FIT3152 Data analytics – 2023: Assignment 2

Lee Zhen Xuan 31860532

Q1 Proportion of days when it is more humid than the previous day compared to those where it is less humid?

Based on the data, the R code "table(WAUS['MHT'])" denotes that there are 999 days that is less humid tomorrow, and 931 days that are more humid tomorrow. Therefore, the proportion of days when it is more humid tomorrow Is 931/1930

Q2 Pre-processing required

One of the pre-processing required is by removing all the NA values from the data for better analysis. Before removing the NA values, there are 2000 rows, and 22 columns. After removing all the NA values, there are 385 rows, and 22 columns. The most NA values appear in the "Sunshine" column, where there are 1175 NA values, and the least NA values are in the "Location" column, where there are no NA values at all. The picture below shows the number of NA values that are present in each column:

```
> na_count <-sapply(WAUS, function(y) sum(length(which(is.na(y)))))
> na_count
         Year      Location       MinTemp       MaxTemp      Rainfall   Evaporation
           28             0            39            26            85          1289
     Sunshine   WindGustDir WindGustSpeed    WindDir9am    WindDir3pm  WindSpeed9am
         1175            60            60           152            37            35
 WindSpeed3pm   Pressure9am   Pressure3pm      Cloud9am      Cloud3pm       Temp9am
           40           295           303          1023          1024            36
      Temp3pm     RainToday       RISK_MM           MHT
           24            77            72            70
```

Another pre-processing done is by using the as.factor() function to convert the MHT column to categorical/factor variables.

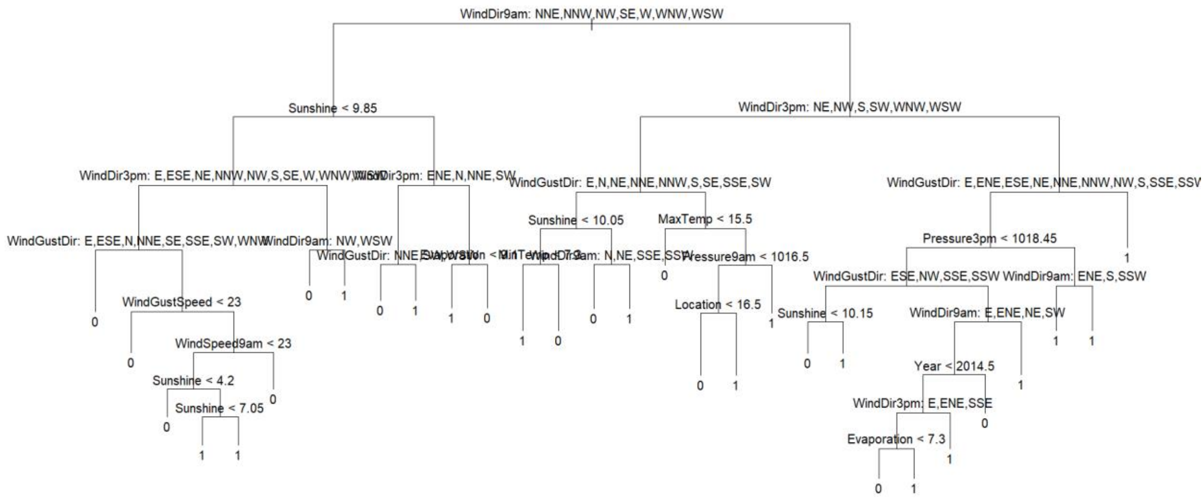
Q3 Divide your data into a 70% training and 30% test set

The data has been divided where 70% of the data is being used to train the models, and 30% of the data used to test the models on predicting unseen data.

Q4 and Q5 Confusion Matrix and Accuracy of each model:

Decision Tree:

The decision tree model has been implemented on the data, which results in 30 terminal nodes, a residual mean deviance of 0.523 (125/239) and a misclassification error rate of 0.108 (29/269). The picture below shows the plot of the decision tree:



The Naïve Bayes method was also implemented as well, using the function naiveBayes(), Bagging method with the bagging() function, boosting with the boosting() function, and Random Forest with randomForest() function. All the data used to train the model is from the subset of the data "WAUS" as "WAUS.train".

The confusion matrices for each of the model is shown below:

Decision Tree:

Predicted_Class	Actual_Class 0	1
0	21	12
1	43	40

Accuracy:0.5259

Naïve Bayes:

Predicted_Class	Actual_Class 0	1
0	20	12
1	44	40

Accuracy: 0.5172

Bagging:

Predicted Class	Observed Class 0	1
0	17	24
1	47	28

Accuracy: 0.3879

Boosting:

Predicted Class	Observed Class 0	1
0	23	16
1	41	36

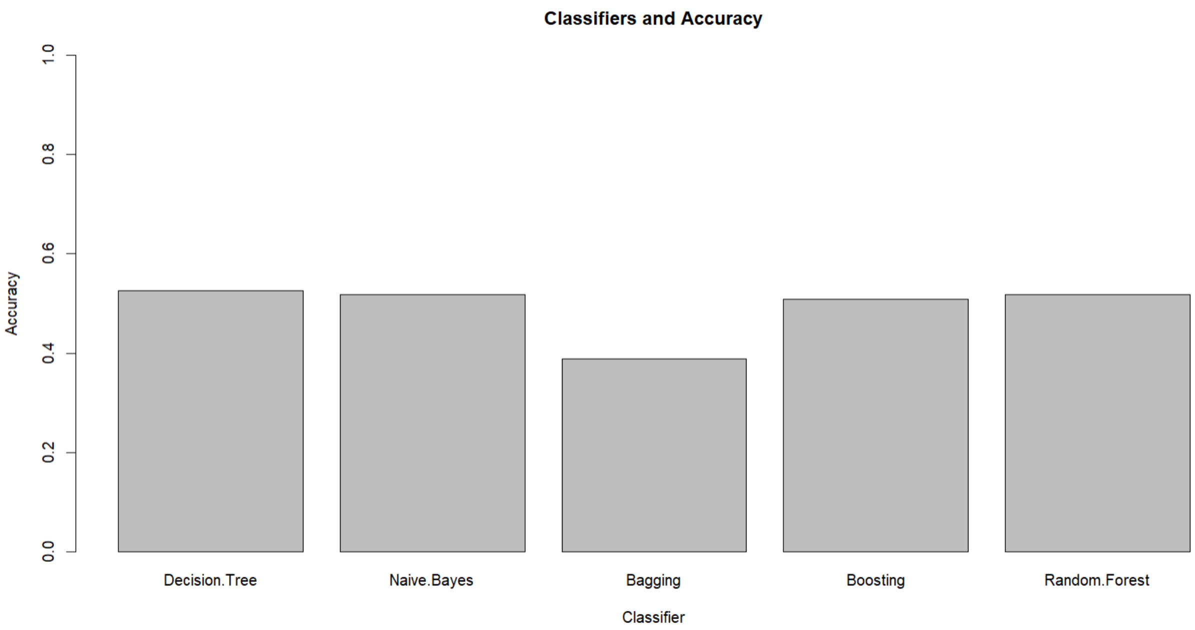
Accuracy: 0.5086

Random Forest:

Predicted_Class	Actual_Class 0	1
0	12	4
1	52	48

Accuracy: 0.5172

The plot below shows the comparison of accuracy between all the models:



As we can see from the accuracy plot and the numbers stated above, the Decision Tree model has a better accuracy in predicting the unseen data, where the Bagging model performs worse than all the models with only a 0.3879 accuracy reported.

