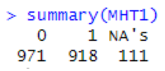
Assignment2 of FIT3152

(1). First, I set MHT attribute as factor type, then use summary function to see how many 0 and 1 in there.

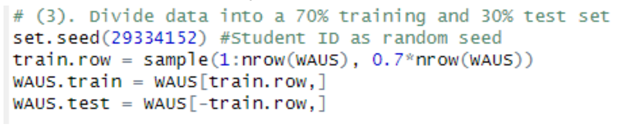


```
> summary(MHT1)
  0    1 NA's
971  918  111
```

1 means more humid than the previous day, 0 means less humid than the previous day. Here we have some NA value, but we just ignore it and focus on 0 and 1 data(971+918=1889). So, proportion for less humid than the previous day 971/1889 (about 0.514) and for more humid than the previous day is 918/1889(about 0.485).
We use summary function to show out the details of each variable. Then take a closer look at each variable's descriptive stats. We can find that some variables are not numeric variables, and it should be factoring variables, like location(should be a place which can be factor to represent a place by number, is not a numeric variables), WindDir9am(not a char, should be factor than we can use it easier), WindDir3pm, etc. these wind directions and location variable might have effect about tomorrow is humidor or not. so, we keep these variables. Then we find the Year variable only for record the time about observation which is not affect for tomorrow is more humid or not, so it can be omitted.

(2). To ensure the integrity of the data, I have preserved the original dataset for potential future use. As a next step, I have excluded the 'Year' column from the analysis since it does not hold any relevance in predicting whether tomorrow will be more humid or not. Furthermore, I have removed all rows that contain missing values (NAs) from the dataset to ensure accurate predictions. Upon further investigation, I have observed that even after increasing the number of sampled rows, the dataset still consists of only 5 locations. However, I consider this acceptable as there are several other variables that play a more significant role in predicting tomorrow's humidity compared to the location variable. Lastly, I have converted the variables to their appropriate factor type(the variable can be directly change to factor: Location, WindDir9am, WindDir3pm, WindGustDir, etc.).

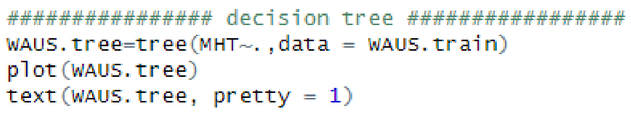
(3). In this question, I just modify the code question provide to me and use this code to divide data into 70 training dataset and 30% test dataset. The code is like below: change all iris to WAUS and seed is my student number.



```
# (3). Divide data into a 70% training and 30% test set
set.seed(29334152) #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,]
```

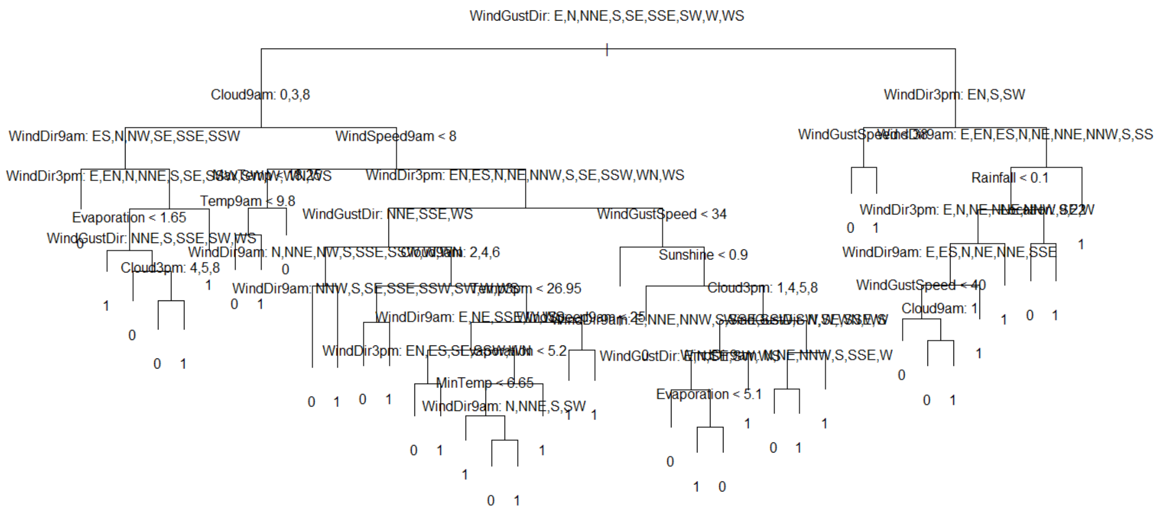
(4). This question asks me to create classification model by each technique.

1. Create model use decision tree approach:



```
################ decision tree #################
WAUS.tree=tree(MHT~.,data = WAUS.train)
plot(WAUS.tree)
text(WAUS.tree, pretty = 1)
```

And plot tree out:



2. Create model use Naïve Bayes approach:

```
################# Naive Bayes #################
WAUS.bayes=naiveBayes(MHT~.,data = WAUS.train)
```

3. Create model use Bagging approach:

```
################# Bagging #################
WAUS.bag = bagging(MHT ~. , data = WAUS.train, mfinal=5)
```

4. Create model use Boosting approach:

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

5. Create model use Random Forest approach:

```
################ Random Forest #################
WAUS.rf <- randomForest(MHT ~. , data = WAUS.train, na.action = na.exclude)
```

(5). use test data to get Confusion matrix and report the accuracy of each model.

1. Predict using the Decision Tree classification model:

```
################# decision tree #################
WAUS.pred = predict(WAUS.tree, WAUS.test, type = "class")
```

Confusion Matrix and accuracy for Decision Tree classification model:

```
t1=table(actual=WAUS.test$MHT, predicted = WAUS.pred)
t1
accuracy1 <- sum(diag(t1)) / sum(t1)
accuracy1 #0.4561, 0.4386, 0.4269, 0.4211, 0.4327, 0.4444, 0.4152
accuracy1=(0.4561+0.4386+0.4269+0.4211+0.4327+0.4444+0.4152)/7 #0.4336
```

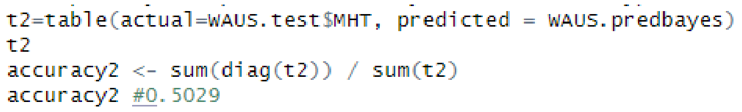
```
> t1
      predicted
actual  0  1
     0 37 50
     1 43 41
```

Each time the code is executed, the variability in accuracy can be attributed to the inherent randomness in certain aspects of the decision tree algorithm. Factors such as the random selection of segmentation points during the model building process contribute to the different outcomes observed. As a result, the accuracy of predictions may vary across different runs of the code. So here I run it lots of times and get a set of different accuracy, then I choose the mean of these set of accuracy as final accuracy. The accuracy here are 0.4336.

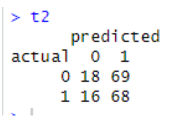
2. Predict using the Naïve Bayes classification model:

```
################ Naive Bayes #################
WAUS.predbayes = predict(WAUS.bayes, WAUS.test, type = "class")
```

Confusion Matrix and accuracy for Naïve Bayes classification model:



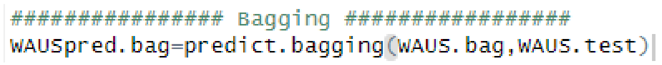
```
t2=table(actual=WAUS.test$MHT, predicted = WAUS.predbayes)
t2
accuracy2 <- sum(diag(t2)) / sum(t2)
accuracy2 #0.5029
```



```
> t2
      predicted
actual  0  1
     0 18 69
     1 16 68
```

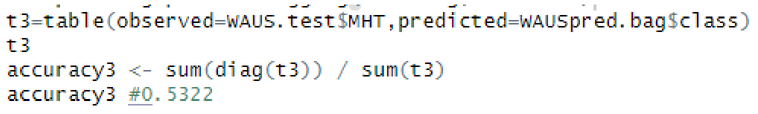
The accuracy here are 0.5029.

3. Predict using the Bagging classification model:

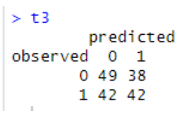


```
################# Bagging ##################
WAUSpred.bag=predict.bagging(WAUS.bag,WAUS.test)
```

Confusion Matrix and accuracy for Bagging classification model:



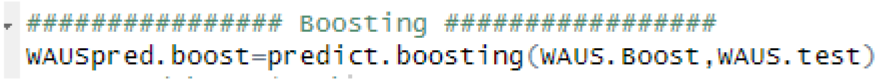
```
t3=table(observed=WAUS.test$MHT,predicted=WAUSpred.bag$class)
t3
accuracy3 <- sum(diag(t3)) / sum(t3)
accuracy3 #0.5322
```



```
> t3
        predicted
observed  0  1
       0 49 38
       1 42 42
```

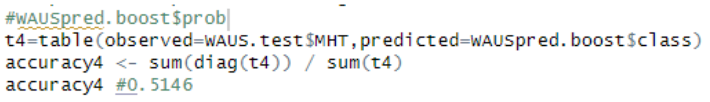
The accuracy here are 0.5322.

