Assignment2

Student ID: 31240291

Student Name: LIANG DIZHEN

```
> rm(list = ls())
> library(e1071)
> library(rpart)
> library(ROCR)
> library(tree)
> library(randomForest)
> library(adabag)
> library(caret)
> setwd("C:/Users/DavidL/OneDrive/CS/FIT3152/A2")
> WAUS <- read.csv("HumidPredict2023D.csv")
> L <- as.data.frame(c(1:49))
> set.seed(31240291) # Your Student ID is the random seed
> L <- L[sample(nrow(L), 10, replace = FALSE),] # sample 10 location
s
> WAUS <- WAUS[(WAUS$Location %in% L),]
> WAUS <- WAUS[sample(nrow(WAUS), 2000, replace = FALSE),] # sample
2000 rows
```

Q1

```
> library(dplyr)
> #length of more humid
> WAUS_01 = aggregate(WAUS[2], WAUS[ncol(WAUS)],length)
> WAUS_01
  MHT Location
1   0      987
2   1      881
> WAUS_proportion = WAUS_01[2,2]/sum(WAUS_01[2])
> WAUS_proportion #proportion = 0.4837 WAUS: 0.4716
[1] 0.472
```

By just using aggregate function by the factor level of MHT, 47.2% of the days, tomorrow is more humid than today in the sample data. 52.8% of the days, tomorrow is less humid than today.

```
> #get real_values attributes, no categorical value
> col_real = sapply(WAUS, is.numeric)
> #just numeric value
> WAUS_num_pre = WAUS[col_real]
> #automatically ignore NA
> summary(WAUS_num_pre)
      Year         Location        MinTemp         MaxTemp          R
ainfall        Evaporation        Sunshine
 Min.   :2008   Min.   : 1.00   Min.   :-3.30   Min.   : 8.30   Min.
   : 0.000   Min.   : 0.000   Min.   : 0.00
 1st Qu.:2011   1st Qu.: 7.00   1st Qu.: 9.00   1st Qu.:19.40   1st
Qu.: 0.000   1st Qu.: 3.200   1st Qu.: 5.00
 Median :2014   Median :19.00   Median :14.10   Median :24.50   Medi
an : 0.000   Median : 5.200   Median : 8.90
 Mean   :2014   Mean   :18.76   Mean   :13.79   Mean   :25.04   Mean
   : 2.241   Mean   : 6.073   Mean   : 7.78
 3rd Qu.:2017   3rd Qu.:28.00   3rd Qu.:18.90   3rd Qu.:30.50   3rd
Qu.: 0.400   3rd Qu.: 7.600   3rd Qu.:10.80
```

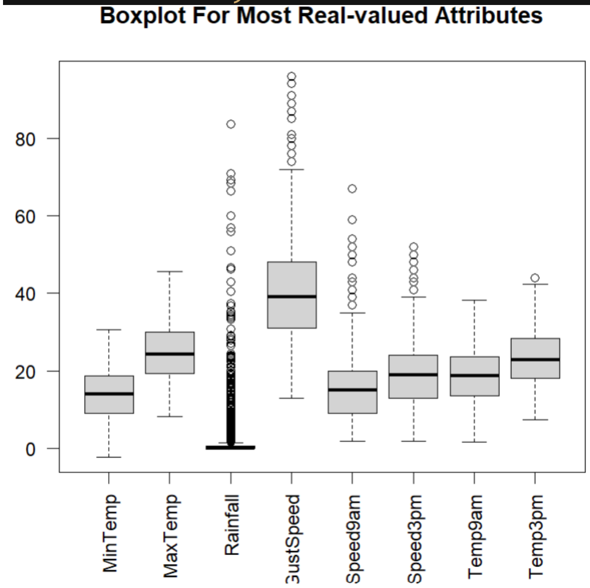
```
 Max.   :2019   Max.   :43.00   Max.   :30.70   Max.   :45.70   Max.
   :83.600   Max.   :81.600   Max.   :13.80
 NA's   :22                     NA's   :73      NA's   :67      NA's
   :127      NA's   :533      NA's   :787
 WindGustSpeed     WindSpeed9am    WindSpeed3pm    Pressure9am
Pressure3pm       Cloud9am        Cloud3pm
 Min.   : 13.00   Min.   : 0.00   Min.   : 0.00   Min.   : 991.2   M
in.   : 989.9   Min.   :0.000   Min.   :0.000
 1st Qu.: 31.00   1st Qu.: 9.00   1st Qu.:13.00   1st Qu.:1012.9   1
st Qu.:1010.1   1st Qu.:1.000   1st Qu.:2.000
 Median : 39.00   Median :15.00   Median :19.00   Median :1017.3   M
edian :1014.5   Median :4.000   Median :5.000
 Mean   : 40.54   Mean   :15.52   Mean   :19.29   Mean   :1017.5   M
ean   :1014.8   Mean   :4.135   Mean   :4.476
 3rd Qu.: 48.00   3rd Qu.:20.00   3rd Qu.:24.00   3rd Qu.:1022.1   3
rd Qu.:1019.5   3rd Qu.:7.000   3rd Qu.:7.000
 Max.   :100.00   Max.   :67.00   Max.   :52.00   Max.   :1036.9   M
ax.   :1035.0   Max.   :8.000   Max.   :8.000
 NA's   :45       NA's   :20      NA's   :20      NA's   :51       N
A's   :40       NA's   :577     NA's   :577
    Temp9am         Temp3pm         RISK_MM             MHT
 Min.   : 1.50   Min.   : 7.50   Min.   :  0.000   Min.   :0.0000
 1st Qu.:13.70   1st Qu.:18.10   1st Qu.:  0.000   1st Qu.:0.0000
 Median :18.70   Median :22.80   Median :  0.000   Median :0.0000
 Mean   :18.78   Mean   :23.33   Mean   :  2.309   Mean   :0.4716
 3rd Qu.:23.70   3rd Qu.:28.27   3rd Qu.:  0.400   3rd Qu.:1.0000
 Max.   :38.30   Max.   :43.90   Max.   :206.200   Max.   :1.0000
 NA's   :72      NA's   :118     NA's   :136       NA's   :132
```

By using combination of sapply, lapply, is.numeric, the real-valued attributes would be extracted and summarised with the summary function.

