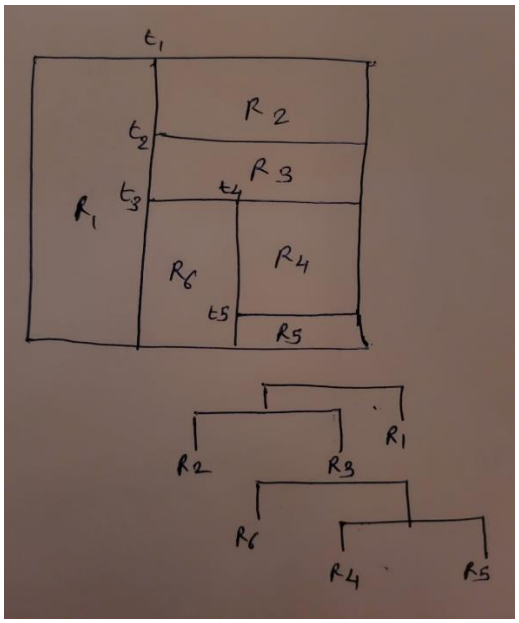
Homework 4

Vaishnavi Mule A20516627

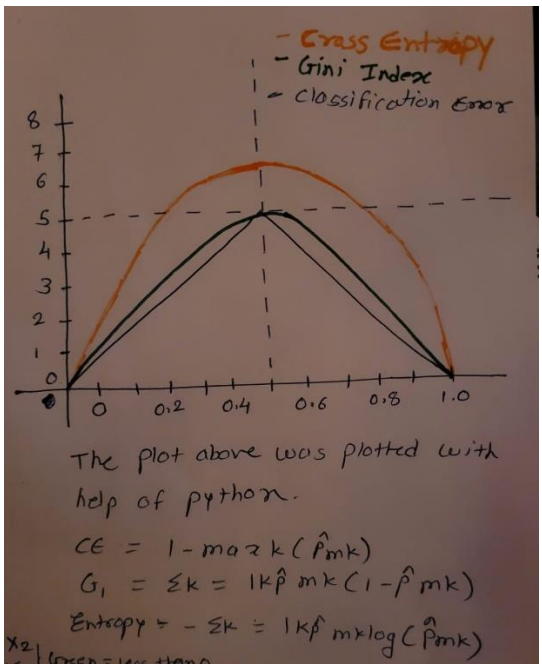
1 Recitation Exercises

Chapter 8

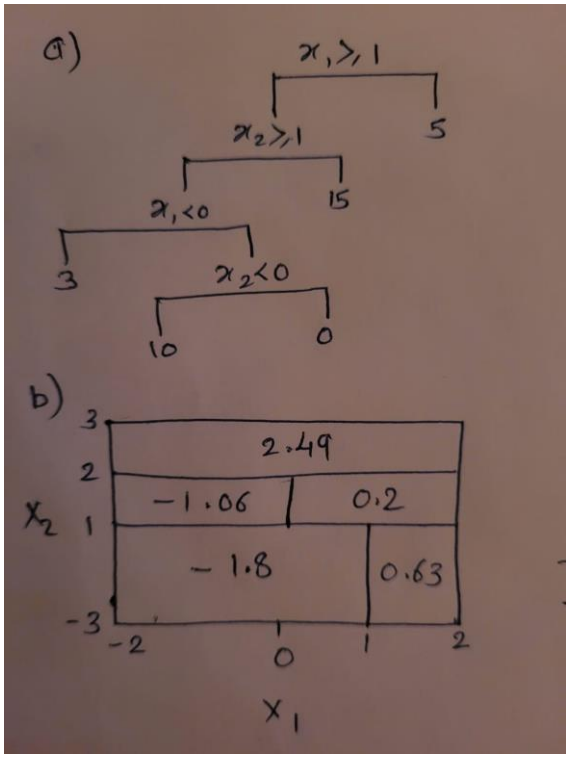
Exercises: 1



Exercises: 3



Exercises: 4



Exercises: 5

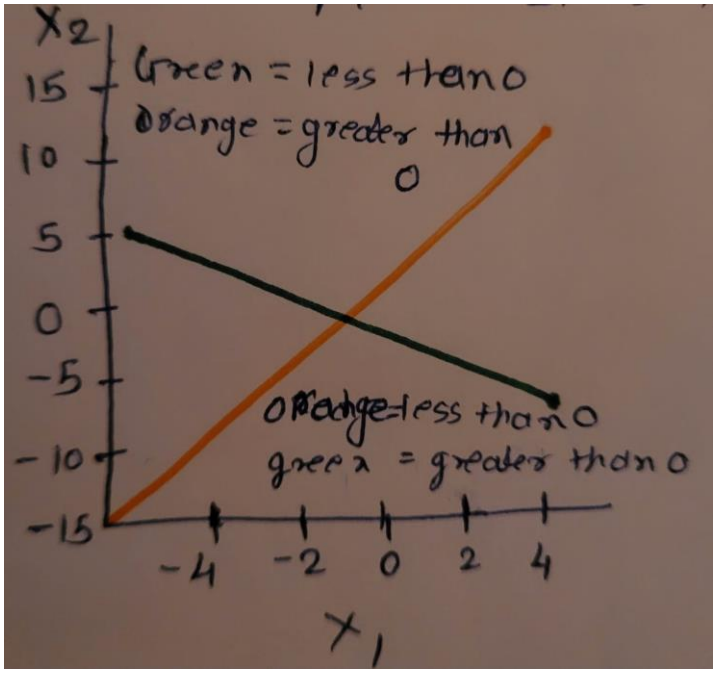
There are two approaches :