4. Predict using the Boosting classification model:

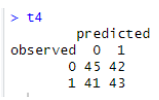


```
################# Boosting ##################
WAUSpred.boost=predict.boosting(WAUS.Boost,WAUS.test)
```

Confusion Matrix and accuracy for Boosting classification model:



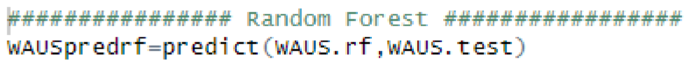
```
#WAUSpred.boost$prob
t4=table(observed=WAUS.test$MHT,predicted=WAUSpred.boost$class)
accuracy4 <- sum(diag(t4)) / sum(t4)
accuracy4 #0.5146
```



```
> t4
        predicted
observed  0  1
       0 45 42
       1 41 43
```

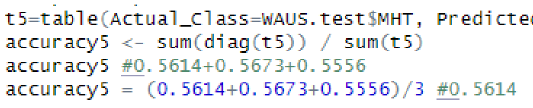
The accuracy here are 0.5146.

5. Predict using the Random Forest classification model:

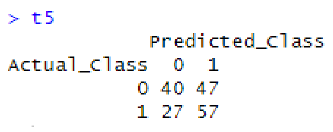


```
################# Random Forest ##################
WAUSpredrf=predict(WAUS.rf,WAUS.test)
```

Confusion Matrix and accuracy for Random Forest classification model:



```
t5=table(Actual_Class=WAUS.test$MHT, Predicted_Class = WAUSpredrf)
accuracy5 <- sum(diag(t5)) / sum(t5)
accuracy5 #0.5614+0.5673+0.5556
accuracy5 = (0.5614+0.5673+0.5556)/3 #0.5614
```

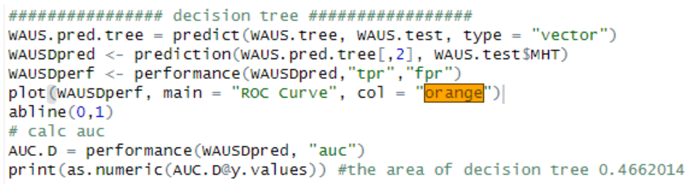


```
> t5
            Predicted_Class
Actual_Class  0  1
           0 40 47
           1 27 57
```

Now due to the randomness of Random Forest is the same situation as what we discussed in decision tree happen. So, it is possible to evaluate the performance of the model and observe its average accuracy over multiple runs by performing methods such as cross-validation. Then we get the accuracy here are 0.5614.

(6). construct an ROC curve for each classifier and calculate the AUC for each classifier.

1. Plot ROC curve for decision tree and calculate the AUC



```
############### decision tree ################
WAUS.pred.tree = predict(WAUS.tree, WAUS.test, type = "vector")
WAUSDpred <- prediction(WAUS.pred.tree[,2], WAUS.test$MHT)
WAUSDperf <- performance(WAUSDpred,"tpr","fpr")
plot(WAUSDperf, main = "ROC Curve", col = "orange")
abline(0,1)
# calc auc
AUC.D = performance(WAUSDpred, "auc")
print(as.numeric(AUC.D@y.values)) #the area of decision tree 0.4662014
```

2. Plot ROC curve for Naïve Bayes and calculate the AUC

```
################ Naive Bayes ################
WAUSpred.bayes = predict(WAUS.bayes, WAUS.test, type = 'raw')
WAUSBpred <- prediction( WAUSpred.bayes[,2], WAUS.test$MHT)
WAUSBperf <- performance(WAUSBpred,"tpr","fpr")
plot(WAUSBperf, add=TRUE, col = "blueviolet")
# calc auc
AUC.N = performance(WAUSBpred, "auc")
print(as.numeric(AUC.N@y.values)) #the area of Naive Bayes 0.5432403
```

3. Plot ROC curve for Bagging and calculate the AUC

```
################ Bagging ################
WAUSBagpred <- prediction( WAUSpred.bag$prob[,2], WAUS.test$MHT)
WAUSBagperf <- performance(WAUSBagpred,"tpr","fpr")
plot(WAUSBagperf, add=TRUE, col = "blue")
# calc auc
AUC.B = performance(WAUSBagpred, "auc")
print(as.numeric(AUC.B@y.values)) #the area of decision tree 0.5476875
```

4. Plot ROC curve for Boosting and calculate the AUC

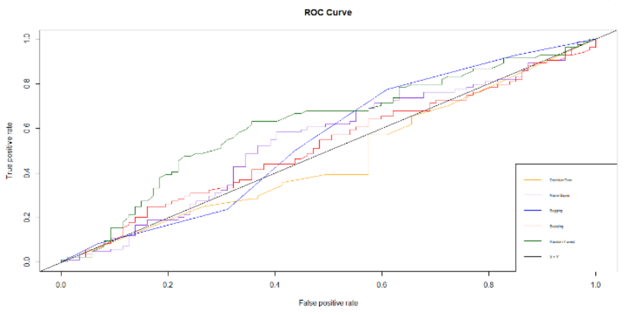
```
################ Boosting ################
WAUSBoostpred <- prediction( WAUSpred.boost$prob[,2], WAUS.test$MHT)
WAUSBoostperf <- performance(WAUSBoostpred,"tpr","fpr")
plot(WAUSBoostperf, add=TRUE, col = "red")
# calc auc
AUC.Boost = performance(WAUSBoostpred, "auc")
print(as.numeric(AUC.Boost@y.values)) #the area of Boosting 0.5191571
```

5. Plot ROC curve for Random Forest and calculate the AUC

```
################ Random Forest ################
WAUSpred.rf <- predict(WAUS.rf, WAUS.test, type="prob")
WAUSFpred <- prediction( WAUSpred.rf[,2], WAUS.test$MHT)
WAUSFperf <- performance(WAUSFpred,"tpr","fpr")
plot(WAUSFperf, add=TRUE, col = "darkgreen")
# calc auc
AUC.F = performance(WAUSFpred, "auc")
print(as.numeric(AUC.F@y.values)) #the area of Random Forest 0.6208949
```

Then each classifier's ROC curve in one plot below(with legend):

```
legend(0.85,0.44,legend = c("Decision Tree","Naive Bayes","Bagging","Boosting","Random Forest","X = Y"),
      lty = 1,col = c("orange","blueviolet","blue","red","darkgreen","black"),inset = c(0,0), cex = 0.5)
```



(7). create a table to compare performance for each classifier.

```
##########################################################################################
# (7). performance table for comparing result
performance_table = data.frame(Classifier = c('Decision Tree','Naive Bayes','Bagging','Boosting','Random Forest'),
                               Accuracy = c(accuracy1,accuracy2,accuracy3,accuracy4,accuracy5),
                               AUC = c(0.4662014,0.5432403,0.5476875,0.5191571,0.6208949)) %>% arrange(desc(Accuracy))
print(performance_table)
```

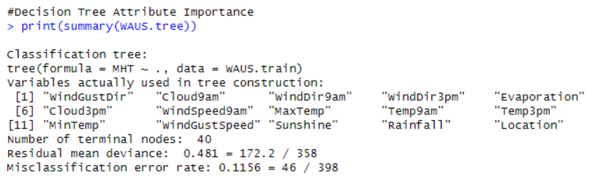
```
> print(performance_table)
     Classifier  Accuracy       AUC
1 Random Forest 0.5614333 0.6208949
2       Bagging 0.5321637 0.5476875
3      Boosting 0.5146199 0.5191571
4   Naive Bayes 0.5029240 0.5432403
5 Decision Tree 0.4335714 0.4662014
```

Accuracy refers to the measure of how accurately a classification model predicts the correct outcome. The ROC curve (receiver operating characteristic curve) is a graph that describes the performance of a classification model at all classification thresholds. AUC is area under the ROC curve, it commonly used to evaluate the performance of binary classification models(like ROC curves). The attribute we focused on is MHT (more humid tomorrow or not), this is a binary classification problem, so look at AUC will show how good the classification in each classifier.

Now look at table here we can see Random Forest have highest accuracy and AUC, so Random Forest is the “best” classifier.