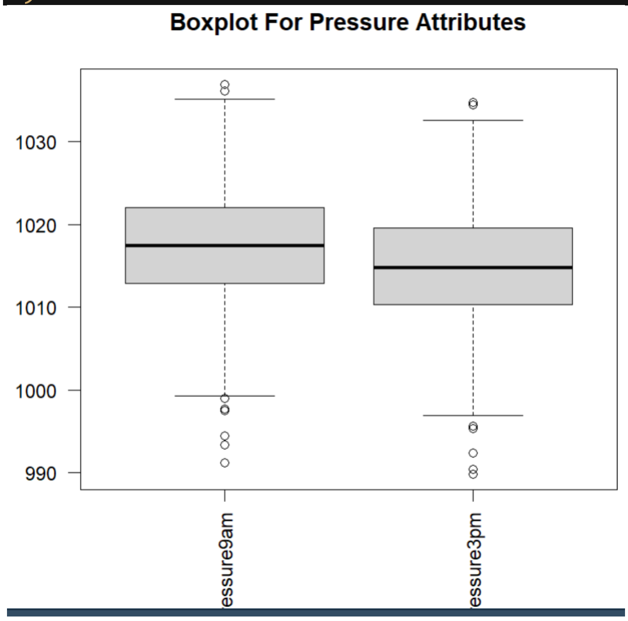
From the summary, there are NAs in most of the columns and those columns will be handled later. Noteworthily, the Location and Year are the categorical variables which mean the statistical distribution is not applicable to them. Those will be pre-processed

Statistical Distribution

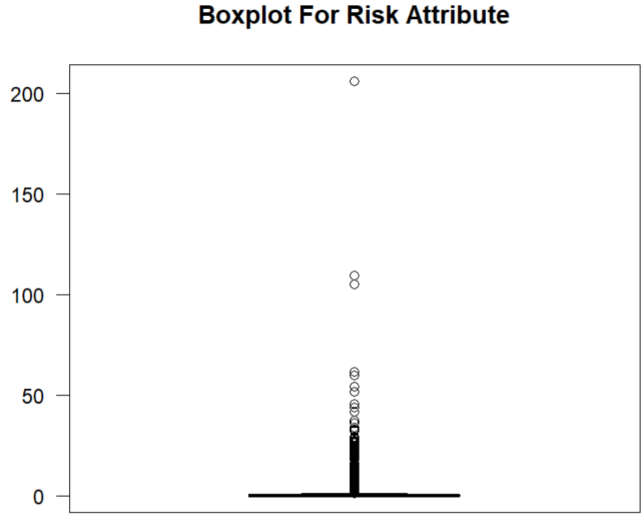
```
> #omit pressure9am and pressure3pm and RISK_MM, values too big for
comparison
> WAUS_NoPressure = WAUS_num_pre %>% select(c(-Pressure9am, -Pressur
e3pm, - RISK_MM))
> boxplot(WAUS_NoPressure, las=2, main = "Boxplot For Most Real-valu
ed Attributes")
```




```
> WAUS_Press = WAUS_num_pre %>% select(c(Pressure9am, Pressure3pm))
> boxplot(WAUS_Press, las=2, main = "Boxplot For Pressure Attributes
")
```



```
> WAUS_Risk = WAUS_num_pre %>% select(RISK_MM)
> boxplot(WAUS_Risk, las=2, main = "Boxplot For Risk Attribute")
```




```
> #reomove the columns that have too many NAs which is not worth for
 analysing
> WAUS$Evaporation <- NULL
> WAUS$Sunshine <- NULL
> WAUS$Cloud9am <- NULL
> WAUS$Cloud3pm <- NULL
> #omit the rows of data if NAs in Character type column and MHT in
hard to be
> #replaced with suitable value, since it is a categorical variable
> WAUS_omit <- na.omit(WAUS,cols=
+                          c("Year","Location", "WindGustDir", "WindDi
r9am"
+                           ,"WindDir3pm","RainToday", "MHT"))
```

Since some columns have too many NAs, which are already not worth for analysing. For example, Sunshine which has 787 NAs over the 2000 observation of sample data set.

All those attributes are discarded for this time analysing. As for those categorical type attributes, since the NAs are hard to be replaced with the suitable value, all the observations that have NA in those categorical attributes would be discarded as well.

Q2

```
> #Q2
> #change categorical non-character type attributes into factor
> WAUS_omit$Year <- as.factor(WAUS_omit$Year)
> WAUS_omit$Location <- as.factor(WAUS_omit$Location)
> WAUS_omit$MHT <- as.factor(WAUS_omit$MHT)

> #change all categorical character type attributes into factor
> col_chr <- sapply(WAUS_omit, is.character)
> #for numeric categorical attribute
> WAUS_omit[col_chr] <- lapply(WAUS_omit[col_chr], factor)


> WAUS_cc <- WAUS_omit

> #improve the performance of many machine learning algorithms by sc
aling
> col_num = sapply(WAUS_cc, is.numeric) #is.numeric count int and do
uble as numeric as well
```

```
> WAUS_cc[col_num] = lapply(WAUS_cc[col_num], scale)
```

After the pre-processing that has already been done, since each value of the categorical attributes should be viewed as one level, the categorical attributes are all factorised to correctly processed by the machine learning algorithm afterward. In addition, since the performance of the machine learning algorithm would be highly affected by the non-scaled large values. All the real-valued attributes are scaled.

```
> #specialised for Artificial neural network
> WAUS_nn <- WAUS_cc
```

Since the ANN would need to pre-process the data in a different way, a copy of processed data is copied and assigned to another variable.

