Exploring Classifiers in Predicting RainTomorrow Variable

Grace Nathania

INTRODUCTION

This report explores which type of classifiers is the best in classifying a dataset. The data used in this exploration is a set of meteorological observations (2000 rows of data) to predict whether it will be raining on the next day in Australia or not. Based on the observation of the data, the proportion of fine days is 80.46% while the proportion of rainy days is 19.54%.

There are 24 predictor variables and 1 predicted variable[1]. Below is the summary of predictor variables*:

Attributes	# of NA	Min.	Max.	Median	Mean	Variance	Std.Dev
MinTemp	24	-4.2	31.4	11.0	11.29	39.55	6.29
MaxTemp	25	6.8	46.8	23.7	23.87	52.21	7.23
Rainfall	44	0.0	147.8	0.0	1.92	57.84	7.61
Evaporation	953	0.0	70.4	5.0	6.08	27.18	5.21
Sunshine	1058	0.0	14.0	9.4	8.44	13.31	3.65
WindGustSpeed	107	11.0	107.0	37.0	39.22	192.78	13.88
WindSpeed9am	85	0.0	63.0	13.0	14.88	86.42	9.30
WindSpeed3pm	88	0.0	65.0	17.0	18.55	84.41	9.19
Humidity9am	31	3.0	100.0	67.0	67.38	392.45	19.81
Humidity3pm	24	0.0	100.0	45.0	46.24	438.02	20.93
Pressure9am	86	996.8	1036.6	1018.5	1018.53	40.99	6.40
Pressure3pm	89	996.2	1034.7	1015.9	1015.91	40.27	6.35
Cloud9am	610	0.0	8.0	5.0	4.22	8.65	2.94
Cloud3pm	628	0.0	8.0	4.0	4.28	7.62	2.76
Temp9am	23	-1.0	37.9	16.4	16.63	42.47	6.52
Temp3pm	23	4.7	45.8	22.1	22.39	49.26	7.02

*Statistics description is rounded to 2 decimal places and **not all attributes** (location, wind direction, and data of the date's rain, and RainToday) are assessed.

Before further exploring the classifiers, from the statistics description of some attributes above, it is noticeable that the data has missing value. Thus, pre-processing is needed to make the dataset suitable for the model:

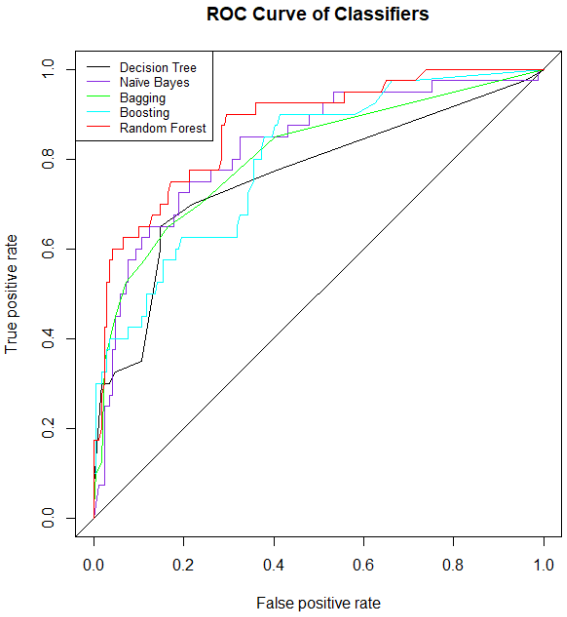
1. Omitting rows with NA value(s)
2. Changing dependent variable's class (RainTomorrow) as factor.

The classifiers used in this analysis are, but not limited to, **Decision Tree**, **Naïve Bayes**, **Bagging**, **Boosting**, and **Random Forest**. This report will analyse which classifiers is the best to classify a dataset by evaluating the accuracy, area under the curve, root-mean-squared error and will try to create the best tree-based classifier by utilising the previous data in R studio.

[1] Detail on attributes is attached in Appendix

CLASSIFIERS EXPLORATION

Classifiers	Decision Tree	Naïve Bayes	Bagging	Boosting	Random Forest
Accuracy (%)	78.95	82.30	85.65	84.69	88.52
AUC	0.76	0.83	0.81	0.80	0.87
RMSE	0.46	0.42	0.38	0.39	0.34



Based on each classifiers' accuracy and area under the curve (AUC), Random Forest is the best classifiers since the model is the most accurate, with the highest AUC value;

- Accuracy tells how many times the model is correctly classified the data.
- AUC value tells the capability of the model to distinguish between classes.
- RMSE measures the difference between predicted and known (train) value.

