FIT3152 Assignment 2 Report

Student name : Ng Chen Ting (31861148)

Question 1

The dataset obtained has 2000 rows with 22 columns, there are 4 columns of type character, 10 columns of type numeric and 8 columns of type int including the class attribute MHT. 921 out of 2000 of the records is where it is more humid than the previous day which takes up to 46.05%. On the other hand, the opposite has 908 records which gives 45.4%. The remaining 171 records is where Humidity column is left empty.

From the summary, it can be observed that Evaporation, Sunshine, Cloud9am and Cloud3pm has over 1000 NA values making them inefficient to be include in the analysis which therefore can be omitted.

Question 2

In this case, it is a good practice to convert all char types to factor type.

In order to use decision trees, the class attribute, MHT has to be a factor type.

Since we are predicting MHT, it is redundant to keep rows where MHT is NA.

Because we are predicting future events, we can also remove the Year column.

Question 3, 4

The dataset is divided into 70% and 30% for training and testing

All models are created at their default settings.

Question 5

A function acc_func is created to calculate the accuracy from confusion matrix. This function takes a confusion matrix as input and gets its TP, TN, FP and FN to perform the calculation below:

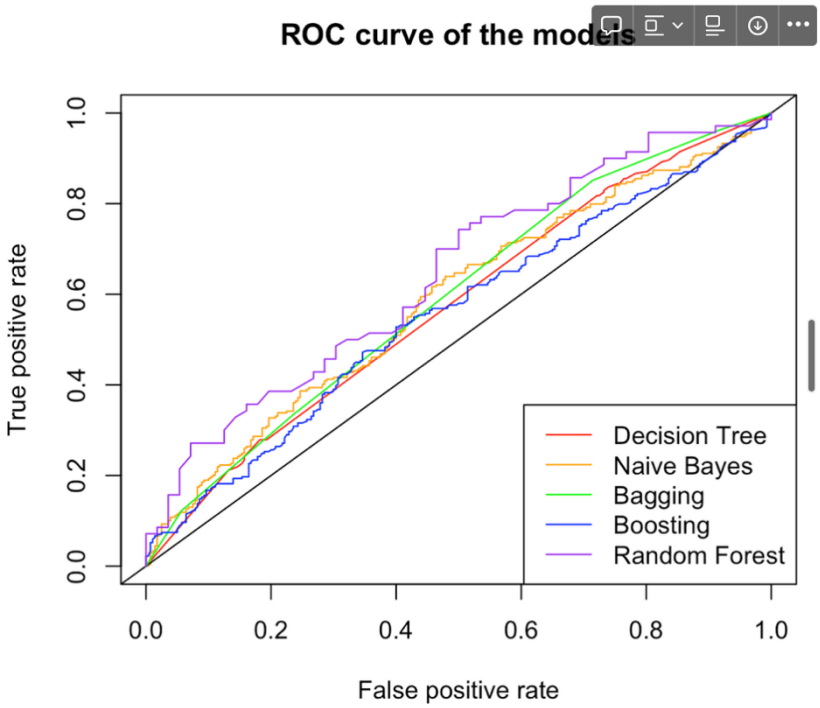
Accuracy = $(TP + TN) / (TP + TN + FP + FN)$

Question 6

An ROC curve is created for all the models and using the ROC curve, the area under the curve (AUC) is calculated. The ROC is used to visualise the model's performance across different classification models by it's TPR and FPR

Question 7

From the ROC curve figure, it can be observed that Random Forest (purple line) has the largest area under the curve. Aside from that, it also has the best accuracy.



From the table, the figures show that random forest outdid the rest of the models from its large difference between its values and the second best classifier which is bagging.

```
> Q7Summary
                ModelAccuracyList AucOfModelList
Decision Tree           0.5428051       0.573274
Naive Bayes             0.5482696      0.5865109
Bagging                 0.5591985      0.5934745
Boosting                0.5500911        0.55626
Random Forest           0.6349206        0.64375
```

From this two observations and evaluations of the model, we can clearly see that Random Forest clearly overpowers all the basic models making it the “best” classifier.

Question 8 – Importance

For decision tree model's summary, Cloud9am, WindGustDir, WindDir3pm, MaxTemp, WindDir9am, WindSpeed3pm, RISK_MM, Sunshine, WindSpeed9am, Pressure9am, Temp9am and Rainfall are the only predictors that are actually used in the tree construction, this makes the other predictors redundant to be included.

```
Classification tree:
tree(formula = MHT ~ ., data = WAUS.train)
Variables actually used in tree construction:
 [1] "Cloud9am"     "WindGustDir"  "WindDir3pm"   "MaxTemp"      "WindDir9am"   "WindSpeed3pm"
 [7] "RISK_MM"      "Sunshine"     "WindSpeed9am" "Pressure9am"  "Temp9am"      "Rainfall"
Number of terminal nodes:  29
Residual mean deviance:  0.3518 = 85.14 / 242
Misclassification error rate: 0.0738 = 20 / 271
```

