FIT3152 Data Analytics
Semester 1 2023
Assignment 2

Name: Tatiana Sutulova

Student ID: 30806151

Part 1: Questions

1.1 Data exploration

Before attempting to explore predictor variables for a given dataset, the data is pre-processed by omitting all the missing values and removing duplicate records, since it can introduce bias and affect the reliability of statistical measures. When exploring the data, it was observed that 340 records out of 730 state that it is more humid than the previous day, whereas 390 records out of 730 state the opposite, which means that the proportion is approximately ~1:1.15, which is represented in Figure 1.

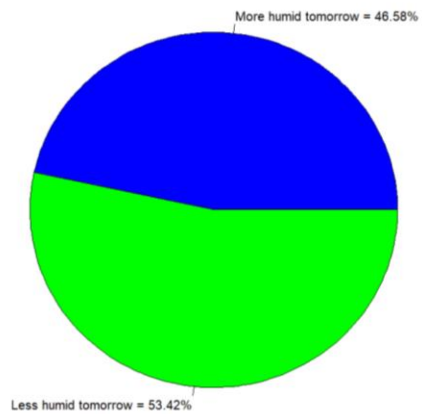

Figure 1. The proportion of days when it is more humid than the previous day compared to those where it is less humid.

In order to further analyze the given dataset, real-valued attributes are to be identified. The real-valued attributes are types of attributes that have an infinite number of possible values within their range. Thus, it was determined that attributes that are to be analyzed are MinTemp, MaxTemp, Rainfall, Evaporation, Sunshine, WindGustSpeed, WindSpeed9am, WindSpeed3pm, Pressure9am, Pressure3pm, Temp9am, Temp3pm, RISK_MM, whereas others are omitted due to its discrete nature. As observed from Table 2, Rainfall, WindGustSpeed and RISK_MM attributes have relatively high standard deviation values which may indicate the presence of outliers. After comparing its max or min values against its mean values presented in Table 1 and building a boxplot shown in Figure 2, it is confirmed that these three attributes contain outlier values, which can significantly impact the overall analysis and prediction result. Thus, additional pre-processing is to be performed in order to eliminate extreme values.

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine
Min.	-4.300	9.70	0.000	0.000	0.000
1st Qu.	8.025	17.60	0.000	2.800	4.200
Median	12.500	22.50	0.000	4.600	8.000
Mean	12.719	23.10	2.858	5.021	7.184
3rd Qu.	17.400	28.07	0.800	6.800	10.375
Max.	26.200	43.10	206.200	16.800	13.900

	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Pressure9am	Pressure3pm	Cloud9am
Min.	13.00	2.00	2.00	991.3	991.3	0.000
1st Qu.	30.00	7.00	11.00	1013.2	1010.9	2.000
Median	37.00	13.00	17.00	1018.2	1015.6	6.000
Mean	38.54	13.62	18.35	1017.9	1015.5	4.668
3rd Qu.	46.00	19.00	24.00	1022.9	1020.3	7.000
Max.	106.00	50.00	56.00	1037.7	1035.6	8.000

	Cloud3pm	Temp9am	Temp3pm	RISK_MM
Min.	0.000	1.30	8.80	0.000
1st Qu.	2.000	12.10	16.23	0.000
Median	5.000	16.75	21.35	0.000
Mean	4.523	17.39	21.58	2.802
3rd Qu.	7.000	22.55	26.25	1.200
Max.	8.000	34.60	41.20	136.600

Table 1. Predictor variables for real-valued attributes

MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am
6.248557	6.769050	11.372620	3.064680	3.876491	14.058345	7.958324

WindSpeed3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm
8.671273	7.168415	7.196547	2.750178	2.719665	6.609260	6.522111