Q3

```
> #Q3
> set.seed(31240291) #Student ID as random seed
> train.row = sample(1:nrow(WAUS_cc), 0.7*nrow(WAUS_cc))
> WAUS.train = WAUS_cc[train.row,]
> WAUS.test = WAUS_cc[-train.row,]
```

Sampling the original data set with my student ID as the random seed

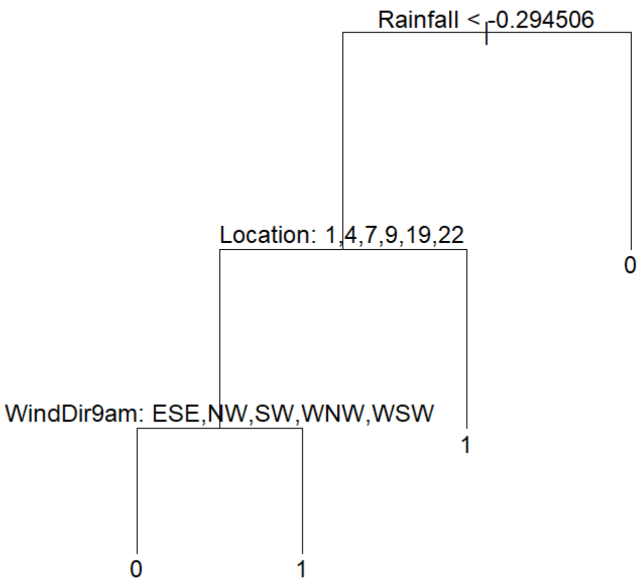
```
> #Q4
> #Accuracy function
> acc <- function(table) {
+   tn = table[1,1]
+   fn = table[1,2]
+   fp = table[2,1]
+   tp = table[2,2]
+   return ((tp + tn)/(tn+fn+fp+tp))
+ }
```

Accuracy function is created in advance for all the machine learning model to evaluate their performance with accuracy

Q5 ~ Q6 – Decision Tree

```
> #Q5
> #Decision Tree
> #str(WAUS.train)#use head can not see its actual type
> WAUS_tree = tree(MHT~., data=WAUS.train)
> #summary(WAUS_tree)
> plot(WAUS_tree)
> text(WAUS_tree, pretty=0)
> #create confusion matrix
> WAUS_predict = predict(WAUS_tree, WAUS.test, type = "class")
> t1 = table(Predicted_Class = WAUS.test$MHT, Actual_Class = WAUS_pr
edict)
> a1 = acc(t1)
> cat("\n Decision Tree \n")

 Decision Tree
> print(t1)
               Actual_Class
Predicted_Class   0   1
              0 102 126
              1  68 134
> #[0,0]: 102, [0,1] = 126, [1,0] = 68, [1,1] = 134
> sprintf("Accuracy: %.3f", a1)
[1] "Accuracy: 0.549"
> #Q6
> WAUS_pred_vec = predict(WAUS_tree, WAUS.test, type = "vector")
> #head(WAUS_pred_vec)
> WAUS_pred_comp = prediction(WAUS_pred_vec[,2], WAUS.test$MHT)
> WAUS_perf <- performance(WAUS_pred_comp, "tpr", "fpr")
> #plot if ROC curve
> plot(WAUS_perf, col="red", main = "Decision Tree vs Bagging
+      vs Naive Bayes vs Boosting vs Random Forest", xlab = "False P
ositive Rate"
+       , ylab = "True Positive Rate")
> abline(0,1)
```




```
> #AUC
> auc1 <- performance(WAUS_pred_comp, "auc")@y.values[[1]] #AUC of p
erformance instance
> auc1 <- round(auc1, 3)
> sprintf("AUC Decision Tree: %.3f", auc1)
[1] "AUC Decision Tree: 0.536"
```

Naïve Bayes

```
> #Naive Bayes Q5 ~ Q6
> #Q5
> WAUS_NB = naiveBayes(MHT~., data=WAUS.train)
> WAUS_pre.NB = predict(WAUS_NB, WAUS.test)
> #create confusion matrix
> t2=table(Predicted_Class = WAUS_pre.NB, Actual_Class = WAUS.test$M
HT)
> cat("\n#NaiveBayes Confusion\n")

#NaiveBayes Confusion
> print(t2)
               Actual_Class
Predicted_Class   0   1
              0 144 108
              1  84  94
> #Accuracy
> a2 = acc(t2)
> sprintf("Accuracy: %.3f", a2)
[1] "Accuracy: 0.553"
```

```
> #Q6
> #output as confidence level
> #obtain class probabilities in vector
> probs.NB <- predict(WAUS_NB, WAUS.test, type = "raw")
> #head(probs.NB)
> WAUS_NB_pred = prediction(probs.NB[,2],WAUS.test$MHT)
> WAUS_NB_perf = performance(WAUS_NB_pred, "tpr", "fpr")
> plot(WAUS_NB_perf, add = T, col = "blue")
> #@y.values[[1]] part of the code extracts the value of the AUC fro
m the
> #performance object. The @y.values slot of the performance object
is a
> #list that contains the calculated performance measures. Since we
only calculated
> #one performance measure (the AUC), we can access it using [[1]]
> #AUC
> #performance(WAUS_NB_pred, "auc") performance instance
> #y.values = valueson of measuament at y-axis
> auc2 <- performance(WAUS_NB_pred, "auc")@y.values[[1]] #AUC of per
formance instance
> auc2 <- round(auc2, 3)
> sprintf("AUC_NB: %.3f", auc2)
[1] "AUC_NB: 0.591"
```

