FIT3152 Assignment 2

Name: Ying Qi Mah

Student ID: 32796765

Description of Data (Refer to appendix [2] for code and output)

First, we can see that for MHT, there are 932 Yes and 928 No, which is almost the same, hence the proportion of days when it is more humid than the previous day is about the same as those where it is less humid. I have also noticed that some responses are NAs, but I will only predict yes and no in this report. For other predictors, most of them are in integer and numeric format, with some in string format, which are converted to factors. There are 12 different years, which ranges from 2008 to 2019 and has 25 NAs. 10 different locations are recorded in the 2000 rows, which are identified with integers. Minimum temperature is recorded as numeric, it ranges from -3.8 to 29.4, with a median of 11.8 and 69 NAs. Maximum temperature is recorded as numeric, it ranges from 8.3 to 45.3, with a median of 22.4 and 67 NAs. Rainfall is recorded as numeric as well, it ranges from 0 to 371, with median 0 and 111 NAs. Evaporation is recorded as numeric, it ranges from 0 to a maximum of 49, with a median of 4.4 and 1272 NAs which is very high. Sunshine is recorded as numeric, it ranges from 0 to 13.7, with a median of 7.7, and 1659 NAs which is a lot. Next, WindGustDir, WindDir9am and WundDir3pm is recorded as string and converted into factor, with 16 different directions. WindGustSpeed is recorded as int, it ranges from 11 to 113, with a median of 37 and 77 NAs. Pressure9am is recorded as numeric, it ranges from 996.4 to 1038.8, with a median of 1017.9 and 59 NAs. Pressure3pm is recorded as numeric, it ranges from 995.2 to 1036.6, with a median of 1015.5 and 59 NAs. Cloud9am s recorded as integer, with median of 6 and 982 NAs. Cloud3pm is recorded as integer, it ranges from 0 to 8, with a median of 6 and 1029 NAs. Temp9am is recorded as numeric, it ranges from 0.8 to 37.7, with a median of 16.7 and 76 NAs. Temp3pm is recorded as numeric, it ranges from 6.2 to 43.5, with a median of 20.7 and 110 NAs. Next, RainToday is recorded as string and converted to factor, with 1454 No, 435 Yes and 111 NAs. RISK_MM is recorded as numeric, it ranges from 0 to 208.5, with median of 208.5 and 104 NAs. Lastly, MHT is recorded as int , it only has 2 values, 0 and 1, and has 140 NAs. 1 is interpreted as Yes, and 0 as No.

Attributes that can be considered omitting are Evaporation, Sunshine, Cloud9am and Cloud3pm as they have very high count of NAs.

Data Preprocessing (Refer to appendix [3] for code and output)

I have converted attributes Location, RainToday, Year, and MHT to factor, this is so that the models can identify the similar factors together. Next, I have also omitted all the rows that contains NA, this is to avoid getting error when we are trying to apply the classification models. After doing all the preprocessing, I am left with 223 rows of data.

Splitting data into train and test set

I have split the data into 70% training set, and 30% test set. (Refer to appendix [4] for code and output)

Implement classification model

I have implemented all 5 classification models, decision tree, naïve bayes, bagging, boosting and random forest using the training set processed above (Refer to appendix [5] for code and output)

Confusion Matrix and Accuracy of each model

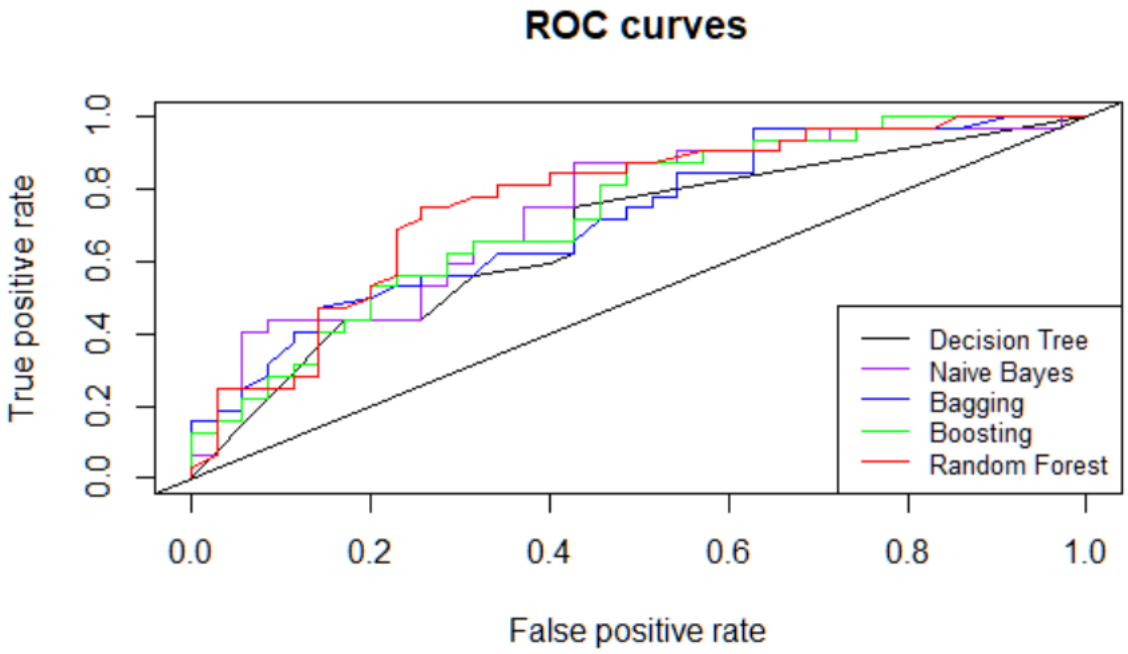
I have tested each model and obtained the confusion matrix and accuracy of each model (Refer to appendix [6] for code)

```
[1] "Decision Tree Confusion Matrix"
      predicted
actual Yes No
   Yes  18 14
   No   11 24
 Accuracy
0.6268657
[1] "Naive Bayes Confusion Matrix"
      predicted
actual Yes No
   Yes  21 11
   No   11 24
 Accuracy
0.6716418
[1] "Bagging Confusion Matrix"
         actual
predicted Yes No
      Yes  14 24
      No   18 11
 Accuracy
0.3731343
[1] "Boosting Confusion Matrix"
         actual
predicted Yes No
      Yes  11 23
      No   21 12
 Accuracy
0.3432836
[1] "Random Forest Confusion Matrix"
         actual
predicted Yes No
      Yes  15  6
      No   17 29
 Accuracy
0.6567164
```

From the result obtained above, we can observe that naïve bayes has the highest accuracy, the second is random forest, and the third is decision tree. Bagging and boosting both have accuracy lower than 0.5, this could be due to the low number of data rows, resulting in bad model.