1. majority Vote approach: with this approach, we would classify X as red since 6 out of 10 belong to class red

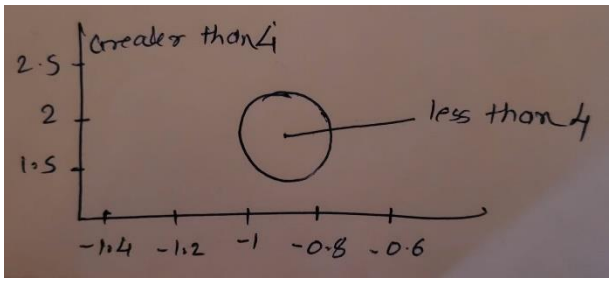
2. Average probability approach: with this, the average of probabilities is 0.45, hence, we classify X as green

1.1 Chapter 9

Exercises: 1

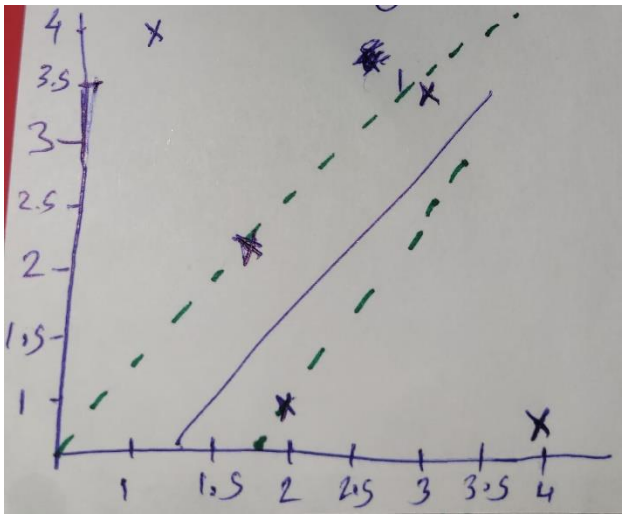


Exercises: 2



a) Sketched above.
b) Marked in red in the plot above. Less than or equal to is within and on the hyperplane boundaries.
c) (0,0) = Blue. (-1,1) = Red. (2,2) = Blue. (3,8) = Blue. (Red-inner circle area, Blue – outer circle area)
d) On expanding the equation of circle, we get X12 + X22 + 2X1 – 4X2 – 1 = 0 which is of the form X1, X12 , X2 and X22 .

Exercises: 3



a) Plotted above
b) Plotted above. X1 – X2 – 0.5 = 0
c) Classifies to green if X1 – X2 – 0,5 < 0, Else Blue
d) Plotted Above (green dotted line)
e) (2,1), (2,2), (4,3) and (4,4)
f) Since (4,1) is not a support vector point, the maximal margin hyperplane won't change.
g) X1 – X2 – 0.4 = 0