Q6 Confidence of predicting ‘more humid tomorrow’ for each case and ROC curve for each classifier

The confidence of predicting ‘more humid tomorrow’ for each case for each of the model for some of the cases is shown below:

Decision Tree

```
            0       1
77780 0.27273 0.72727
94161 0.80000 0.20000
26006 0.00000 1.00000
42264 0.00000 1.00000
39573 0.33333 0.66667
93719 0.00000 1.00000
10167 0.04762 0.95238
96396 0.04762 0.95238
76685 0.00000 1.00000
21709 0.00000 1.00000
5704  0.80000 0.20000
73219 0.33333 0.66667
26537 0.00000 1.00000
35181 0.00000 1.00000
93547 0.28571 0.71429
73881 0.83333 0.16667
99524 1.00000 0.00000
6332  0.50000 0.50000
43345 0.94444 0.05556
98392 0.04762 0.95238
8822  0.00000 1.00000
11767 0.00000 1.00000
32768 1.00000 0.00000
9612  0.80000 0.20000
98915 0.37209 0.62791
22343 0.80000 0.20000
79609 0.94444 0.05556
33946 0.00000 1.00000
45321 0.83333 0.16667
65231 0.00000 1.00000
328   0.00000 1.00000
50470 0.33333 0.66667
72171 0.04762 0.95238
83716 0.50000 0.50000
48985 0.37209 0.62791
```

Naïve Bayes

```
             0        1
 [1,] 0.870177 0.129823
 [2,] 0.025432 0.974568
 [3,] 0.005320 0.994680
 [4,] 0.107256 0.892744
 [5,] 0.031514 0.968486
 [6,] 0.053239 0.946761
 [7,] 0.729058 0.270942
 [8,] 0.121632 0.878368
 [9,] 0.081739 0.918261
[10,] 0.578852 0.421148
[11,] 0.010377 0.989623
[12,] 0.037418 0.962582
[13,] 0.031222 0.968778
[14,] 0.034697 0.965303
[15,] 0.064140 0.935860
[16,] 0.978999 0.021001
[17,] 0.139264 0.860736
[18,] 0.659984 0.340016
[19,] 0.691296 0.308704
[20,] 0.897810 0.102190
[21,] 0.052596 0.947404
[22,] 0.005291 0.994709
[23,] 0.993802 0.006198
[24,] 0.119116 0.880884
[25,] 0.810275 0.189725
[26,] 0.891201 0.108799
[27,] 0.604453 0.395547
[28,] 0.058170 0.941830
[29,] 0.028726 0.971274
[30,] 0.083014 0.916986
```

Boosting:

Bagging:

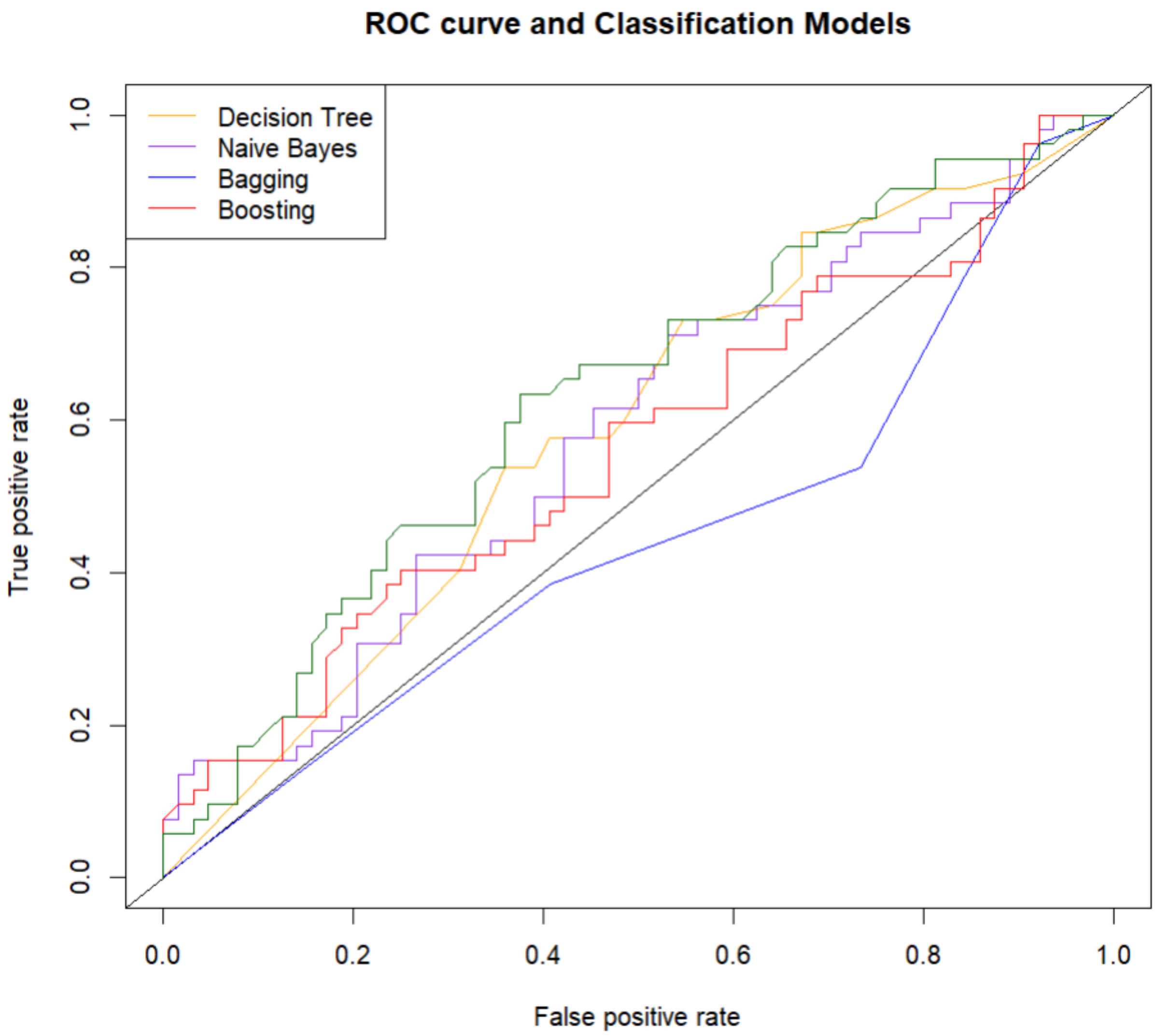
	[,1]	[,2]
[1,]	0.59176	0.4082
[2,]	0.31700	0.6830
[3,]	0.00000	1.0000
[4,]	0.31248	0.6875
[5,]	0.49162	0.5084
[6,]	0.21342	0.7866
[7,]	0.52366	0.4763
[8,]	0.70536	0.2946
[9,]	0.38542	0.6146
[10,]	0.41219	0.5878
[11,]	0.60016	0.3998
[12,]	0.48322	0.5168
[13,]	0.09458	0.9054
[14,]	0.19379	0.8062
[15,]	0.27168	0.7283
[16,]	0.47219	0.5278
[17,]	0.31073	0.6893
[18,]	0.53148	0.4685
[19,]	0.47872	0.5213
[20,]	0.68547	0.3145
[21,]	0.30420	0.6958
[22,]	0.09211	0.9079
[23,]	0.88766	0.1123
[24,]	0.71260	0.2874
[25,]	0.49918	0.5008
[26,]	0.41451	0.5855
[27,]	0.78883	0.2112
[28,]	0.58192	0.4181
[29,]	0.31248	0.6875
[30,]	0.62546	0.3745
[31,]	0.27666	0.7233
[32,]	0.38661	0.6134

	[,1]	[,2]
[1,]	1.0	0.0
[2,]	0.2	0.8
[3,]	0.4	0.6
[4,]	0.2	0.8
[5,]	0.2	0.8
[6,]	0.2	0.8
[7,]	0.8	0.2
[8,]	0.6	0.4
[9,]	0.0	1.0
[10,]	0.6	0.4
[11,]	0.2	0.8
[12,]	0.2	0.8
[13,]	0.0	1.0
[14,]	0.2	0.8
[15,]	0.4	0.6
[16,]	1.0	0.0
[17,]	0.0	1.0
[18,]	0.4	0.6
[19,]	0.4	0.6
[20,]	0.8	0.2
[21,]	0.2	0.8
[22,]	0.2	0.8
[23,]	0.8	0.2
[24,]	0.2	0.8
[25,]	0.2	0.8
[26,]	0.4	0.6
[27,]	1.0	0.0
[28,]	0.4	0.6
[29,]	0.2	0.8
[30,]	0.4	0.6
[31,]	0.4	0.6
[32,]	0.6	0.4

Random Forest:

	0	1
77780	0.500	0.500
94161	0.144	0.856
26006	0.146	0.854
42264	0.466	0.534
39573	0.362	0.638
93719	0.160	0.840
10167	0.450	0.550
96396	0.400	0.600
76685	0.396	0.604
21709	0.280	0.720
5704	0.242	0.758
73219	0.364	0.636
26537	0.192	0.808
35181	0.268	0.732
93547	0.330	0.670
73881	0.664	0.336
99524	0.386	0.614
6332	0.458	0.542
43345	0.404	0.596
98392	0.596	0.404
8822	0.276	0.724
11767	0.082	0.918
32768	0.708	0.292
9612	0.532	0.468
98915	0.384	0.616
22343	0.460	0.540
79609	0.622	0.378
33946	0.328	0.672
45321	0.302	0.698

The ROC curve for each of the classifier is shown below:



The AUC values from the curves above are shown below:

	Decision.Tree	Naive.Bayes	Bagging	Boosting	Random.Forest
1	0.588	0.5841	0.4474	0.5639	0.6313

The AUC values from the ROC curves summarizes the overall performance of a binary classification model, to differentiate the positive and negative classes, therefore Random Forest indicates a better discrimination power and can predict the unseen data better.