Confidence of predicting “More Humid Tomorrow”, Constructing ROC curve and Calculate AUC

I have computed, the confidence of predicting “More Humid Tomorrow” for each case, and constructed, the ROC curve with the result (Refer to appendix [7] for code)



After constructing the curves, I have also obtained the AUC for each curve (Refer to appendix [8] for the code)

Classifier <chr>	AUC <dbl>	Accuracy <dbl>
Decision Tree	0.6705357	0.6268657
Naive Bayes	0.7187500	0.6716418
Bagging	0.7410714	0.3731343
Boosting	0.7294643	0.3432836
Random Forest	0.7678571	0.6567164

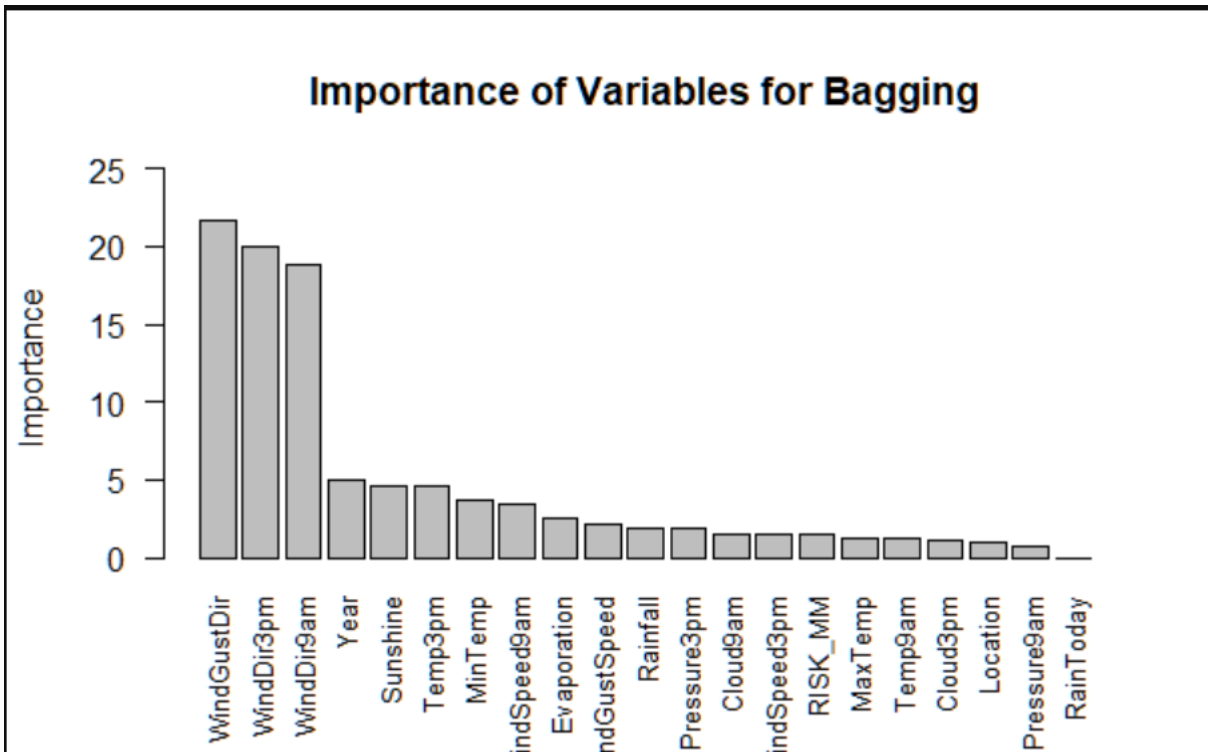
From the summary table above, we can see that Naïve Bayes has the highest accuracy, which mean it is better in predicting this testing data set. Random forest has the highest AUC, which can be interpreted as it performs better at distinguishing Yes and No classes. We can also observe that the accuracy of random forest is just slightly lower than naïve bayes, which is only a difference of 0.02. As a conclusion, random forest is the best classifier among the 5 of them.