RISK_MM
9.683898

Table 2. Standard deviation for real-valued attributes

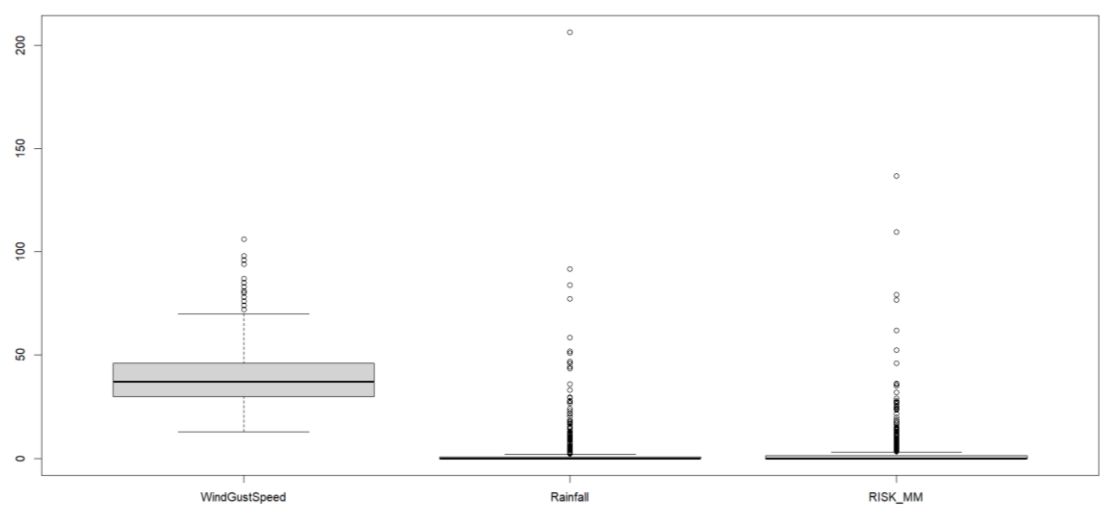

Figure 2. Boxplot for WindGustSpeed, Rainfall and RISK_MM attributes

1.2 Pre-processing

Pre-processing data is a vital step in preparing it for machine learning, since it ensures that it is in a suitable format for the learning algorithm to understand and make accurate predictions. The first step of pre-processing data is handling missing values which is performed by removing data records that contain any NA values. The next step is handling outliers for the attributes with a high standard deviation, as determined in 1.1. First, the Interquartile Range (IQR) method was used to detect and remove the outliers for the WindGustSpeed attribute, whereas for Rainfall and RISK_MM the outliers were removed based on the data determined in the boxplot. The last step of pre-processing includes encoding data, which means that any categorical or textual data is converted to numerical representation, since it is required by the machine learning algorithms, that performed in the following sections.

1.3 Implementation of classification models. Confusion matrixes & accuracy

After successfully dividing the data into a 70% training and 30% testing subsets and implementing a classification model, using the following techniques: decision tree, Naïve Bayes, bagging, boosting and random forest, the test data was used to classify each of the test cases as 'more humid tomorrow' or 'less humid tomorrow'. Table 3 illustrates confusion matrixes determined for each classifier and the accuracy of the model prediction, which is calculated using the formula in Figure 3.

	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

Most widely-used metric:

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

Also:

Precision = TP/(TP + FP)

Sensitivity = TP /(TP + FN)

Figure 3. Formula for accuracy of the model prediction using confusion matrix.

	Confusion matrix	Accuracy
Decision Tree	predicted observed 0 1 0 52 64 1 35 56	52.17%
Naïve Bayes	predicted observed 0 1 0 31 85 1 25 66	46.86%
Bagging	predicted observed 0 1 0 71 45 1 51 40	53.62%
Boosting	predicted observed 0 1 0 56 60 1 33 58	55.07%
Random forest	predicted observed 0 1 0 67 49 1 48 43	53.14%

Table 3. Confusion matrix and accuracy for each classifier

1.4 ROC Curve & AUC value for each classifier. Best classifier

The test data was utilized to calculate the confidence in predicting "increased humidity tomorrow" for each model. As depicted in Figure 4, an ROC curve was generated for each classifier, as well as Table 4 presents the accuracy and AUC values obtained for each classifier.

According to the determined data, we can conclude that most classifiers exhibit comparable accuracy and AUC, except for Naïve Bayes, which has noticeably lower performance. Among them, Random Forest stands out as the classifier with the highest AUC value of 58.53%. It is worth mentioning that the "Boosting" classifier also achieves a relatively high AUC value of 57.65% and a slightly higher accuracy of 55.07%. This indicates that both Random Forest and Boosting classifiers perform exceptionally well, making them the top performers and, therefore, the two best classifiers.