Q7 Table comparing the results

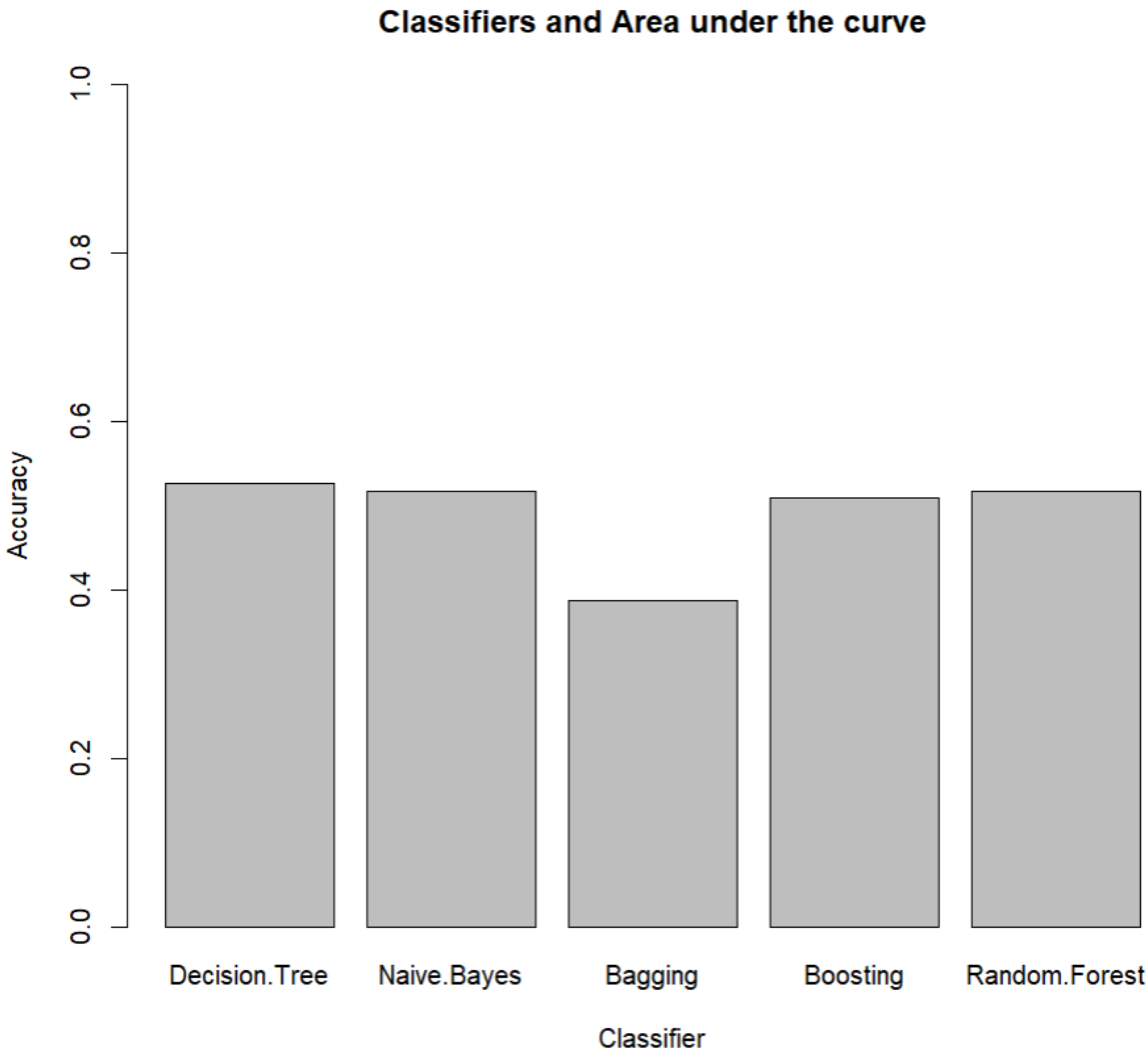
Based on the accuracy of each of the classification model and their AUC values shown below, it seems that the Random Forest classifier performs the best overall in terms of accuracy, and the AUC value based on its ROC curve, with a high acccuracy of 0.5172, with the highest AUC value of 0.6313.

Accuracy values:

	Decision.Tree	Naive.Bayes	Bagging	Boosting	Random.Forest
1	0.5259	0.5172	0.3879	0.5086	0.5172

AUC values:

	Decision.Tree	Naive.Bayes	Bagging	Boosting	Random.Forest
1	0.588	0.5841	0.4474	0.5639	0.6313



Q8 Most important variables in predicting whether it will be more humid tomorrow or not

The most important variables for each of the classification models can be shown below:

Decision Tree:

"WindDir9am", "Sunshine”, "WindDir3pm”, "WindGustDir", "WindGustSpeed", "WindSpeed9am", "Evaporation","MinTemp", "MaxTemp", "Pressure9am", "Location", "Pressure3pm", "Year"

Bagging:

WindGustDir: 24.808 WindDir3pm: 23.788 WindDir9am: 21.607 Cloud9am: 9.296 Pressure3pm: 3.106 Pressure9am: 3.683 WindSpeed3pm: 2.217 Evaporation: 2.166	Sunshine: 2.128 Temp9am: 1.675 Location: 1.198 Cloud3pm: 1.161 Year: 1.160 WindSpeed9am: 1.009 MaxTemp: 0 MinTemp: 0 RainFall: 0 RainToday: 0 RISK_MM: 0 Temp3pm: 0 WindGustSpeed: 0

Boosting:

WindDir3pm: 21.1322 WindGustDir: 20.1133 WindDir9am: 17.1757 MinTemp: 6.1306 Sunshine: 5.3431 MaxTemp: 5.3114 Evaporation: 4.8913 WindSpeed3pm: 4.5770 Pressure3pm: 4.0845	Pressure9am: 3.6728 Cloud9am: 2.0815 Cloud3pm: 1.4920 WindGustSpeed: 1.4331 Temp9am: 1.0303 Temp3pm: 0.9584 Location: 0.5728 Rainfall: 0 RainToday: 0 RISK_MM: 0 WindSpeed9am: 0 Year: 0

Random Forest:

WindDir9am : 17.2405735679168 WindDir3pm : 16.3722134285823 WindGustDir : 13.8150203701903 Sunshine : 7.11033720608023 Pressure9am : 6.62256244945334 MinTemp : 6.21497541583388 Pressure3pm : 6.17765315595554 MaxTemp : 6.05332192881562 Cloud9am : 5.72345107010616 Temp9am : 5.69855690516244 Evaporation : 5.58340143252628	Temp3pm : 5.56745210420568 WindSpeed3pm : 4.92771982841906 WindGustSpeed : 4.78664322712217 Year : 4.33447316418825 WindSpeed9am : 3.99891458064103 Cloud3pm : 3.53748859773141 Rainfall : 2.71455454147681 RISK_MM : 2.22471506413356 Location : 1.40823322600317 RainToday : 0.522851498776918

Based on the classification models with their important variables above, the importance of variables of the classification models of Bagging, Boosting and Random Forest are ordered by their importance, where the higher the number for each of the variables, the higher importance of the variables used in the classification model. We can see that WindDir9am, WindDir3pm, WindGustDir appears very consistently as important variables by all the classification model used, and they highly affect the outcome of the classification model, where particularly the variables RainFall, RainToday and RISK_MM have low importance or no importance at all when developing each of the classification models. Therefore, since RainFall, RainToday, and RISK_MM as very little effect on performance, they should be omitted from the data. The classification model of Naïve Bayes on the other hand, does not have a straightforward way of calculating the importance of variables, where the model treats each of the features independently and assigns its weights based on their individual properties. Therefore, the variable importance is not included.