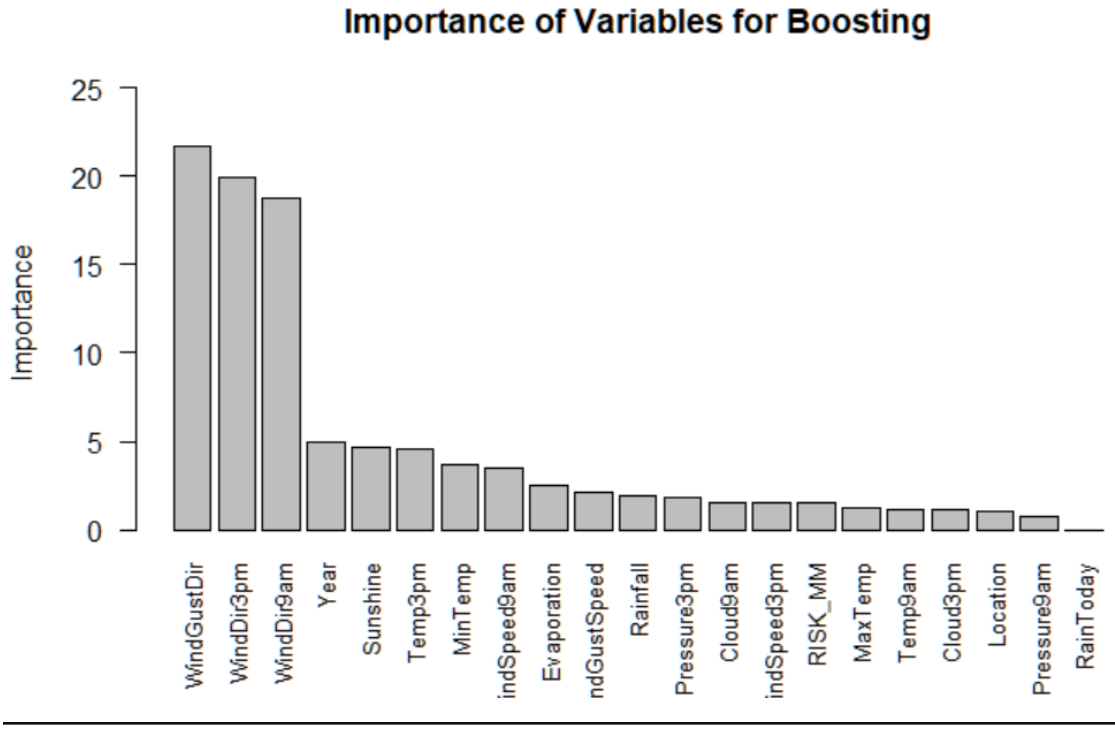
Determining Most Important Variables

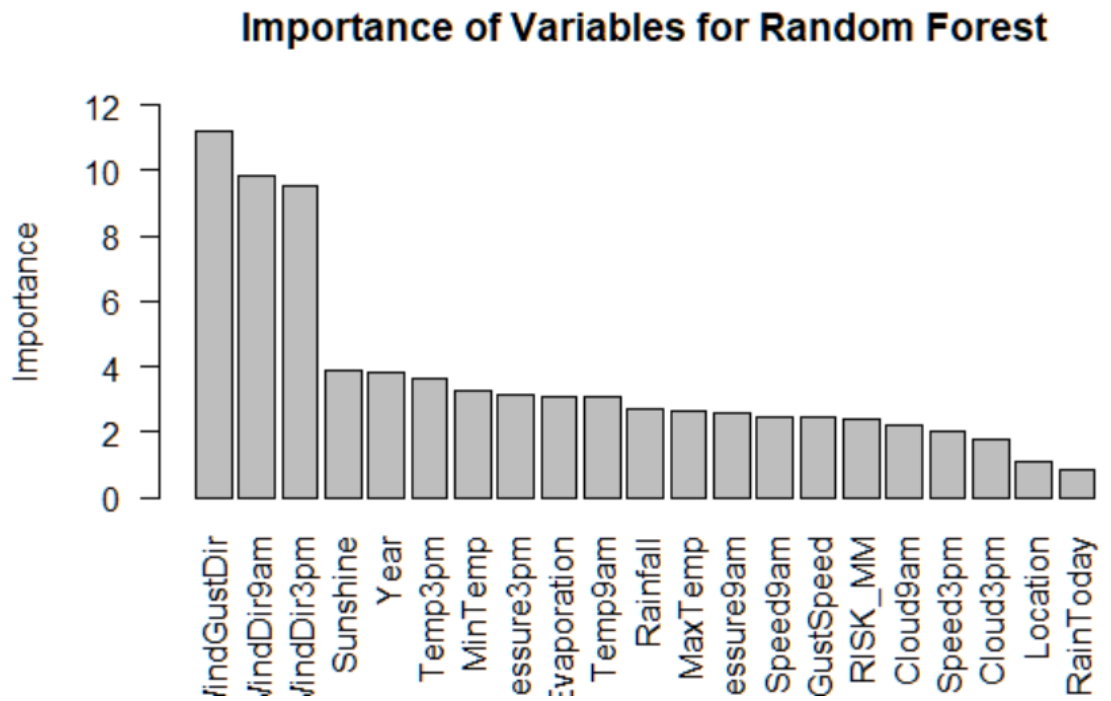
To determine the most important variables, I have obtained the importance of each variable in each model, and plot them side by side in order to identify their relative importance. (Refer to appendix [9] for the code)

Variables Used in Decision Tree

```
Classification tree:
tree(formula = MHT ~ ., data = WAUS.train)
Variables actually used in tree construction:
 [1] "WindGustDir"   "Cloud9am"      "WindDir9am"    "WindSpeed9am"
 [5] "WindGustSpeed" "WindDir3pm"    "MaxTemp"       "Year"
 [9] "Location"      "MinTemp"
Number of terminal nodes:  18
Residual mean deviance:  0.3937 = 54.33 / 138
Misclassification error rate: 0.08333 = 13 / 156
```



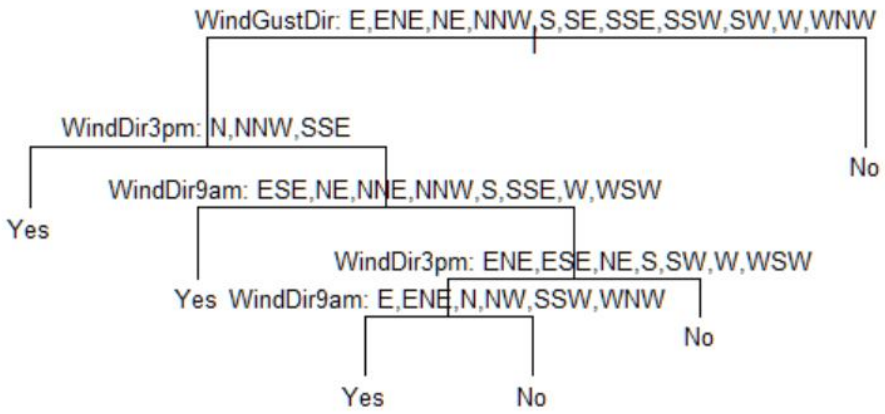




From the result plotted above, we can see that there are 3 variables that always ranked the highest, which are WindGustDir, WinDir9am and WindDir3pm. With this information, we can say that these 3 are the most important variables in our prediction. In terms of the variables that could be omitted, Location, Cloud3pm and RainToday are the three common variables that have lowest importance in the three plots. Hence, they could be omitted as they have very little effect on the performance.