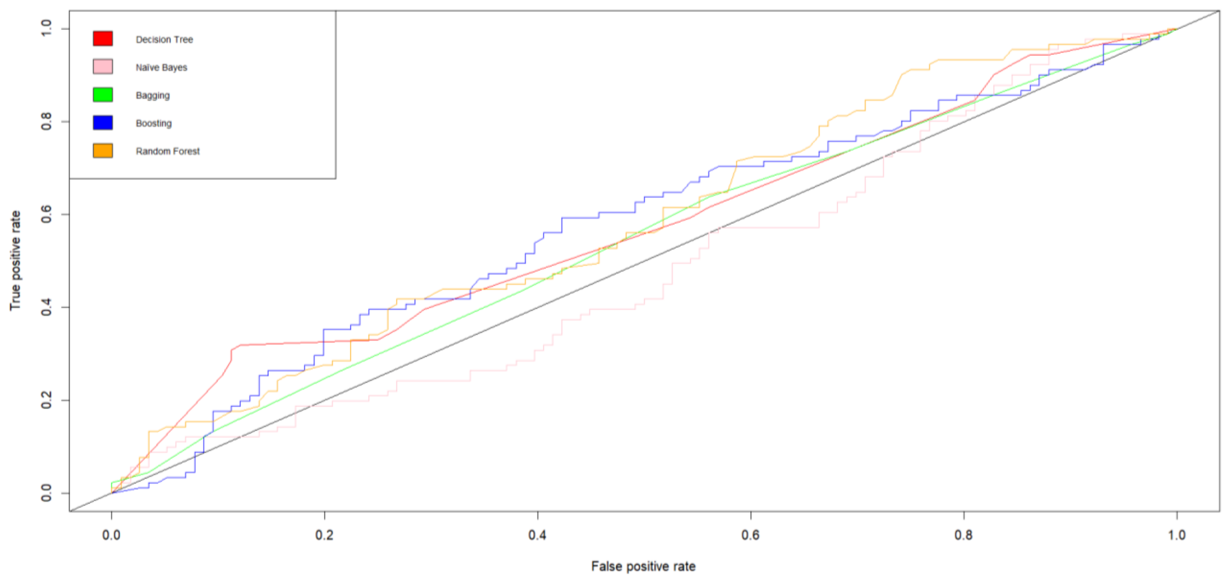

Figure 4. ROC curve for each classifier

	Accuracy	AUC
Decision Tree	52.17%	57.28%
Naïve Bayes	46.86%	48.08%
Bagging	53.62%	54.27%
Boosting	55.07%	57.65%
Random forest	53.14%	58.53%

Table 4. Accuracy and AUC values for each classifier

Part 2: Investigative Tasks

2.1 Most important variables in predicting whether it will be more humid tomorrow.

After examining each of the models, excluding Naïve Bayes due to its nature, the most important variables in predicting whether it will be more humid tomorrow or not were determined and presented in Table 5. Analyzing table data, it can be concluded that the most important variables for prediction are *Pressure9am*, *MinTemp*, *Pressure3pm*, which are vital for all four classification models, as well as *WindGustSpeed*, *Evaporation*, *Sunshine*, which are important for at least three models. On the other hand, the least important variables that can be omitted from the data with very little effect on performance are *RainToday*, *Year* and *Cloud9am*, since they appear to be the least important variables for all of the classification models. Other variables, such as *Location*, *RISK_MM*, etc. cannot be omitted since they are least important for some classifiers, whereas quite valuable for others.

Models	The most important variables	The least important variables
Decision Tree	WindSpeed9am, Rainfall, Pressure9am, RISK_MM, Temp3pm, WindGustSpeed, Evaporation, MinTemp, WindDir9am, Sunshine, Cloud3pm, Location, WindSpeed3pm, Pressure3pm	The rest
Bagging	MaxTemp (8.637486) WindSpeed9am (7.302275) WindGustSpeed (6.880136) WindDir3pm (6.805115) Pressure9am (6.391129) WindGustDir (6.351241) MinTemp (6.620729) Pressure3pm (5.974392)	RainToday (0.000000) Year (1.349824) Location (1.461447) Cloud9am (1.996047) RISK_MM (2.026906)
Boosting	Pressure9am (10.3216040) Evaporation (8.3380680) Sunshine (7.3928582) MaxTemp (6.4038287) MinTemp(5.9730602) Pressure3pm (5.8977199) Temp9am (5.7774648) WindGustSpeed (5.6218668)	RainToday (0.000000) Cloud3pm (0.7125081) Year (2.3525870) RISK_MM (3.1256603) Cloud9am (3.2193881) Temp3pm (3.3345127)
Random forest	Pressure9am (16.152054) Sunshine (15.893534) Temp3pm (15.429312) MinTemp (15.379348) MaxTemp (15.344142) Pressure3pm (15.275486) Temp9am (14.968517) Evaporation (14.747849)	RainToday (1.647964) Rainfall (7.175908) Cloud9am (7.214533) Cloud3pm (7.521364) RISK_MM (7.157995) Year (9.933745) Location (8.320598)

Conclusions	Pressure9am, MintTemp, Pressure3pm, WindGustSpeed, Evaporation, Sunshine	RainToday, Year, Cloud9am

Table 5. The most and least important variables for all the models

2.2 Human-readable classifier creation

In order to develop a classifier that allows a person to manually determine if tomorrow's weather will be more humid or not, a Decision Tree classifier was selected as the base and constructed using only the most significant attributes listed in table 5. The resulting simplified model, as depicted in Figure 5, encompasses only 9 attributes. This reduction in complexity was accomplished through cross-validation and post-pruning techniques applied to the decision tree.

Upon evaluating the performance of the new "simple" model, it was observed that its accuracy improved slightly to 53.14% compared to the original model. However, the AUC value experienced a slight decrease to 56.94%. The primary objective of creating this model was to decrease the complexity of the decision tree while ensuring that both accuracy and AUC metrics remained above 50%. This goal was successfully accomplished.