Q9 Classifier that is simple enough for a person to be able to classify whether it will be more humid tomorrow or not by hand

First, the attributes chosen to be included in the classifier are WindDir9am, WindDir3pm and WindGustDir as stated in Question 8, there are the most important variables in all the classification models. A threshold is applied to all of the unique values in each of these variables, and the threshold is calculated by using the proportion of 1s in MHT that appear in each unique value of these 3 variables. For example, if WAUS$WindGust is "E" and the MHT is 1 appears 10 times, and WAUS$WindGust is "E" and the MHT is 0 appears 20 times, the threshold is 10/30. Therefore, using this threshold, we can use the WAUS.test to find if all of these threshold added together and averaged is more than 0.5, then the classifier will predict as 1, else it will predict as 0. The confusion matrix is shown below:

```
               Actual_Class
Predicted_Class  0  1
              0 23 19
              1 41 33
```

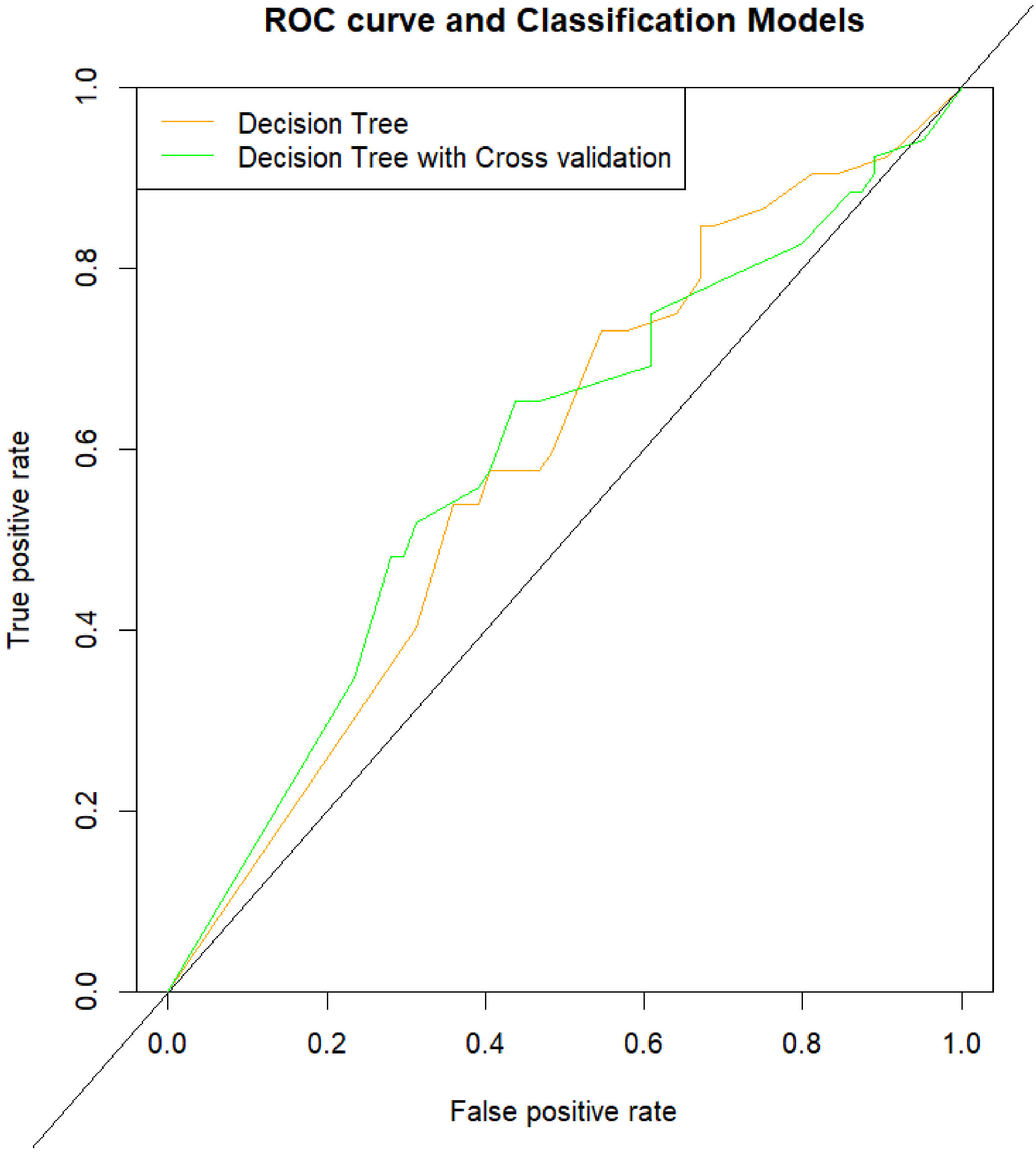
The accuracy of this simple classifer by hand is 0.4828, which is slightly lower than the accuracies for the classifier models performed above.

Q10 Best tree-based classifier

The best tree-based classifer has been created by using cross-validation, and post pruning of a decision tree model. However, even though I have tried different values for the pruned tree minumum node size, complexity parameter to extreme values, the accuracy of the cross-validation decision tree is still the same compared to the normal decision tree with an accuracy of 0.5259. However, the AUC value for the improved decision tree tends to be slightly higher than with a value of 0.5934 compared to 0.588.

A cross-validation model of a random forest is also performed as well, but the accuracy of the cross-validation model after tuning all the parameters only goes up as high as 0.5259 as well, which is slightly higher than the accuracy of a normal Random Forest model of an accuracy value of 0.5172.

Here is a plot of the ROC curve for the Cross Validation Decision Tree against the the normal Decision Tree:



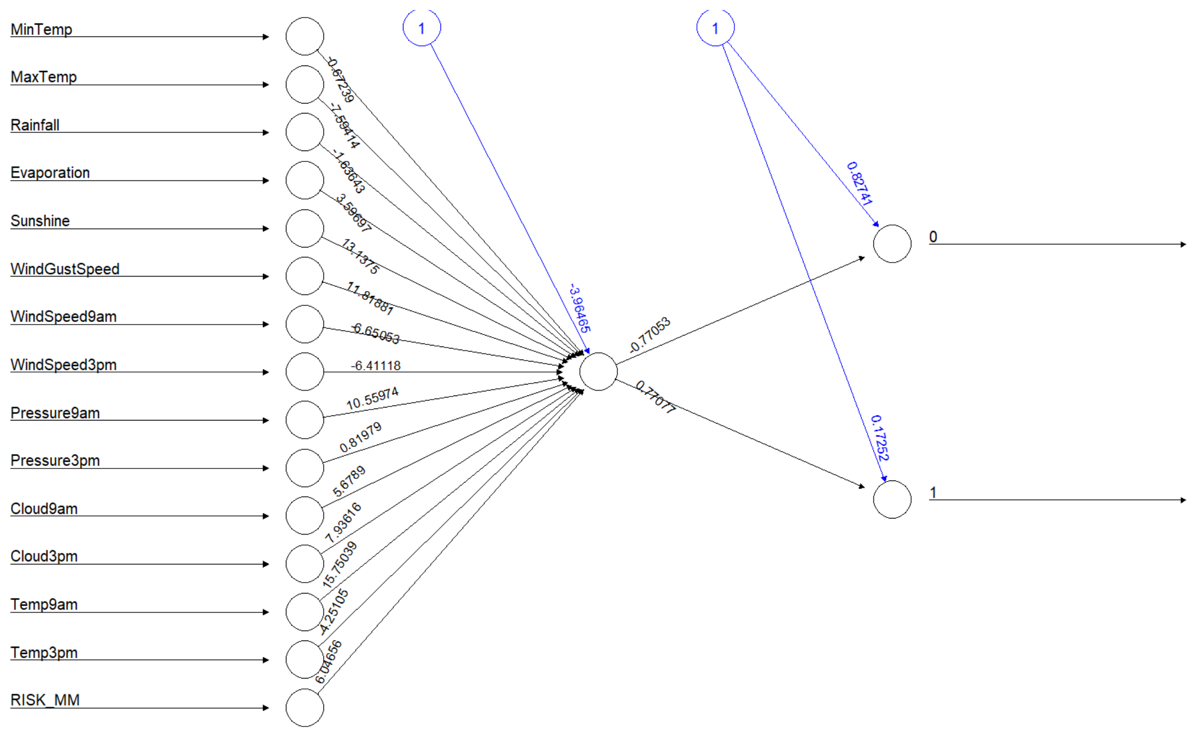
Q11 Artificial Neural Network classifier and its performance

The attributes used in the Artificial Neural Network classfier are:

"MinTemp", "MaxTemp", "Rainfall", "Evaporation", "Sunshine", "WindGustSpeed", "WindSpeed9am", "WindSpeed3pm", "Pressure9am", "Pressure3pm", "Cloud9am", "Cloud3pm", "Temp9am", "Temp3pm", "RISK_MM"

The predictor variables in the training and test sets of the Artificial Neural Network are first normalised in order for the model to work correctly, and to make predictions on the test set.