For Bagging, the most important attributes are WindDir3pm, WindDir9am and WindGustDir as they are all above 10.0 while Location, RainToday and WindSpeed3pm have an importance value of 0 meaning they can be omitted because they have no effect on the model.

```
> BaggingModel$importance
     Cloud3pm      Cloud9am   Evaporation      Location       MaxTemp       MinTemp   Pressure3pm
     4.812253      5.180857      3.851189      0.000000      4.320972      1.265293      2.546838
  Pressure9am      Rainfall     RainToday       RISK_MM      Sunshine       Temp3pm       Temp9am
     1.571834      7.207951      0.000000      2.822135      1.022504      2.687748      1.735863
   WindDir3pm    WindDir9am   WindGustDir WindGustSpeed  WindSpeed3pm  WindSpeed9am
    15.737037     22.026921     17.038481      2.243871      0.000000      3.928252
```

For Boosting, it has the same important attributes as bagging but only RainToday has a value of 0.

```
> BoostingModel$importance
     Cloud3pm      Cloud9am   Evaporation      Location       MaxTemp       MinTemp   Pressure3pm
    0.7593554     2.9081988     2.7167505     3.1103164     4.1750585     5.3022058     1.0946488
  Pressure9am      Rainfall     RainToday       RISK_MM      Sunshine       Temp3pm       Temp9am
    1.8730024     4.4605468     0.0000000     1.9947180     3.8756842     3.3937856     6.5702893
   WindDir3pm    WindDir9am   WindGustDir WindGustSpeed  WindSpeed3pm  WindSpeed9am
   17.6635293    14.0201802    16.9111505     2.2266461     3.9833285     2.9606048
```

For Random Forest, the important attributes are WindDir9am, WindDir3pm and WindGustDir. While RainToday and Location seems to have the lowest importance value where the figure can be found in Question 10 sorted by importance.

In summary, Location and RainToday seems to have the lowest importance from the observations of all the models meaning they are not likely to affect the performance and can therefore be omitted. On the other hand, the most important variables observed is the direction of the wind at 9 am and 3 pm and also the speed of the strongest wind gust over the day.

Question 9 – Simple Tree

I started off with Decision Tree since it is the easiest classifier to visualise and simplify. The figure is a simple decision tree model generated by picking specific attributes from the original basic decision tree model.

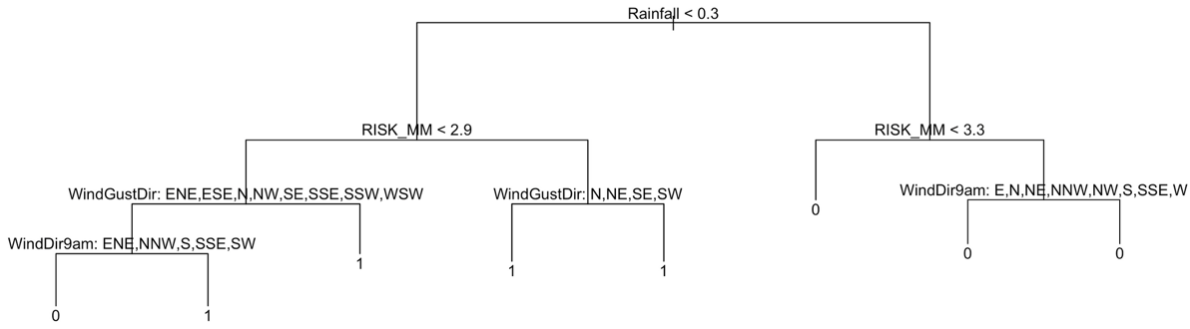
Rainfall being the top indicates that the amount of rainfall plays the largest role in determining it's humidity, as it goes down it's leaf nodes, it gets more specific to predict it's humidity based on their rules.

The attributes were chosen based on their importance and how the tree looks after each time it is ran and checking its accuracy. By removing certain attributes like WindDir8am, the accuracy rose a lot meaning it could have been simplified.

Performance

It has an accuracy of 0.5919854 and an AUC of 0.5935608.

Compared to the basic decision tree model made in question 4, it has a better accuracy and area under the curve but it is still not as good as random forest.



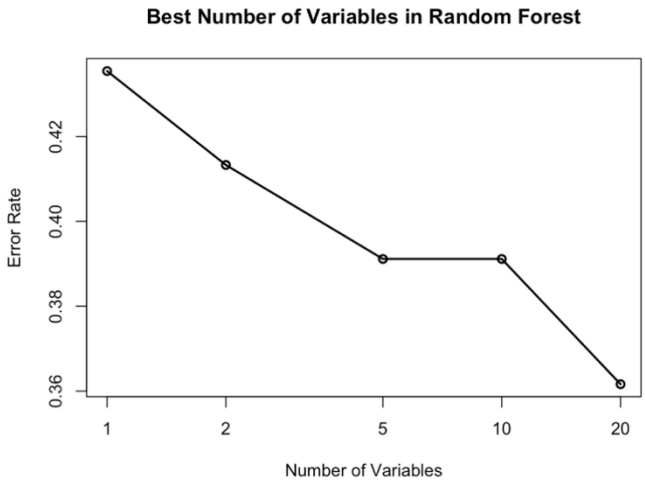
The model above is simple enough for a person to trace. We can predict if it will be more humid tomorrow by tracing it until the leaf nodes where 1 means it will be more humid tomorrow and 0 meaning it won't.