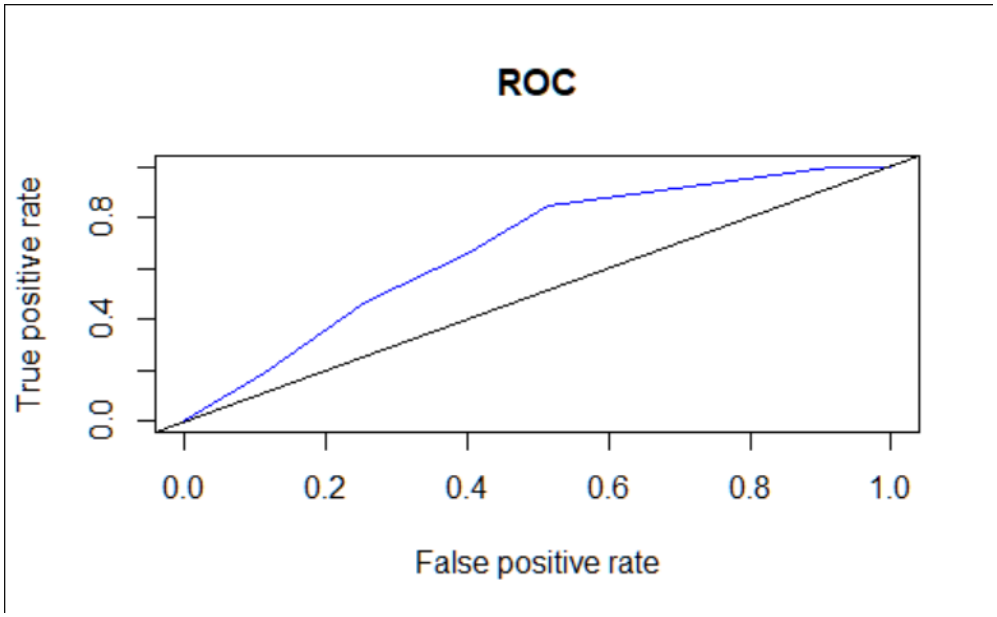
Create a Simple classifier.

To create a very simple classifier, I have chosen a subset of the training data set, which only contains the column WindGustDir, WindDir9am, WindDir3pm and MHT. This is because these are the three most important variables from the last part. Then, I fit this dataset into a tree. After obtaining the tree, I pruned it down to make it more easily interpretable. (Refer to appendix[9]) Here is the resulting tree:



It can be used easily as it has only a few splits in the tree.

Now lets evaluate the performance of this model, I have calculated the accuracy and AUC of the model. I have also plotted the ROC curve for this model. (Refer to appendix[10] for the code)




```
  Accuracy
0.6268657
[1] "AUC"
[1] 0.678125
```

From the result about, we can see that the performance of this model is even better than the decision tree we made in question 3, as it has same accuracy and higher AUC.

Create Best tree Classifier

To create the best classifier, I aim to improve the random forest model by removing some non-important columns from the dataset to create a larger dataset. However, after numerous tries of sub-setting the columns, I have found that not removing any columns provides the best prediction.

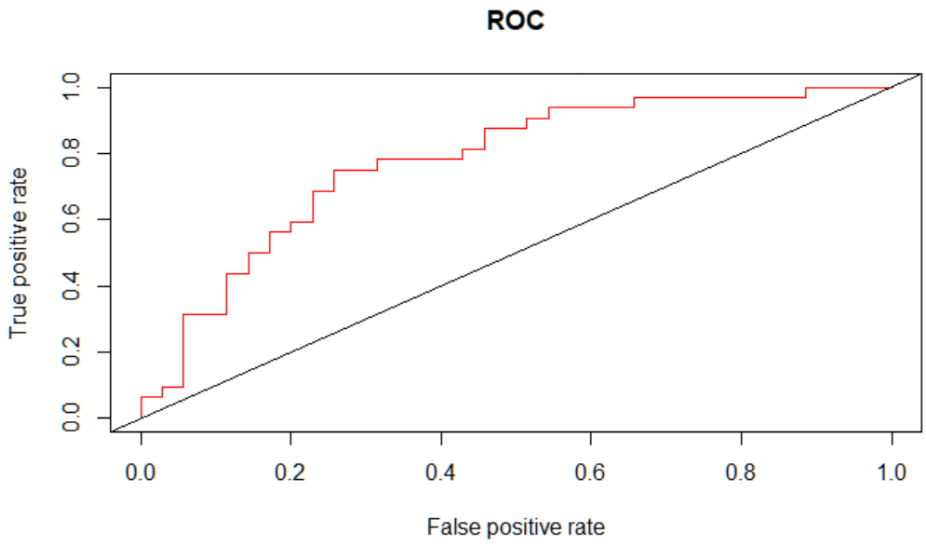
I have computed a cross validation of random forest, the result is shown below. (Refer to appendix[11] for the code)

```
$n.var
[1] 21 10  5  3  1

$error.cv
       21        10         5         3         1
0.4166667 0.4679487 0.4935897 0.5320513 0.4807692
```

The result of cross validation shows that taking all the columns provides less error, hence, I will not be removing attributes. As I am not removing attributes, I tried to tune the parameters of random forest. The result after tuning is shown below. (Refer to appendix [12] for the code)

```
[1] "AUC of random forest"
[1] 0.7758929
 Accuracy
0.7014925
```



I have adjusted the parameter of random forest by changing the number of tree to 1000, and number of features considered at each split to 4. By doing so, I was able to improve the model by a little bit, and obtained the better model than all 5 I have created earlier. The reason that I chose to improve the random forest model among all 5 is because it is already the best among all 5, also it does not overfit, and it is not sensitive to outliers.