Bagging

```
> #Q5
> WAUS.bag <- bagging(MHT~., data=WAUS.train, mfinal=5)
> WAUS_pre.bag <- predict.bagging(WAUS.bag, WAUS.test)
> #use the predicted probability to predict the whether it Y or No
> WAUS_Bagpred <- prediction(WAUS_pre.bag$prob[,2], WAUS.test$MHT) #
test has the actual
> #Bagging
> #Q5
> t3 = WAUS_pre.bag$confusion
> a3 = acc(t3)
> cat("\n#Bagging Confusion\n")

#Bagging Confusion
> print(t3)
               Observed Class
Predicted Class   0   1
              0 138 110
              1  90  92
> sprintf("Accuracy: %.3f", a3)
[1] "Accuracy: 0.535"

#Q6
> auc3 <- performance(WAUS_Bagpred, "auc")@y.values[[1]] #AUC of per
formance instance
> auc3 <- round(auc3, 3)
> sprintf("AUC_Bag: %.3f", auc3)
[1] "AUC_Bag: 0.556"

> WAUS_bag_perf <- performance(WAUS_Bagpred, "tpr", "fpr")
> plot(WAUS_bag_perf, add=TRUE, col = "darkgreen")
```

Boosting

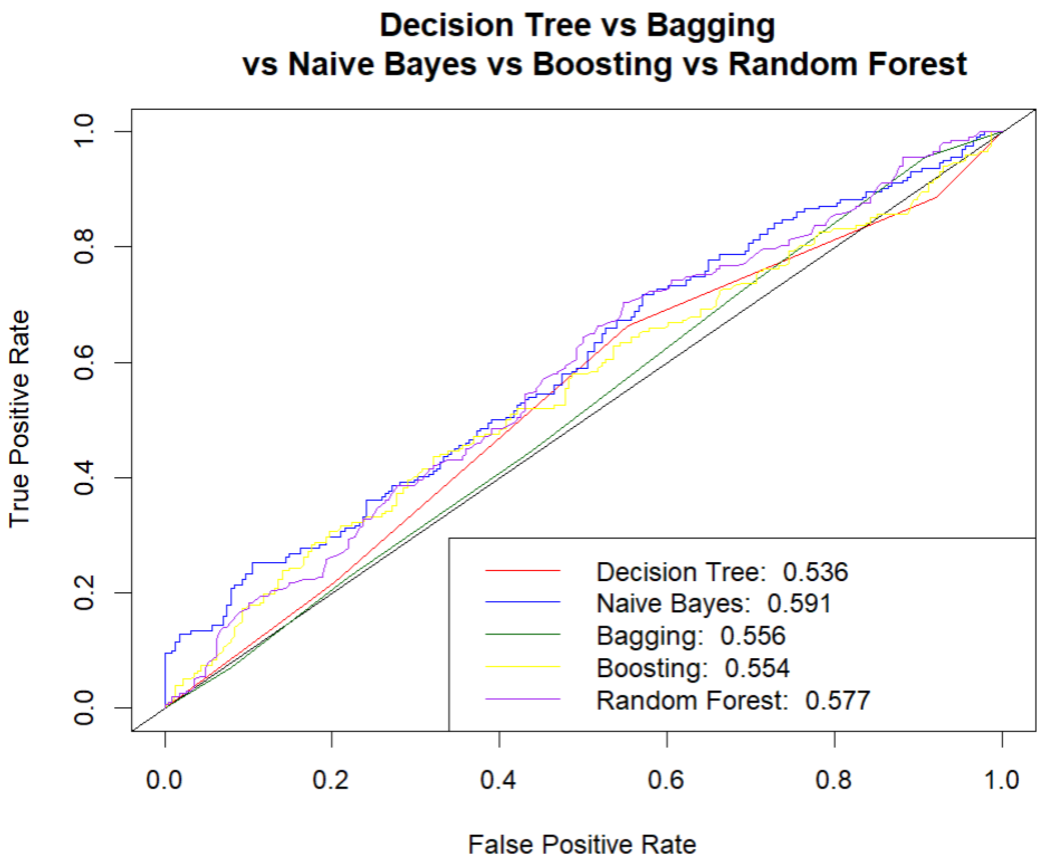
```
> WAUS_pred.boost <- predict.boosting(WAUS.Boost, newdata=WAUS.test)
> t4 = WAUS_pred.boost$confusion
> a4 = acc(t4)
> #Boosting Q5 ~ Q6
> #Q5
> WAUS.Boost <- boosting(MHT ~. , data = WAUS.train, mfinal=10)
> WAUS_pred.boost <- predict.boosting(WAUS.Boost, newdata=WAUS.test)
> t4 = WAUS_pred.boost$confusion
> a4 = acc(t4)
> cat("\n#Boosting Confusion\n")

#Boosting Confusion
> print(t4)
               Observed Class
Predicted Class   0   1
              0 128 104
              1 100  98
> sprintf("Accuracy: %.3f", a4)
[1] "Accuracy: 0.526"
> #Q6
> WAUS_Boostpred <- prediction(WAUS_pred.boost$prob[,2], WAUS.test$M
HT)
> WAUS_Boostperf <- performance(WAUS_Boostpred,"tpr","fpr")
> plot(WAUS_Boostperf, add=TRUE, col = "yellow")
> #AUC
> auc4 <- performance(WAUS_Boostpred, "auc")@y.values[[1]] #AUC of p
erformance instance
> auc4 <- round(auc4, 3)
> sprintf("AUC_Boost: %.3f", auc4)
[1] "AUC_Boost: 0.542"
```