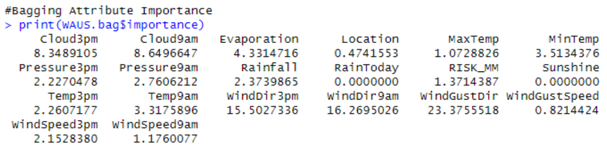
(8). Attribute importance for each classifier

1. Decision Tree Attribute Importance (in sequence, 1st windGustDir, 2nd could9am etc.)



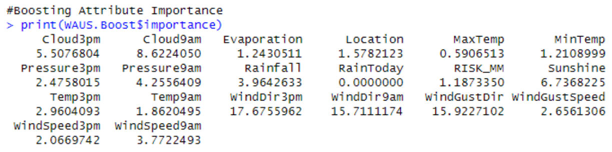
```
#Decision Tree Attribute Importance
> print(summary(WAUS.tree))

Classification tree:
tree(formula = MHT ~ ., data = WAUS.train)
Variables actually used in tree construction:
 [1] "WindGustDir"   "Cloud9am"      "WindDir9am"    "WindDir3pm"    "Evaporation"
 [6] "Cloud3pm"      "WindSpeed9am"  "MaxTemp"       "Temp9am"       "Temp3pm"
[11] "MinTemp"       "WindGustSpeed" "Sunshine"      "Rainfall"      "Location"
Number of terminal nodes:  40
Residual mean deviance:  0.481 = 172.2 / 358
Misclassification error rate: 0.1156 = 46 / 398
```

2. Bagging Attribute Importance (bigger number under attributes’ name means more important)



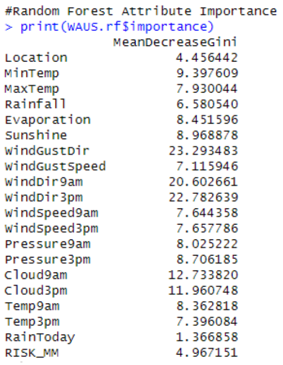
```
#Bagging Attribute Importance
> print(WAUS.bag$importance)
     Cloud3pm      Cloud9am   Evaporation      Location       MaxTemp       MinTemp
    8.3489105     8.6496647     4.3314716     0.4741553     1.0728826     3.5134376
  Pressure3pm   Pressure9am      Rainfall     RainToday       RISK_MM      Sunshine
    2.2270478     2.7606212     2.3739865     0.0000000     1.3714387     0.0000000
      Temp3pm       Temp9am    WindDir3pm    WindDir9am   WindGustDir WindGustSpeed
    2.2607177     3.3175896    15.5027336    16.2695026    23.3755518     0.8214424
 WindSpeed3pm  WindSpeed9am
    2.1528380     1.1760077
```

3. Boosting Attribute Importance (bigger number under attributes’ name means more important)



```
#Boosting Attribute Importance
> print(WAUS.Boost$importance)
     Cloud3pm      Cloud9am   Evaporation      Location       MaxTemp       MinTemp
    5.5076804     8.6224050     1.2430511     1.5782123     0.5906513     1.2108999
  Pressure3pm   Pressure9am      Rainfall     RainToday       RISK_MM      Sunshine
    2.4758015     4.2556409     3.9642633     0.0000000     1.1873350     6.7368225
      Temp3pm       Temp9am    WindDir3pm    WindDir9am   WindGustDir WindGustSpeed
    2.9604093     1.8620495    17.6755962    15.7111174    15.9227102     2.6561306
 WindSpeed3pm  WindSpeed9am
    2.0669742     3.7722493
```

4. Random Forest Attribute Importance (bigger number under attributes’ name means more important)



```
#Random Forest Attribute Importance
> print(WAUS.rf$importance)
              MeanDecreaseGini
Location              4.456442
MinTemp               9.397609
MaxTemp               7.930044
Rainfall              6.580540
Evaporation           8.451596
Sunshine              8.968878
WindGustDir          23.293483
WindGustSpeed         7.115946
WindDir9am           20.602661
WindDir3pm           22.782639
WindSpeed9am          7.644358
WindSpeed3pm          7.657786
Pressure9am           8.025222
Pressure3pm           8.706185
Cloud9am             12.733820
Cloud3pm             11.960748
Temp9am               8.362818
Temp3pm               7.396084
RainToday             1.366858
RISK_MM               4.967151
```

5. Naïve Bayes Attribute Importance (not able to do):

There is not way to directly assess feature importance or generate coefficients associated with the features used to train the model in Naïve Bayes classifier. This is because Naïve Bayes calculate conditional and unconditional probabilities for the features, then forecast the class with highest probabilities. So, it can not generate coefficients associated with the features used to train the model.

Determine the most important variables and consider about which variables could be omitted from the data:

It is commonly understood that variables with higher importance coefficients have a greater influence on predicting whether tomorrow will be more humid or not. These coefficients

provide an indication of the relative importance or impact of each variable in the prediction process.
Look at the value of coefficients in Bagging, Boosting and Random Forest Classifiers, the variable “WindDir3pm”, “WindDir9am” and “WindGustDir” have significantly higher coefficients value than coefficients value of all the other variables. And they are also be used to construct the decision tree. So, these three attributes are most important.

The coefficient of “Cloud9am” are also highest, it only below “WindDir3pm”, “WindDir9am” and “WindGustDir” in Bagging, Boosting and Random Forest classifiers, and it is also used to construct the decision tree. This suggests that this variable play a significant role in predicting tomorrow's humidity levels according to these three models. So, “Cloud9am” are also important variables.

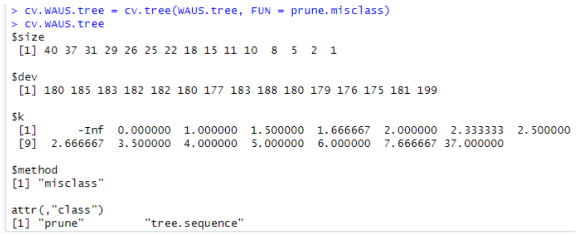
Although other variables may have lower importance coefficients compared to "WindDir3pm", "WindDir9am", "WindGustDir" and "Cloud9am" it would be inaccurate to deem them unimportant and exclude them from the analysis. When considering the collective impact of these variables, even those with relatively lower coefficients, they can still contribute significantly to the prediction of whether tomorrow will be more humid or not. Therefore, it is essential to consider most variables in the model to ensure a comprehensive analysis. However, we can omit the variables which have the lowest importance coefficients in all of classifiers and not used to construct the decision tree.

Not in decision tree	**Bagging**	**Boosting**	**Random Forest**
"WindSpeed3pm"	2.1528380	2.0669742	7.657786
"Pressure9am"	2.7606212	4.2556409	8.025222
"Pressure3pm"	2.2270478	2.4758015	8.706185
"RainToday"	0	0	1.366858
"RISK_MM"	1.3714387	1.1873350	4.967151

I decided to omit the 3 variables with lowest importance coefficients which are “RainToday”, “RISK_MM” and “WindSpeed3pm”. Because they are not used to construct the decision tree and have lowest importance coefficients, so omit them will have very little impact on performance of predict more humid or not in tomorrow.

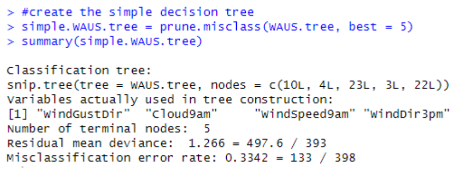
To conclusion, the most important variables are "WindDir3pm", "WindDir9am", "WindGustDir", "Cloud9am". The least important variables are “RainToday”, “RISK_MM” and “WindSpeed3pm”(which we can omit). The other variables are moderately important variables, so we just keep it here.