Examining the ROC curve of classifiers, Random Forest's curve is closed to ideal model (TPR of 1 and FPR of 0).

In a further ado, evaluating independent attributes is needed to know which attribute(s) is important in determining rain on the next day. Below is the data*:

Classifiers	Decision Tree	Naïve Bayes	Bagging	Boosting	Random Forest
Day	9.21	N/A	1.31	3.02	3.99
Month	4.53		0.80	0.45	2.60
Year	N/A		0.37	1.48	2.34
Location	0.07		0.00	0.50	2.21
MinTemp	4.13		0.00	3.54	4.43
MaxTemp	2.75		1.02	2.01	3.60
Rainfall	9.80		0.88	1.28	6.27
Evaporation	1.51		0.00	2.24	3.93
Sunshine	24.36		13.83	11.86	16.75
WindGustDir	19.47		17.36	14.55	12.37
WindGustSpeed	2.67		1.09	3.08	5.19
WindDir9am	20.84		10.54	13.47	12.44
WindDir3pm	23.53		16.26	16.03	13.31
WindSpeed9am	N/A		0.00	1.05	2.58
WindSpeed3pm	3.55		1.25	0.94	3.45
Humidity9am	6.17		0.00	2.32	5.49
Humidity3pm	38.72		31.95	14.27	17.85
Pressure9am	5.88		0.36	0.75	4.53
Pressure3pm	6.31		0.80	0.56	4.71
Cloud9am	1.30		0.37	0.45	5.42
Cloud3pm	8.41		0.78	4.39	10.02
Temp9am	3.91		0.00	1.34	3.60
Temp3pm	7.13		1.00	0.48	5.22
RainToday	N/A		0.00	0.00	1.42

*Top 5 attributes are marked in red while top 6-10 attributes are marked in yellow
*We cannot tell importance variables used in Naïve's Bayes since it is not tree-based classifiers

Based on each classifier, the hours of bright sunshine (Sunshine), the direction and the speed of strongest wind gust (WindGustDir and WindGustSpeed), the direction of wind at 9am and 3pm (WinDir9am and WindDir3pm), and the humidity at 3pm (Humidity3pm) over the day play important role in determining the rain of the day. These 5 attributes should not be omitted from the data since they are significant.

However, on predictors where the value of their importance is 0 or not available, they can be omitted from the classifiers. For an instance, RainToday can be omitted from Decision Tree, Bagging, and Boosting data since its score is not available in Decision Tree and 0 in Bagging and Boosting. On the other hand, RainToday cannot be omitted from the data when Random Forest is used since its importance value is greater than 0. A test is done to know whether removing RainToday will impact the accuracy of all classifiers except Naïve Bayes:

Classifiers	**Decision Tree**	**Bagging**	**Boosting**	**Random Forest**
Accuracy with RainToday	78.95	85.65	84.69	88.52
Accuracy without RainToday	78.95	88.04	86.12	87.56

Referring to the accuracy after removing RainToday, we can conclude that when a predictor:

1. has no data of importance, it will not improve the accuracy
2. has 0 as a value of importance, it will increase the accuracy
3. has a value of importance greater than 0, it will decrease the accuracy

To sum up, when the importance value of a predictor is **not** 0 or unavailable, it should not be omitted from the model since it takes part in classifying the data.

CREATING THE BEST TREE-BASED CLASSIFIER

Referred to the previous observation, Random Forest is the best classifier among all considering its high accuracy, AUC, and small RMSE value. Random Forest is the best since it is an ensemble of Decision Tree of different predictors and it decides based on the majority of the outputs from the decision trees. Furthermore, a random forest can reduce the high variance from the other flexible models by combining many trees into one ensemble model.

In order to improve the model, the number of tree has to be increased (default value is 500). Therefore, it will be varied from 501 to 1000 and we will see which number of tree has higher accuracy with minimum value. Based on the observation, 523 trees inside Random Forest are enough to improve the accuracy. In addition, the parameter for importance is also set to be TRUE.