The conditions needs for tomorrow to be more humid is that rainfall has to be more than 0.3 and amount of rain is less than 2.9. If amount of rain is more than 2.9, the strongest wind gust direction has to be either ENE, ESE, N, NW, SE, SSE, SSW or WSW, if not, the wind direction at 9 am has to be ENE, NNW, S, SSE or SW.

If those conditions aren't met, it would be 0 indicating it won't be more humid tomorrow.

Question 10 – Best Tree-Based Classifier

To build the best classifier, I started from random forest since it is the best classifier out of the rest. In random forest's cross validation, a random forest is generated from all the attributes and are then ordered by Gini importance rank. It can be observed that as the number of variables increase, the error rate decreases for random forest model of this seed decreases. This means the model works well with most of the predictors. In order to further improve the model, we can only remove and test predictors one by one.



We can order the model by importance in Question 5 and pick variables based on their importance.

The 4 chosen attributes to be removed are Location, the minimum and maximum temperature and wind direction at 3pm. These attributes are obtained by experimenting and going through trial error of omitting each variable. Wind direction at 3pm seems to have the most impact on the prediction quality as removing it caused the largest changes even though it has quite a high importance. This can be due to overfitting or multicollinearity which means it could have been highly correlated with other variables in the model.

	MeanDecreaseGini
WindDir9am	15.015297
WindDir3pm	13.407851
WindGustDir	11.902294
Sunshine	9.341037
MinTemp	9.017229
Evaporation	7.030737
Rainfall	6.697839
Cloud9am	6.686017
Pressure9am	6.591006
Pressure3pm	6.231624
Temp9am	5.955494
MaxTemp	5.632378
Temp3pm	5.421775
WindSpeed9am	5.291251
WindGustSpeed	5.186378
WindSpeed3pm	4.382637
Cloud3pm	3.536153
RISK_MM	3.404283
RainToday	2.215668
Location	1.274329

The new random forest model has an accuracy of 0.6746032 and an AUC value of 0.6442602 which is better than all the other models in terms of both accuracy and Area Under Curve.

```
> acc_func(RandomForestConf)
[1] 0.6746032

> as.numeric(cauc@y.values)
[1] 0.6442602
```

Question 11 – Artificial Neural Network

Pre-processing

In order to create an Artificial Neural Network, pre-processing of the WAUS data has to be done. The pre-processing steps include data cleaning, data normalisation, splitting training and testing data and creating indicators for categorical variable. Since ANN does not work with NA values, we having to remove rows containing NA using na.omit(). Data normalisation is needed to ensure that all the numeric datas have a similar scale to avoid the larger values dominating the ANN learning process. Now that we have a clean and normalised WAUS data frame, we have to split 70% into training data and 30% testing data. Lastly, one-hot-encoding is performed to convert factor variables to indicators where each level in a column becomes its own binary variable. This is because ANN requires the input format to be numeric.

After pre-processing, the new data frame called WAUSANN which was split into WAUSANN.train and WAUSANN.test has 67 columns, 277 and 120 columns respectively but they are then resampled to 700 and 300 rows for more rows to be processed.

Attributes used

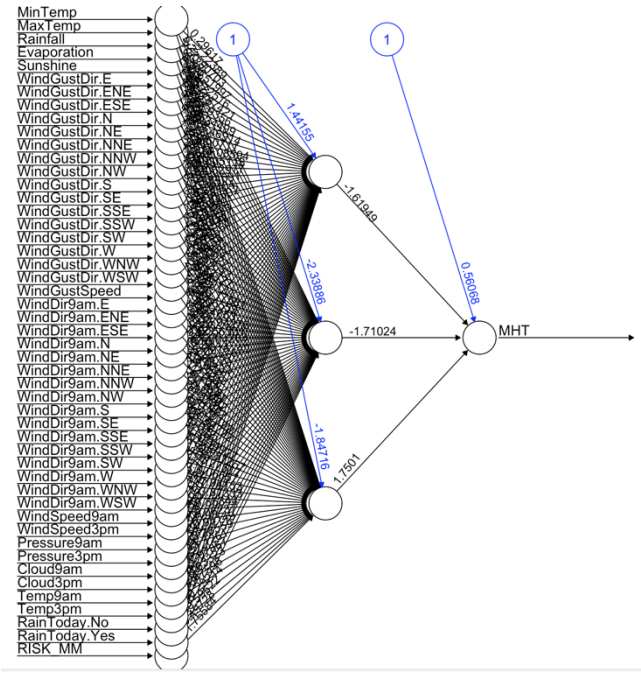
When computing predictions, they are returned in a numerical value because MHT was in numeric format, we can then create a barrier by separating all values less than 0.5 to be considered as the model predicting them to be MHT = 0 while the rest are 1. 0.5 is obtained by experimenting and adjusting for the best cut off point.