(9). Here I choose use decision tree to create a classifier which simple enough for a person to be able to classify whether it will be more humid tomorrow or not by hand. So, my aim is to create a simple decision tree, so I prune the original decision tree to reduce its size and number of variables it uses. First, I use cross validation to see what is best size to prune my decision tree.



```
> cv.WAUS.tree = cv.tree(WAUS.tree, FUN = prune.misclass)
> cv.WAUS.tree
$size
 [1] 40 37 31 29 26 25 22 18 15 11 10  8  5  2  1

$dev
 [1] 180 185 183 182 182 180 177 183 188 180 179 176 175 181 199

$k
 [1]      -Inf  0.000000  1.000000  1.500000  1.666667  2.000000  2.333333  2.500000
 [9]  2.666667  3.500000  4.000000  5.000000  6.000000  7.666667 37.000000

$method
[1] "misclass"

attr(,"class")
[1] "prune"         "tree.sequence"
```

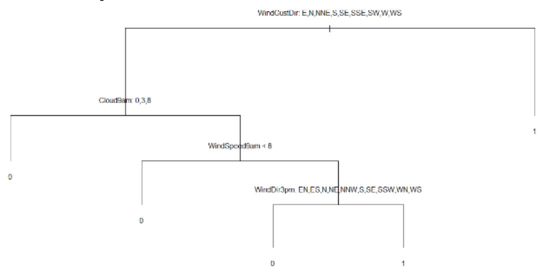
When examining the cross-validation results, I observed that when setting the tree size to 5, I obtained the smallest value for the total deviance ($dev) and a lower value for the cost-complexity pruning parameter ($k). The total deviance represents the overall measure of the model's fit to the data, while the cost-complexity pruning parameter controls the trade-off between tree complexity and model performance. By selecting a smaller tree size, we aim to find a simpler and more interpretable model while still maintaining good predictive accuracy.



```
> #create the simple decision tree
> simple.WAUS.tree = prune.misclass(WAUS.tree, best = 5)
> summary(simple.WAUS.tree)

Classification tree:
snip.tree(tree = WAUS.tree, nodes = c(10L, 4L, 23L, 3L, 22L))
Variables actually used in tree construction:
[1] "WindGustDir"  "Cloud9am"     "WindSpeed9am" "WindDir3pm"  
Number of terminal nodes:  5 
Residual mean deviance:  1.266 = 497.6 / 393 
Misclassification error rate: 0.3342 = 133 / 398 
```

After pruning the original tree, a smaller-sized tree is obtained. This pruned tree consists of the three most important variables that were previously selected, namely "WindGustDir," "Cloud9am," and "WindDir3pm," along with the inclusion of one moderately important variable, "WindSpeed9am." However, the variable "WindDir9am," which was also identified as highly important, is not included in the pruned tree. This omission could be attributed to the pruning process, which aims to control and limit the complexity of the tree to prevent overfitting. During pruning, the decision tree considers the importance and contribution of each variable, selecting those that offer the greatest enhancement to the overall model performance.

The simple decision tree below:


WindGustDir: E,N,NNE,S,SE,SSE,SW,W,WS
Cloud9am: 0,3,8
1
WindSpeed9am < 8
0
WindDir3pm: EN,ES,N,NE,NNW,S,SE,SSW,WN,WS
0
0
1

Predict 1: which is tomorrow is more humid. It happens when:

! WindGustDir: E,N,NNE,S,SE,SSE,SW,W,WS

WindGustDir: E,N,NNE,S,SE,SSE,SW,W,WS & ! Cloud9am: 0,3,8 & WindSpeed9am > 8 & !WindDir3pm: EN,ES,N,NE,NNW,S,SE,SSW,WN,WS

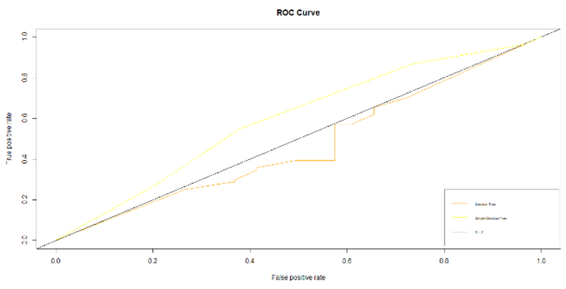
Predict 0: which is tomorrow is no more humid. It happens when:

WindGustDir: E,N,NNE,S,SE,SSE,SW,W,WS & Cloud9am: 0,3,8

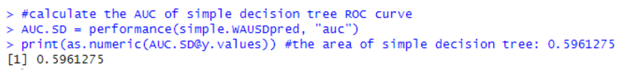
WindGustDir: E,N,NNE,S,SE,SSE,SW,W,WS & ! Cloud9am: 0,3,8 & WindSpeed9am < 8

WindGustDir: E,N,NNE,S,SE,SSE,SW,W,WS & ! Cloud9am: 0,3,8 & WindSpeed9am > 8 & WindDir3pm: EN,ES,N,NE,NNW,S,SE,SSW,WN,WS

Then plot ROC of decision tree (we create in question 4) and ROC of simple decision tree:

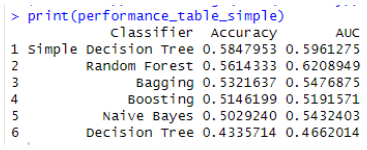


And calculate the AUC of it:



```
> #calculate the AUC of simple decision tree ROC curve
> AUC.SD = performance(simple.WAUSDpred, "auc")
> print(as.numeric(AUC.SD@y.values)) #the area of simple decision tree: 0.5961275
[1] 0.5961275
```

Then create a table to compare them：



```
> print(performance_table_simple)
           Classifier  Accuracy       AUC
1 Simple Decision Tree 0.5847953 0.5961275
2        Random Forest 0.5614333 0.6208949
3              Bagging 0.5321637 0.5476875
4             Boosting 0.5146199 0.5191571
5          Naive Bayes 0.5029240 0.5432403
6        Decision Tree 0.4335714 0.4662014
```

As we observe the table above, the simplified decision tree is pruned from the original decision tree and exhibits higher accuracy and AUC compared to the original tree. This improvement is attributed to the pruning process, where we optimize the size of the decision tree, resulting in lower deviance and a reduced value of k. This pruning technique helps mitigate the issue of overfitting, ultimately enhancing the performance of the decision tree model.

The improvement is significant as the original decision tree exhibits the lowest accuracy and AUC values. In contrast, the simplified decision tree shows the highest accuracy and the second highest AUC value, with only the Random Forest surpassing it. The AUC metric is influenced by a model's capacity to classify positive and negative samples effectively. As a composite model, the Random Forest combines the predictive outcomes of multiple decision trees, potentially resulting in stronger classification capabilities and superior performance in terms of AUC. Boosting, bagging, and random forest are ensemble learning methods that using multiple decision trees and aggregating their predictions to reduce individual tree bias and variance, thereby enhancing the overall model's performance, they supposed to have better accuracy. However, due to the complexity and randomness of ensemble methods, they may perform slightly worse than pruned decision trees in certain cases.

To conclusion, simple decision tree is much better than original decision tree (we create at question 4).

(10). Create the best tree-based classifier based on what we create in Q4.

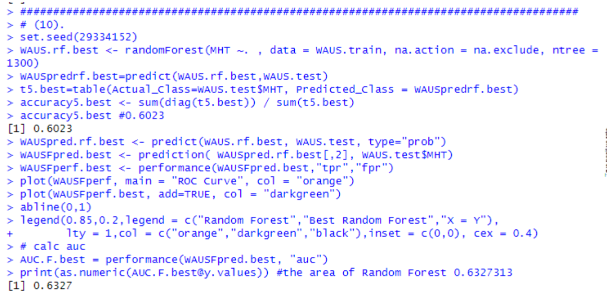
Here, I have chosen to focus on improving the performance of the random forest classifier that was created in question 4. The rationale behind this selection is based on the performance_table_simple provided above, which indicates that the original "Random Forest" classifier has the second highest accuracy and the highest AUC. Since it already performs well, improving it further could potentially lead to the best possible classifier.

The Random Forest is an ensemble method that combines multiple decision trees and uses bootstrapping and feature randomness to improve performance and reduce overfitting. Which means combine a greater number of trees will make it performance better. So here I change the ntree to create a new model, the default value for the ntree parameter in the randomForest function is 500. Then we want to use more trees to combine, so we start from 1000 trees and end with 2000(if more than 2000 trees, it will spend too long time to generate the model).

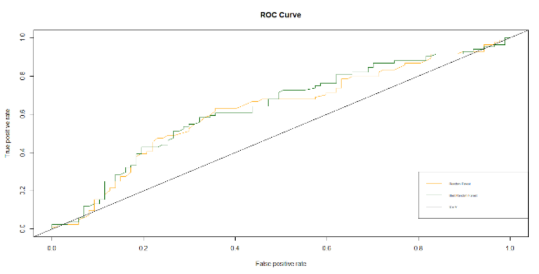
The table below presents the accuracy achieved by the Random Forest classifier for different numbers of trees:

Number of trees	1000	1100	1200	1300	1400	1500	1600	1700	1800	1900	2000
accuracy	0.5847	0.5964	0.5964	0.6023	0.6023	0.5906	0.5906	0.5964	0.5964	0.5906	0.5906