Random Forest

```
> #Random Forest Q5 ~ Q6
> #Q5
> WAUS.rf <- randomForest(MHT ~. , data = WAUS.train, na.action = n
a.exclude)
> WAUSpred.rf <- predict(WAUS.rf, WAUS.test)
> t5=table(Predicted_Class = WAUSpred.rf, Actual_Class = WAUS.test$M
HT)
> #Accuracy
> a5 = acc(t5)
> cat("\n#Random Forest Confusion")

#Random Forest Confusion> sprintf("Accuracy: %.3f", a5)
[1] "Accuracy: 0.565"
> #Q6
> #these are all from ROCR package
> #output as vector of predicted probabilities
> WAUS_Ppred.rf <- predict(WAUS.rf, WAUS.test, type="prob")
> WAUS_rf_pred <- prediction(WAUS_Ppred.rf[,2], WAUS.test$MHT)
> str(WAUS.test$MHT)
 Factor w/ 2 levels "0","1": 1 1 2 2 2 1 1 2 1 2 ...
> WAUS_rf_perf <- performance(WAUS_rf_pred,"tpr","fpr")
> options(digits = 3)
> #AUC from
> auc5 <- performance(WAUS_rf_pred, "auc")@y.values[[1]] #AUC of per
formance instance
> auc5 <- round(auc5, 3)
> sprintf("AUC_Boost: %.3f", auc5)
[1] "AUC_Boost: 0.576"
> plot(WAUS_rf_perf, add=TRUE, col = "purple")
> legend("bottomright", legend = c(paste("Decision Tree: ", as.chara
cter(auc1))
+                                    ,paste("Naive Bayes: ", as.charac
ter(auc2))
+                                    ,paste("Bagging: ", as.character
(auc3))
+                                    ,paste("Boosting: ", as.character
(auc4))
+                                    ,paste("Random Forest: " , as.cha
racter(auc5)))
+                                    ,col = c("red","blue","darkgreen
","yellow","purple"), lty = 1)
```



Q7

```
> #Q7 Table for all models
> # Precision
> pre <- function(table) {
+   tn = table[1,1]
+   fn = table[1,2]
+   fp = table[2,1]
+   tp = table[2,2]
+   return(tp / (tp + fp))
+ }
> # Sensitivity (True Positive Rate)
> tpr <- function(table) {
+   tn = table[1,1]
+   fn = table[1,2]
+   fp = table[2,1]
+   tp = table[2,2]
+   return(tp / (tp + fn))
+ }
> # Specificity (True Negative Rate)
> tnr <- function(table) {
+   tn = table[1,1]
+   fn = table[1,2]
+   fp = table[2,1]
+   tp = table[2,2]
+   return (tn / (tn + fp))
+ }
> # False Positive Rate
> fpr <- function(table) {
+   tn = table[1,1]
+   fn = table[1,2]
+   fp = table[2,1]
+   tp = table[2,2]
```

```
+   return (fp / (fp + tn))
+ }
> tab <- matrix(c(a1,a2,a3,a4,a5
+                 ,auc1,auc2,auc3,auc4,auc5
+                 ,tpr(t1),tpr(t2),tpr(t3),tpr(t4),tpr(t5)
+                 ,fpr(t1),fpr(t2),fpr(t3),fpr(t4),fpr(t5)
+                 ,tnr(t1),tnr(t2),tnr(t3),tnr(t4),tnr(t5)
+                 ,pre(t1),pre(t2),pre(t3),pre(t4),pre(t5)), ncol=6,
 byrow = FALSE)
> colnames(tab) <- c('Accuracy', 'AUC', 'TPR', 'FPR', 'TNR', 'Precis
ion')
> rownames(tab) <- c('Decision Tree','Naive Bayes','Bagging','Boosti
ng','Random Forest')
> tab <- as.table(tab)
> tab
              Accuracy   AUC   TPR   FPR   TNR Precision
Decision Tree    0.549 0.536 0.515 0.400 0.600     0.663
Naive Bayes      0.553 0.591 0.465 0.368 0.632     0.528
Bagging          0.535 0.556 0.455 0.395 0.605     0.505
Boosting         0.556 0.554 0.450 0.351 0.649     0.532
Random Forest    0.565 0.576 0.629 0.491 0.509     0.531
```

From the table created, noteworthily, Random Forest is performing the best on those important metrics: Accuracy, TPR, FPR metrics. Especially, TPR's performance is well beyond other models, which mean the Random Forest is performing well on classifying the true positive out of all the positive instances. Apart from it, the Naïve Bayes has highest AUC which mean with different confidence threshold of classifying into Yes or No (predicting more humid tomorrow or not), the Naïve Bayes has the overall highest performance on TPR and FPR by considering different confidence threshold.

By considering just the most-used metrics: Accuracy, TPR and FPR, random forest would be the best classifier.

#Q8

```
> #Q8
> #Attribute Importance
> cat("\n#Decision Tree Attribute Importance\n")

#Decision Tree Attribute Importance
> print(summary(WAUS_tree))

Classification tree:
tree(formula = MHT ~ ., data = WAUS.train)
Variables actually used in tree construction:
[1] "Rainfall"   "Location"    "WindDir9am"
Number of terminal nodes:  4
Residual mean deviance:  1.32 = 1320 / 997
Misclassification error rate: 0.396 = 396 / 1001
> #feature has a high conditional probability for one value and a lo
w conditional
> #probability for another value, it means that the feature can help
 distinguish
> #between instances of the class and instances of other classes. On
 the other hand,
> #if a feature has similar conditional probabilities for all value
s, it means that
> #the feature is not very useful for predicting the class
> cat("\n#Naive Bayes Attribute Importance\n")

#Naive Bayes Attribute Importance
> #Determine importance of each variable
> varImp(WAUS.caret)
ROC curve variable importance

              Importance
Rainfall          100.00
Temp3pm            85.80
Location           78.08
MaxTemp            76.06
Temp9am            74.81
RainToday          68.15
WindSpeed9am       48.39
WindGustSpeed      47.00
WindGustDir        26.89
WindDir9am         25.96
Pressure3pm        22.78
WindSpeed3pm       16.39
WindDir3pm         11.98
MinTemp             9.99
Year                8.99
RISK_MM             6.66
Pressure9am         0.00
> cat("\n#Baging Attribute Importance\n")
```