The first model of ANN was made with all the predictors and it resulted in an accuracy of 0.57.

As observed from previous models, Wind Direction at 3pm seems to have a lot of impact on the accuracy of the model and Location because from most of the models, it has the lowest importance. By removing it from neural network, the accuracy improved from 0.57 to 0.5833333.

For the final model, all the attributes except Wind Direction at 3pm is used.

The figure on the right is the Artificial Neural Network created with 66 input nodes and 1 output node.

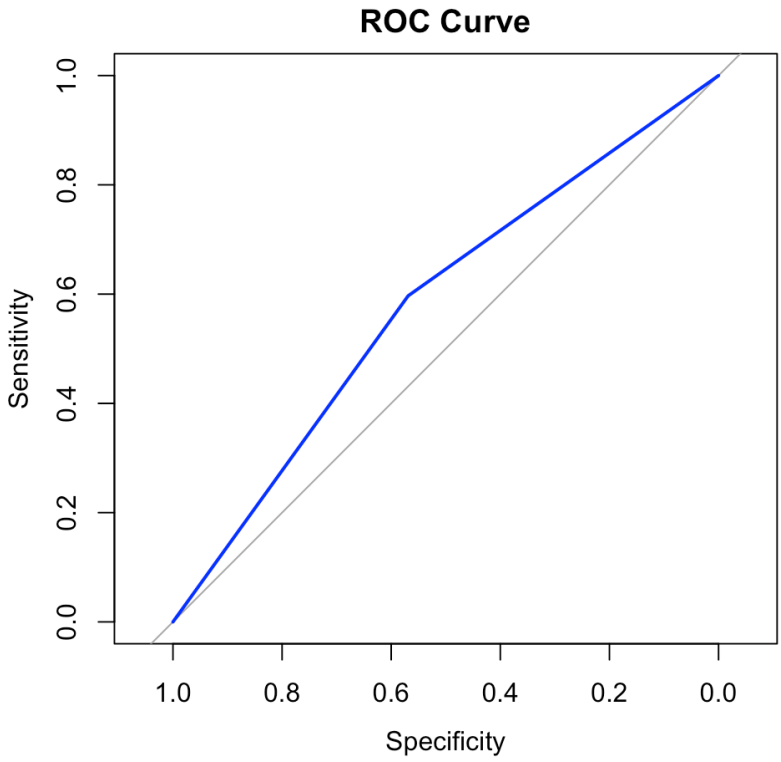


Performance

The final Artificial Neural Network model has an accuracy of 0.5833333 which is better than all the basic models made in Question 4 except Random Forest. This shows that ANN is powerful but may not always outperform random forest. This is because ANN has the ability to learn meaningful representations of the data through hidden layers meaning they can extract patterns from the input data, leading to an improved performance compared to basic models that rely on pre-defined features. It may not always outperform random forest because random forest requires less hyperparameter tuning and works best in scenarios where there's noisy and unbalanced data.

```
> acc_func(nnConf)
[1] 0.5833333
```

```
> auc
Area under the curve: 0.5829
```



Question 12 – New Classifier (XGBoost)

The new classifier chosen for this question is Extreme Gradient Boosting and the package used is xgboost.

Web Link : https://cran.r-project.org/web/packages/xgboost/xgboost.pdf

Description model type

The new classifier used is a machine learning algorithm based on the gradient boosting framework, an optimised and distributed version. It uses shallow decision trees as its base learners, they are shallow to prevent overfitting and also to capture linear and non-linear relationships. It works by combining weak decision trees to create a an ensemble of models where each subsequent model is made by correcting the mistakes of the previous models. It is a type of decision tree ensemble learning algorithm that is similar to random forest but they differ in the way they are built and combined.

How it works?

To build the model, pre-processing also has to be made. We can reuse the pre-processings made in Question 11 with the addition of changing Humidity variable (MHT) back to factor type.

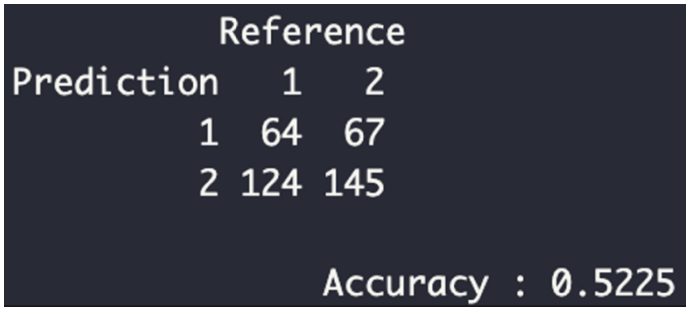
To further improve the model, we can first perform cross validation by setting the basic grid tuning and running cross validation to get. We can provide a basic parameter grid using expand.grid() to specify values for the hyper-parameters to be tuned later on. TrainControl() is then used define the cross-validation settings to allow train() to perform cross validation. It is specified in method as "cv" which means k-fold cross validation with 3 folds. Train() is used to perform cross-validation for this classifier where it takes in x (predictors) and y (class = MHT) and settings previously set in train_control. TuneGrid is set to the previously made basic grid tuning to indicate the grid of parameters and lastly verbose is used to display progress information.