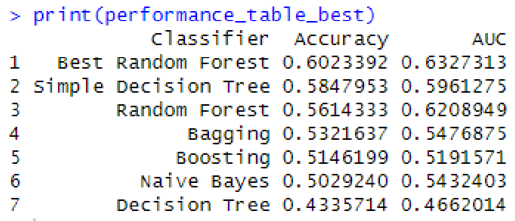
Here we can see accuracy are highest when tree number is 1300 or 1400. So, we choose 1300 because it spends less time to generate the model. Then I use this code which are what I used in Q4 and a bit change with variable name and number of trees. Also, I print the ROC curve with Random Forest and best random forest in one plot.



```
> ####################################################################################
> # (10).
> set.seed(29334152)
> WAUS.rf.best <- randomForest(MHT ~. , data = WAUS.train, na.action = na.exclude, ntree =
1300)
> WAUSpredrf.best=predict(WAUS.rf.best,WAUS.test)
> t5.best=table(Actual_Class=WAUS.test$MHT, Predicted_Class = WAUSpredrf.best)
> accuracy5.best <- sum(diag(t5.best)) / sum(t5.best)
> accuracy5.best #0.6023
[1] 0.6023
> WAUSpred.rf.best <- predict(WAUS.rf.best, WAUS.test, type="prob")
> WAUSFpred.best <- prediction( WAUSpred.rf.best[,2], WAUS.test$MHT)
> WAUSFperf.best <- performance(WAUSFpred.best,"tpr","fpr")
> plot(WAUSFperf, main = "ROC Curve", col = "orange")
> plot(WAUSFperf.best, add=TRUE, col = "darkgreen")
> abline(0,1)
> legend(0.85,0.2,legend = c("Random Forest","Best Random Forest","X = Y"),
+         lty = 1,col = c("orange","darkgreen","black"),inset = c(0,0), cex = 0.4)
> # calc auc
> AUC.F.best = performance(WAUSFpred.best, "auc")
> print(as.numeric(AUC.F.best@y.values)) #the area of Random Forest 0.6327313
[1] 0.6327
```



And I make a performance table below:



```
> print(performance_table_best)
             Classifier  Accuracy       AUC
1   Best Random Forest 0.6023392 0.6327313
2 Simple Decision Tree 0.5847953 0.5961275
3        Random Forest 0.5614333 0.6208949
4              Bagging 0.5321637 0.5476875
5             Boosting 0.5146199 0.5191571
6          Naive Bayes 0.5029240 0.5432403
7        Decision Tree 0.4335714 0.4662014
```

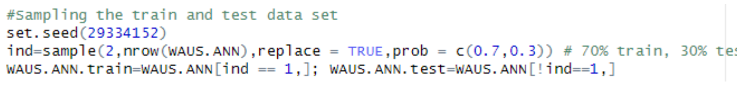
See the picture above:

We can find that the “Best Random Forest” (which is the improved Random Forest) have both highest accuracy and AUC. Which means what we success to make a best classifier here.

(11). Create ANN classifier and compare with previous classifiers.

The process for data before fit ANN: 1. Remove rows containing missing values 2. Recode the output Class as numeric 3. Create training and test sets 4. Fit neural network and test accuracy etc.

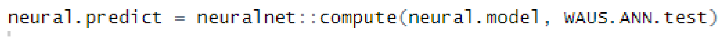
First, before performing the ANN operation, we need to first process the data so that it is all numerical variables and as large as possible. Therefore, I have decided to omit the three least important variables("RainToday", "RISK_MM", "WindSpeed3pm") we obtained in question 8 (because these variables have little impact on predicting whether tomorrow will be wetter), and “Year” is needed to omit because it is not useful in this predict. Also, there are some variables("WindDir3pm", "WindDir9am", "WindGustDir") is not numeric variables but it cannot be omitted, because we find that are most important variables in question 8, so we make it be binary variable which can be used in ANN and combine it with dataset. Then omit the rows with NA values, so that we can obtain more data.



```
#Sampling the train and test data set
set.seed(29334152)
ind=sample(2,nrow(WAUS.ANN),replace = TRUE,prob = c(0.7,0.3)) # 70% train, 30% test
WAUS.ANN.train=WAUS.ANN[ind == 1,]; WAUS.ANN.test=WAUS.ANN[!ind==1,]
```

Then just sampling the data as 70% for train data and 30% for test data.

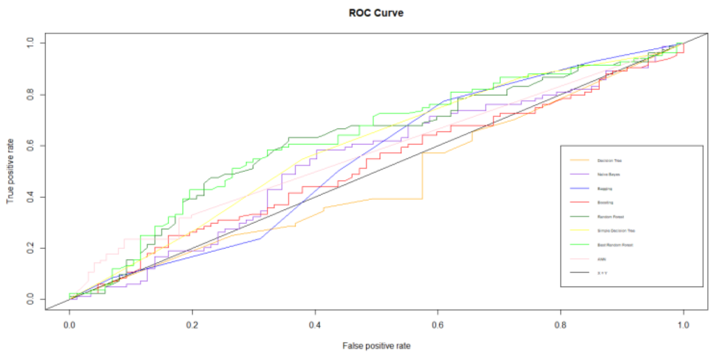
Then use the neuralnet method to make ANN model, the MHT is the target variable that I trying to predict, and the other variables are all numeric independent variables(and choose hidden as 9,5,4,3 which have four layer and number of neurons in each layer. We make sure number of nodes are not too much, if it has too many nodes, the neural network becomes like a memory bank which means it will work well at training set, but not work well for other data). Use the WAUS.ANN.train as training dataset to make model.



```
neural.predict = neuralnet::compute(neural.model, WAUS.ANN.test)
```

Then use compute method and test dataset to predict it and make a confusion matrix table for it and calculate its accuracy. We can see accuracy below is 0.5914.

```
> # now round these down to integers
> neural.predict.label = as.data.frame(round(neural.predict$net.result,0))
> neural.cm = table(observed = WAUS.ANN.test$MHT, predicted = neural.predict.label$V1)
> neural.cm
        predicted
observed  0  1
       0 95  6
       1 70 15
> #calculate the accuracy and print it out
> neural.accuracy <- sum(diag(neural.cm)) / sum(neural.cm)
> neural.accuracy # 0.5914
[1] 0.5914
```



And then we calculate the AUC and make a table to compare it with all the classifier we created before. Also put ROC curve of all of them in one plot.

```
> print(performance_table_best)
            Classifier  Accuracy       AUC
1   Best Random Forest 0.6023392 0.6327313
2                  ANN 0.5913978 0.5708212
3 Simple Decision Tree 0.5847953 0.5961275
4        Random Forest 0.5614333 0.6208949
5              Bagging 0.5321637 0.5476875
6             Boosting 0.5146199 0.5191571
7          Naive Bayes 0.5029240 0.5432403
8        Decision Tree 0.4335714 0.4662014
```

Here we can see "ANN" are better than most of classifiers except the "Best Random Forest" because of it has second higher accuracy and 4th higher AUC(because there are two improved classifiers: "Simple Decision Tree", "Best Random Forest", it we do not count these, it should be highest accuracy and second higher AUC). And our data are balance data, accuracy here would be a better measure of the model's performance. So, we can say it is best classifier in all classifiers(no improved classifiers).

The reason why this happens might because ANN have nonlinear relationship modelling ability which means ANN can learn and capture nonlinear relationships, while certain traditional classifiers (such as decision trees or naive Bayes) are more suitable for linear relationships. So, our data sets contain complex non-linear relationships, then ANN be able to better adapt the data and provide more accurate classifications. Also, ANN has highly flexible structure which means ANNs can have multiple hidden layers and many neurons, allowing them to adapt to a variety of complex patterns and features. In contrast, other classifiers may have simpler structures that do not provide the same flexibility and adaptability. These are the reason why ANN better than all traditional classifiers this time.

(12). Use the new classifier which are not in course and do the same thing before.

I'm study how to use it from these web https://discuss.xgboost.ai/, https://towardsdatascience.com/the-notorious-xgboost-c7f7adc4c183

Here I choose to use the XGBoost which is a tree-based classifier. This is a new classifier outside of our course. The reason why I choose this classifier is that XGBoost have these advantages: Efficiency, Flexibility, High predictive performance.

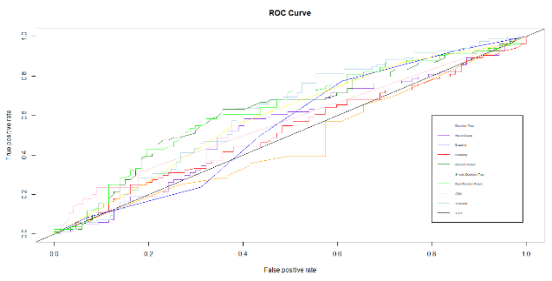
The Efficiency: XGBoost has high efficiency in training and prediction. It accelerates the training process of the model through parallel computing(concurrent create trees) and approximation algorithm. Also, it uses a specific data structure to improve the speed of prediction.