Implementing Artificial Neural Network Classifier

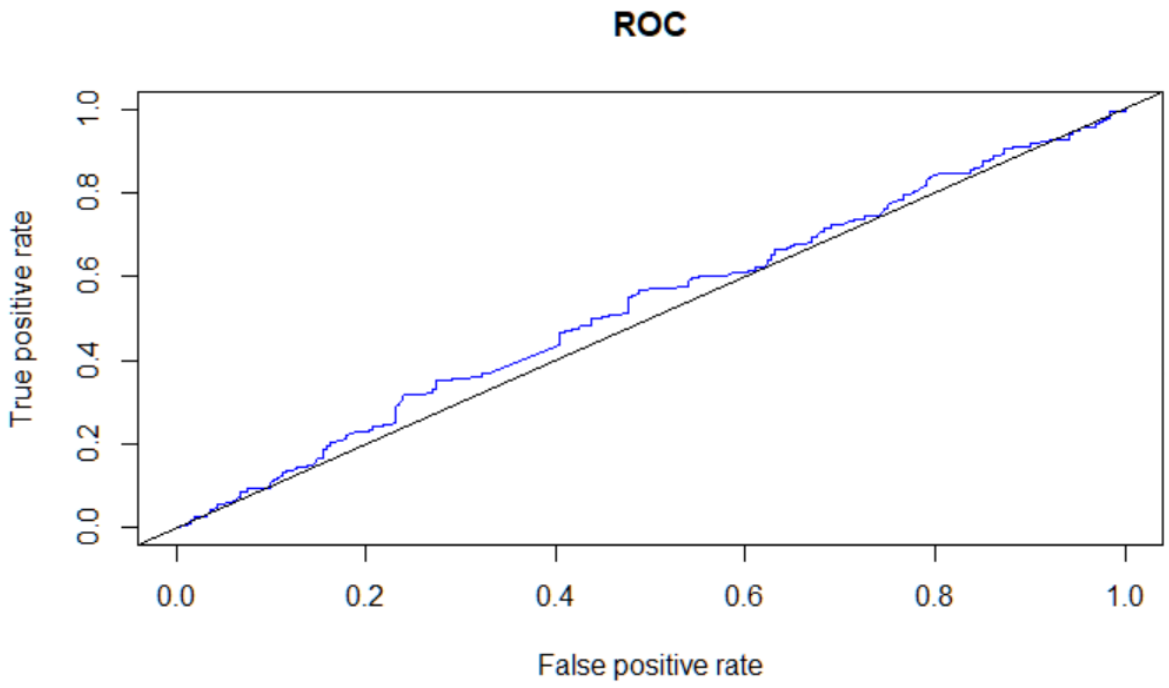
To implement the artificial neural network classifier, I have taken the top 3 predictors from previous insights to fit them in the classifier. In order to fit them, the pre-processing required are converting all the factors into numeric, hence, I have converted all of the factor attributes into numeric. (Refer to appendix[13] for the code) Then, I have fitted the data into the model and tested the model using test data set. (Refer to appendix [14] for the code) The confusion matrix and accuracy for the prediction is shown below. (Refer to appendix[15] for the code):

```
Confusion Matrix and Statistics

       predicted
actual Yes  No
   Yes 171  66
   No  194  58

               Accuracy : 0.4683
```

And lastly, I have plotted out the ROC curve for the classifier and calculated the AUC. (Refer to appendix[16] for the code)



AUC:

```
[1] 0.5274513
```

In conclusion, we can see that the ANN model does not perform better than the best model we obtained before, it has lower AUC than all others but have higher accuracy than bagging and boosting. This could be due to the dataset is biased, or the number of data rows is too few.

Fit a New Classifier

The new classifier that I have chosen to fit is SVM classifier, which stands for support vector machine. The svm classifier that I have used is included in the e1071 package. Here's a link to the documentation of the svm classifer https://www.rdocumentation.org/packages/e1071/versions/1.7-13/topics/svm. SVM is a supervised learning model, it classifies items by separating objects using a hyperplane, and then classifies objects that are on the same side as the same class.

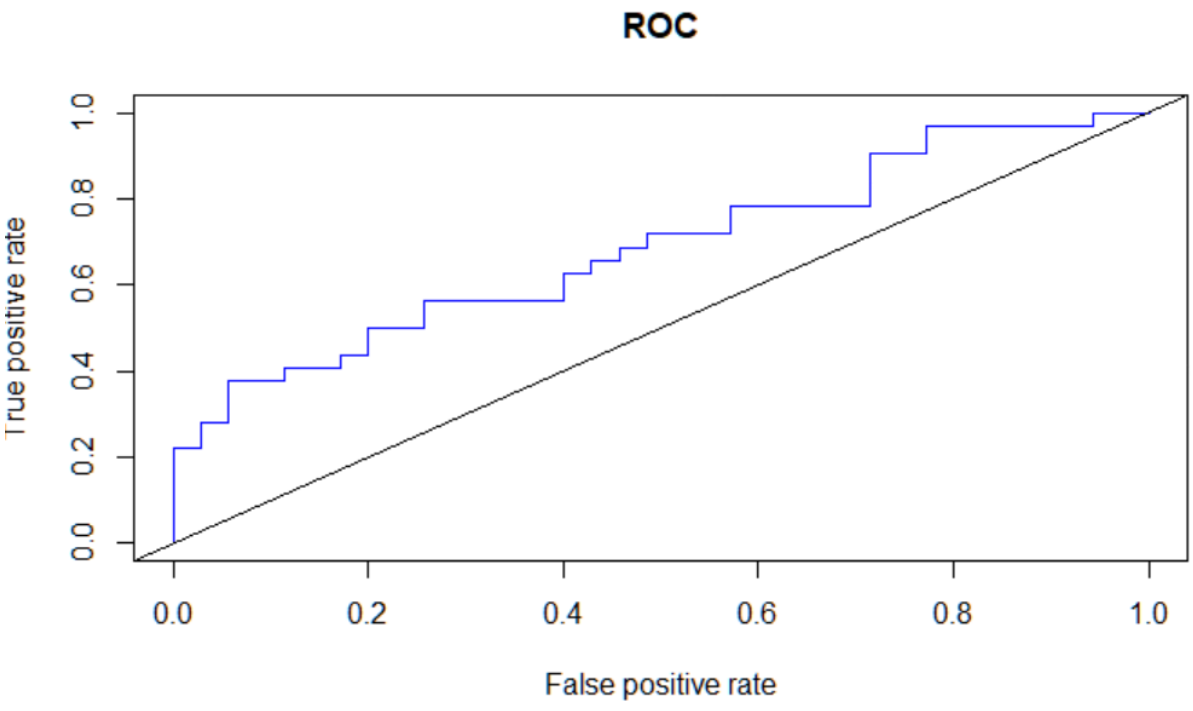
I have fitted my training data into the SVM model, and obtained the confusion matrix by testing the classifier with the test data. (Refer to appendix[17] for the code)

```
Confusion Matrix and Statistics

          predicted
actual    Yes No
   Yes     22 10
   No      17 18

                  Accuracy : 0.597
```

Next, I have constructed the ROC curve and obtained the AUC value. (Refer to appendix[18] for the code)




```
[1] "AUC"
[1] 0.6848214
```

As a result, we can see that the AUC is 0.684 and accuracy is 0.597, the performance of this classifier is not better than the best classifier that we have had, but the accuracy is higher than some of them. I believe that it can be improved after tuning some parameters.