Cross validation is not needed in improving the model. Reason is Random Forest itself randomizes the variable selection during each tree split. Considering tree split, the number of best split to improve the model is also calculated. Based on further exploration, this data is obtained:

Classifier	**Number of Tree**	**Number of Split**	**Accuracy (%)**	**RMSE**
Random.Forest.1	500	4	88.52	0.34
Random.Forest.2	523	4	89.47	0.32

*Random.Forest.1 indicates previous random forest's model while Random.Forest.2 indicates improved version of the model

Observing the accuracy of improved version of previous random forest's model, we can clearly see that it is higher than the previous model; number of tree is increased by 23 while number of split remains the same. In addition, the RMSE value is decreased which indicates that the predicted value is closer to actual value. Therefore, we can conclude that increasing the number of tree and setting the importance parameter to True give us more accurate model.

ARTIFICIAL NEURAL NETWORK CLASSIFIER

Classifying the data with tree-based classifiers and Naïve Bayes, it is no harm to fit the data into a human brain-based classifier: Artifical Neural Network (ANN) to know whether this model gives better accuracy in predicting the rain. ANN is chosen since it is claimed to be accurate and able to handle redundant attributes and noisy data.

Knowing that all predictors take part in classifying the data, we will use all predictors by first doing required pre-processing of the data;

1. changing the value of "Yes" into 1 and "No" into 0 in RainToday and RainTomorrow
2. encoding non-numeric data into numeric with one hot encoding such as for wind direction
3. normalising the data with min-max normalisation

Once all necessary pre-processing processes are done, we will create a formula to be fitted to the model with as.formula function. Afterwards, we will try different number of hidden layer(s). A single, double and triple hidden layers have been tried by utilising the 2/3 principle for number of "node" at each hidden layer. A single hidden layer gives the highest **accuracy of 84.21%** (hidden = 1; a different number of hidden from 1 to 23, since there are 24 predictors, has been tried). Another combination of multiple layers such as 24:10:4:2:1 or 24:10:6:1 (red color indicates the hidden layers) do not give us better accuracy than 1 single hidden layer.

The most fundamental difference between ANN and the other 5 classifiers is: ANN is a technique of deep learning while the others are machine learning's. ANN utilises layers made up of interconnected node to compute the output of the network. Hidden layer(s) is also used in ANN to perform non-linear transformations for the inputs to enter the network. Inputs for ANN will also receive a weight.

In addition, ANN is able to analyse image, audio and text while other classifiers might not have this ability. However, ANN is not really suitable in working with tabular data, like the data we are current using for this report, since long pre-processing is needed, high complexity of model, low number of data and a possibility of bias. Therefore, ANN is more suitable in working with image, audio and text while the other machine learning models, such as Random Forest and Bagging, works better with tabular data.

CONCLUSION

Referred to the accuracy, AUC, and RMSE value of all classifiers in predicting the rain in Australia, Random Forest has the highest accuracy and AUC and the smallest RMSE value which it can be concluded that Random Forest is a single best classifier. Since Random Forest is the best classifier, we try to improve the model by adjusting its parameters: importance, mtry, and ntree. By increasing the number of tree (ntree) and set importance to TRUE, the accuracy is improved and the RMSE value is decreased. Based on the analysis, mtry does not impact the accuracy much since the same number of split is enough.

Furthermore, importance of predictors is examined for each model except Naïve Bayes since it is not a tree-based model. Sunshine, WindGustDir, WindGustSpeed, WinDir9am, WinDir3pm and Humidity3pm are the top 5 most important variables for all classifiers. There are some attributes that could be omitted from the data since they have very little effect on performance such as those with importance value of 0 or no importance value. However, predictors with a value greater than 0 must not be omitted from the model since they still have effect on model's performance.

On the other hands, another classifier is tried to predict the rain in Australia: Artificial Neural Network (ANN). ANN is a type of deep learning and in order to use this model, we have to do pre-processing for our data: one-hot-encoding and min-max normalisation. This model gives better accuracy, 84.21%, than Decision Tree and Naïve Bayes model but not for Bagging, Boosting, and Random Forest. The main reason is ANN is not suitable enough to work with tabular data while the data we have is tabular.

In conclusion, Random Forest tends to be better than other classifiers in working with tabular data. Random Forest is an ensemble of Decision Tree and each Decision Tree has different predictors which makes it consider many factors to produce an output. In Random Forest, all predictors commonly are important (importance value > 0) and there is no way to remove any of them since it will affect its performance.