2 Practicum Problems

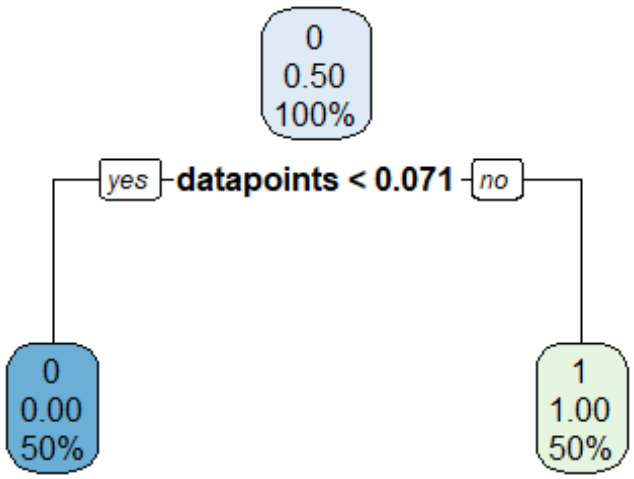
2.1 Problem 1

Vaishnavi

2023-11-12

Setting up the dataframe

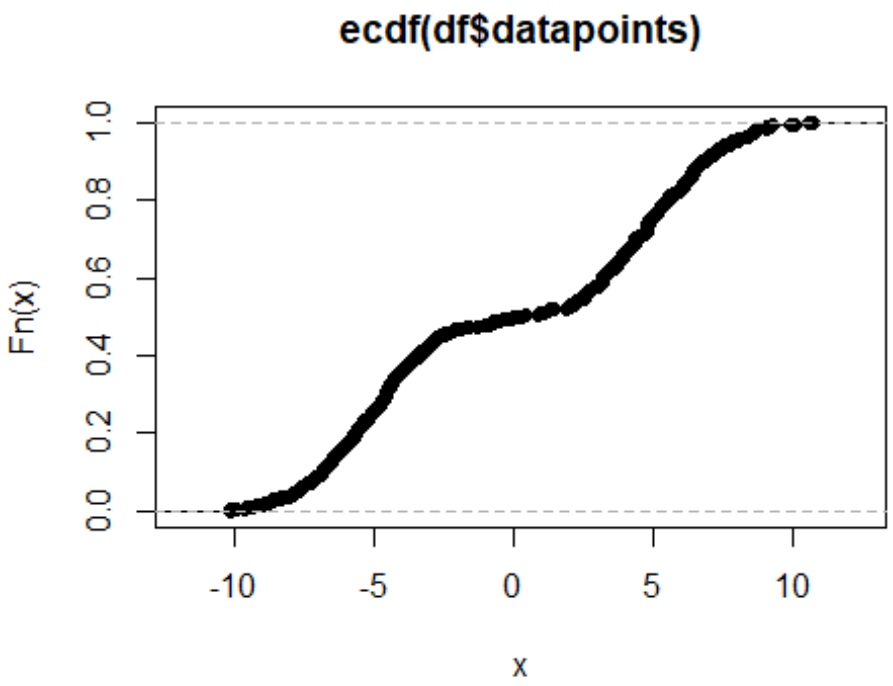
```
df1 = data.frame(datapoints = rnorm(300, 5,2), class=rep(1,300))
df2 = data.frame(datapoints = rnorm(300, -5,2), class=rep(0,300))
df = rbind(df1,df2)

library(rpart)

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

library(rpart.plot)

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

df$class = as.factor(df$class)
bdt_df = rpart(class ~ ., data=df)
rpart.plot(bdt_df)
```



Threshold Value in first split is 0.46

Empirical Distribution of the feature is as below.

```
plot(ecdf(df$datapoints))
```



Root node = 1 Leaf nodes = 2

```
tail(bdt_df$cptable[,'nsplit'],1)

## 2
## 1
```

Finding entropy and Gini Values

```
calc_gini_index = function(prob)
{
  gi = 2 * (prob) * (1  - prob)
  return (gi)
}


entropy <- function(prob)
{

 entropy_val = (prob * log(prob) + (1 - prob) * log(1 - prob))

 return (entropy_val)

}

prob = c(0.5, 0, 1)
gi = sapply(prob, calc_gini_index)
gi
```

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

Entropy is 0.5

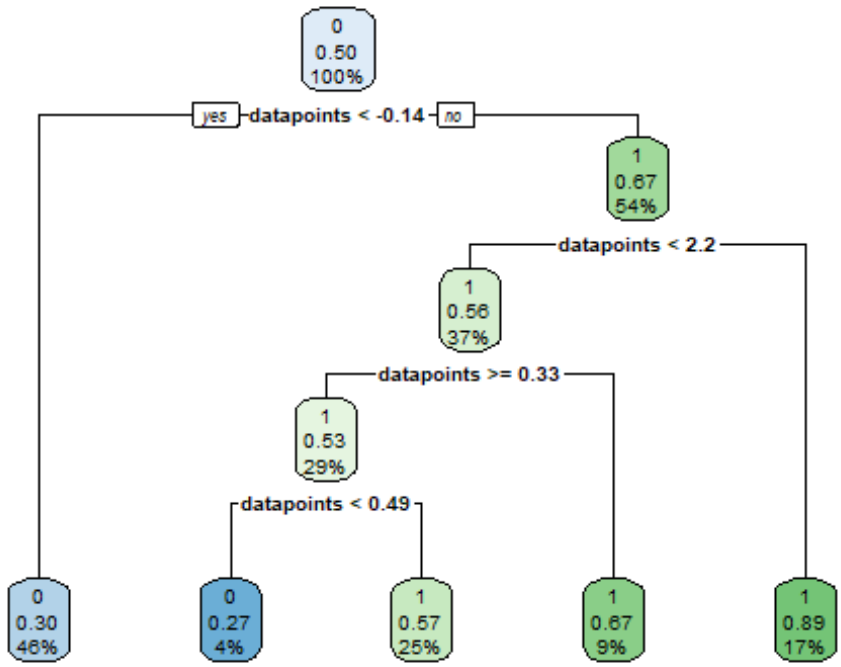
```
entropy_val = sapply(prob, entropy)
entropy_val
```

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

Gini is -0.6931472

Repeating for N.D of (1,2) and (-1,2)

```
temp1 = data.frame(datapoints= rnorm(300, 1,2), class=rep(1,300))
temp2 = data.frame(datapoints = rnorm(300, -1,2), class=rep(0,300))
df2 = rbind(temp1,temp2)

library(rpart)
library(rpart.plot)
df2$class = as.factor(df2$class)
bdt_df2 = rpart(class ~ ., data=df2)
rpart.plot(bdt_df2)
```