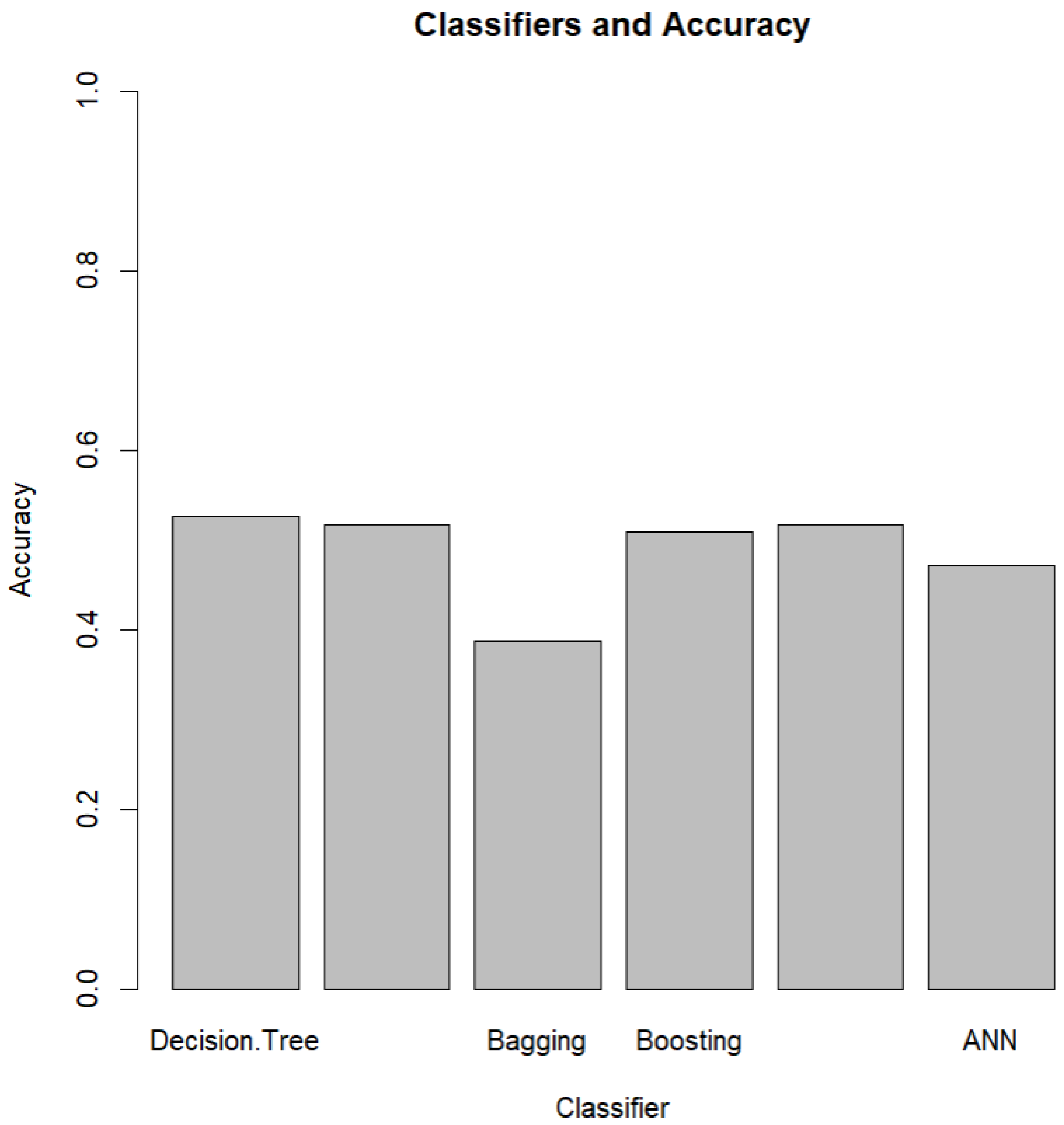
After the ANN model with 1 hidden layer has been set up, a plot of the ANN model is shown below:



The Aritificial Neural Network model has reported an accuracy of 0.4715 which is lower than the previous classifiers models, where the confusion matrix is shown below:

```
           predicted
observed   0  1
         0 28 51
         1 51 63
```

The plot of the accuracy of ANN and other classifiers can be seen below:



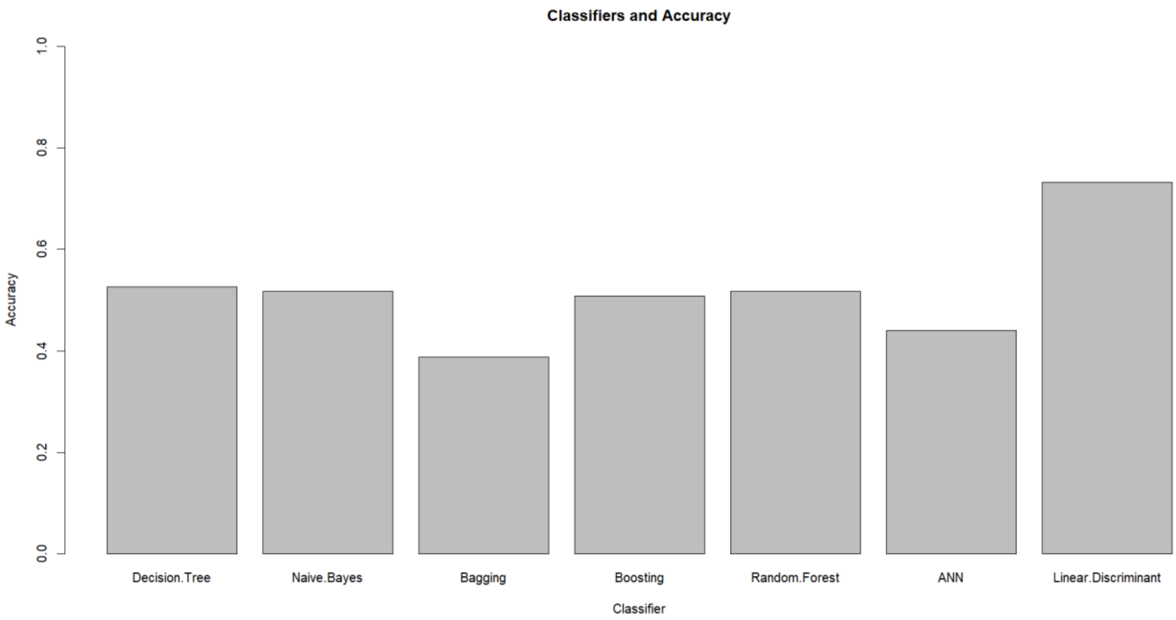
	Decision.Tree	Naive.Bayes	Bagging	Boosting	Random.Forest	ANN
1	0.5259	0.5172	0.3879	0.5086	0.5172	0.4715

Q12 New classifier to the data: Linear Discriminant Analysis (LDA).

The new classifier used in the data is Linear Discriminant Analysis (LDA). It is a dimenisionality reduction technique of classification used to reduce the number of dimensions (variables) in a dataset while still retaining as much information as possible. It helps to find the linear combination of the original variables that provide the best possible separation betweeen the groups.