APPENDIX – Data's Description and R Code

Description of the data:

Attributes 1:3 - Day, Month, Year of the observation

Attribute 4 - Location: the location of the observation

Attribute 5 - MinTemp: the daily minimum temperature in degrees celsius

Attribute 6 - MaxTemp: the daily maximum temperature in degrees celsius

Attribute 7 - Rainfall: the rainfall recorded for the day in mm

Attribute 8 - Evaporation: the evaporation (mm) in the 24 hours to 9am

Attribute 9 - Sunshine: hours of bright sunshine over the day.

Attribute 10 - WindGust: direction of the strongest wind gust over the day.

Attribute 11 - WindGustSpeed: speed (km/h) of the strongest wind gust over the day.

Attribute 12 - WindDir9am: direction of the wind at 9am

Attribute 13 - WindDir3pm: direction of the wind at 3pm

Attribute 14 - WindSpeed9am: speed (km/hr) averaged over 10 minutes prior to 9am

Attribute 15 - WindSpeed3pm: speed (km/hr) averaged over 10 minutes prior to 3pm

Attribute 16 - Humidity9am: humidity (percent) at 9am

Attribute 17 - Humidity3pm: humidity (percent) at 3pm

Attribute 18 - Pressure9am: atmospheric pressure (hpa) reduced to mean sea level at 9am

Attribute 19 - Pressure3pm: atmospheric pressure (hpa) reduced to mean sea level at 3pm

Attribute 20 - Cloud9am: fraction of sky obscured by cloud at 9am. This is measured in "oktas", which are a unit of eigths. It records how many eigths of the sky are obscured by cloud. A 0 measure indicates completely clear sky whilst an 8 indicates that it is completely overcast.

Attribute 21 - Cloud3pm: fraction of sky obscured by cloud at 3pm.

Attribute 22 - Temp9am: temperature (degrees C) at 9am

Attribute 23 - Temp3pm: temperature (degrees C) at 3pm

Attribute 24 - RainToday: boolean: 1 if precipitation (mm) in the 24 hours to 9am exceeds 1mm, otherwise 0

Attribute 25 - RainTomorrow: the target variable. Did it rain tomorrow?