Appendix

[1]
```
library(ROCR)
library(e1071)
library(rpart)
library(ggplot2)
library(caret)
library(adabag)
library(randomForest)
library(tree)
rm(list = ls())
WAUS <- read.csv("HumidPredict2023D.csv",stringsAsFactors = TRUE)
L <- as.data.frame(c(1:49))
set.seed(32796765) # 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
WAUS2 = WAUS
```

[2]
```
str(WAUS)
summary(WAUS)
```

```
'data.frame':   2000 obs. of  22 variables:
 $ Year         : Factor w/ 12 levels "2008","2009",..: 10 NA 8 7 5 5 4 8 6
10  ...
 $ Location     : Factor w/ 10 levels "3","11","12",..: 6 9 1 6 2 2 1 5 4 10
 ...
 $ MinTemp      : num  12.6 3.2 19.7 7.9 7.3 7.4 11.3 7.3 25.1 18.6  ...
 $ MaxTemp      : num  20.6 18.4 30.7 13 21.3 25.3 24.7 10.8 39.4 23.4  ...
 $ Rainfall     : num  0 0 10.2 0 0 0 0 3.2 0 24.6  ...
 $ Evaporation  : num  NA 9 NA 0.8 3.4 NA NA NA 5.8 NA  ...
 $ Sunshine     : num  NA NA NA 1.8 NA NA NA NA NA NA  ...
 $ WindGustDir  : Factor w/ 16 levels "E","ENE","ESE",..: 6 15 15 3 15 13 16
4 8 12  ...
 $ WindGustSpeed: int  37 35 35 24 19 22 35 37 33 59  ...
 $ WindDir9am   : Factor w/ 16 levels "E","ENE","ESE",..: 4 8 8 10 6 1 2 6 8
13  ...
 $ WindDir3pm   : Factor w/ 16 levels "E","ENE","ESE",..: 12 15 NA 3 3 10 9
1 5 12  ...
 $ WindSpeed9am : int  9 15 15 11 13 9 6 9 15 17  ...
 $ WindSpeed3pm : int  2 17 19 9 7 7 11 17 11 39  ...
 $ Pressure9am  : num  1022 1022 1014 1027 1020  ...
 $ Pressure3pm  : num  1022 1018 1013 1026 NA  ...
 $ Cloud9am     : int  8 2 NA 8 2 0 NA NA 1 8  ...
 $ Cloud3pm     : int  8 1 NA 8 1 0 1 NA 4 8  ...
 $ Temp9am      : num  18.1 8.7 23.2 10.3 14.5 19.1 17.9 8.7 31.9 19.3  ...
 $ Temp3pm      : num  16.2 18 29.6 11.8 20.7 24.5 23.6 10.1 39 22.3  ...
 $ RainToday    : Factor w/ 2 levels "No","Yes": 1 1 2 1 1 1 1 2 1 2  ...
 $ RISK_MM      : num  43.6 0 0 0 0 0 0 11.2 0 1.2  ...
 $ MHT          : Factor w/ 2 levels "Yes","No": 2 1 1 2 1 1 2 1 NA 1  ...
```

```
     Year          Location       MinTemp          MaxTemp
2018   :239   11      :252   Min.   :-3.80   Min.   : 8.30
2013   :226   48      :235   1st Qu.: 7.90   1st Qu.:18.00
2014   :222   46      :231   Median :11.80   Median :22.40
2017   :204   3       :219   Mean   :12.16   Mean   :23.23
2019   :202   34      :214   3rd Qu.:16.50   3rd Qu.:27.50
(Other):882   35      :212   Max.   :29.40   Max.   :45.30
NA's   : 25   (Other):637   NA's   :69      NA's   :67
   Rainfall        Evaporation        Sunshine       WindGustDir
Min.   :  0.00   Min.   : 0.000   Min.   : 0.000   WNW    : 171
1st Qu.:  0.00   1st Qu.: 2.400   1st Qu.: 4.300   SSW    : 148
Median :  0.00   Median : 4.400   Median : 7.700   SE     : 146
Mean   :  2.78   Mean   : 5.454   Mean   : 7.009   NE     : 145
3rd Qu.:  0.80   3rd Qu.: 7.200   3rd Qu.: 9.900   S      : 142
Max.   :371.00   Max.   :49.000   Max.   :13.700   (Other):1173
NA's   :111      NA's   :1272     NA's   :1659     NA's   :  75
WindGustSpeed     WindDir9am     WindDir3pm    WindSpeed9am
Min.   : 11.0   NW     : 142   SE     : 167   Min.   : 0.00
1st Qu.: 30.0   WNW    : 138   SSE    : 156   1st Qu.: 7.00
Median : 37.0   SW     : 136   ESE    : 150   Median :13.00
Mean   : 39.7   SSW    : 127   NE     : 150   Mean   :13.85
3rd Qu.: 48.0   N      : 119   S      : 150   3rd Qu.:19.00
Max.   :113.0   (Other):1142   (Other):1169   Max.   :52.00
NA's   :77      NA's   : 196   NA's   :  58   NA's   :52
 WindSpeed3pm    Pressure9am      Pressure3pm        Cloud9am
Min.   : 0.00   Min.   : 996.4   Min.   : 995.2   Min.   :0.000
1st Qu.:13.00   1st Qu.:1013.4   1st Qu.:1010.7   1st Qu.:2.000
Median :17.00   Median :1017.9   Median :1015.5   Median :6.000
Mean   :18.68   Mean   :1018.0   Mean   :1015.6   Mean   :5.026
3rd Qu.:24.00   3rd Qu.:1022.8   3rd Qu.:1020.4   3rd Qu.:8.000
Max.   :63.00   Max.   :1038.8   Max.   :1036.6   Max.   :8.000
NA's   :46      NA's   :56       NA's   :59       NA's   :982
   Cloud3pm        Temp9am        Temp3pm      RainToday
Min.   :0.000   Min.   : 0.80   Min.   : 6.20   No  :1454
1st Qu.:2.000   1st Qu.:12.70   1st Qu.:16.60   Yes : 435
Median :6.000   Median :16.70   Median :20.70   NA's: 111
Mean   :4.993   Mean   :16.92   Mean   :21.46
3rd Qu.:8.000   3rd Qu.:20.90   3rd Qu.:25.40
Max.   :8.000   Max.   :37.70   Max.   :43.50
NA's   :1029    NA's   :76      NA's   :110
   RISK_MM          MHT
Min.   :  0.000   Yes :932
1st Qu.:  0.000   No  :928
Median :  0.000   NA's:140
Mean   :  2.504
3rd Qu.:  1.000
Max.   :208.500
NA's   :104
```