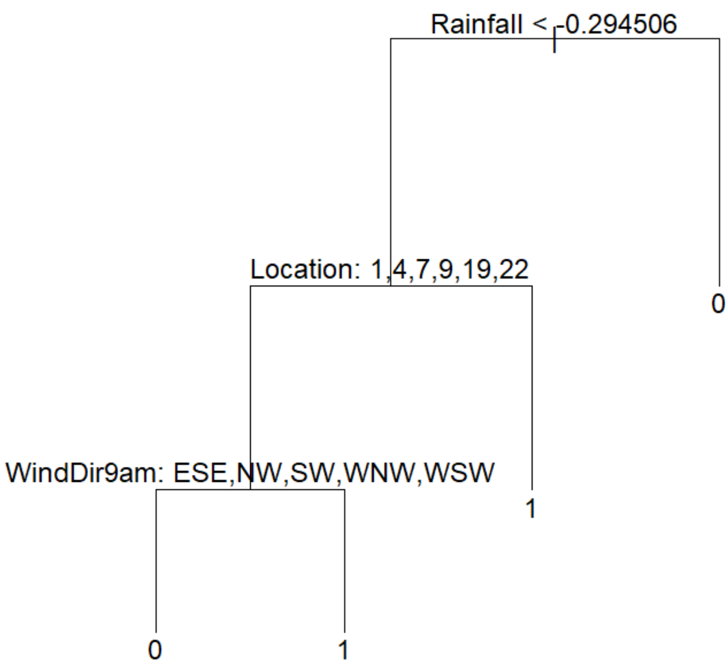
```
#Baging Attribute Importance
> print(WAUS.bag$importance)
     Location        MaxTemp        MinTemp    Pressure3pm    Pressure9a
m       Rainfall
        7.543          2.634          3.644          2.670          2.40
9          7.567
    RainToday        RISK_MM        Temp3pm        Temp9am     WindDir3p
m     WindDir9am
        0.000          0.884          1.258          1.583         17.96
4         14.208
  WindGustDir WindGustSpeed  WindSpeed3pm  WindSpeed9am           Yea
r
       18.425          0.889          2.394          3.897         12.03
0
> cat("\n#Boosting Attribute Importance\n")

#Boosting Attribute Importance
> print(WAUS.Boost$importance)
     Location        MaxTemp        MinTemp    Pressure3pm    Pressure9a
m       Rainfall
         7.33           2.40           3.83           3.19           2.4
7           5.07
    RainToday        RISK_MM        Temp3pm        Temp9am     WindDir3p
m     WindDir9am
         0.00           2.10           2.30           4.20          15.0
7          17.00
  WindGustDir WindGustSpeed  WindSpeed3pm  WindSpeed9am           Yea
r
        13.75           1.40           1.56           2.04          16.2
8
> cat("\n#Random Forest Attribute Importance\n")

#Random Forest Attribute Importance
> print(WAUS.rf$importance)
              MeanDecreaseGini
Year                     47.27
Location                 32.15
MinTemp                  25.11
MaxTemp                  25.75
Rainfall                 18.44
WindGustDir              55.82
WindGustSpeed            21.27
WindDir9am               56.41
WindDir3pm               55.52
WindSpeed9am             20.83
WindSpeed3pm             20.49
Pressure9am              24.70
Pressure3pm              25.85
Temp9am                  27.26
Temp3pm                  26.45
RainToday                 4.16
RISK_MM                  10.48
```

Among all the model, the most important attributes would be the WindDir9am since it plays a quite important role for all the models. In comparison, Pressure9am and Pressure3pm since it barely has effect on predicting the MHT.

Q9



Since the decision tree is already simple enough for a person to traverse the node to classify or predict MHT, the decision tree model can be just used by hand. Firstly, value of the Rainfall should be checked whether it is <-0.294506. If no, MHT = 0 (tomorrow is not more humid than today); If yes, it is required to traverse to the sub-node Location to check whether Location value is one of the 1,4,7,9,19,22. If yes, MHT = 1 (tomorrow is more humid than today); If no, it is required to traverse to the sub-node WindDir9am to check whether WinDir9am is one of the ESE, NW, SW, WNW, WSW. If no, MHT = 0 (tomorrow is not more humid than today); If yes, MHT = 1 (tomorrow is more humid than today).

For the decision model, Rainfall, Location and WindDir9am are relatively more important attribute to classify MHT. In every step of classifying MHT dependent on the factors of each chosen attribute, the attribute would be only chosen if it would bring the most information gain at that current stage (Greedy Approach). For example, the start, Rainfall is chosen to be root node as it brings more information gain and most homogeneity among all other attributes. Since the hand model just implement the original decision tree, performance would be expected to be the same as the decision developed in question4 on all metrics.

Q10

```
> #Q10
> WAUS_tree_imp = tree(MHT ~ .- Location - WindDir9am - WindGustSpee
d, data=WAUS.train)
> plot(WAUS_tree_imp)
> text(WAUS_tree_imp, pretty=0)
> #create confusion matrix
> WAUS_predict = predict(WAUS_tree_imp, WAUS.test, type = "class")
> ptt = table(Predicted_Class = WAUS.test$MHT, Actual_Class = WAUS_p
redict)
> #accuracy
> a6 = acc(ptt)
> a6 = round(a6, 3)
> #neuralnet would conflit with the ROCR package
> detach(package:neuralnet)
> WAUS_pred_vec = predict(WAUS_tree_imp, WAUS.test, type = "vector")
> WAUS_pred_comp = prediction(WAUS_pred_vec[,2], WAUS.test$MHT)
> WAUS_perf <- performance(WAUS_pred_comp, "tpr", "fpr")
> #AUC
> auc6 <- performance(WAUS_pred_comp, "auc")@y.values[[1]] #AUC of p
erformance instance
> auc6 <- round(auc6, 3)
> sprintf("AUC Improved Tree: %.3f", auc6)
[1] " AUC Improved Tree: 0.606"
```

```
> #cv tree
> WAUS_cvtree = cv.tree(WAUS_tree_imp, FUN = prune.misclass)
> print(WAUS_cvtree)
$size
[1] 4 3 1

$dev
[1] 465 462 481

$k
[1] -Inf 13.0 36.5

$method
[1] "misclass"

attr(,"class")
[1] "prune"         "tree.sequence"
> #prune the tree
> WAUS_prune_tree = prune.misclass(WAUS_tree_imp, best = 4)
> plot(WAUS_prune_tree)
> text(WAUS_prune_tree, pretty = 0)
> #create confusion matrix
> WAUS_ppredict = predict(WAUS_prune_tree, WAUS.test, type = "class
")
> pt = table(predicted_values = WAUS_ppredict, actual_value = WAUS.t
est$MHT)
> print(pt)
                actual_value
predicted_values   0   1
               0 194 149
               1  34  53
```

```
> acc(pt)
[1] 0.574
> cat("\n# Improved Decision Tree Attribute Importance\n")

#Pruned Decision Tree Attribute Importance
> print(summary(WAUS_tree_imp))

Classification tree:
tree(formula = MHT ~ . - Location - WindDir9am - WindGustSpeed,
    data = WAUS.train)
Variables actually used in tree construction:
[1] "Rainfall"    "WindGustDir" "Temp9am"
Number of terminal nodes:  4
Residual mean deviance:  1.32 = 1320 / 997
Misclassification error rate: 0.383 = 383 / 1001
```

```
> tab
              Accuracy   AUC   TPR   FPR   TNR Precision
Decision Tree    0.549 0.536 0.515 0.400 0.600     0.663
Naive Bayes      0.553 0.591 0.465 0.368 0.632     0.528
Bagging          0.535 0.556 0.455 0.395 0.605     0.505
Boosting         0.556 0.554 0.450 0.351 0.649     0.532
Random Forest    0.565 0.576 0.629 0.491 0.509     0.531
Improved Tree    0.574 0.606 0.609 0.434 0.566     0.262
```