R Code

```
library(pastecs)
library(rpart)
library(e1071)
library(adabag)
library(randomForest)
library(ROCR)
library(data.table)
library(mltools)

rm(list = ls())
WAUS = read.csv("WAUS2020.csv")
L = as.data.frame(c(1:49))
set.seed(30241510) # 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

##########First Step##########
#Cleaning the data from NA value
WAUS = na.omit(WAUS)

#Set RainTomorrow column as factor
WAUS$RainTomorrow = as.factor(WAUS$RainTomorrow)
str(WAUS)

##########Question 1##########
# a. What is the proportion of rainy days to fine days
# analysing number of yes and no under RainToday
raindays_proportion = as.data.frame(table(WAUS$RainTomorrow)) # No = 560; Yes = 136
fine_days = round((raindays_proportion[1,2]/sum(raindays_proportion[,2]))*100, digits = 2)
rainy_days = round((raindays_proportion[2,2]/sum(raindays_proportion[,2]))*100, digits = 2)
cat("Proportion of fine days: ",fine_days) #80.46
cat("Proportion of rainy days: ",rainy_days) #19.54

# b. descriptions of the predictor (independent) variables
data_summary = as.data.frame(round(stat.desc(WAUS[,-c(1,2,3,4,10,12,13,24,25)]), 2))
data_summary = t(data_summary[c(3,4,5,8,9,12,13),])

##########Question 2##########
#Question 2 has been done before doing question 1. I remove the NA value because
#it is unnecessary to be included. I have also change RainTomorrow as factor

##########Question 3##########
#Divide data into 70% training and 30% test
set.seed(30241510)
train.row = sample(1:nrow(WAUS), 0.7*nrow(WAUS))
train.data = WAUS[train.row,]
test.data = WAUS[-train.row,]

##########Question 4##########
#Implement a classification model using each of the following techniques
#Decision Tree
tree.waus = rpart(RainTomorrow~., data = train.data, method = "class")
tree.pred = predict(tree.waus, test.data, type = "class")

#Naïve Bayes
nb.waus = naiveBayes(RainTomorrow~., data = train.data)
nb.pred = predict(nb.waus, test.data)

#Bagging
ba.waus = bagging(RainTomorrow~., data = train.data, mfinal = 10)
ba.pred = predict.bagging(ba.waus, newdata = test.data)

#Boosting
bo.waus = boosting(RainTomorrow~., data = train.data, mfinal = 10)
bo.pred = predict.boosting(bo.waus, newdata = test.data)

#Random Forest
rf.waus = randomForest(RainTomorrow~., data = train.data)
rf.pred = predict(rf.waus, test.data)

##########Question 5##########
#Create confusion matrix and calculate accuracy
#Decision Tree - should I do cv?
tree.cm = table(actual = test.data$RainTomorrow, predicted = tree.pred)
tree.cm.accuracy = round(mean(tree.pred == test.data$RainTomorrow)*100, digits = 2)
cat("Accuracy of Decision Tree:",tree.cm.accuracy,"%")

#Naïve Bayes
nb.cm = table(actual = test.data$RainTomorrow, predicted = nb.pred)
nb.cm.accuracy = round(mean(nb.pred == test.data$RainTomorrow)*100, digits = 2)
cat("Accuracy of Naïve Bayes is:", nb.cm.accuracy,"%")
```

```
#Bagging
ba.cm = ba.pred$confusion
ba.cm.accuracy = round(mean(ba.pred$class == test.data$RainTomorrow)*100, digits = 2)
cat("Accuracy of Bagging is:", ba.cm.accuracy,"%")

#Boosting
bo.cm = bo.pred$confusion
bo.cm.accuracy = round(mean(bo.pred$class == test.data$RainTomorrow)*100, digits = 2)
cat("Accuracy of Boosting is:", bo.cm.accuracy,"%")

#Random Forest
rf.cm = table(actual = test.data$RainTomorrow, predicted = rf.pred)
rf.cm.accuracy = round(mean(rf.pred == test.data$RainTomorrow)*100, digits = 2)
cat("Accuracy of Random Forest is:", rf.cm.accuracy,"%")

##########Question 6##########
#Calculate confidence, construct ROC curve

#Decision tree
tree.waus.vector = predict(tree.waus, test.data, type = "prob")
tree.waus.pred = prediction(tree.waus.vector[,2], test.data$RainTomorrow)
tree.waus.perf = performance(tree.waus.pred,"tpr","fpr")
plot(tree.waus.perf, main = "ROC Curve of Classifiers")
abline(0,1)

#Naive Bayes
nb.waus.raw = predict(nb.waus, test.data, type = "raw")
nb.waus.pred = prediction(nb.waus.raw[,2], test.data$RainTomorrow)
nb.waus.perf = performance(nb.waus.pred, "tpr", "fpr")