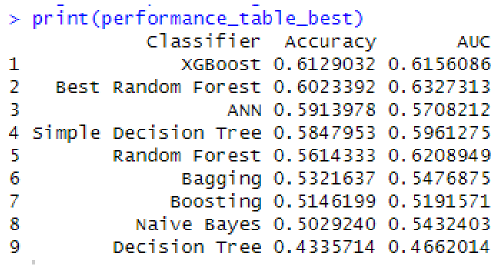
The High predictive performance: XGBoost typically achieves high performance in prediction tasks by optimizing the objective function and using gradient enhancement algorithms.

Here we can use the WAUS.important which are the variable of the we create before in Q11, it omits the not important variables, not useful variables and drop all the row with NA value. This thing will make our work easier. And consider to the important variables are characteristic type, so we do the same thing in Q11, because it cannot be just omitted when we predict either tomorrow more humid or not.

After we fix the data type, we need to set seed as student number and sampling the train and test data set. Due to XGBoost need to use matrix type variables, so we convert data frame to matrix and data is all the things without MHT and label is MHT (it is like MHT ~ .). now we can use this to build XGBoost model, (WAUS.XGBoost.model = xgboost::xgboost(data = train.other, label = train.MHT, nrounds = 200, objective = "binary:logistic", verbose = 0, max_depth = 1, eta = 0.59, colsample_bytree = 0.6)) use the xgboost function. And change number of rounds(nrounds), max depth, learning rate(eta) and column subsampling by tree(colsample_bytree) to make model be better. The approach I use is that all this start with small number and add it up see how it change, if it increases make accuracy and AUC increase, we increase it, if it increases make accuracy or AUC decrease, we do increase. Then try this more round make we find most fitted one for the data.

After above step, the last thing we need to do is just do the same thing we have done before. Test model performance, make confusion matrix table and calculate the accuracy. And know it performance, plot the ROC curve of it compare with other classifiers:




```
> print(performance_table_best)
             Classifier  Accuracy       AUC
1               XGBoost 0.6129032 0.6156086
2   Best Random Forest 0.6023392 0.6327313
3                   ANN 0.5913978 0.5708212
4 Simple Decision Tree 0.5847953 0.5961275
5         Random Forest 0.5614333 0.6208949
6               Bagging 0.5321637 0.5476875
7              Boosting 0.5146199 0.5191571
8           Naive Bayes 0.5029240 0.5432403
9         Decision Tree 0.4335714 0.4662014
```

Last thing is making a performance table to see how good is XGBoost classifier. The table shows us that XGBoost is the best classifier here, because I try lots of the Configuration of model parameters and keep the best one. It has highest accuracy and third highest AUC, so we can say this is the best Tree-Based classifier we create in this assignment.