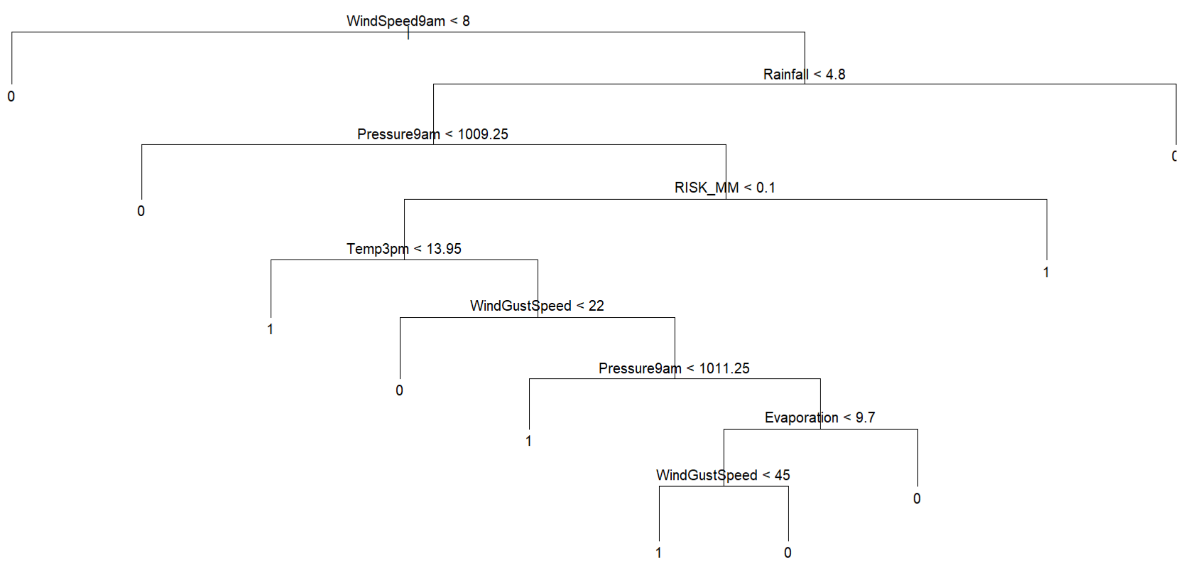

Figure 5. The simplified decision tree model

2.3 The best tree-based classifier

In order to develop an optimal tree-based classifier, the Random Forest basic model was chosen as the foundation due to its outstanding performance as mentioned in section 1.4. It was identified as one of the top classifiers in terms of accuracy and AUC values, making it an ideal starting point for enhancement.

I have made attempts to enhance the Random Forest classifier using two different methods: cross-validation and parameter adjustment. The cross-validation approach involved selecting 20 folds to strike a balance between computational efficiency and reliability. This number ensured that each fold had sufficient data for training and evaluation, thereby guaranteeing robust evaluation while keeping the computational process efficient. However, this approach yielded an accuracy of 53.6% and an AUC value of 59.36%, which did not significantly outperform other classifiers in terms of accuracy. Therefore, it did not qualify as the "best" classifier.

The second approach involved adjusting the parameters, which resulted in a more favorable outcome. The accuracy improved to 56.52% and the AUC value rose to 60.13%, suggesting the achievement of the expected "best" result. The approach closely followed that of the basic model, with one difference being the removal of the *Raintoday* attribute. As seen in 2.1, this attribute was determined to be the least important and did not contribute any value to the classification model. Additionally, the parameter *ntree* was set to 1000, indicating the number of trees to be grown in the random forest model. This value was chosen to enhance performance and algorithm robustness, avoiding diminishing returns or overfitting that can occur with higher *ntree* values.

2.3 Artificial Neural Network classifier

Using the insights from the previous analysis, an Artificial Neural Network classifier was implemented. To begin with, the data pre-processed as described in section 1.2 was used with one additional pre-processing step: standardization. Standardization involves transforming the input data in such a way that it has a mean of zero and a standard deviation of one, which allows to avoid bias towards certain features, accelerates convergence and ensures numerical stability. With standardization all features are transformed to a similar scale, preventing any feature from overwhelming the learning process an reducing the changes of overflow/underflow. Based on the results determined in section 2.1 and trial and error, it was decided that the RainToday and Year attributes will not be added to the prediction process, since they were determined to be the least important and thus do not add any value to the model. The visualization of the neural network is presented in Figure 5.