The results of cross-validation are now stored in xgb_tune. To retrieve the best tuning hyperparameter values found during cross-validation, we can obtain it by $bestTune.

Now that we have the best hyper parameters from cross validation we can refit the model with the new information by replacing the settings in train_control and final_grid. The final model of XGboost is obtained and is then used to predict the test data.

Performance

The final model has an accuracy of 0.5225, which is worse than the rest of the models made previously.



Appendix

```
# set working directory
setwd("/Users/aliciang/Downloads/FIT3152-DATA-
ANALYTICS/FIT3152_Assignment2")

install.packages("rpart")
install.packages("tree")
install.packages("e1071")     # for naivebayes in Q4
install.packages("adabag")    # for bagging in Q4
install.packages("ROCR")      # for prediction in Q6
install.packages("randomForest")   # for randomForest in Q4
install.packages("pROC")     # for roc in NN

library(tree)
library(e1071)
library(adabag)
library(rpart)
library(ROCR)
library(randomForest)
library(pROC)


rm(list = ls())
WAUS <- read.csv("HumidPredict2023D.csv")
L <- as.data.frame(c(1:49))

set.seed(31861148) # 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

# Question 1 ----------------------------------------------------
------------

dim(WAUS)           # Output : 2000 rows 22 columns
str(WAUS)           # Output : 4 char, 10 num, 8 int

# Proportion of Humidity values
sum(WAUS$MHT == 1, na.rm = TRUE) / 20 # Output : 46.05% of rows
where MHT is 1
sum(WAUS$MHT == 0, na.rm = TRUE) / 20  # Output : 45.4% of rows
where MHT is 0

# Descriptions of independent variables
summary(WAUS)

# Question 2 ----------------------------------------------------
------------

# Turns all char type to factors
WAUS <- as.data.frame(unclass(WAUS),stringsAsFactors=T)

# Change Class to a factor (not needed for NN)
WAUS$MHT <- as.factor(WAUS$MHT)
```

```
# Remove all rows where MHT is NA (not needed for NN)
WAUS <- WAUS[!is.na(WAUS$MHT), ]
dim(WAUS)           # Output : 1829 rows 22 columns

# Remove Year Column as redundant for data training
WAUS <- WAUS[,2:22]


# Question 3 ------------------------------------------------------
------------

set.seed(31861148) #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,]

# Question 4 ------------------------------------------------------
------------

# Decision Tree
DecisionTreeModel <- tree(MHT ~ ., data = WAUS.train)

# Naive Bayes
NaiveBayesModel <- naiveBayes(MHT ~. - MHT, data = WAUS.train)

# Bagging
set.seed(31861148)
BaggingModel <- bagging(MHT ~ ., data = WAUS.train, mfinal = 5)

# Boosting
set.seed(31861148)
BoostingModel <- boosting(MHT ~ .-MHT, data = WAUS.train, mfinal =
10)


# Random Forest
WAUS.train <- na.omit(WAUS.train) # because random forest doesn't
work on NA
set.seed(31861148)
RandomForestModel <- randomForest(MHT ~ . -MHT, data = WAUS.train)
# reset
WAUS.train = WAUS[train.row,]

# Question 5 ------------------------------------------------------
------------

# List to display accuracy in Q7
ModelAccuracyList = list()

# function to calculate accuracy
acc_func = function(cm) {

  TP <- cm[2, 2]
  TN <- cm[1, 1]
  FP <- cm[1, 2]
  FN <- cm[2, 1]
```

```
  # Calculate the accuracy
  accuracy <- (TP + TN) / (TP + TN + FP + FN)

  return(accuracy)
}

# Decision Tree
DecisionTreePred <- predict(DecisionTreeModel, newdata = WAUS.test,
                            type = "class")
DecisionTreeConf <- table(observed = WAUS.test$MHT,
                          predicted = DecisionTreePred)
ModelAccuracyList = append(ModelAccuracyList,
acc_func(DecisionTreeConf))

# Naive Bayes
NaiveBayesPred <- predict(NaiveBayesModel, WAUS.test)
NaiveBayesConf <- table(actual = WAUS.test$MHT, predicted =
NaiveBayesPred)
ModelAccuracyList = append(ModelAccuracyList,
acc_func(NaiveBayesConf))

# Bagging
BaggingPred = predict.bagging(BaggingModel, newdata = WAUS.test)
ModelAccuracyList = append(ModelAccuracyList,
acc_func(BaggingPred$confusion))

# Boosting
BoostingPred = predict.boosting(BoostingModel, newdata = WAUS.test)
ModelAccuracyList = append(ModelAccuracyList,
acc_func(BoostingPred$confusion))

# Random Forest
WAUS.test = na.omit(WAUS.test)
RandomForestPred <- predict(RandomForestModel, newdata = WAUS.test,
                            type = "class")