Appendix:

```
"
Project name: Assignment2 of FIT3152
Author: Chenlongjie Weng
Student number:29334152
"
# import the package we need at start
#install.packages("tree")
library(tree)
#install.packages("e1071")
library(e1071)
#install.packages(("ROCR"))
library(ROCR)
#install.packages("randomForest")
library(randomForest)
#install.packages("adabag")
library(adabag)
#install.packages("rpart")
library(rpart)
#install.packages("neuralnet")
#library(neuralnet)
#detach("package:neuralnet", unload = TRUE)
library(car)
library(dplyr)

################################################################################
#######
# First clear the work space and read file
setwd("C:/Users/WENGCHENLONGJIE/Desktop/FIT3152/Assignment2")
rm(list = ls())
WAUS <- read.csv("HumidPredict2023D.csv")
L <- as.data.frame(c(1:49))
set.seed(29334152) # 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

################################################################################
#####
# (1). Explore the data
MHT1=as.factor(WAUS$MHT)
summary(MHT1)
```

```
################################################################################
#######
# （2). pre-processing to make the data set suitable for the model fitting
WAUS_original = WAUS #keep the original data set just in case
#remove all the variable I decide to remove as what I say in report
WAUS <- WAUS %>% dplyr::select(-c("Year")) #drop Year
#remove all the Na's
WAUS = WAUS[complete.cases(WAUS),]
#show how many location I have after remove NAs.
unique(WAUS$Location)
#then change the type of variables which should be a factor variables
WAUS$WindGustDir <- as.factor(WAUS$WindGustDir) #convert non-numeric variables to factor
variables
WAUS$WindDir9am <- as.factor(WAUS$WindDir9am) #convert non-numeric variables to factors
variables
WAUS$WindDir3pm <- as.factor(WAUS$WindDir3pm) #convert non-numeric variables to factors
variables
WAUS$Cloud3pm <- as.factor(WAUS$Cloud3pm) #convert non-numeric variables to factors
variables
WAUS$Cloud9am <- as.factor(WAUS$Cloud9am) #convert non-numeric variables to factors
variables
WAUS$Location <- as.factor(WAUS$Location) #convert non-numeric variables to factors
variables
WAUS$MHT = as.factor(WAUS$MHT) #set this be factor which means more humid or not
WAUS$RainToday = as.factor(WAUS$RainToday) #convert non-numeric variables to factors
variables

################################################################################
#######
# (3). Divide data into a 70% training and 30% test set
set.seed(29334152) #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,]

################################################################################
#######
# (4). create classification model by each technique
################ decision tree #################
WAUS.tree=tree(MHT~.,data = WAUS.train)
plot(WAUS.tree)
text(WAUS.tree, pretty = 1)

################ Naive Bayes #################
```

```
WAUS.bayes=naiveBayes(MHT~.,data = WAUS.train)

################ Bagging #################
WAUS.bag = bagging(MHT ~. , data = WAUS.train, mfinal=5)

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

################ Random Forest #################
WAUS.rf <- randomForest(MHT ~. , data = WAUS.train, na.action = na.exclude)

################################################################################
#######
# (5). use test data to get Confusion matrix and report the accuracy of each model
set.seed(29334152)
################ decision tree #################
WAUS.pred = predict(WAUS.tree, WAUS.test, type = "class")
t1=table(actual=WAUS.test$MHT, predicted = WAUS.pred)
t1
accuracy1 <- sum(diag(t1)) / sum(t1)
accuracy1 #0.4561, 0.4386, 0.4269, 0.4211, 0.4327, 0.4444, 0.4152
accuracy1=(0.4561+0.4386+0.4269+0.4211+0.4327+0.4444+0.4152)/7 #0.4336

################ Naive Bayes #################
WAUS.predbayes = predict(WAUS.bayes, WAUS.test, type = "class")
t2=table(actual=WAUS.test$MHT, predicted = WAUS.predbayes)
t2
accuracy2 <- sum(diag(t2)) / sum(t2)
accuracy2 #0.5029

################ Bagging #################
WAUSpred.bag=predict.bagging(WAUS.bag,WAUS.test)
t3=table(observed=WAUS.test$MHT,predicted=WAUSpred.bag$class)
t3
accuracy3 <- sum(diag(t3)) / sum(t3)
accuracy3 #0.5322

################ Boosting #################
WAUSpred.boost=predict.boosting(WAUS.Boost,WAUS.test)
#WAUSpred.boost$prob
t4=table(observed=WAUS.test$MHT,predicted=WAUSpred.boost$class)
accuracy4 <- sum(diag(t4)) / sum(t4)
accuracy4 #0.5146
```

```
################ Random Forest #################
WAUSpredrf=predict(WAUS.rf,WAUS.test)
t5=table(Actual_Class=WAUS.test$MHT, Predicted_Class = WAUSpredrf)
accuracy5 <- sum(diag(t5)) / sum(t5)
accuracy5 #0.5614+0.5673+0.5556
accuracy5 = (0.5614+0.5673+0.5556)/3 #0.5614


################################################################################
#######
# (6).
################ decision tree #################
WAUS.pred.tree = predict(WAUS.tree, WAUS.test, type = "vector")
WAUSDpred <- prediction(WAUS.pred.tree[,2], WAUS.test$MHT)
WAUSDperf <- performance(WAUSDpred,"tpr","fpr")
plot(WAUSDperf, main = "ROC Curve", col = "orange")
abline(0,1)
# calc auc
AUC.D = performance(WAUSDpred, "auc")
print(as.numeric(AUC.D@y.values)) #the area of decision tree 0.4662014


################ Naive Bayes #################
WAUSpred.bayes = predict(WAUS.bayes, WAUS.test, type = 'raw')
WAUSBpred <- prediction( WAUSpred.bayes[,2], WAUS.test$MHT)
WAUSBperf <- performance(WAUSBpred,"tpr","fpr")
plot(WAUSBperf, add=TRUE, col = "blueviolet")
# calc auc
AUC.N = performance(WAUSBpred, "auc")
print(as.numeric(AUC.N@y.values)) #the area of Naive Bayes 0.5432403

################ Bagging #################
WAUSBagpred <- prediction( WAUSpred.bag$prob[,2], WAUS.test$MHT)
WAUSBagperf <- performance(WAUSBagpred,"tpr","fpr")
plot(WAUSBagperf, add=TRUE, col = "blue")
# calc auc
AUC.B = performance(WAUSBagpred, "auc")
print(as.numeric(AUC.B@y.values)) #the area of decision tree 0.5476875

################ Boosting #################
WAUSBoostpred <- prediction( WAUSpred.boost$prob[,2], WAUS.test$MHT)
WAUSBoostperf <- performance(WAUSBoostpred,"tpr","fpr")
plot(WAUSBoostperf, add=TRUE, col = "red")
# calc auc
```

```
AUC.Boost = performance(WAUSBoostpred, "auc")
print(as.numeric(AUC.Boost@y.values)) #the area of Boosting 0.5191571


################ Random Forest #################
WAUSpred.rf <- predict(WAUS.rf, WAUS.test, type="prob")
WAUSFpred <- prediction( WAUSpred.rf[,2], WAUS.test$MHT)
WAUSFperf <- performance(WAUSFpred,"tpr","fpr")
plot(WAUSFperf, add=TRUE, col = "darkgreen")
# calc auc
AUC.F = performance(WAUSFpred, "auc")
print(as.numeric(AUC.F@y.values)) #the area of Random Forest 0.6208949

legend(0.85,0.44,legend = c("Decision Tree","Naive Bayes","Bagging","Boosting","Random
Forest","X = Y"),
      lty = 1,col = c("orange","blueviolet","blue","red","darkgreen","black"),inset = c(0,0), cex =
0.5)



################################################################################
#######
# (7). performance table for comparing result
performance_table = data.frame(Classifier = c('Decision Tree','Naive
Bayes','Bagging','Boosting','Random Forest'),
                              Accuracy = c(accuracy1,accuracy2,accuracy3,accuracy4,accuracy5),
                              AUC = c(0.4662014,0.5432403,0.5476875,0.5191571,0.6208949)) %>%
arrange(desc(Accuracy))
print(performance_table)



################################################################################
#######
# (8). determine the most important variables in predicting whether it will be more humid
tomorrow or not
#Attribute importance for each one(bigger value is more important)
cat("\n#Decision Tree Attribute Importance\n")
print(summary(WAUS.tree))
cat("\n#Bagging Attribute Importance\n")
print(WAUS.bag$importance)
cat("\n#Boosting Attribute Importance\n")
print(WAUS.Boost$importance)
```

```
cat("\n#Random Forest Attribute Importance\n")
print(WAUS.rf$importance)


#################################################################################
#######
# (9). create a simple classifier (decision tree)
set.seed(29334152) #set seed to make sure each time get same result
cv.WAUS.tree = cv.tree(WAUS.tree, FUN = prune.misclass)
cv.WAUS.tree
#create the simple decision tree
simple.WAUS.tree = prune.misclass(WAUS.tree, best = 5)
summary(simple.WAUS.tree)
plot(simple.WAUS.tree)
text(simple.WAUS.tree, pretty = 1)
#then predict the simple decision tree model using test data
simple.WAUS.tree.pred = predict(simple.WAUS.tree, WAUS.test, type = "class")
#confusion matrix
simple.WAUS.tree.cm = table(actual = WAUS.test$MHT, predicted = simple.WAUS.tree.pred)
simple.WAUS.tree.cm
accuracy.simple <- sum(diag(simple.WAUS.tree.cm)) / sum(simple.WAUS.tree.cm)
accuracy.simple #The Accuracy for Simple Decision Tree: 0.5847953
#then find simple decision tree confidence
simple.WAUS.tree.pred.confidence = predict(simple.WAUS.tree, WAUS.test, type = "vector")
#plot ROC curve for decision tree and simple decision tree
plot(WAUSDperf, main = "ROC Curve", col = "orange")
abline(0,1)
simple.WAUSDpred <- prediction(simple.WAUS.tree.pred.confidence[,2], WAUS.test$MHT)
simple.WAUSDperf <- performance(simple.WAUSDpred,"tpr","fpr")
plot(simple.WAUSDperf, add=T, col = "yellow")
legend(0.8,0.25,legend = c("Decision Tree","Simple Decision Tree","X = Y"),
    lty = 1,col = c("orange","yellow","black"),inset = c(0,0), cex = 0.5)
#calculate the AUC of simple decision tree ROC curve
AUC.SD = performance(simple.WAUSDpred, "auc")
print(as.numeric(AUC.SD@y.values)) #the area of simple decision tree: 0.5961275
#now create a table to easy compare each method
performance_table_simple = data.frame(Classifier = c('Decision Tree','Naive
Bayes','Bagging','Boosting','Random Forest','Simple Decision Tree'),
                              Accuracy =
c(accuracy1,accuracy2,accuracy3,accuracy4,accuracy5,accuracy.simple),
                              AUC =
c(0.4662014,0.5432403,0.5476875,0.5191571,0.6208949,0.5961275)) %>%
arrange(desc(Accuracy))
print(performance_table_simple)
```

```
##################################################################################
#######
# (10).
set.seed(29334152)
WAUS.rf.best <- randomForest(MHT ~. , data = WAUS.train, na.action = na.exclude, ntree = 1300)