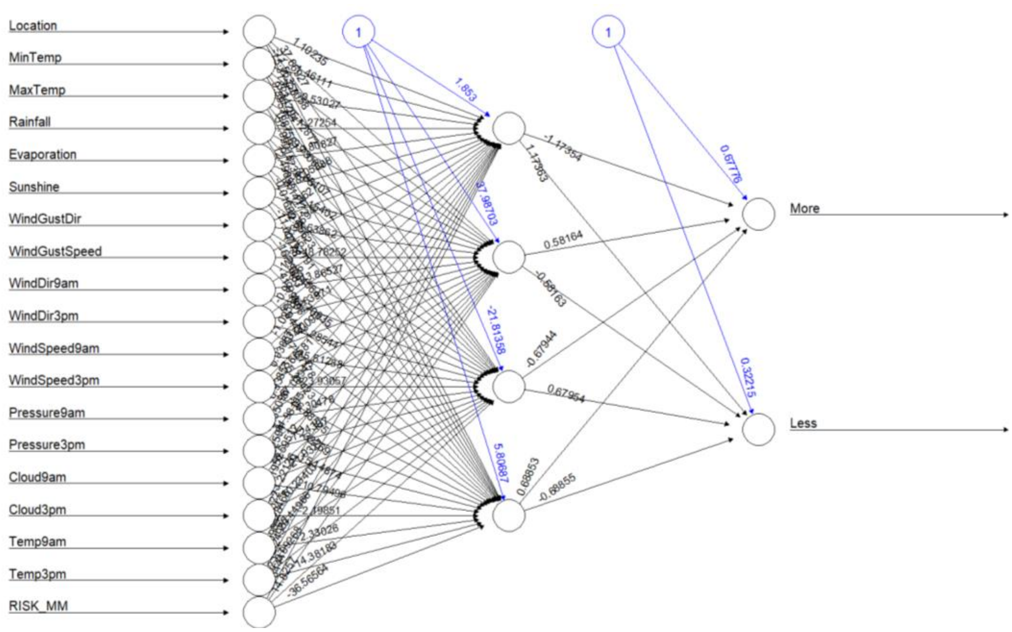

Figure 6. Neuralnet training model

Comparing the performance of neuralnet model with others described in section 1.4, it may be observed that it's accuracy of 54.06% is higher than most of the models except boosting, whereas its AUC of 53.01% is significantly lower than that of other models except for Naïve Bayes. The fact that a neural network model has higher accuracy, but lower AUC (Area Under the ROC Curve) compared to other models suggests that the neural network model might be better at correctly classifying the majority class instances but struggles with correctly classifying the minority class instances or achieving a balanced prediction.

2.4 New classifier

The new classifier that was fit to the data is an XGBoost, which stands for Extreme Gradient Boosting and available in an *xgboost* package [1]. The basic idea behind this method is to combine weak prediction models, such as decision trees, in an additive manner to create a strong predictive mode. It follows a boosting framework, where each subsequent model is trained to correct mistakes made by the previous ones.

Here is a basic overview of how XGBoost works step-by-step:

1. **Initialization**: XGBoost starts with a single decision tree, which serves as the initial weak learner
2. **Gradient Boosting**: XGBoost iteratively improves the model's performance by adding new decision trees. At each iteration, it calculates the gradients of a chosen loss function with respect to the current predictions. These gradients represent the direction in which the model needs improvement.
3. **Tree Construction:** XGBoost constructs a new decision tree to capture the patterns and relationships in the data that were not well captured by the existing model. It uses gradient boosting, where the new tree is built to minimize the loss function by fitting the negative gradients (residuals) of the previous trees.
4. **Regularization:** XGBoost incorporates regularization techniques to control the complexity of the model and prevent overfitting. It adds penalties to the loss function that discourage the creation of overly complex trees or limit the impact of individual trees.
5. **Ensemble Combination**: The predictions from all the individual trees are combined to obtain the final prediction. XGBoost uses a weighted sum of the predictions, where the weights are determined by the model's learning rate and the contribution of each tree.
6. **Iteration**: Steps 2-5 are repeated for a specified number of iterations or until a stopping criterion is met. The model continues to improve by minimizing the loss function and refining the predictions [2].

XGBoost is a powerful machine learning algorithm, which is known for its efficiency, scalability and ability to achieve highly accurate result. Comparing its accuracy of 58.94% and AUC of 59.23% with these of other models (Table 4) it outperforms other models, due to its iterative learning nature and regularization techniques.