Two leafs and 1 Root Node.

```
tail(bdt_df2$cptable[,'nsplit'],1)
```

```
## 3
## 4
```

```
prob2 = c(0.5, 0.22, 0.64, 0.57, 0.51,0.14,0.58,0.51,0.33,0.64,0.68,0.61,0
.29,0.63,0.87)
gi = sapply(prob2, calc_gini_index)
gi
```

```
##  [1] 0.5000 0.3432 0.4608 0.4902 0.4998 0.2408 0.4872 0.4998 0.4422 0.4
608
## [11] 0.4352 0.4758 0.4118 0.4662 0.2262
```

```
entropy_val = sapply(prob2, entropy)
entropy_val
```

```
##  [1] -0.6931472 -0.5269080 -0.6534182 -0.6833149 -0.6929472 -0.4049635
##  [7] -0.6802920 -0.6929472 -0.6341786 -0.6534182 -0.6268695 -0.6687481
## [13] -0.6021517 -0.6589557 -0.3863867
```

Pruning

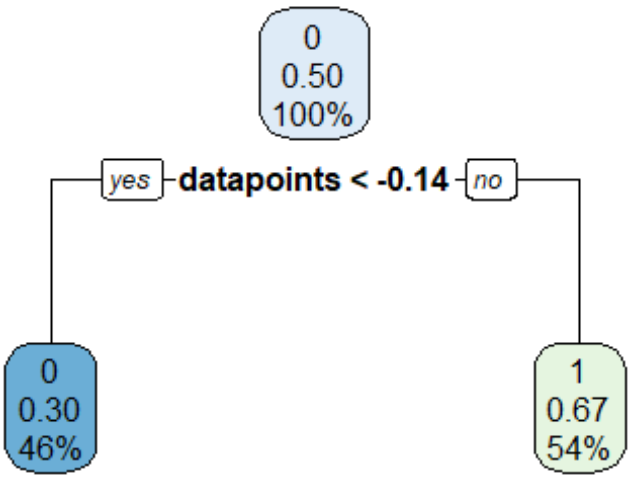
```
bdt_df2$cptable[which.min(bdt_df2$cptable[,"xerror"]),"CP"]
```

```
## [1] 0.01
```

```
prunetree = prune(bdt_df2, cp=0.1)
prunetree
```

```
## n= 600
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 600 300 0 (0.5000000 0.5000000)
##   2) datapoints< -0.1370024 273  82 0 (0.6996337 0.3003663) *
##   3) datapoints>=-0.1370024 327 109 1 (0.3333333 0.6666667) *
```

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



```
prob3 = c(0.5, 0.22, 0.64)
gi = sapply(prob3, calc_gini_index)
gi
```

```
## [1] 0.5000 0.3432 0.4608
```

```
entropy_val = sapply(prob3, entropy)
entropy_val
```

```
## [1] -0.6931472 -0.5269080 -0.6534182
```

The tree seems to be the same even after pruning.

2.2 Problem 2

Vaishnavi

2023-11-12

#Loading Dataset

```
r_df = read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases
/wine-quality/winequality-red.csv", header=TRUE, sep=";")
w_df = read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases
/wine-quality/winequality-red.csv", header=TRUE, sep=";")
```

Create 80-20 Split

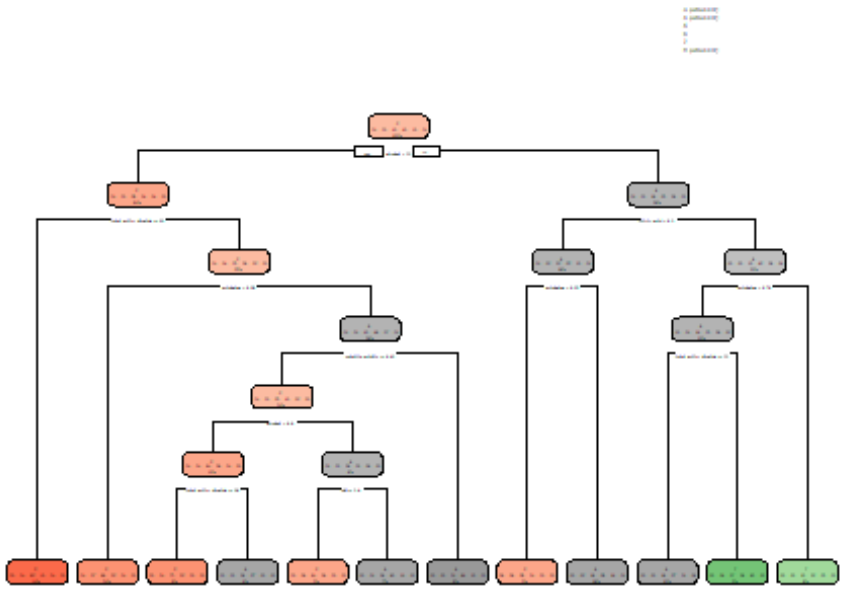
```
library(ggplot2)
library(lattice)
```

```
## Warning: package 'lattice' was built under R version 4.2.3
```