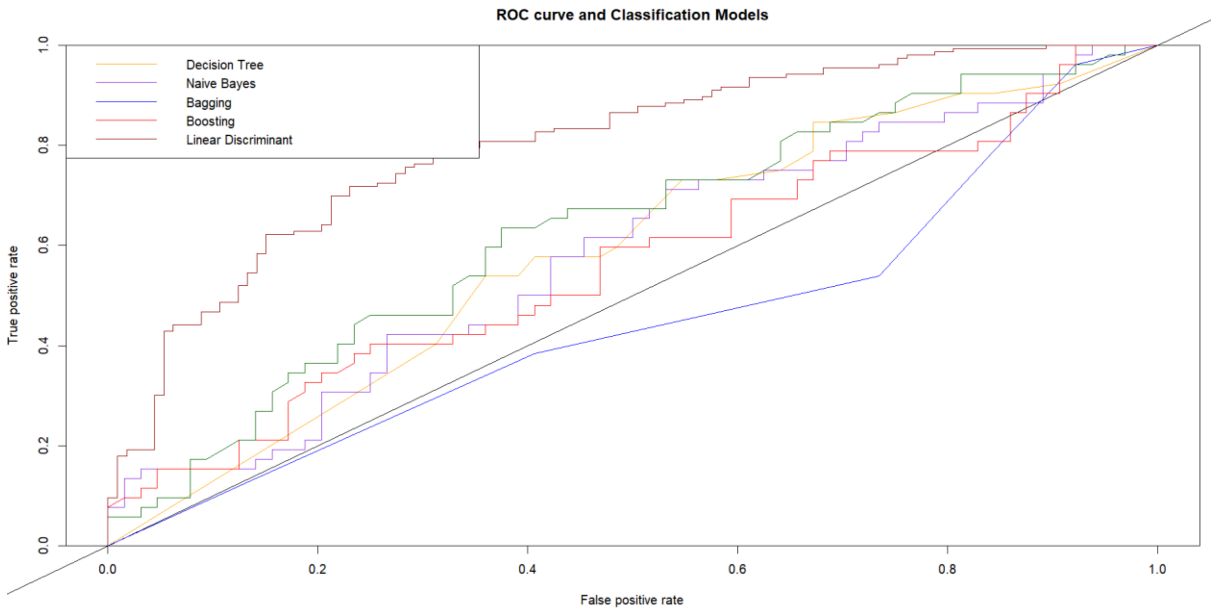
After the model has been implemented, the model has reported an accuracy of as high as 0.7323 with an AUC of the ROC curve of 0.8019, which is an significant improvement compared to the other classifiers. Therefore, the Linear Discriminant Analysis model is the best classification model in order to predict the unseen data of whether or not it will be more himid tomorrow.

The accuracy comparision plot of all the classifiers can be seen below:



	Decision.Tree	Naive.Bayes	Bagging	Boosting	Random.Forest	ANN	Linear.Discriminant
1	0.5259	0.5172	0.3879	0.5086	0.5172	0.4404	0.7323

The ROC curve plot can also be seen below:



In conclusion, there are various methods that can be untilized to predict the values of MHT, and even other larger datasets out there. However, it is important to choose the best classification method to best suit the dataframe with the appropriate parameter tunings in order to get the best results of predicting the unseen data.