[3] WAUS$Location = as.factor(WAUS$Location)

WAUS$RainToday = as.factor(WAUS$RainToday)

```
WAUS$Year = as.factor(WAUS$Year)
WAUS$MHT = as.factor(ifelse(WAUS$MHT==1,"Yes","No"))
levels(WAUS$MHT) = c("Yes","No")
WAUS = na.omit(WAUS)
dim(WAUS)
```

```
[1] 223  22
```

[4]
```
set.seed(32796765) #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,]
```

[5]
```
set.seed(32796765)
WAUS.tree = tree(MHT~.,data = WAUS.train)

set.seed(32796765)
WAUS.bayes = naiveBayes(MHT~.,data = WAUS.train)

set.seed(32796765)
WAUS.bag = bagging(MHT~.,data=WAUS.train)

set.seed(32796765)
WAUS.boost = boosting(MHT~.,data=WAUS.train)

set.seed(32796765)
WAUS.rf = randomForest(MHT~.,data=WAUS.train)
```

[6]
```
WAUS.treePredict = predict(WAUS.tree, WAUS.test,type="class")
tree.c = confusionMatrix(table(actual = WAUS.test$MHT, predicted = WAUS.treePredict))
WAUS.tree.accuracy=tree.c$overall[1]

WAUS.bayesPredict = predict(WAUS.bayes,WAUS.test)
bayes.c=confusionMatrix(table(actual = WAUS.test$MHT,predicted= WAUS.bayesPredict))
```

```
WAUS.bayes.accuracy=bayes.c$overall[1]



WAUS.bagPred = predict.bagging(WAUS.bag,WAUS.test)

WAUS.bagPred$class = as.factor(WAUS.bagPred$class)

levels(WAUS.bagPred$class) = c("Yes","No")

bag.c= confusionMatrix(table(predicted =WAUS.bagPred$class,actual = WAUS.test$MHT))

WAUS.bag.accuracy=bag.c$overall[1]



WAUS.boostPred = predict.boosting(WAUS.boost, newdata = WAUS.test)

WAUS.boostPred$class = as.factor(WAUS.boostPred$class)

levels(WAUS.boostPred$class) = c("Yes","No")

boost.c = confusionMatrix(table(predicted =WAUS.boostPred$class,actual =
WAUS.test$MHT))

WAUS.boost.accuracy=boost.c$overall[1]



WAUS.rfPred = predict(WAUS.rf, WAUS.test)

rf.c = confusionMatrix(table(predicted = WAUS.rfPred, actual = WAUS.test$MHT))

WAUS.rf.accuracy=rf.c$overall[1]

print("Decision Tree Confusion Matrix")

print(tree.c$table)

WAUS.tree.accuracy

print("Naive Bayes Confusion Matrix")

print(bayes.c$table)

WAUS.bayes.accuracy

print("Bagging Confusion Matrix")

print(bag.c$table)

WAUS.bag.accuracy
```

```
print("Boosting Confusion Matrix")
print(boost.c$table)
WAUS.boost.accuracy

print("Random Forest Confusion Matrix")
print(rf.c$table)
WAUS.rf.accuracy
```

[7]
```
plot.new()
WAUS.pred.tree = predict(WAUS.tree,WAUS.test,type="vector")
WAUS.tree.pred = prediction(WAUS.pred.tree[,1],WAUS.test$MHT)
WAUS.tree.perf = performance(WAUS.tree.pred,"tpr","fpr")
WAUS.tree.auc = as.numeric(performance(WAUS.tree.pred,"auc")@y.values)
print("AUC of decision tree")
WAUS.tree.auc
plot(WAUS.tree.perf, main = "ROC curves")

WAUS.pred.bayes = predict(WAUS.bayes,WAUS.test,type="raw")
WAUS.bayes.pred = prediction(WAUS.pred.bayes[,1],WAUS.test$MHT)
WAUS.bayes.perf = performance(WAUS.bayes.pred,"tpr","fpr")
WAUS.bayes.auc = as.numeric(performance(WAUS.bayes.pred,"auc")@y.values)
print("AUC of naive bayes")
WAUS.bayes.auc
plot(WAUS.bayes.perf,add=TRUE,col="purple")

WAUS.bag.pred = prediction(WAUS.bagPred$prob[,1],WAUS.test$MHT)
WAUS.bag.perf = performance(WAUS.bag.pred,"tpr","fpr")
WAUS.bag.auc = as.numeric(performance(WAUS.bag.pred,"auc")@y.values)
print("AUC of bagging")
WAUS.bag.auc
plot(WAUS.bag.perf,add=TRUE,col="blue")
```

```
WAUS.boost.pred = prediction(WAUS.boostPred$prob[,1],WAUS.test$MHT)

WAUS.boost.perf = performance(WAUS.boost.pred,"tpr","fpr")

WAUS.boost.auc = as.numeric(performance(WAUS.boost.pred,"auc")@y.values)

print("AUC of boosting")

WAUS.boost.auc

plot(WAUS.boost.perf,add=TRUE,col="green")


WAUS.pred.rf = predict(WAUS.rf,WAUS.test,type="prob")

WAUS.rf.pred = prediction(WAUS.pred.rf[,1],WAUS.test$MHT)

WAUS.rf.perf = performance(WAUS.rf.pred,"tpr","fpr")

WAUS.rf.auc = as.numeric(performance(WAUS.rf.pred,"auc")@y.values)

print("AUC of random forest")

WAUS.rf.auc

plot(WAUS.rf.perf, add=TRUE,col = "red")

legend("bottomright",legend=c("Decision Tree","Naive
Bayes","Bagging","Boosting","Random
Forest"),col=c("black","purple","blue","green","red"),lty = 1,cex=0.8)

abline(0,1)
```

[8]
```
summaryDf = data.frame(c("Decision Tree","Naive Bayes","Bagging","Boosting","Random
Forest"),c(WAUS.tree.auc,WAUS.bag.auc,WAUS.bayes.auc,WAUS.boost.auc,WAUS.rf.auc
),c(WAUS.tree.accuracy,WAUS.bayes.accuracy,WAUS.bag.accuracy,WAUS.boost.accuracy,
WAUS.rf.accuracy))

colnames(summaryDf) = c("Classifier","AUC","Accuracy")

summaryDf
```