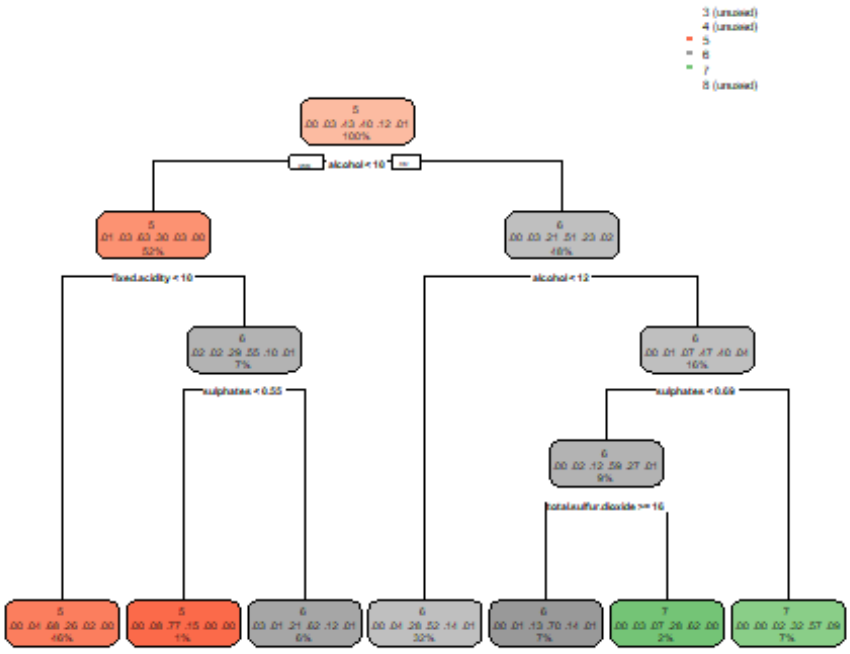
```
library(caret)

r_index = createDataPartition(r_df$quality, p =0.80, list=FALSE)
w_index = createDataPartition(r_df$quality, p =0.80, list=FALSE)

rw_train = r_df[r_index,]
rw_test = r_df[-r_index,]

ww_train = w_df[w_index,]
ww_test = w_df[-w_index,]
```

Induce Decision Tree for Both Wines while targetting Quality as the output variable.

```
library(rpart)

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

rw_train$quality = factor(rw_train$quality)
rw_test$quality = factor(rw_test$quality)

ww_train$quality = factor(ww_train$quality)
ww_test$quality = factor(ww_test$quality)

redwine_decisiontree = rpart(quality ~ . , data=rw_train, method="class")
whitewine_decisiontree =  rpart(quality ~ . , data=ww_train , method="clas
s")

library(rpart.plot)

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

rpart.plot(redwine_decisiontree)
```



```
rpart.plot(whitewine_decisiontree)
```



Confusion Matrix and Accuracies

```
redwine_predict = predict(redwine_decisiontree, newdata = rw_test, type='class')
whitewine_predict = predict(whitewine_decisiontree, newdata = ww_test, type='class')
redwine_dt_cm = confusionMatrix(table(redwine_predict,rw_test$quality))
whitewine_dt_cm = confusionMatrix(table(ww_test$quality, whitewine_predict))

redwine_dt_cm
```

```
## Confusion Matrix and Statistics
##
##
## redwine_predict   3   4   5   6   7   8
##               3   0   0   0   0   0   0
##               4   0   0   0   0   0   0
##               5   1   8 102  36   2   0
##               6   1   5  27  69  19   1
##               7   0   0   4  22  20   1
##               8   0   0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.6006
##                  95% CI : (0.5445, 0.6549)
##     No Information Rate : 0.4182
##     P-Value [Acc > NIR] : 4.584e-11
##
##                   Kappa : 0.3678
##
```