R code

```
setwd("C:/Monash/FIT3152/Assignment2")

library(dplyr)

library(tree)
library(e1071)
library(ROCR)
library(ggplot2)
library(gridExtra)

library(randomForest)
library(adabag)
```

```
library(rpart)

library(caret)

detach("package:neuralnet", unload = TRUE)

#install.packages('rpart')

library(rpart)



rm(list = ls())

WAUS <- read.csv("HumidPredict2023D.csv")

L <- as.data.frame(c(1:49))

set.seed(31860532) # Your Student ID is the random seed

L <- L[sample(nrow(L), 10, replace = FALSE),] # sample 10 locations

WAUS <- WAUS[(WAUS$Location %in% L),]

WAUS <- WAUS[sample(nrow(WAUS), 2000, replace = FALSE),] # sample 2000 rows


dim(WAUS)


str(WAUS)

na_count <-sapply(WAUS, function(y) sum(length(which(is.na(y)))))

na_count

data.frame(na_count)

nrow(WAUS)


summary(WAUS)


str(WAUS)

dim(WAUS) # 22 columns from the dataset

ncol(WAUS) #2000 rows from the dataset

nrow(WAUS)

sapply(WAUS, function(x) sum(is.na(x)))
```

```
#Counting the number of 1s and 0s in MHT
table(WAUS['MHT'])
#999 0s
#931 1s



#preprocessing by removing NA values from the dataset
WAUS = WAUS[complete.cases(WAUS),]

dim(WAUS)
na_count <-sapply(WAUS, function(y) sum(length(which(is.na(y)))))

#the dimension of the dataset after preprocessing
WAUS$MHT <- recode(WAUS$MHT, `0` = "0", `1` = "1")
WAUS$MHT <- as.factor(WAUS$MHT)

WAUS$WindGustDir = as.factor(WAUS$WindGustDir)
WAUS$WindDir9am = as.factor(WAUS$WindDir9am)
WAUS$WindDir3pm = as.factor(WAUS$WindDir3pm)

str(WAUS)
dim(WAUS) # 22 columns from the dataset
ncol(WAUS) #2000 rows from the dataset
nrow(WAUS)
sapply(WAUS, function(x) sum(is.na(x)))

#Counting the number of 1s and 0s in MHT
table(WAUS['MHT'])
#999 0s
#931 1s
```

```
set.seed(31860532) #Student ID as random seed

train.row = sample(1:nrow(WAUS), 0.7*nrow(WAUS))

WAUS.train = WAUS[train.row,]

WAUS.test = WAUS[-train.row,]

nrow(WAUS.train)

WAUS.test


class(WAUS$MHT)

# Decision tree model

WAUS.tree = tree(MHT ~., data = WAUS.train)

print(summary(WAUS.tree))

plot(WAUS.tree)

text(WAUS.tree, pretty = 0)


unique(WAUS.tree$frame$var)


#do predictions for decision tree

WAUS.predtree = predict(WAUS.tree, WAUS.test, type = "class")

WAUS.predtree

DT_classification <- ifelse(WAUS.predtree == 1, "more humid tomorrow", "less humid tomorrow")

DT_CM=table(Predicted_Class = WAUS.predtree, Actual_Class = WAUS.test$MHT)

DT_CM


#calculating accuracy for decision tree

TP = DT_CM[1,1]

FP = DT_CM[1,2]

FN = DT_CM[2,1]

TN = DT_CM[2,2]

acc_DT = (TP + TN) / (TP+TN+FP+FN)

acc_DT
```

```
#Plotting ROC curve for Decision Tree
WAUS.pred.tree = predict(WAUS.tree, WAUS.test, type = "vector")
WAUS.pred.tree
prediction( WAUS.pred.tree[,2], WAUS.test$MHT)
DT_pred <- prediction( WAUS.pred.tree[,2], WAUS.test$MHT)

DT_perf <- performance(DT_pred,"tpr","fpr")
plot(DT_perf, col = "orange", main = "ROC curve and Classification Models")
abline(0,1)

#AUC
auc_DT <- performance(DT_pred, "auc")@y.values[[1]]
auc_DT



# Calculate naive bayes
WAUS.bayes = naiveBayes(MHT ~. , data = WAUS.train)

WAUS.predbayes = predict(WAUS.bayes, WAUS.test)
NB_classification <- ifelse(WAUS.predbayes == 1, "more humid tomorrow", "less humid tomorrow")
NB_CM=table(Predicted_Class = WAUS.predbayes, Actual_Class = WAUS.test$MHT)
NB_CM
#calculating accuracy for naive bayes
TP = NB_CM[1,1]
FP = NB_CM[1,2]
FN = NB_CM[2,1]
TN = NB_CM[2,2]
acc_NB = (TP + TN) / (TP+TN+FP+FN)
acc_NB
```

```
#Plotting ROC curve for naive bayes
WAUSpred.bayes = predict(WAUS.bayes, WAUS.test, type = 'raw')
WAUSpred.bayes
NB_pred <- prediction( WAUSpred.bayes[,2], WAUS.test$MHT)
NB_perf <- performance(NB_pred,"tpr","fpr")
plot(NB_perf, add=TRUE, col = "blueviolet", main = "AUC curve for each of the models")



#AUC
auc_NB <- performance(NB_pred, "auc")@y.values[[1]]
auc_NB

# Bagging
WAUS.bag <- bagging(MHT ~. , data = WAUS.train, mfinal=5)
WAUSpred.bag <- predict.bagging(WAUS.bag, WAUS.test)
WAUSpred.bag
BAG_CM = WAUSpred.bag$confusion
BAG_CM

#calculating accuracy for bagging
TP = BAG_CM[1,1]
FP = BAG_CM[1,2]
FN = BAG_CM[2,1]
TN = BAG_CM[2,2]
acc_BAG = (TP + TN) / (TP+TN+FP+FN)
acc_BAG

#Plotting ROC curve for Bagging
BAG_pred <- prediction( WAUSpred.bag$prob[,2], WAUS.test$MHT)
BAG_perf <- performance(BAG_pred,"tpr","fpr")
plot(BAG_perf, add=TRUE, col = "blue")
```

```
#AUC

auc_BAG <- performance(BAG_pred, "auc")@y.values[[1]]

auc_BAG


#Finding important variables

BAG_imp = WAUS.bag$importance

BAG_imp[order(BAG_imp, decreasing = TRUE)]



#Boosting

WAUS.Boost <- boosting(MHT ~. , data = WAUS.train, mfinal=10)

WAUSpred.boost <- predict.boosting(WAUS.Boost, newdata=WAUS.test)

WAUSpred.boost

BST_classification <- ifelse(WAUSpred.boost$class == 1, "more humid tomorrow", "less humid
tomorrow")

BST_CM = WAUSpred.boost$confusion

BST_CM


#calculating accuracy for boosting

TP = BST_CM[1,1]

FP = BST_CM[1,2]

FN = BST_CM[2,1]

TN = BST_CM[2,2]

acc_BST = (TP + TN) / (TP+TN+FP+FN)

acc_BST


#Plotting ROC Curve for boosting

BST_pred <- prediction( WAUSpred.boost$prob[,2], WAUS.test$MHT)

BST_perf <- performance(BST_pred,"tpr","fpr")

plot(BST_perf, add=TRUE, col = "red")
```

```
#AUC
auc_BST <- performance(BST_pred, "auc")@y.values[[1]]
auc_BST

#Finding important variables
BST_imp = WAUS.Boost$importance
BST_imp[order(BST_imp, decreasing = TRUE)]


# Random Forest
WAUS.rf <- randomForest(MHT ~. , data = WAUS.train, na.action = na.exclude)
WAUSpredrf <- predict(WAUS.rf, WAUS.test)
WAUSpredrf
RF_classification <- ifelse(WAUSpredrf == 1, "more humid tomorrow", "less humid tomorrow")
RF_CM=table(Predicted_Class = WAUSpredrf, Actual_Class = WAUS.test$MHT)
RF_CM

#calculating accuracy for random forest
TP = RF_CM[1,1]
FP = RF_CM[1,2]
FN = RF_CM[2,1]
TN = RF_CM[2,2]
acc_RF = (TP + TN) / (TP+TN+FP+FN)
acc_RF

#Plotting ROC curve for random forest
WAUSpred.rf <- predict(WAUS.rf, WAUS.test, type="prob")
WAUSpred.rf
WAUSRFpred <- prediction( WAUSpred.rf[,2], WAUS.test$MHT)
WAUSRFperf <- performance(WAUSRFpred,"tpr","fpr")
```

```
plot(WAUSRFperf, add=TRUE, col = "darkgreen")

legend(x = "topleft",
       col = c( "orange","blueviolet","blue","red"), lty = 1, lwd = 1,
       legend = c('Decision Tree', 'Naive Bayes', 'Bagging','Boosting'))

#AUC
auc_RF <- performance(WAUSRFpred, "auc")@y.values[[1]]
auc_RF


#Important variables for Random Forest
importance_values <- WAUS.rf$importance[,"MeanDecreaseGini"]
variable_names <- names(importance_values)

# Order the importance in decreasing order
importance_order <- order(importance_values, decreasing = TRUE)
importance_order

importance_sorted <- importance_values[importance_order]
variable_names_sorted <- variable_names[importance_order]
# Print the importance in decreasing order
for (i in 1:length(importance_sorted)) {
  print(paste(variable_names_sorted[i], ":", importance_sorted[i]))
}


# Define the decision rule
classify_humidity <- function(data) {
  max_temp <- data$MaxTemp
  rainfall <- data$Rainfall
```

```
  wind_speed <- data$WindSpeed9am
  cloud_cover <- data$Cloud9am

  humidity <- ifelse(max_temp >= 25 & rainfall < 10 & wind_speed < 20 & cloud_cover < 5, 1, 0)
  return(humidity)
}

# Apply the classifier to the test data
WAUS.test$Predicted_MHT <- classify_humidity(WAUS.test)

# Evaluate model performance
TP <- sum(WAUS.test$Predicted_MHT == 1 & WAUS.test$MHT == 1)
FP <- sum(WAUS.test$Predicted_MHT == 1 & WAUS.test$MHT == 0)
TN <- sum(WAUS.test$Predicted_MHT == 0 & WAUS.test$MHT == 0)
FN <- sum(WAUS.test$Predicted_MHT == 0 & WAUS.test$MHT == 1)

accuracy <- (TP + TN) / (TP + FP + TN + FN)
accuracy


#Comparison of accuracy for each model
accuracy_table <- data.frame("Decision Tree" = acc_DT, "Naive Bayes" = acc_NB, "Bagging" = acc_BAG, "Boosting" = acc_BST, "Random Forest" = acc_RF)
accuracy_table
classifiers <- colnames(accuracy_table)
accuracy_values <- unlist(accuracy_table)
barplot(accuracy_values, names.arg = classifiers, ylim = c(0, 1), ylab = "Accuracy", xlab = "Classifier", main = "Classifiers and Accuracy")

#Comparison of auc for each model
auc_table <- data.frame("Decision Tree" = auc_DT, "Naive Bayes" = auc_NB, "Bagging" = auc_BAG, "Boosting" = auc_BST, "Random Forest" = auc_RF)
```

```
auc_table

classifiers_auc <- colnames(accuracy_table)

auc_values <- unlist(accuracy_table)

barplot(auc_values, names.arg = classifiers_auc, ylim = c(0, 1), ylab = "Accuracy", xlab = "Classifier",
main = "Classifiers and Area under the curve")


View(auc_table)



#Classification model by hand


# Count the occurrences of each value in WAUS$WindDir9am for MHT = 1

count_1 <- table(WAUS$WindDir9am[WAUS$MHT == 1])