plot(nb.waus.perf, add = TRUE, col = "blueviolet")

#Bagging
ba.waus.pred = prediction(ba.pred$prob[,2], test.data$RainTomorrow)
ba.waus.perf = performance(ba.waus.pred, "tpr","fpr")
plot(ba.waus.perf, add = TRUE, col = "green")

#Boosting
bo.waus.pred = prediction(bo.pred$prob[,2], test.data$RainTomorrow)
bo.waus.perf = performance(bo.waus.pred,"tpr","fpr")
plot(bo.waus.perf, add = TRUE, col = "cyan")

#Random Forest
rf.waus.prob = predict(rf.waus, test.data, type = "prob")
rf.waus.pred = prediction(rf.waus.prob[,2], test.data$RainTomorrow)
rf.waus.perf = performance(rf.waus.pred, "tpr","fpr")
plot(rf.waus.perf, add = TRUE, col = "red")

#add legend to plot
legend("topleft", legend= c("Decision Tree","Naïve Bayes","Bagging","Boosting","Random Forest"),col =
c("black","blueviolet","green","cyan","red"), lty=1, cex=0.8)

#Calculate AUC
#Decision Tree
tree.waus.auc = performance(tree.waus.pred,"auc")
tree.waus.auc = round(tree.waus.auc@y.values[[1]],2)
cat("Area Under the Curve (AUC) of Decision Tree is:", tree.waus.auc)

#Naive Baiyes
nb.waus.auc = performance(nb.waus.pred,"auc")
nb.waus.auc = round(nb.waus.auc@y.values[[1]],2)
cat("Area Under the Curve (AUC) of Naïve Bayes is:", nb.waus.auc)

#Bagging
ba.waus.auc = performance(ba.waus.pred, "auc")
ba.waus.auc = round(ba.waus.auc@y.values[[1]],2)
cat("Area Under the Curve (AUC) of Bagging is:", ba.waus.auc)

#Boosting
bo.waus.auc = performance(bo.waus.pred, "auc")
bo.waus.auc = round(bo.waus.auc@y.values[[1]],2)
cat("Area Under the Curve (AUC) of Boosting is:", bo.waus.auc)

#Random Forest
rf.waus.auc = performance(rf.waus.pred, "auc")
rf.waus.auc = round(rf.waus.auc@y.values[[1]],2)
cat("Area Under the Curve (AUC) of Random Forest is:", rf.waus.auc)

##########Question 7##########
#Calculate RMSE as another indicator
#Decision Tree
tree.rmse = round(sqrt(mean((as.numeric(tree.pred) - as.numeric(test.data$RainTomorrow))^2)),2)

#Naive Bayes
nb.rmse = round(sqrt(mean((as.numeric(nb.pred) - as.numeric(test.data$RainTomorrow))^2)),2)
```

```
#Bagging
ba.rmse = round(sqrt(mean((ifelse(ba.pred$class == "Yes",2,1) - as.numeric(test.data$RainTomorrow))^2)),2)

#Boosting
bo.rmse = round(sqrt(mean((ifelse(bo.pred$class == "Yes",2,1) - as.numeric(test.data$RainTomorrow))^2)),2)

#Random Forest
rf.rmse = round(sqrt(mean((as.numeric(rf.pred) - as.numeric(test.data$RainTomorrow))^2)),2)

classifiers.data  = data.frame(Classifiers = c("Decision Tree","Naïve Bayes","Bagging","Boosting","Random
Forest"),Accuracy = c(tree.cm.accuracy,nb.cm.accuracy,ba.cm.accuracy,bo.cm.accuracy,rf.cm.accuracy), AUC =
c(tree.waus.auc, nb.waus.auc, ba.waus.auc,  bo.waus.auc, rf.waus.auc), RMSE = c(tree.rmse, nb.rmse, ba.rmse,
bo.rmse, rf.rmse))

classifiers.data.accuracy = classifiers.data [order(-classifiers.data$Accuracy),]
classifiers.data.auc = classifiers.data [order(-classifiers.data$AUC),]
classifiers.data.rmse = classifiers.data [order(classifiers.data$RMSE),]

##########Question 8##########
#DecisionTree
tree.waus$variable.importance

#Naive Bayes - cannot tell

#Bagging
print(ba.waus$importance)

#Boosting
print(bo.waus$importance)

#RandomForest
print(rf.waus$importance)

#try removing RainToday from train and test data
train.data.rt = train.data[,-24]
test.data.rt = test.data[,-24]

#fit new data to all classifiers except Naive Bayes and calculate their accuracy
#Decision Tree
tree.waus.rt = rpart(RainTomorrow~., data = train.data.rt, method = "class")
tree.pred.rt = predict(tree.waus.rt, test.data.rt, type = "class")
tree.acc.rt = round(mean(tree.pred.rt == test.data.rt$RainTomorrow)*100, digits = 2)

#Bagging
ba.waus.rt = bagging(RainTomorrow~., data = train.data.rt, mfinal = 10)
ba.pred.rt = predict.bagging(ba.waus.rt, newdata = test.data.rt)
ba.acc.rt = round(mean(ba.pred.rt$class == test.data.rt$RainTomorrow)*100, digits = 2)

#Boosting
bo.waus.rt = boosting(RainTomorrow~., data = train.data.rt, mfinal = 10)
bo.pred.rt = predict.boosting(bo.waus.rt, newdata = test.data.rt)
bo.acc.rt = round(mean(bo.pred.rt$class == test.data.rt$RainTomorrow)*100, digits = 2)