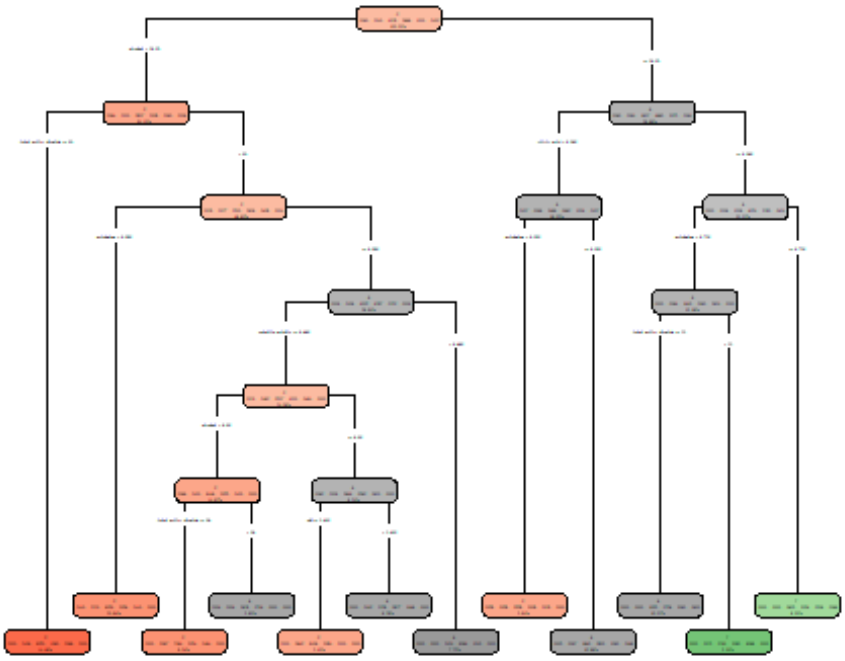
```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class
: 8
## Sensitivity          0.000000  0.00000   0.7669   0.5433  0.48780 0.000
000
## Specificity          1.000000  1.00000   0.7459   0.7225  0.90253 1.000
000
## Pos Pred Value             NaN      NaN   0.6846   0.5656  0.42553
NaN
## Neg Pred Value       0.993711  0.95912   0.8166   0.7041  0.92251 0.993
711
## Prevalence           0.006289  0.04088   0.4182   0.3994  0.12893 0.006
289
## Detection Rate       0.000000  0.00000   0.3208   0.2170  0.06289 0.000
000
## Detection Prevalence 0.000000  0.00000   0.4686   0.3836  0.14780 0.000
000
## Balanced Accuracy    0.500000  0.50000   0.7564   0.6329  0.69517 0.500
000
```

```
whitewine_dt_cm
```

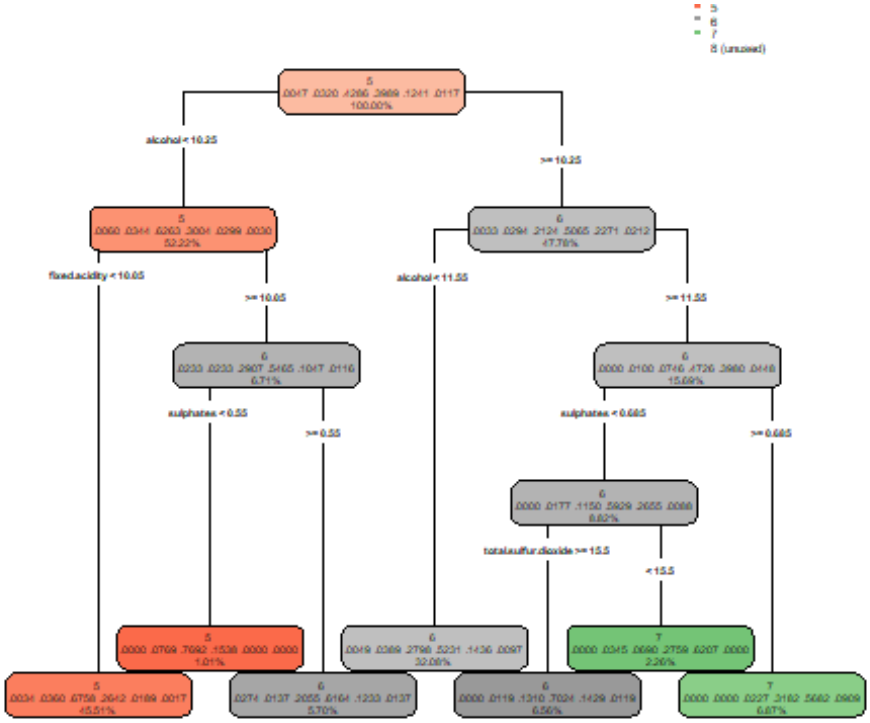
```
## Confusion Matrix and Statistics
##
##    whitewine_predict
##      3  4  5  6  7  8
##   3  0  0  2  2  0  0
##   4  0  0  4  8  0  0
##   5  0  0 93 37  2  0
##   6  0  0 52 63 12  0
##   7  0  0  4 21 15  0
##   8  0  0  0  0  3  0
##
## Overall Statistics
##
##                Accuracy : 0.5377
##                  95% CI : (0.4812, 0.5935)
##     No Information Rate : 0.4874
##     P-Value [Acc > NIR] : 0.04103
##
##                   Kappa : 0.255
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class
: 8
## Sensitivity               NA       NA   0.6000   0.4809  0.46875
NA
## Specificity           0.98742  0.96226   0.7607   0.6578  0.91259 0.990
566
## Pos Pred Value            NA       NA   0.7045   0.4961  0.37500
```

```
NA
## Neg Pred Value              NA       NA    0.6667   0.6440  0.93885
NA
## Prevalence            0.00000  0.00000    0.4874   0.4119  0.10063 0.000
000
## Detection Rate        0.00000  0.00000    0.2925   0.1981  0.04717 0.000
000
## Detection Prevalence  0.01258  0.03774    0.4151   0.3994  0.12579 0.009
434
## Balanced Accuracy           NA       NA    0.6804   0.5693  0.69067
NA
```

```
rpart.plot(redwine_decisiontree, digits = 4, fallen.leaves = TRUE, type =
4)
```



```
rpart.plot(whitewine_decisiontree, digits = 4, fallen.leaves = TRUE, type
= 4)
```




```
varImp(whitewine_decisiontree)
```

```
##                         Overall
## alcohol               104.329757
## citric.acid            21.592608
## density                27.960380
## fixed.acidity          31.531286
## free.sulfur.dioxide     6.322233
## pH                      5.304934
## residual.sugar          7.263290
## sulphates              84.151122
## total.sulfur.dioxide   61.010146
## volatile.acidity       66.719407
## chlorides               0.000000
```

```
varImp(redwine_decisiontree)
```

```
##                         Overall
## alcohol               136.680649
## chlorides               5.910383
## citric.acid            22.516824
## density                52.562204
## fixed.acidity          30.094305
## free.sulfur.dioxide    17.156556
## pH                     15.291086
## residual.sugar          5.937411
## sulphates              97.941859
## total.sulfur.dioxide   75.258402
## volatile.acidity       97.711513
```