count_1

# Count the occurrences of each value in WAUS$WindDir9am for MHT = 0

count_0 <- table(WAUS$WindDir9am[WAUS$MHT == 0])

count_0


# Calculate the thresholds for predicting 1 for each value in WAUS$WindDir9am

thresholds <- NULL


# Loop through each value in WAUS$WindDir9am

for (value in unique(WAUS$WindDir9am)) {

  # Calculate the threshold for the current value

  threshold <- count_1[value] / (count_0[value] + count_1[value])


  # Store the threshold in the thresholds vector

  WindDir9amthresholds <- c(thresholds, threshold)

}
```

```
# Count the occurrences of each value in WAUS$WindDir9am for MHT = 1
count_1 <- table(WAUS$WindDir3pm[WAUS$MHT == 1])
count_1
# Count the occurrences of each value in WAUS$WindDir9am for MHT = 0
count_0 <- table(WAUS$WindDir3pm[WAUS$MHT == 0])
count_0

for (value in unique(WAUS$WindDir3pm)) {
  # Calculate the threshold for the current value
  threshold <- count_1[value] / (count_0[value] + count_1[value])

  # Store the threshold in the thresholds vector
  WindDir3pmthresholds <- c(thresholds, threshold)
}

count_1 <- table(WAUS$WindGustDir[WAUS$MHT == 1])
count_1
# Count the occurrences of each value in WAUS$WindDir9am for MHT = 0
count_0 <- table(WAUS$WindGustDir[WAUS$MHT == 0])
count_0

for (value in unique(WAUS$WindGustDir)) {
  # Calculate the threshold for the current value
  threshold <- count_1[value] / (count_0[value] + count_1[value])

  # Store the threshold in the thresholds vector
  WindGustDirthresholds <- c(thresholds, threshold)
}


predictions <- vector("integer", length = nrow(WAUS.test))
```

```r
for (i in 1:nrow(WAUS.test)) {
  if ((WindDir9amthresholds[WAUS.train[i,]$WindDir9am] +
WindDir3pmthresholds[WAUS.train[i,]$WindDir3pm] +
WindGustDirthresholds[WAUS.test[i,]$WindGustDir]) / 3 >= 0.5) {
    predictions[i] <- 1
  } else {
    predictions[i] <- 0
  }
}
predictions
WAUS.test$MHT

#Confusion matrix for classifier by hand
CBH_CM = table(Predicted_Class = predictions, Actual_Class = WAUS.test$MHT)
TP = CBH_CM[1,1]
FP = CBH_CM[1,2]
FN = CBH_CM[2,1]
TN = CBH_CM[2,2]
acc_CBH = (TP + TN) / (TP+TN+FP+FN)
acc_CBH




#Best tree-based classifier - cross validation Decision Tree
#using cv and decision tree
set.seed(31860532)
# Fit the classification tree model
WAUS.improvedtree <- tree(MHT ~ ., data = WAUS.train)
# Perform cross-validation using deviance as the pruning criterion
testptfit <- cv.tree(WAUS.improvedtree, FUN = prune.tree, method = "deviance")
# Prune the tree to the optimal complexity level determined by cross-validation
prunedtree <- prune.tree(WAUS.improvedtree, best = testptfit$k[which.min(testptfit$dev)])
```

```
# Adjust the minimum node size
prunedtree$control$mincut = 10
# Set the complexity parameter
prunedtree$control$cp <- 0.01
# Predict the class labels for new data using the pruned tree
predictions <- predict(prunedtree, newdata = WAUS.test, type = "class")
predictions
improvedDT_CM =table(Predicted_Class = predictions, Actual_Class = WAUS.test$MHT)
improvedDT_CM

#accuracy of Cross validation Decision Tree
TP = improvedDT_CM[1,1]
FP = improvedDT_CM[1,2]
FN = improvedDT_CM[2,1]
TN = improvedDT_CM[2,2]
acc_IDT = (TP + TN) / (TP+TN+FP+FN)
acc_IDT

probs <- predict(prunedtree, newdata = WAUS.test, type = "vector")

# Create the prediction object
DT_improvedpred <- prediction(probs[, 2], WAUS.test$MHT)

# Create the performance object
DT_improvedperf <- performance(DT_improvedpred, measure = "tpr", x.measure = "fpr")

# Plot the ROC curve
plot(DT_improvedperf, main = "ROC Curve for Improved Decision tree", add = TRUE, col = "green")
auc_improvedDT <- performance(DT_improvedpred, "auc")@y.values[[1]]
auc_improvedDT
```

```
legend(x = "topleft",
       col = c( "orange", "green"), lty = 1, lwd = 1,
       legend = c('Decision Tree', 'Decision Tree with Cross validation'))



#Best tree-based classifier 2 - cross validation Random Forest
rfcv_results <- rfcv(
  trainx = WAUS.train[, -ncol(WAUS.train)],
  trainy = WAUS.train$MHT,
  cv.fold = 10,
  scale = "log",
  step = 0.5,
  mtry = function(p) max(1, floor(sqrt(p))),
  recursive = FALSE
)


# Fit the random forest model with the optimal number of variables
rf_model <- randomForest(
  x = WAUS.train[, -ncol(WAUS.train)],
  y = WAUS.train$MHT

)
# Make predictions using the random forest model
predictions <- predict(rf_model, newdata = WAUS.test[, -ncol(WAUS.test)])
improvedRF_CM=table(Predicted_Class = predictions, Actual_Class = WAUS.test$MHT)
improvedRF_CM
```

```
# ANN model

options(digits=4)


set.seed(31860532) #Student ID as random seed

train.row = sample(1:nrow(WAUS), 0.7*nrow(WAUS))

WAUS.train = WAUS[train.row,]

WAUS.test = WAUS[-train.row,]

WAUScombined = rbind(WAUS.train, WAUS.test)

WAUSmm = model.matrix(~WindGustDir+WindDir9am+WindDir3pm+RainToday,
data=WAUScombined)


WAUScombined = cbind(WAUScombined, WAUSmm)

str(WAUScombined)



nrow(WAUScombined)

WAUStest = WAUScombined[1:193,]

WAUStrain = WAUScombined[194:385,]

# Select the predictor columns

predictor_cols <- c("MinTemp", "MaxTemp", "Rainfall", "Evaporation", "Sunshine",
"WindGustSpeed", "WindSpeed9am", "WindSpeed3pm", "Pressure9am", "Pressure3pm",
"Cloud9am", "Cloud3pm", "Temp9am", "Temp3pm", "RISK_MM")

WAUScombined <- WAUScombined[, c(predictor_cols, "MHT")]


# Normalize the predictor columns in the training and test sets

preproc <- preProcess(WAUStrain[, predictor_cols], method = c("center", "scale"))

WAUStrain[, predictor_cols] <- predict(preproc, WAUStrain[, predictor_cols])

WAUStest[, predictor_cols] <- predict(preproc, WAUStest[, predictor_cols])
```

```
set.seed(31860532)

WAUStrain = WAUStrain[sample(nrow(WAUStrain), 100, replace = TRUE),]

WAUStrain = as.data.frame(WAUStrain)

str(WAUStrain)


library(neuralnet)


# Train the neural network

WAUS.nn <- neuralnet(MHT ~ MinTemp + MaxTemp + Rainfall + Evaporation + Sunshine +
WindGustSpeed +

                     WindSpeed9am + WindSpeed3pm + Pressure9am + Pressure3pm + Cloud9am +
Cloud3pm +

                     Temp9am + Temp3pm + RISK_MM, WAUStrain, hidden = 1)

plot(WAUS.nn, rep="best")

# Make predictions on the test set

WAUS.pred <- compute(WAUS.nn, WAUStest[, predictor_cols])

WAUS.predr <- round(WAUS.pred$net.result, 0)

# Create the confusion matrix

WAUStest$MHT


ANN_CM = table(observed = WAUStest$MHT, predicted = WAUS.predr[,1])

ANN_CM

TP =ANN_CM[1,1]

FP =ANN_CM[1,2]

FN =ANN_CM[2,1]

TN =ANN_CM[2,2]

acc_ANN = (TP + TN) / (TP+TN+FP+FN)

acc_ANN
```

```
accuracy_table <- data.frame("Decision Tree" = acc_DT, "Naive Bayes" = acc_NB, "Bagging" =
acc_BAG, "Boosting" = acc_BST, "Random Forest" = acc_RF, "ANN" = acc_ANN)

accuracy_table

classifiers <- colnames(accuracy_table)

accuracy_values <- unlist(accuracy_table)

barplot(accuracy_values, names.arg = classifiers, ylim = c(0, 1), ylab = "Accuracy", xlab = "Classifier",
main = "Classifiers and Accuracy")

View(accuracy_table)

#new classifier

#load the detach function to detach the library neuralnet

detach("package:neuralnet", unload = TRUE)

library(klaR)

library(psych)

library(MASS)

library(ROCR)

library(devtools)


nrow(WAUS.train)

linear <- lda(MHT~., WAUS.train)


p<- predict(linear, WAUS.train)


#ldahist(data = p$x[,1], g = WAUS.train$MHT)

p1 <- predict(linear, WAUS.train)$class

p1

LM_CM<- table(Predicted = p1, Actual = WAUS.train$MHT)

LM_CM

#calculating accuracy for random forest

TP =  LM_CM[1,1]

FP =  LM_CM[1,2]

FN =  LM_CM[2,1]

TN =  LM_CM[2,2]
```

```
acc_LM = (TP + TN) / (TP+TN+FP+FN)

acc_LM



# Create the prediction object

LMpred <- prediction(p$x[, 1], WAUS.train$MHT)


# Create the performance object

LMperf <- performance(LMpred, measure = "tpr", x.measure = "fpr")


# Plot the ROC curve

plot(LMperf,add=TRUE, main = "ROC Curve", col = "darkred")

auc_LM <- performance(LMpred, "auc")@y.values[[1]]

auc_LM


accuracy_table <- data.frame("Decision Tree" = acc_DT, "Naive Bayes" = acc_NB, "Bagging" = acc_BAG, "Boosting" = acc_BST, "Random Forest" = acc_RF, "ANN" = acc_ANN, "Linear Discriminant" = acc_LM)

accuracy_table

classifiers <- colnames(accuracy_table)

accuracy_values <- unlist(accuracy_table)

barplot(accuracy_values, names.arg = classifiers, ylim = c(0, 1), ylab = "Accuracy", xlab = "Classifier", main = "Classifiers and Accuracy")

View(accuracy_table)


legend(x = "topleft",

    col = c( "orange","blueviolet","blue","red", "darkred"), lty = 1, lwd = 1,

    legend = c('Decision Tree', 'Naive Bayes', 'Bagging','Boosting', 'Linear Discriminant'))
```