#Random Forest
rf.waus.rt = randomForest(RainTomorrow~., data = train.data.rt)
rf.pred.rt = predict(rf.waus.rt, test.data.rt)
rf.acc.rt = round(mean(rf.pred.rt == test.data.rt$RainTomorrow)*100, digits = 2)

#Create comparison accuracy table
comparison_rt = data.frame (Classifiers = c("Previous Accuracy", "Current Accuracy"), Decision.Tree =
c(tree.cm.accuracy, tree.acc.rt), Bagging = c(ba.cm.accuracy, ba.acc.rt), Boosting = c(bo.cm.accuracy,
bo.acc.rt), Random.Forest = c(rf.cm.accuracy, rf.acc.rt))

##########Question 9##########
#Since Random Forest has the highest accuracy and AUC, and the lowest RMSE, we take random forest
#classifiers as our tree-based classifier. We will be trying to improve the model by
#1 - setting importance parameter to TRUE
#2 - increasing ntree of random forest
#3 - searching the best number of split

#Looking for better ntree
num.tree = c(550, 600, 650, 700, 750, 800, 850, 900, 950, 1000)
best.rf.acc = rf.cm.accuracy
best.rf = rf.waus
best.rf.pred = rf.pred

for (i in 501:1000){
  set.seed(30241510)
  rf.waus.new = randomForest(RainTomorrow~., data = train.data, importance = TRUE, ntree = i)
  rf.pred.new = predict(rf.waus.new, test.data)
  acc = round(mean(rf.pred.new == test.data$RainTomorrow)*100, digits = 2)
  if (acc > best.rf.acc){
    best.rf.acc = acc
    best.rf = rf.waus.new
    best.rf.pred = rf.pred.new }
}
```

```
cat("best number of tree is", best.rf$ntree, "with accuracy of", best.rf.acc,"%")

#looking for better mtry starting from minimum split + 1 to the number of predictors
for (i in (best.rf$mtry+1):24){
  set.seed(30241510)
  rf.mtry = randomForest(RainTomorrow~., data = train.data, importance = TRUE, ntree = best.rf$ntree, mtry = i)
  rf.mtry.pred = predict(rf.mtry, test.data)
  acc = round(mean(rf.mtry.pred == test.data$RainTomorrow)*100, digits = 2)
  if (acc > best.rf.acc){
    best.rf.acc = acc
    best.rf = rf.mtry
    best.rf.pred = rf.mtry.pred
  }
}
cat("best number of split is", best.rf$mtry, "with accuracy of", best.rf.acc, "%")

#Calculating RMSE of better random forest
best.rf.rmse = round(sqrt(mean((as.numeric(best.rf.pred) - as.numeric(test.data$RainTomorrow))^2)),2)

#Creating data frame to compare previous and current model
rf.comparison = data.frame(Classifier = c("Random.Forest.1", "Random.Forest.2"), ntree = c(rf.waus$ntree,
best.rf$ntree), mtry = c(rf.waus$mtry, best.rf$mtry), Accuracy = c(rf.cm.accuracy, best.rf.acc), RMSE =
c(rf.rmse, rf.rmse.rt))

##########Question 10##########
#implements ANN - we will include all attributes and
#we will first do one hot encoding for non-numerical data and change
#the value of "Yes" to 1 and "No" to 0 for RainToday and RainTomorrow

library(neuralnet)
train.data[,24:25] = ifelse(train.data[,24:25] == "Yes", 1, 0)
test.data[,24:25] = ifelse(test.data[,24:25] == "Yes", 1, 0)

train.data = one_hot(as.data.table(train.data))
test.data = one_hot(as.data.table(test.data))

#Normalised the data with min max normalisation
normalise = function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

train.data = as.data.frame(lapply(train.data, FUN = normalise))
test.data = as.data.frame(lapply(test.data, normalise))

#Create a formula
predictors_name = colnames(train.data)
f = as.formula(paste("RainTomorrow ~",paste(predictors_name[!predictors_name %in% "RainTomorrow"], collapse = " +
")))

#Create a for loop to know the best number of hidden layer(s)
nn.acc = 0
hidden.layer = 0
best.nn = 0
for (i in 1:23){
  set.seed(30241510)
  waus.nn = neuralnet(f, train.data, hidden = i, linear.output = FALSE)
  waus.nn.pred = compute(waus.nn, test.data[,1:69])
  waus.nn.pred = as.data.frame(round(waus.nn.pred$net.result,0))
  current_acc = round(mean(waus.nn.pred$V1 == test.data$RainTomorrow)*100, digits = 2)
  if (current_acc > nn.acc){
    nn.acc = current_acc
    hidden.layer = i
    best.nn = waus.nn
  }
}

cat("Accuracy of ANN: ", nn.acc, "%\n")
cat("Number of hidden layer(s): ", hidden.layer)
```