RandomForestConf <- table(observed = WAUS.test$MHT,
                          predicted = RandomForestPred)
ModelAccuracyList = append(ModelAccuracyList,
acc_func(RandomForestConf))
# reset due to Random Forest
WAUS.test = WAUS[-train.row,]

# Question 6 ------------------------------------------------------
------------

# List to display AUC in Q7
AucOfModelList = list()

WAUS.test$MHT <- as.numeric(WAUS.test$MHT) # dont need for DT

# Decision Tree's ROC
DecisionTreeConf <- predict(DecisionTreeModel, WAUS.test, type =
"vector")
DecisionTreePredObj <- prediction(DecisionTreeConf[,2],
WAUS.test$MHT)
```

```
DecisionTreePerf <- performance(DecisionTreePredObj, "tpr", "fpr")
plot(DecisionTreePerf, col = "red", main = "ROC curve of the
models")
abline(0,1)
cauc = performance(DecisionTreePredObj, "auc")
AucOfModelList = append(AucOfModelList, as.numeric(cauc@y.values))

# Naive Bayes' ROC
NaiveBayesConf <- predict(NaiveBayesModel, WAUS.test, type = 'raw')
NaiveBayesPredObj <- prediction( NaiveBayesConf[,2], WAUS.test$MHT)
NaiveBayesPerf <- performance(NaiveBayesPredObj,"tpr","fpr")
plot(NaiveBayesPerf, col ="orange", add = TRUE)
cauc = performance(NaiveBayesPredObj, "auc")
AucOfModelList = append(AucOfModelList, as.numeric(cauc@y.values))

# Bagging's ROC
BaggingConf <- BaggingPred$prob
BaggingPredObj <- prediction( BaggingConf[,2], WAUS.test$MHT)
BaggingPerf <- performance(BaggingPredObj,"tpr","fpr")
plot(BaggingPerf, col ="green", add = TRUE)
cauc = performance(BaggingPredObj, "auc")
AucOfModelList = append(AucOfModelList, as.numeric(cauc@y.values))

# Boosting's ROC
BoostingConf <- BoostingPred$prob
BoostingPredObj <- prediction( BoostingConf[,2], WAUS.test$MHT)
BoostingPerf <- performance(BoostingPredObj,"tpr","fpr")
plot(BoostingPerf, col ="blue", add = TRUE)
cauc = performance(BoostingPredObj, "auc")
AucOfModelList = append(AucOfModelList, as.numeric(cauc@y.values))

# Random Forest's ROC
WAUS.test = na.omit(WAUS.test)
RandomForestConf = predict(RandomForestModel, WAUS.test, type =
"prob")
RandomForestPredObj <- prediction( RandomForestConf[,2],
WAUS.test$MHT)
RandomForestPerf <- performance(RandomForestPredObj,"tpr","fpr")
plot(RandomForestPerf, col ="purple", add = TRUE)
cauc = performance(RandomForestPredObj, "auc")
AucOfModelList = append(AucOfModelList, as.numeric(cauc@y.values))

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

# To add legend
legend("bottomright",
       legend = c("Decision Tree","Naive Bayes", "Bagging",
"Boosting",
                  "Random Forest"),
       col = c("red", "orange", "green","blue","purple"),
       lty = 1)

# Function to get back the ROC plot for Decision Tree and Random
Forest
getBackPlot = function() {
```

```
  plot(DecisionTreePerf, col = "red", main = "ROC curve of the
models")
  plot(RandomForestPerf, col ="purple", add = TRUE)
  abline(0,1)

}
# Question 7 ----------------------------------------------------
------------

# table summary
ModelNames = c("Decision Tree", "Naive Bayes", "Bagging",
"Boosting",
               "Random Forest")
Q7Summary = cbind(ModelAccuracyList, AucOfModelList)
Q7Summary = cbind(ModelNames, Q7Summary)
Q7Summary <- data.frame(Q7Summary[,-1], row.names = Q7Summary[,1])
Q7Summary

# Question 8 ----------------------------------------------------
------------

# Important attributes for Decision tree
summary(DecisionTreeModel)
# Explanation : Given the summary these 12 are meaningful

BaggingModel$importance
BoostingModel$importance
RandomForestModel$importance

import <- RandomForestModel$importance
import[order(import, decreasing=TRUE),,drop = FALSE]

# WAUS.train w/o Location, RainToday, RISKMM
WAUS.trainQ10 = WAUS.train[,-c(1)]
colnames(WAUS.trainQ10)
# Question 9 ----------------------------------------------------
------------

# Simplified Decision Tree
SimpleDecisionTreeModel <- tree(MHT ~  WindGustDir + MaxTemp +
    WindDir9am + WindSpeed3pm + RISK_MM  + WindSpeed9am +
Pressure9am +
    Temp9am + Rainfall , data = WAUS.train)

plot(DecisionTreeModel)
text(DecisionTreeModel, pretty = 0)
summary(DecisionTreeModel)

# type = class for classification
SimpleDecisionTreePred <- predict(SimpleDecisionTreeModel, newdata =
WAUS.test,
                                  type = "class")
acc_func(table(observed = WAUS.test$MHT, predicted =
SimpleDecisionTreePred))