References

[1] XGBoost R Tutorial

[2] XGBoost: A Scalable Tree Boosting System

Appendix

```r
# Author: Sutulova Tatiana, 30806151
# Assignment 2
# The objective of this assignment is to gain familiarity with classification models using R.
# We want to obtain a model that may be used to predict whether or not it will be more humid
# tomorrow than it is today for 10 locations in Australia.

# Libraries and packages to be used
library(dplyr)
library(tree) # For decision tree
library(e1071) # For Naive Bayes 1
library(adabag) # For bagging
library(rpart) # For bagging and boosting
library(randomForest) # For random forest
library(ROCR) # For ROCR and AUC


detach("package:neuralnet", unload = TRUE)
# Read the data from the file into a data frame
rm (list = ls())
WAUS <- read.csv("HumidPredict2023D.csv", header = TRUE)
L <- as.data.frame(c(1:49))
set.seed(30806151)
L <- L[sample(nrow(L), 15, replace = FALSE),] # sample 15 locations (reduced when processing)
WAUS <- WAUS[(WAUS$Location %in% L),]
WAUS <- WAUS[sample(nrow(WAUS), 2000, replace = FALSE),] # sample 2000 rows


# Q2. Pre-processing:
WAUS <- unique(WAUS) # Clear out duplicates
WAUS <- na.omit(WAUS) # Removing rows with NA values

unique_val <- unique(WAUS$Location) # checking whether there are 10 distinct locations

#################################################################################
# Q1: Proportion of proportion of days when it is more humid than the previous day compared to
those where it is less humid?
length(WAUS$MHT[WAUS$MHT == 1]) # More humid tomorrow
length(WAUS$MHT[WAUS$MHT == 0]) # Less humid tomorrow

# Viaualising with pie chart
pct <- c(length(WAUS$MHT[WAUS$MHT == 1]), length(WAUS$MHT[WAUS$MHT == 0]))
shades <- c("blue", "green")
categories <- c("More humid tomorrow", "Less humid tomorrow" )
pie_labels <- paste0(categories, " = ", round(100*(pct/nrow(WAUS)),2), "%")
pie(pct, labels = pie_labels, col = shades)
```

```r
# Selecting real-valued attributes
real_attributes <- WAUS[, c("MinTemp", "MaxTemp", "Rainfall", "Evaporation", "Sunshine",
                            "WindGustSpeed", "WindSpeed9am", "WindSpeed3pm", "Pressure9am",
                            "Pressure3pm", "Cloud9am", "Cloud3pm", "Temp9am", "Temp3pm",
                            "RISK_MM")]
summary(real_attributes) # Mean, median, etc
apply(real_attributes,2,sd) #standard deviations

################################################################################
# Q2. Pre-processing contd.:
# Handling the outliers:
boxplot( WAUS$WindGustSpeed, as.integer(WAUS$Rainfall), WAUS$RISK_MM,
names=c("WindGustSpeed","Rainfall","RISK_MM") )

# Remove outliers for WindGustSpeed (IQR Method)
quartiles <- quantile(WAUS$WindGustSpeed,probs=c(.25, .75), na.rm = FALSE)
IQR <- IQR(WAUS$WindGustSpeed)
Lower <- quartiles[1] - 1.5*IQR
Upper <- quartiles[2] + 1.5*IQR
WAUS <- subset(WAUS, WAUS$WindGustSpeed > Lower & WAUS$WindGustSpeed < Upper)

# Remove outliers for Rainfall and RISK_MM based on the boxplot
WAUS <- subset(WAUS, WAUS$Rainfall<40)
WAUS <- subset(WAUS,WAUS$RISK_MM<35)


# Encode data
WAUS$RainToday <-ifelse(WAUS$RainToday=="Yes",1, 0 )
WAUS$WindDir3pm <- as.numeric(factor(WAUS$WindDir3pm))
WAUS$WindGustDir <- as.numeric(factor(WAUS$WindGustDir))
WAUS$WindDir9am <- as.numeric(factor(WAUS$WindDir9am))
WAUS$MHT <- as.factor(WAUS$MHT)

################################################################################
# Q3: Divide data into a 70% training and 30% test set
set.seed(30806151)  #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,]

################################################################################
# Q4: Implement a classification model using each of the following techniques
# Decision Tree
decTreeModel = tree(MHT~., data = WAUS.train)

# Naïve Bayes
naiveBayesModel = naiveBayes(MHT~., data = WAUS.train)

# Bagging
baggingModel <- bagging(MHT~., data = WAUS.train, mfinal = 10)

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

```
# Random Forest
randomForestModel <- randomForest(MHT~., data = WAUS.train)

################################################################################
# Q5: Using the test data, classify each of the test cases as ‘more humid tomorrow’ or ‘less
humid tomorrow’. # Create a confusion matrix and report the accuracy of each model.

# Decision Tree
decTreePredict = predict(decTreeModel, WAUS.test, type = "class" )
table(observed = WAUS.test$MHT, predicted = decTreePredict)

# Naïve Bayes
naiveBayesPredict = predict(naiveBayesModel, WAUS.test)
table(observed = WAUS.test$MHT, predicted = naiveBayesPredict)

# Bagging
baggingPredict <- predict.bagging(baggingModel, WAUS.test)
cat("\n#Bagging Confusion\n")
print(baggingPredict$confusion)

# Boosting
boostingPredict <- predict.boosting(boostingModel, WAUS.test)
cat("\n#Boosting Confusion\n")
print(boostingPredict$confusion)

# Random Forest
randomForestPredict <- predict(randomForestModel, WAUS.test)
table(observed = WAUS.test$MHT, predicted = randomForestPredict)