Repeat using Random Forest and comparing results

```
library(randomForest)

## randomForest 4.7-1.1

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

library(ggplot2)
redwine_randomforest = randomForest(quality ~ . , data=rw_train)
whitewine_randomforest = randomForest(quality ~ . , data=ww_train)

redwine_randomforest_predict = predict(redwine_randomforest, newdata = rw_
test)
whitewine_randomforest_predict = predict(redwine_randomforest, newdata = w
w_test)

library(caret)
redwine_rf_cm = confusionMatrix(table(redwine_randomforest_predict, rw_tes
t$quality))
redwine_rf_cm

## Confusion Matrix and Statistics
##
##
## redwine_randomforest_predict   3   4   5   6   7   8
##                            3   0   0   0   0   0   0
##                            4   1   0   0   0   0   0
##                            5   1   8 109  24   0   0
##                            6   0   5  21  97  23   0
##                            7   0   0   3   6  18   2
##                            8   0   0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.7044
##                  95% CI : (0.6509, 0.754)
##     No Information Rate : 0.4182
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5217
##
##  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   0.8195   0.7638  0.43902 0.000
000
## Specificity          1.000000 0.996721   0.8216   0.7435  0.96029 1.000
```

```
000
## Pos Pred Value            NaN 0.000000   0.7676   0.6644  0.62069
NaN
## Neg Pred Value       0.993711 0.958991   0.8636   0.8256  0.92042 0.993
711
## Prevalence           0.006289 0.040881   0.4182   0.3994  0.12893 0.006
289
## Detection Rate       0.000000 0.000000   0.3428   0.3050  0.05660 0.000
000
## Detection Prevalence 0.000000 0.003145   0.4465   0.4591  0.09119 0.000
000
## Balanced Accuracy    0.500000 0.498361   0.8206   0.7536  0.69966 0.500
000
```

```
whitewine_rf_cm = confusionMatrix(table(ww_test$quality, whitewine_randomf
orest_predict ))
whitewine_rf_cm
```

```
## Confusion Matrix and Statistics
##
##    whitewine_randomforest_predict
##       3   4   5   6   7   8
##   3   2   1   1   0   0   0
##   4   0   8   3   1   0   0
##   5   0   0 127   5   0   0
##   6   0   0   8 118   1   0
##   7   0   0   0   4  36   0
##   8   0   0   0   0   0   3
##
## Overall Statistics
##
##                Accuracy : 0.9245
##                  95% CI : (0.8898, 0.951)
##     No Information Rate : 0.4371
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8824
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class
: 8
## Sensitivity          1.000000  0.88889   0.9137   0.9219   0.9730 1.000
000
## Specificity          0.993671  0.98706   0.9721   0.9526   0.9858 1.000
000
## Pos Pred Value       0.500000  0.66667   0.9621   0.9291   0.9000 1.000
000
## Neg Pred Value       1.000000  0.99673   0.9355   0.9476   0.9964 1.000
000
## Prevalence           0.006289  0.02830   0.4371   0.4025   0.1164 0.009
434
## Detection Rate       0.006289  0.02516   0.3994   0.3711   0.1132 0.009
434
```

```
## Detection Prevalence 0.012579  0.03774   0.4151   0.3994   0.1258 0.009
434
## Balanced Accuracy    0.996835  0.93797   0.9429   0.9373   0.9794 1.000
000
```

```
cat("Random Forest- Red Wine Accuracy : ",redwine_rf_cm$overall['Accuracy'
])
```

```
## Random Forest- Red Wine Accuracy :  0.7044025
```

```
cat("\nDecision Tree- Red Wine Accuracy : ", redwine_dt_cm$overall['Accura
cy'])
```

```
##
## Decision Tree- Red Wine Accuracy :  0.6006289
```

```
cat("Random Forest- White Wine Accuracy : ",whitewine_rf_cm$overall['Accur
acy'])
```

```
## Random Forest- White Wine Accuracy :  0.9245283
```

```
cat("\nDecision Tree- White Wine Accuracy : ", whitewine_dt_cm$overall['Ac
curacy'])
```

```
##
## Decision Tree- White Wine Accuracy :  0.5377358
```