```

```
# Getting Back Plot
getBackPlot()

SimpleDecisionTreeConf <- predict(SimpleDecisionTreeModel,
WAUS.test,
                                  type = "vector")
SimpleDecisionTreePredObj <- prediction(SimpleDecisionTreeConf[,2],
                                        WAUS.test$MHT)
SimpleDecisionTreePerf <- performance(SimpleDecisionTreePredObj,
"tpr", "fpr")
plot(SimpleDecisionTreePerf, col = "green", add = TRUE)
abline(0,1)
cauc = performance(SimpleDecisionTreePredObj, "auc")
as.numeric(cauc@y.values)

legend("bottomright",
       legend = c("Decision Tree","Random Forest","Simple Tree"),
       col = c("red","purple", "green"),
       lty = 1)

# Question 10 ------------------------------------------------------
------------

# Cross Validation for Random Forest - plot shows that the model
# does well with most of the predictors
set.seed(31861148)
WAUS.trainOmitted = na.omit(WAUS.train)
set.seed(31861148)
RandomForestCV <- rfcv(WAUS.trainOmitted[,1:20],
WAUS.trainOmitted$MHT)

with(RandomForestCV, plot(n.var, error.cv, log="x", type="o", lwd=2,
                          xlab="Number of Variables", ylab="Error
Rate",
                          main = "Best Number of Variables in Random
Forest"))

# The best choice is the remove predictors from model one by one
# Order importance of predictors
import <- RandomForestModel$importance
import[order(import, decreasing=TRUE),,drop = FALSE]

### ACTUAL RANDOM FOREST MODEL ###
WAUS.train = na.omit(WAUS.train)
# Removed NA rows, Location and WindDir3pm , maybe 3
WAUS.trainQ10 = WAUS.train[,-c(1,2,3,10)]
# Restore back original value
WAUS.train = WAUS[train.row,]

set.seed(31861148)
RandomForestModel <- randomForest(MHT~., data = WAUS.trainQ10, ntree
= 2500,
                                  max.depth = 150, min.node.size =
2)

WAUS.test = na.omit(WAUS.test)
```

```
RandomForestPred <- predict(RandomForestModel, newdata = WAUS.test,
                            type = "class")
# Confusion Matrix, Accuracy = 0.674603
RandomForestConf <- table(observed = WAUS.test$MHT,
                          predicted = RandomForestPred)
acc_func(RandomForestConf)
WAUS.test = WAUS[-train.row,]

# Area under curve = 0.6442602
getBackPlot()
WAUS.test = na.omit(WAUS.test)
RandomForestCVConf = predict(RandomForestModel, WAUS.test, type =
"prob")
RandomForestPredObj <- prediction( RandomForestCVConf[,2],
WAUS.test$MHT)
RandomForestPerf <- performance(RandomForestPredObj,"tpr","fpr")
plot(RandomForestPerf, col ="blue", add =TRUE)
cauc = performance(RandomForestPredObj, "auc")
AucOfModelList = append(AucOfModelList, as.numeric(cauc@y.values))


# Question 11 ------------------------------------------------------
------------

# Note :it can't be placed at the top as it has the prediction()
# function too and it will mask the function in Question 6
install.packages("neuralnet")
install.packages("caret")
library(neuralnet)
library(caret)

# Artificial Neural Network
# Pre-processing
WAUSANN = WAUS[complete.cases(WAUS),] # remove NA rows
WAUSANN$MHT = as.numeric(WAUSANN$MHT) # change MHT to numeric

numeric_cols <- sapply(WAUSANN, is.numeric)
set.seed(31861148) #Student ID as random seed
WAUSANN[, numeric_cols] <- scale(WAUSANN[, numeric_cols])

# train and test set
set.seed(31861148) #Student ID as random seed
train.row = sample(1:nrow(WAUSANN), 0.7*nrow(WAUSANN))
WAUSANN.train = WAUSANN[train.row,]
WAUSANN.test = WAUSANN[-train.row,]

# Creating indicators for training set
MHT = WAUSANN.train[,21]
WAUSANN.train = WAUSANN.train[,1:20]
dummy <- dummyVars(" ~ .", data = WAUSANN.train)
newWAUSANN.train <- data.frame(predict(dummy, newdata =
WAUSANN.train))
WAUSANN.train = cbind(newWAUSANN.train, MHT)

# Creating indicators for testing set
MHT = WAUSANN.test[,21]
```

```
WAUSANN.test = WAUSANN.test[,1:20]
dummy <- dummyVars(" ~ .", data = WAUSANN.test)
newWAUSANN.test <- data.frame(predict(dummy, newdata =
WAUSANN.test))
WAUSANN.test = cbind(newWAUSANN.test, MHT)

dim(WAUSANN.train)
dim(WAUSANN.test)

### NEURAL NETWORK MODEL ###
set.seed(31861148)
WAUS.nn = neuralnet(MHT ~  MinTemp+ MaxTemp+ Rainfall+ Evaporation+
                      Sunshine+ WindGustDir.E+ WindGustDir.ENE+
WindGustDir.ESE+
                      WindGustDir.N+ WindGustDir.NE+
WindGustDir.NNE+
                      WindGustDir.NNW+ WindGustDir.NW+
WindGustDir.S+