################################################################################
# Q6: Using the test data, calculate the confidence of predicting ‘more humid tomorrow’ for
# each case

# Decision Tree
decTreePredictV = predict(decTreeModel, WAUS.test, type = "vector" )
decTreePredictV[,2]
WAUS.test$MHT
DCTpred = prediction(decTreePredictV[,2], WAUS.test$MHT)
DCTperf <-performance(DCTpred, "tpr", "fpr")
plot(DCTperf, col = 'red')
abline(0,1)

DCTauc = performance(DCTpred, "auc") # Calculate the AUC
print(as.numeric(DCTauc@y.values))

# Naïve Bayes
naiveBayesPredictV = predict(naiveBayesModel, WAUS.test, type = "raw")
NBpred <- prediction(naiveBayesPredictV[,2], WAUS.test$MHT)
NBperf <- performance(NBpred, "tpr", "fpr")
plot(NBperf, add = TRUE, col = 'pink')

NBauc = performance(NBpred, "auc") # Calculate the AUC
print(as.numeric(NBauc@y.values))
```

```r
# Bagging
bagPred <- prediction(baggingPredict$prob[,2], WAUS.test$MHT)
bagPerf <- performance(bagPred, "tpr", "fpr")
plot(bagPerf, add = TRUE, col = 'green')

bagAuc = performance(bagPred, "auc") # Calculate the AUC
print(as.numeric(bagAuc@y.values))

# Boosting
boostPred <- prediction(boostingPredict$prob[,2], WAUS.test$MHT)
boostPerf <- performance(boostPred, "tpr", "fpr")
plot(boostPerf, add = TRUE, col = 'blue')

boostAuc = performance(boostPred, "auc") # Calculate the AUC
print(as.numeric(boostAuc@y.values))

# Random Forest
randomForestPredictV <- predict(randomForestModel, WAUS.test, type = "prob")
RFpred <- prediction(randomForestPredictV[,2], WAUS.test$MHT)
RFperf <- performance(RFpred, "tpr", "fpr")
plot(RFperf, add = TRUE, col = 'orange')

RFauc = performance(RFpred, "auc") # Calculate the AUC
print(as.numeric(RFauc@y.values))

# Creating a legened for ROC
legend( x = "bottomright", legend = c("Decision Tree","Naïve
Bayes","Bagging","Boosting","Random Forest"), fill = c("red","pink","green","blue","orange"),
cex=0.6)

################################################################################
# Q8:Examining each of the models, determine the most important variables in predicting
whether it will be more humid tomorrow or not

cat("\n#Decision Tree Attribute Importance\n")
print(summary(decTreeModel))

cat("\n#Baging Attribute Importance\n")
print(baggingModel$importance)

cat("\n#Boosting Attribute Importance\n")
print(boostingModel$importance)

cat("\n#Random Forest Attribute Importance\n")
print(randomForestModel$importance)

################################################################################
# Q9: Starting with one of the Q4 classifiers, create a classifier that is simple enough for a
person to be able to classify whether it will be more humid tomorrow or not by hand.
simpleModel=tree(MHT~ WindSpeed9am + Rainfall + Pressure9am + RISK_MM + Temp3pm +
WindGustSpeed + Evaporation + MinTemp + WindDir9am + Sunshine + Cloud3pm + Location +
WindSpeed3pm + Pressure3pm, data = WAUS.train)
# Cross Validation
cvModel = cv.tree(simpleModel, FUN = prune.misclass)
```

```
print(cvModel)
# Post-pruning
pruneModel = prune.misclass(simpleModel, best = 10)
plot(pruneModel)
text(pruneModel, pretty = 0)
#Building confusion matrix to determine accuracy
simplePredict = predict(pruneModel, WAUS.test, type = "class" )
table(observed = WAUS.test$MHT, predicted = simplePredict)

# Testing and determining the AUC value
simpleV = predict(pruneModel, WAUS.test, type = "vector" )
simplepred <- prediction(simpleV[,2], WAUS.test$MHT)
simplepref <-performance(simplepred, "tpr", "fpr")
simpleAUC = performance(simplepred, "auc") # Calculate the AUC
print(as.numeric(simpleAUC@y.values))


################################################################################
# Q10: Creating the best tree-based classifier
#Approach 1: Cross validation method
WAUS.train <- data.frame(WAUS.train)
ctrl <- trainControl(method = "cv", number = 10)

model <- train(MHT~., data = WAUS.train, method = "cforest", trControl = ctrl)
prediction <- predict(model, WAUS.test)
confusion_matrix <- table(prediction, WAUS.test$MHT)
confusion_matrix

V <- predict(model, WAUS.test, type = "prob")
pred <- prediction(V[,2], WAUS.test$MHT)
perf <- performance(pred, "tpr", "fpr")
auc = performance(pred, "auc") # Calculate the AUC
print(as.numeric(auc@y.values))