[9]
```
WAUS.simple.tree =
tree(MHT~.,data=WAUS.train[,c("WindGustDir","WindDir9am","WindDir3pm","MHT")])

WAUS.tree.pruned = prune.misclass(WAUS.simple.tree, best = 3)

plot(WAUS.tree.pruned)

text(WAUS.tree.pruned, pretty=0, cex=0.8)
```

[10]
```
WAUS.pruned.predict = predict(WAUS.tree.pruned, WAUS.test, type = "class")

c = confusionMatrix(table(actual = WAUS.test$MHT, predicted = WAUS.pruned.predict))

c$overall[1]

WAUS.pred.tree.pruned = predict(WAUS.tree.pruned,WAUS.test,type="vector")

WAUS.tree.pred.pruned = prediction(WAUS.pred.tree.pruned[,1],WAUS.test$MHT)
```

```
WAUS.tree.perf.pruned = performance(WAUS.tree.pred.pruned,"tpr","fpr")
plot(WAUS.tree.perf.pruned, col="blue", main = "ROC")
abline(0,1)
print("AUC")
as.numeric(performance(WAUS.tree.pred.pruned,"auc")@y.values)
```

[11]
```
WAUS2 = na.omit(WAUS3)
WAUS2$MHT = as.factor(ifelse(WAUS2$MHT==1,"Yes","No"))
WAUS2$Year = as.factor(WAUS2$Year)
WAUS2$Location = as.factor(WAUS2$Location)
WAUS$RainToday = as.factor(WAUS$RainToday)
levels(WAUS2$MHT) = c("Yes","No")
set.seed(32796765) #Student ID as random seed
train.row = sample(1:nrow(WAUS2), 0.7*nrow(WAUS2))
WAUS2.train = WAUS2[train.row,]
WAUS2.test = WAUS2[-train.row,]
summary(WAUS2)
W.cvforest = rfcv(WAUS2.train[,1:21], WAUS2.train[,22], cv.fold=10, scale="log",
        step=0.5, mtry=function(p) max(1, floor(sqrt(p))), recursive=FALSE)
W.cvforest
```

[12]
```
set.seed(32796765)
WAUS2.rf = randomForest(MHT~.,data=WAUS2.train,ntree=1000,mtry=4)
WAUS2.rfPred = predict(WAUS2.rf, WAUS2.test)
rf2.c = confusionMatrix(table(predicted = WAUS2.rfPred, actual = WAUS2.test$MHT))
WAUS2.rf.accuracy=rf2.c$overall[1]
WAUS2.pred.rf = predict(WAUS2.rf,WAUS2.test,type="prob")
WAUS2.rf.pred = prediction(WAUS2.pred.rf[,1],WAUS2.test$MHT)
WAUS2.rf.perf = performance(WAUS2.rf.pred,"tpr","fpr")
WAUS2.rf.auc = as.numeric(performance(WAUS2.rf.pred,"auc")@y.values)
print("AUC of random forest")
WAUS2.rf.auc
WAUS2.rf.accuracy
plot.new()
```

```
plot(WAUS2.rf.perf,col="red",main="ROC")
abline(0,1)
```

[13]
```
library(neuralnet)
WAUS2 = subset(WAUS3,
select=c("WindGustDir","WindDir9am","WindDir3pm","MHT"))
WAUS2 = na.omit(WAUS2)
WAUS2$MHT = as.numeric(WAUS2$MHT)
WAUS2$WindGustDir = as.numeric(WAUS2$WindGustDir)
WAUS2$WindDir3pm = as.numeric(WAUS2$WindDir3pm)
WAUS2$WindDir9am = as.numeric(WAUS2$WindDir9am)
set.seed(32796765) #Student ID as random seed
train.row = sample(1:nrow(WAUS2), 0.7*nrow(WAUS2))
WAUS2.train = WAUS2[train.row,]
WAUS2.test = WAUS2[-train.row,]
```

[14]
```
set.seed(32796765)
WAUS.nn = neuralnet(MHT~., WAUS2.train, hidden=7)
WAUS.nn.pred = compute(WAUS.nn, WAUS2.test)
WAUS.nn.pred2 = as.data.frame(ifelse(WAUS.nn.pred$net.result>0.5,1,0))
```

[15]
```
WAUS2.predicted = as.factor(ifelse(WAUS.nn.pred2$V1==1,"Yes","No"))
levels(WAUS2.predicted) = c("Yes","No")
WAUS2.test$MHT = as.factor(ifelse(WAUS2.test$MHT==1,"Yes","No"))
levels(WAUS2.test$MHT) = c("Yes","No")
t = table(actual = WAUS2.test$MHT, predicted = WAUS2.predicted)
confusionMatrix(t)
```

[16]
```
library(ROCR)
WAUS2.rf.prob <- as.numeric(WAUS.nn.pred$net.result[,1])
WAUS2.rf.pred = ROCR::prediction(WAUS2.rf.prob,WAUS2.test$MHT)
WAUS2.rf.perf = performance(WAUS2.rf.pred,"tpr","fpr")
WAUS2.rf.auc = as.numeric(performance(WAUS2.rf.pred,"auc")@y.values)
plot.new()
plot(WAUS2.rf.perf, col="blue",main = "ROC")
abline(0,1)
WAUS2.rf.auc
```

[17]
```
set.seed(32796765)
WAUS.svm = svm(MHT~.,data=WAUS.train,probability = TRUE)
WAUS.svm.pred = predict(WAUS.svm, WAUS.test)
svm.c = confusionMatrix(table(actual = WAUS.test$MHT, predicted = WAUS.svm.pred))
svm.c
```

[18]
```
WAUS.pred.svm = predict(WAUS.svm,WAUS.test, decision.values = TRUE)
WAUS.pred.svm<-attr(WAUS.pred.svm,"decision.values")
WAUS.svm.prediction = ROCR::prediction(WAUS.pred.svm,WAUS.test$MHT)
WAUS.svm.perf = performance(WAUS.svm.prediction,"tpr","fpr")
plot(WAUS.svm.perf, col="blue", main = "ROC")
abline(0,1)
print("AUC")
as.numeric(performance(WAUS.svm.prediction,"auc")@y.values[[1]])
```