                      WindGustDir.SE+ WindGustDir.SSE+
                      WindGustDir.SSW+ WindGustDir.SW+
WindGustDir.W+
                      WindGustDir.WNW+ WindGustDir.WSW+
WindGustSpeed+
                      WindDir9am.E+ WindDir9am.ENE+
WindDir9am.ESE+WindDir9am.N+
                      WindDir9am.NE+ WindDir9am.NNE+WindDir9am.NNW+
                      WindDir9am.NW+ WindDir9am.S+ WindDir9am.SE+
                      WindDir9am.SSE+ WindDir9am.SSW+ WindDir9am.SW+
                      WindDir9am.W+ WindDir9am.WNW+ WindDir9am.WSW +
                      WindSpeed9am+ WindSpeed3pm+
                      Pressure9am+ Pressure3pm+ Cloud9am+ Cloud3pm+
Temp9am+
                      Temp3pm+ RainToday.No+ RainToday.Yes+ RISK_MM,
                    WAUSANN.train,
                    hidden=3, err.fct ="sse", threshold =  0.05,
                    linear.output = TRUE)

plot(WAUS.nn, rep="best")
WAUS.nn$result.matrix

WAUS.pred = compute(WAUS.nn, WAUSANN.test[,1:66])
# Cut off point to label them to 0 and 1
WAUS.predRescaled <- ifelse(WAUS.pred$net.result <= 0.5, 0, 1)

# Confusion matrix
nnConf <- table(observed = WAUSANN.test$MHT, predicted =
        WAUS.predRescaled)
acc_func(nnConf)

# ROC
# Note : Install package pROC for this to work
auc <- roc(WAUSANN.test$MHT, WAUS.predRescaled)$auc
NeuralNetworkROC <- roc(WAUSANN.test$MHT, WAUS.predRescaled)
plot(NeuralNetworkROC, col = "blue", main = "ROC Curve")
```

```
# Question 12 ------------------------------------------------------
------------

# XGBoost packages
install.packages("caret")
install.packages("xgboost")
install.packages("mltools")
install.packages("data.table")

library(caret) # Machine Learning Library, for dummyVars
library(xgboost) # XGBoost library
library(mltools)
library(data.table)

# Can reused the indicators made in Q11 but have to change them back
to factors
WAUSANN.train$MHT <- as.factor(WAUSANN.train$MHT)

# Resampling for 400 test data
set.seed(31861148)
WAUSANN.test <- WAUSANN.test[sample(nrow(WAUSANN.test), size = 400,
                                    replace = TRUE), ]
WAUSANN.test$MHT <- as.factor(WAUSANN.test$MHT)

### Cross Validation for XGBoost ###
# Basic Grid Tuning
grid_tune <- expand.grid(
  nrounds = c(500,1000,1500), # number of trees
  max_depth = c(2,4,6),
  eta = 0.3, #Learning rate
  gamma = 0, # pruning
  colsample_bytree = 1, # subsample ratio of columns for tree
  min_child_weight = 1,
  subsample = 1 # to prevent overfitting by sampling X% training
)

# Cross Validation settings
train_control <- trainControl(method = "cv",
                              number=3,
                              verboseIter = TRUE,
                              allowParallel = TRUE)
# Training Cross Validation model
set.seed(31861148)
xgb_tune <- train(x = WAUSANN.train[,-67],
                  y = WAUSANN.train[,67],
                  trControl = train_control,
                  tuneGrid = grid_tune,
                  method= "xgbTree",
                  verbose = TRUE)
# Best tune parameters
xgb_tune$bestTune

### Getting best model based on Cross Validation ###
# Update model settings for final model
train_control <- trainControl(method = "none",
                              verboseIter = TRUE,
```

```
                              allowParallel = TRUE)

# New settings based on bestTune from Cross Validation
final_grid <- expand.grid(nrounds = xgb_tune$bestTune$nrounds,
                          eta = xgb_tune$bestTune$eta,
                          max_depth = xgb_tune$bestTune$max_depth,
                          gamma = xgb_tune$bestTune$gamma,
                          colsample_bytree =
xgb_tune$bestTune$colsample_bytree,
                          min_child_weight =
xgb_tune$bestTune$min_child_weight,
                          subsample = xgb_tune$bestTune$subsample)
# Training final XGBoost model
set.seed(31861148)
xgb_model <- train(x = WAUSANN.train[,-67],
                   y = WAUSANN.train[,67],
                   trControl = train_control,
                   tuneGrid = final_grid,
                   method = "xgbTree",
                   verbose = TRUE)

# Predicting test data
xgb.pred <- predict(xgb_model, WAUSANN.test)

# Confusion Matrix
confusionMatrix(as.factor(as.numeric(xgb.pred)),
                as.factor(as.numeric(WAUSANN.test$MHT)))
```