As inspired by the high performance and mechanism of random forest (selecting different attributes to build the tree and predict on all created tree, the outcome of them are averaged and output) and acknowledgement of the weakness of decision (Greed Approach), The attributes WindDir9am and Location which were used before are removed to re-build the decision tree to reduce the effect due to Greedy Approach. Afterward, this improved tree is passed to cross-validation check, which would be better off not to prune the tree. The improved tree has higher performance on the major metrics (Accuracy, AUC, TPR, FPR) compared to the original decision tree and all the others.

Q11

```
> #Q11
> library(neuralnet)
> options(digit=3)
> WAUS.nn.data <- WAUS_nn
> #WAUS.nn.data = cbind(WAUS.nn.data, MHT)
> #WAUS.nn.data$MHT <- NULL
> WAUS.nn.data$Location <- NULL
> WAUS.nn.data$Year <- NULL
> col_num = sapply(WAUS.nn.data, is.numeric)
> WAUS.nn.num <- WAUS.nn.data[,col_num]
> #one-hot encoding to convert the Location column into multiple bin
ary columns,
> #one for each level of the factor. This can be done using the mode
l.matrix function in R
> WAUS_ptmm = model.matrix(~WindGustDir + WindDir9am + WindDir3pm, d
ata=WAUS.nn.data)
> WAUS.com = cbind(WAUS_ptmm, WAUS.nn.num)
> str(WAUS.com)
> # Remove columns by names
> #WAUS.com <- WAUS.com %>% select(-c("WindGustDir","WindDir9am","Wi
ndDir3pm"))
> WAUS.com$MHT <- WAUS.nn.data$MHT
> #sample train and test data
> set.seed(31240291)
> nn_train.row = sample(1:nrow(WAUS.com), 0.8*nrow(WAUS.com))
```

```
> WAUS.nn.train = WAUS.com[nn_train.row,]
> WAUS.nn.test = WAUS.com[-nn_train.row,]
> MHT.nn = WAUS.nn.test$MHT
> WAUS.nn.test.pre = WAUS.nn.test[,1:ncol(WAUS.nn.test)-1]
> #input variables must be explicit since for the input neuron of AN
N
> #Location - WindDir9am - WindGustSpeed can be discarded
> WAUS.nn = neuralnet(MHT ~ WindGustDirENE+WindGustDirESE+WindGustDi
rN+WindGustDirNE
+                      +WindGustDirNNE+WindGustDirNNW+WindGustDirNW+W
indGustDirS+WindGustDirSE
+                      +WindGustDirSSE+WindGustDirSSW+WindGustDirSW+W
indGustDirW+WindGustDirWNW
+                      +WindGustDirWSW+WindDir9amENE+WindDir9amESE+Wi
ndDir9amN+WindDir9amNE
+                      +WindDir9amNNE+WindDir9amNNW+WindDir9amNW+Wind
Dir9amS+WindDir9amSE
+                      +WindDir9amSSE+WindDir9amSSW+WindDir9amSW+Wind
Dir9amW+WindDir9amWNW
+                      +WindDir9amWSW+WindDir3pmENE+WindDir3pmESE+Win
dDir3pmN+WindDir3pmNE
+                      +WindDir3pmNNE+WindDir3pmNNW+WindDir3pmNW+Wind
Dir3pmS
+                      +WindDir3pmSE+WindDir3pmSSE+WindDir3pmSSW+Wind
Dir3pmSW
+                      +WindDir3pmW+WindDir3pmWNW+WindDir3pmWSW+MinTe
mp+MaxTemp
+                      +Rainfall+WindSpeed9am+WindSpeed3pm+Pressure9a
m
+                      +Pressure3pm+Temp9am+Temp3pm+RISK_MM,WAUS.nn.t
rain,hidden=3,linear.output=FALSE)
> #[,-MHT] to ignore response variable since it is not one of the ou
tput variable
> #in output neuron of  ANN model
> WAUS.nn.comp = compute(WAUS.nn, WAUS.nn.test)
> ann.tfpr = WAUS.nn.comp$net.result[,2]
> #round to integer if < 0.5 -> = 0
> ann.predr = round(ann.tfpr, 0)
> t7 = table(Predicted_Class = ann.predr, Actual_Class = WAUS.nn.tes
t$MHT)
> print(t7)
               Actual_Class
Predicted_Class  0  1
              0 81 63
              1 75 68
> #Accuracy
> a7 = acc(t7)
> a7 = round(a7,3)
> sprintf("Accuracy: %.3f", a7)
[1] "Accuracy: 0.519"
> #Q6
> #conflict between ROCR and neuralnet package
> detach(package:neuralnet)
> # Calculate the predicted values
> WAUS_pred.ann = predict(WAUS.nn, newdata = WAUS.nn.test)
> WAUS.nn.test = as.data.frame(WAUS.nn.test)
> WAUS_value = as.data.frame(WAUS.nn.comp$net.result)
> WAUS_ANNpred = prediction(WAUS_value$V1, WAUS.nn.test$MHT)
> WAUS_ANNperf = performance(WAUS_ANNpred,"tpr","fpr")
> # Plot the ROC curve
> plot(WAUS_ANNperf, add=TRUE, col = "orange", main = "ROC Curve for
 ANN Model"
+       , xlab = "False Positive Rate", ylab = "True Positive Rate")
> abline(0,1)
> #AUC
> auc7 <- performance(WAUS_ANNpred, "auc")@y.values[[1]] #AUC of per
formance instance
> auc7 <- round(auc7, 3)
> sprintf("AUC_ANN: %.3f", auc7)
```

```
[1] "AUC_ANN: 0.448"
> #Q7
> tab <- rbind(tab, c(a7,auc7,tpr(t7),fpr(t7),tnr(t7),pre(t7)))
> rownames(tab)[nrow(tab)] <- "ANN"
> tab
               Accuracy   AUC   TPR   FPR   TNR Precision
Decision Tree     0.549 0.536 0.515 0.400 0.600     0.663
Naive Bayes       0.553 0.591 0.465 0.368 0.632     0.528
Bagging           0.535 0.556 0.455 0.395 0.605     0.505
Boosting          0.556 0.554 0.450 0.351 0.649     0.532
Random Forest     0.565 0.576 0.629 0.491 0.509     0.531
Improved Tree     0.574 0.606 0.609 0.434 0.566     0.262
ANN               0.519 0.448 0.519 0.481 0.519     0.476
```