WAUSpredrf.best=predict(WAUS.rf.best,WAUS.test)
t5.best=table(Actual_Class=WAUS.test$MHT, Predicted_Class = WAUSpredrf.best)
accuracy5.best <- sum(diag(t5.best)) / sum(t5.best)
accuracy5.best #0.6023
WAUSpred.rf.best <- predict(WAUS.rf.best, WAUS.test, type="prob")
WAUSFpred.best <- prediction( WAUSpred.rf.best[,2], WAUS.test$MHT)
WAUSFperf.best <- performance(WAUSFpred.best,"tpr","fpr")
plot(WAUSFperf, main = "ROC Curve", col = "orange")
plot(WAUSFperf.best, add=TRUE, col = "darkgreen")
abline(0,1)
legend(0.8,0.3,legend = c("Random Forest","Best Random Forest","X = Y"),
    lty = 1,col = c("orange","darkgreen","black"),inset = c(0,0), cex = 0.5)
# calc auc
AUC.F.best = performance(WAUSFpred.best, "auc")
print(as.numeric(AUC.F.best@y.values)) #the area of Random Forest 0.6327313
#now create a table to easy compare each method
performance_table_best = data.frame(Classifier = c('Decision Tree','Naive
Bayes','Bagging','Boosting','Random Forest','Simple Decision Tree','Best Random Forest'),
                              Accuracy =
c(accuracy1,accuracy2,accuracy3,accuracy4,accuracy5,accuracy.simple,accuracy5.best),
                              AUC =
c(0.4662014,0.5432403,0.5476875,0.5191571,0.6208949,0.5961275,0.6327313)) %>%
arrange(desc(Accuracy))
print(performance_table_best)


##################################################################################
#######
# (11). create ANN and compare it with all other classifier
library(car)
library(neuralnet)
set.seed(29334152) #student ID as random seed

#first we omit less important variables
less.important = c("RainToday", "RISK_MM", "WindSpeed3pm")
# use dplyr:: to make sure the select method is from this package
# minus sign (-) is used in the select() function from the "dplyr" package to exclude specific
columns from a data frame.
WAUS.important = WAUS_original %>% dplyr::select(-less.important)
#then we omit the no useful variables
WAUS.important <- WAUS.important %>% dplyr::select(-c("Year")) #drop Year

#below is convert the all the non numeric important variable to factors,
str(WAUS.important) #use to make sure the most important variables: "WindDir3pm",
"WindDir9am", "WindGustDir", "Cloud9am" in correct type
WAUS.important$WindGustDir <- as.factor(WAUS.important$WindGustDir) #convert non-
numeric variables to factor variables
WAUS.important$WindDir9am <- as.factor(WAUS.important$WindDir9am) #convert non-
numeric variables to factors variables
WAUS.important$WindDir3pm <- as.factor(WAUS.important$WindDir3pm) #convert non-
numeric variables to factors variables

#remove all the Na's
WAUS.important = WAUS.important[complete.cases(WAUS.important),]
WAUS.ANN = WAUS.important

#for now we can only contain the numeric variable then we can use it to fit ANN
#but WindGustDir+WindDir9am+WindDir3pm these three are not numeric
#we convert all categorical variables to binary variables, each indicating if the current
#data entry belongs to a category, being either 1 or 0.
#because these three are most important so we convert in to binary
#then we can use it
WindDir_numberic = model.matrix(~WindGustDir+WindDir9am+WindDir3pm, data=WAUS.ANN)
WAUS.ANN = cbind(WAUS.ANN, WindDir_numberic)

#Sampling the train and test data set
set.seed(29334152)
ind=sample(2,nrow(WAUS.ANN),replace = TRUE,prob = c(0.7,0.3)) # 70% train, 30% test
WAUS.ANN.train=WAUS.ANN[ind == 1,]; WAUS.ANN.test=WAUS.ANN[!ind==1,]

str(WAUS.ANN) #see all variables type, only use numeric variables
set.seed(29334152)
neural.model = neuralnet::neuralnet(MHT ~ Location + MinTemp + MaxTemp + Rainfall +
Evaporation
                        + Sunshine + WindGustSpeed + WindSpeed9am + Pressure9am +
Pressure3pm
                        + Cloud9am + Cloud3pm + Temp9am + Temp3pm + WindGustDirENE
                        + WindGustDirESE + WindGustDirN + WindGustDirNE + WindGustDirNNE
                        + WindGustDirNNW + WindGustDirNW + WindGustDirS
                        + WindGustDirSE  + WindGustDirSSE + WindGustDirSSW
```

```
                    + WindGustDirSW + WindGustDirW + WindGustDirWNW +
WindGustDirWSW
                    + WindDir9amENE + WindDir9amESE + WindDir9amN + WindDir9amNE
                    + WindDir9amNNE + WindDir9amNNW + WindDir9amNW + WindDir9amS
                    + WindDir9amSE + WindDir9amSSE + WindDir9amSSW + WindDir9amSW
                    + WindDir9amW + WindDir9amWNW + WindDir9amWSW +
WindDir3pmENE
                    + WindDir3pmESE + WindDir3pmN + WindDir3pmNE + WindDir3pmNNE
                    + WindDir3pmNNW  + WindDir3pmNW + WindDir3pmS + WindDir3pmSE
                    + WindDir3pmSSE + WindDir3pmSSW + WindDir3pmSW + WindDir3pmW
                    + WindDir3pmWNW + WindDir3pmWSW, data = WAUS.ANN.train,
                    hidden = c(9,5,4,3), #number of hidden neurons (vertices) in each layer
                    linear.output = FALSE) #making a binary classification model

neural.predict = neuralnet::compute(neural.model, WAUS.ANN.test)

# now round these down to integers
neural.predict.label = as.data.frame(round(neural.predict$net.result,0))
neural.cm = table(observed = WAUS.ANN.test$MHT, predicted = neural.predict.label$V1)
neural.cm
#calculate the accuracy and print it out
neural.accuracy <- sum(diag(neural.cm)) / sum(neural.cm)
neural.accuracy # 0.5914

#the ROC for the ANN
neural.prediction = ROCR::prediction(neural.predict$net.result, WAUS.ANN.test$MHT)
neural.performance = performance(neural.prediction, "tpr","fpr")

#plot ROC for all classifiers
plot(WAUSDperf, main = "ROC Curve", col = "orange")
abline(0,1)
plot(WAUSBperf, add=TRUE, col = "blueviolet")
plot(WAUSBagperf, add=TRUE, col = "blue")
plot(WAUSBoostperf, add=TRUE, col = "red")
plot(WAUSFperf, add=TRUE, col = "darkgreen")
plot(simple.WAUSDperf, add=T, col = "yellow")
plot(WAUSFperf.best, add=TRUE, col = "green")
plot(neural.performance,col = 'pink',add = TRUE) #the ANN here

legend(0.8,0.6,legend = c("Decision Tree","Naive Bayes","Bagging","Boosting","Random
Forest","Simple Decision Tree","Best Random Forest","ANN","X = Y"),
    lty = 1,col =
c("orange","blueviolet","blue","red","darkgreen","yellow","green","pink","black"),inset = c(0,0),
cex = 0.45)
```

```
#calculate the AUC of ANN
neural.AUC = performance(neural.prediction, 'auc') #calculate AUC
neural.AUC = neural.AUC@y.values[[1]] #extract only the AUC value
neural.AUC #0.5708212

#then compare ANN with the classifiers we create before
performance_table_best = data.frame(Classifier = c('Decision Tree','Naive
Bayes','Bagging','Boosting','Random Forest','Simple Decision Tree','Best Random Forest','ANN'),
                              Accuracy =
c(accuracy1,accuracy2,accuracy3,accuracy4,accuracy5,accuracy.simple,accuracy5.best,neural.acc
uracy),
                              AUC =
c(0.4662014,0.5432403,0.5476875,0.5191571,0.6208949,0.5961275,0.6327313,0.5708212)) %>
% arrange(desc(Accuracy))
print(performance_table_best)

detach(package:neuralnet) #detach the neuralnet package to avoid conflicts with other packages



################################################################################
#######
# (12). use new model from web, which are XGBoost.
#install.packages("xgboost")
library(xgboost)
WAUS.XGboost = WAUS.important

WindDir_numberic = model.matrix(~WindGustDir+WindDir9am+WindDir3pm,
data=WAUS.XGboost)
WAUS.XGboost = cbind(WAUS.XGboost, WindDir_numberic)

WAUS.XGboost = WAUS.XGboost %>%
  dplyr::select(-c("WindGustDir",
            "WindDir9am",
            "WindDir3pm"))

#Sampling the train and test data set
set.seed(29334152)
ind=sample(2,nrow(WAUS.XGboost),replace = TRUE,prob = c(0.7,0.3)) # 70% train, 30% test
WAUS.XGboost.train=WAUS.XGboost[ind == 1,]; WAUS.XGboost.test=WAUS.XGboost[!ind==1,]

#convert data frame to matrix
train.MHT = as.matrix(WAUS.XGboost.train$MHT)
train.other = as.matrix(WAUS.XGboost.train[,-15])
test.MHT = as.matrix(WAUS.XGboost.test$MHT)
test.other = as.matrix(WAUS.XGboost.test[,-15])

#build XGBoost model
WAUS.XGBoost.model = xgboost::xgboost(data = train.other, label = train.MHT, nrounds = 200,
objective = "binary:logistic", verbose = 0, max_depth = 1, eta = 0.59, colsample_bytree = 0.6)

#test the model's performance
WAUS.XGBoost.pred = predict(WAUS.XGBoost.model, test.other)
prediction = as.numeric(WAUS.XGBoost.pred > 0.5) #here we see greater than 0.5 as 1, else 0
XGBoost.cm = table(predicted = prediction, Actual = test.MHT)
XGBoost.cm
#calculate accuracy
XGBoost.accuracy <- sum(diag(XGBoost.cm)) / sum(XGBoost.cm)
XGBoost.accuracy # 1

#XGBoost
xgboost.pred <- prediction(WAUS.XGBoost.pred, WAUS.XGboost.test$MHT)
xgboost.perf <- ROCR::performance(xgboost.pred,"tpr","fpr")

#ROC curve of all classifier in one plot to compare
plot(WAUSDperf, main = "ROC Curve", col = "orange")
abline(0,1)
plot(WAUSBperf, add=TRUE, col = "blueviolet")
plot(WAUSBagperf, add=TRUE, col = "blue")
plot(WAUSBoostperf, add=TRUE, col = "red")
plot(WAUSFperf, add=TRUE, col = "darkgreen")
plot(simple.WAUSDperf, add=T, col = "yellow")
plot(WAUSFperf.best, add=TRUE, col = "green")
plot(neural.performance,col = 'pink',add = TRUE) #the ANN here
plot(xgboost.perf, col = 'lightblue',add = TRUE) #the XGBoost here

legend(0.8,0.6,legend = c("Decision Tree","Naive Bayes","Bagging","Boosting","Random
Forest","Simple Decision Tree","Best Random Forest","ANN","XGBoost","X = Y"),
    lty = 1,col =
c("orange","blueviolet","blue","red","darkgreen","yellow","green","pink","lightblue","black"),ins
et = c(0,0), cex = 0.45)

#calculate auc
xgboost_AUC = ROCR::performance(xgboost.pred, "auc")
xgboost_AUC <- as.numeric(format(xgboost_AUC@y.values[[1]]))
xgboost_AUC #0.5414094
```

```
#the table to show how it performance in all classifier.
performance_table_best = data.frame(Classifier = c('Decision Tree','Naive
Bayes','Bagging','Boosting','Random Forest','Simple Decision Tree','Best Random
Forest','ANN','XGBoost'),
                              Accuracy =
c(accuracy1,accuracy2,accuracy3,accuracy4,accuracy5,accuracy.simple,accuracy5.best,neural.acc
uracy,XGBoost.accuracy),
                              AUC =
c(0.4662014,0.5432403,0.5476875,0.5191571,0.6208949,0.5961275,0.6327313,0.5708212,xgbo
ost_AUC)) %>% arrange(desc(Accuracy))
print(performance_table_best)
```