#Approach 2: Adjusting parameters
bestTBCModel <- randomForest(MHT~ .- RainToday , data = WAUS.train, ntree = 1000)
bestTBCPredict <- predict(bestTBCModel, WAUS.test)
bestTBCPredict
# Confusion matrix to calculate accuracy
table(observed = WAUS.test$MHT, predicted = bestTBCPredict)

# Calculating AUC
bestTBCPredictV <- predict(bestTBCModel, WAUS.test, type = "prob")

bestTBCpred <- prediction(bestTBCPredictV[,2], WAUS.test$MHT)
bestTBCperf <- performance(bestTBCpred, "tpr", "fpr")

bestTBCauc = performance(bestTBCpred, "auc") # Calculate the AUC
print(as.numeric(bestTBCauc@y.values))


################################################################################
# Q11: Implement an Artificial Neural Network classifier
library(neuralnet)
```

```r
# Clearing environment except for some variables
rm(list= ls()[!(ls() %in% c('WAUS','WAUS.train', 'WAUS.test'))])

# Making a copy of both sets to use for nn
nnWAUS.train = WAUS.train
nnWAUS.test = WAUS.test

# Adding separate columns for more humid tomorow: if 1 then More, if 0 then Les
nnWAUS.train$More=nnWAUS.train$MHT == 1
nnWAUS.train$Less=nnWAUS.train$MHT == 0

# Data pre-processing: standardization: all values have mean of 0 and standard deviation of 1
preproc <- preProcess(nnWAUS.train, method = c("center", "scale"))  # Adjust input_columns as
per your dataset
nnWAUS.train <- predict(preproc, nnWAUS.train)
nnWAUS.test <- predict(preproc, nnWAUS.test)


# Building the ANN model
waus.nn = neuralnet(More + Less~ .- RainToday - Year - MHT, nnWAUS.train, hidden = 4, rep = 2)
plot(waus.nn, rep = "best") #plotting
waus.nn$result.matrix

# Predicting
wausnn.pred = compute(waus.nn, nnWAUS.test[,!names(nnWAUS.test) %in% c("RainToday", "Year",
"MHT")])
wausnn.predr = round(wausnn.pred$net.result, 0) #Rounding result
wausnn.predrdf = as.data.frame(as.table(wausnn.predr)) #Converting to a dataframe
# Leave only ones
wausnn.predrdfs = wausnn.predrdf[!wausnn.predrdf$Freq == 0,]
wausnn.predrdfs = wausnn.predrdfs[!wausnn.predrdfs$Freq == -1,]


wausnn.predrdfs
# Changing the format to get the confusion matrix
wausnn.predrdfs$Freq=NULL

wausnn.predrdfs$Var2 <-ifelse(wausnn.predrdfs$Var2=="A",1, 0 )

colnames(wausnn.predrdfs) = c("Obs", "MHT")
wausnn.predrdfs = wausnn.predrdfs[order(wausnn.predrdfs$Obs),]

#Getting confusion matrix
table(observed = nnWAUS.test$MHT, predicted = wausnn.predrdfs$MHT)


# Calculate AUC value
auc_value <- roc(nnWAUS.test$MHT, wausnn.predrdfs$MHT)$auc

auc_value

################################################################################
```

```r
# Q12: Fit a new classifier to the data, test and report its performance in the same way as
for previous models.
rm(list= ls()[!(ls() %in% c('WAUS','WAUS.train', 'WAUS.test'))])

require(xgboost)

#define predictor and response variables in training set
train_x = data.matrix(WAUS.train[,!names(WAUS.train) %in% c("MHT")])
train_y = WAUS.train$MHT

#define predictor and response variables in testing set
test_x = data.matrix(WAUS.test[,!names(WAUS.test) %in% c("MHT")])
test_y = WAUS.test$MHT

# Converting to numerical 0 and 1 from factor
train_y <- ifelse(as.numeric(train_y) == "1", 0, 1)
test_y <- ifelse(as.numeric(test_y) == "1", 0, 1)

#define final training and testing sets (converting to a proper matrix format)
xgb_train = xgb.DMatrix(data = train_x, label = train_y)
xgb_test = xgb.DMatrix(data = test_x, label = test_y)


# Set the parameters for the XGBoost model
params <- list(
  objective = "binary:logistic", #binary classification using logistic regression
  eval_metric = "logloss" # evaluation metric
)

# Train the XGBoost model with 100 boosting rounds
xgb_model <- xgboost(params = params, data = xgb_train, nrounds = 50)

# Predict on the test set using the trained model
predictions <- round((predict(xgb_model, xgb_test)),0)

# Getting a confusion matrix
table(observed = WAUS.test$MHT, predicted = predictions)

# Calculating the AUC value
pred <- prediction(predictions, WAUS.test$MHT)
boostAuc = performance(pred, "auc")
print(as.numeric(boostAuc@y.values))
```