2.3 Problem 3

Vaishnavi

2023-11-12

Loading SMS Spam Dataset

```
ss_df = read.csv("SMSSpamCollection", sep="\t", stringsAsFactors = FALSE,h
eader=FALSE, quote="")
colnames(ss_df) = c('type', 'message')
```

Creating Corpus

```
library(tm)

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

## Loading required package: NLP

library(ggplot2)

##
## Attaching package: 'ggplot2'

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

library(caret)

## Loading required package: lattice

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

ss_df$type = factor(ss_df$type)
ss_corpus = VCorpus(VectorSource(ss_df$message))
ss_corpus

## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 5574
```

Applying Transformations on the Corpus

```
ss_corpus = tm_map(ss_corpus,content_transformer(tolower))

ss_corpus = tm_map(ss_corpus, removeWords ,stopwords('english'))

ss_corpus = tm_map(ss_corpus, stripWhitespace)

ss_corpus = tm_map(ss_corpus, removePunctuation)

as.character(ss_corpus[1:3])

## [1] "list(list(content = \"go jurong point crazy available bugis n grea
t world la e buffet cine got amore wat\", meta = list(author = character(0
```

```
), datetimestamp = list(sec = 22.5528988838196, min = 25, hour = 23, mday
= 9, mon = 3, year = 123, wday = 0, yday = 98, isdst = 0), description = c
haracter(0), heading = character(0), id = \"1\", language = \"en\", origin
= character(0))), list(content = \"ok lar joking wif u oni\", meta = list(
author = character(0), datetimestamp = list(sec = 22.5532701015472, min =
25, \n    hour = 23, mday = 9, mon = 3, year = 123, wday = 0, yday = 98, i
sdst = 0), description = character(0), heading = character(0), id = \"2\",
language = \"en\", origin = character(0))), list(content = \"free entry 2
wkly comp win fa cup final tkts 21st may 2005 text fa 87121 receive entry
questionstd txt ratetcs apply 08452810075over18s\", meta = list(author = c
haracter(0), datetimestamp = list(sec = 22.5533020496368, min = 25, hour =
23, mday = 9, mon = 3, year = 123, wday = 0, yday = 98, isdst = 0), \n
description = character(0), heading = character(0), id = \"3\", language =
\"en\", origin = character(0))))"
## [2] "list()
"
## [3] "list()"

dtm_ss = DocumentTermMatrix(ss_corpus)
dtm_ss

## <<DocumentTermMatrix (documents: 5574, terms: 8879)>>
## Non-/sparse entries: 44937/49446609
## Sparsity           : 100%
## Maximal term length: 51
## Weighting          : term frequency (tf)

ss_train = dtm_ss[0:4000,]
ss_test = dtm_ss[4000:5574,]

smspam_train_labels = ss_df[0:4000,]$type
smspam_test_labels = ss_df[4000:5574,]$type
```

#Construct Features from Words that have more than 10 occurances and Split the Data

```
smsspam_freqterm = findFreqTerms(ss_train,10)
```

#Boolean Representation

```
freq_terms_sms_train = ss_train[,smsspam_freqterm]
freq_terms_sms_test = ss_test[,smsspam_freqterm]

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

ss_train = apply(freq_terms_sms_train, MARGIN = 2, convert_binary)
ss_test =  apply(freq_terms_sms_test, MARGIN = 2, convert_binary)
```

Fit to SVM Model

```
library(e1071)
smsspam_svm_model = svm(smspam_train_labels ~. , data=ss_train)
print(smsspam_svm_model)

##
## Call:
```

```
## svm(formula = smspam_train_labels ~ ., data = ss_train)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  1476

smsspam_svm_train_pred = predict(smsspam_svm_model, ss_train)
smsspam_svm_test_pred = predict(smsspam_svm_model, ss_test)

ss_train_cm = confusionMatrix(smsspam_svm_train_pred,smspam_train_labels)
ss_test_cm = confusionMatrix(smsspam_svm_test_pred,smspam_test_labels)

ss_train_accuracy = ss_train_cm$overall[1]
cat("Train dataset Accuracy :", ss_train_accuracy)

## Train dataset Accuracy : 0.9925

ss_test_accuracy = ss_test_cm$overall[1]
cat("\nTest dataset Accuracy :", ss_test_accuracy)

##
## Test dataset Accuracy : 0.9707937
```