Since ANN (Artificial Neural Network) would require explicit attributes levels while train the model, these following attributes are used:

WindGustDir, WindDir9am, WindDir3pm, MinTemp,MaxTemp, Rainfall,WindSpeed9am, WindSpeed3pm, Pressure9am, Pressure3pm, Temp9am, Temp3pm, RISK_MM

Due to the large amount of level in Location and Year and relatively low impact that thses attributes could have on prediction, those attributes are neglected before training the ANN. In comparison with the others, ANN is probably one of the poorest performing models. One of the reasons would be the low number of hidden layers, with the increasing number of hidden layers, the ANN would perform much better. At the same time, the need of the computation resource would be dramatically increased. Therefore, the performance of ANN is bounded by the low computation resource of my laptop. Another reason would be the activation function used in each neuron is not fitted very well. The performance of ANN is highly bounded by the synergistic effect not enough layer to fit with the best activation function.

Q12

```
> #Q5 ~ Q6
> #install.packages("party")
> library(party)
> #always use ? to see the document
> WAUS_cf <- cforest(MHT ~ .,data = WAUS.train)
> #can not ignore newdata
> WAUS.cf.pred <- predict(WAUS_cf, newdata = WAUS.test)
> length(WAUS.cf.pred)
[1] 430
> str(WAUS.cf.pred)
 Factor w/ 2 levels "0","1": 1 1 2 2 1 1 1 1 1 2 ...
> t8 = table(predicted = WAUS.cf.pred, actual = WAUS.test$MHT)
> a8 = acc(t8)
> a8 <- round(a8, 3)
> sprintf("CForest Accuracy: %.3f", a8)
[1] "CForest Accuracy: 0.572"
> #Q6
> WAUS.cf.conf <- predict(WAUS_cf, newdata = WAUS.test, type = "prob
")
> #dictionary data type
> avector <- vector(mode = "numeric", length = 0)
> #loop through the dictionary data type
> for (name in names(WAUS.cf.conf)) {
+   prob_1 <- WAUS.cf.conf[[name]][2]
+   avector <- c(avector, prob_1)
+ }
> WAUS_cf_pred <- prediction(avector, WAUS.test$MHT)
```

```
> WAUS_cf_perf <- performance(WAUS_cf_pred,"tpr","fpr")
> plot(WAUS_cf_perf)
> abline(0,1)
> #AUC from
> auc8 <- performance(WAUS_cf_pred, "auc")@y.values[[1]] #AUC of per
formance instance
> auc8 <- round(auc8, 3)
> sprintf("CForest AUC: %.3f", auc8)
[1] "CForest AUC: 0.615"
> #add into table
> tab <- rbind(tab, c(a8,auc8,tpr(t8),fpr(t8),tnr(t8),pre(t8)))
> rownames(tab)[nrow(tab)] <- "CForest"
> tab
              Accuracy   AUC   TPR   FPR   TNR Precision
Decision Tree    0.549 0.536 0.515 0.400 0.600     0.663
Naive Bayes      0.553 0.591 0.465 0.368 0.632     0.528
Bagging          0.535 0.556 0.455 0.395 0.605     0.505
Boosting         0.556 0.554 0.450 0.351 0.649     0.532
Random Forest    0.565 0.576 0.629 0.491 0.509     0.531
Improved Tree    0.574 0.606 0.609 0.434 0.566     0.262
ANN              0.519 0.448 0.519 0.481 0.519     0.476
CForest          0.572 0.615 0.490 0.355 0.645     0.550
```

Since random forest is performing well on this data set, the variation of the random forest is my direction to search for the possibly better classification – cforest.

The cforest function is from the party package, which uses conditional inference trees to

create random forests.

Conditional inference trees use statistical tests to determine the best split at each node. At each node, the response variable is tested with each predictor variable with null hypothesis test to test whether response variable is independent from each predictor variable.

The predictor variable with the lowest p-value (strongest association with the response variable) is selected for splitting, then the cforest use statistical test to determine best split point for the selected predictor which would bias toward factor variable with many levels.

From the table, cforest has done a better job than the traditional random forest in terms of Accuracy, TNR and Precision. Among all the models, it is one of best performing models.