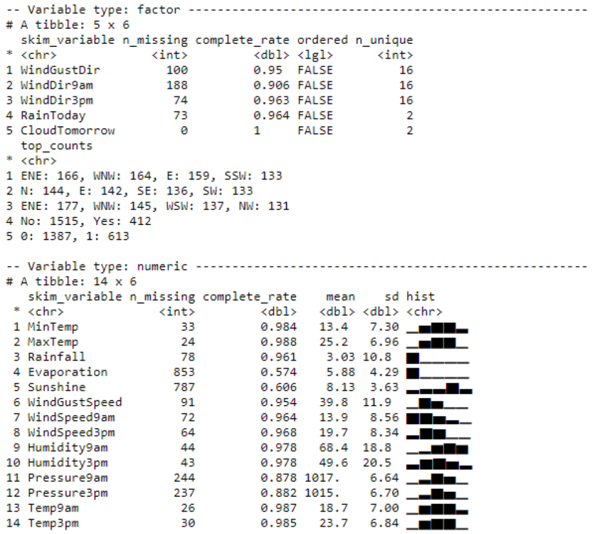
1. Exploratory of variables (Q1)



```
-- Variable type: factor -------------------------------------------------------
# A tibble: 5 x 6
  skim_variable n_missing complete_rate ordered n_unique
* <chr>             <int>         <dbl> <lgl>      <int>
1 WindGustDir         100         0.95  FALSE         16
2 WindDir9am          188         0.906 FALSE         16
3 WindDir3pm           74         0.963 FALSE         16
4 RainToday            73         0.964 FALSE          2
5 CloudTomorrow         0         1     FALSE          2
  top_counts
* <chr>
1 ENE: 166, WNW: 164, E: 159, SSW: 133
2 N: 144, E: 142, SE: 136, SW: 133
3 ENE: 177, WNW: 145, WSW: 137, NW: 131
4 No: 1515, Yes: 412
5 0: 1387, 1: 613

-- Variable type: numeric ------------------------------------------------------
# A tibble: 14 x 6
   skim_variable n_missing complete_rate    mean    sd hist
 * <chr>             <int>         <dbl>   <dbl> <dbl> <chr>
 1 MinTemp              33         0.984   13.4   7.30 ▁▃▇▇▂
 2 MaxTemp              24         0.988   25.2   6.96 ▁▃▇▇▁
 3 Rainfall             78         0.961    3.03 10.8  ▇▁▁▁▁
 4 Evaporation         853         0.574    5.88  4.29 ▇▁▁▁▁
 5 Sunshine            787         0.606    8.13  3.63 ▂▂▂▇▂
 6 WindGustSpeed        91         0.954   39.8  11.9  ▁▇▃▁▁
 7 WindSpeed9am         72         0.964   13.9   8.56 ▇▇▃▂▁
 8 WindSpeed3pm         64         0.968   19.7   8.34 ▂▇▆▁▁
 9 Humidity9am          44         0.978   68.4  18.8  ▁▁▃▇▆
10 Humidity3pm          43         0.978   49.6  20.5  ▂▇▇▅▂
11 Pressure9am         244         0.878 1017.    6.64 ▁▂▇▅▁
12 Pressure3pm         237         0.882 1015.    6.70 ▁▂▇▅▁
13 Temp9am              26         0.987   18.7   7.00 ▁▅▇▇▂
14 Temp3pm              30         0.985   23.7   6.84 ▁▅▇▇▁
```

Figure 1.1. The statistical summary of the dataset

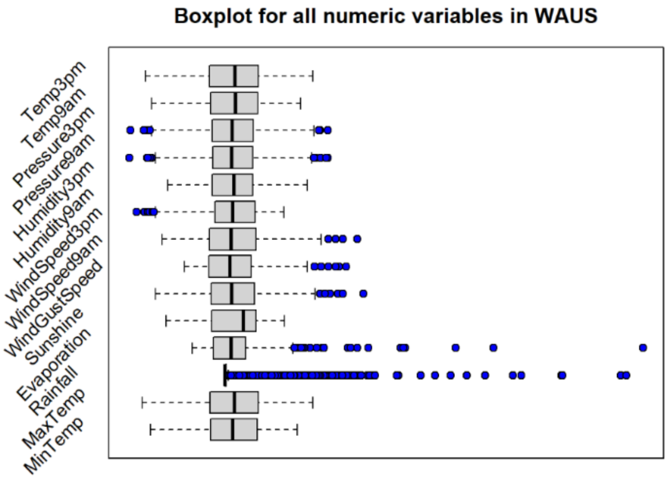


Figure 1.2. The boxplot of the dataset

Based on the statistical summary of the dataset (Figure 1.1), there are serveral noteworthy insights explained below:

- The proportion of cloudy days to clear days is 30.65% (Cloudy =1387, Not Cloudy = 613).
- The mean of Pressure9am and Pressure3pm are notably high.
- Most of the variables are skewed. For example, Rainfall and Evaporation are extremely right-skewed. That shows that its median is larger than its mean. Contrariwise, Humidity9am and Sunshine are left-skewed.
- A majority of variables contain NA values. Especially, Evaporation and Sunshine have 43% and 39% missing values.
- Since this analysis is predicting whether the following day will be cloudy, some variables are not essential to be predictors, namely Day, Month, and Year; hence they are omitted.

On top of the insights, based on (Figure 1.2), Rainfall is the most extreme variable as it has the highest range and abundant outliers. Few variables like Temp3pm and MaxTemp are of similar distribution.

2. Data Cleaning and Pre-processing (Q 2,3)

Before modelling, the data pre-processing should be conducted which is gleaned from the part 1 statistical summary.

2.1. Dealing with Categorical values

The data type of multiple variables is “Char” which is specified to be categorical. Using as.factor ( ) can categorise the variables of all wind direction, RainToday, location, and CloudTomorrow.

2.2. Handling missing values and outliers

All rows with NA values are removed, as a compromise between complexity for modelling and information loss. Generally, it is assumed that these data points are missing at random (MAR), meaning that the tendency for a data point to be missing is not related to the missing data. The cause perhaps was due to human errors during data collection, or some points of sensors functioned improperly on those days that weather cannot be detected. Moreover, the analysis involves the comparison between models. Some of the classifiers like Boosting and ANN cannot handle missing values while others can. This context must be ensured that the dataset is consistent when comparing models, therefore, removing missing values is necessary.

Furthermore, since all models are robust to outliers, they are not removed.

2.3. Data Splicing

At this stage, the dataset is cleaned and ready to model. The dataset is split into 70% for training and 30% for testing, with a random seed set as 31084222.

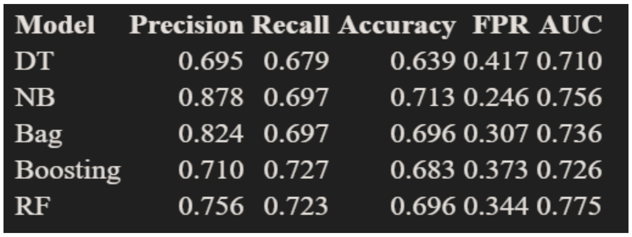
3. The comparison of the original models (Q 4,5,6,7)

The following is the code for five classifiers, all of which are at their default settings.

```
#Decision Tree
  tree.train = tree( CloudTomorrow ~ ., data=WAUS.train )
#Naïve Bayes
  NB.train = naiveBayes( CloudTomorrow~.,data=WAUS.train )
#Bagging
  bag.train = bagging( CloudTomorrow ~. ,data = WAUS.train, mfinal = 6 )
#Boosting
  boosting.train = boosting( CloudTomorrow ~. ,data = WAUS.train, mfinal = 7 )
#Random Forest
  rf.train = randomForest( CloudTomorrow ~. ,data = WAUS.train )
```

Assumption for the Naïve Bayes model: Naïve means that all variables are independent; however, since this can hardly be achieved in real life. Therefore, it is assumed that all variables are independent.

After testing each model, Random Forrest (RF) is the best model. The reasons are explained below:



Model	Precision	Recall	Accuracy	FPR	AUC
DT	0.695	0.679	0.639	0.417	0.710
NB	0.878	0.697	0.713	0.246	0.756
Bag	0.824	0.697	0.696	0.307	0.736
Boosting	0.710	0.727	0.683	0.373	0.726
RF	0.756	0.723	0.696	0.344	0.775

Figure 3.1. The summary of the performance for the models

Firstly, although the accuracy is lower than Naïve Bayes model, the difference is minor, and the value for RF is sufficiently high (Figure 3.1). That means there is 70 % for RF to classify if tomorrow is cloudy or not cloudy correctly. Other values also are higher than other models overall.

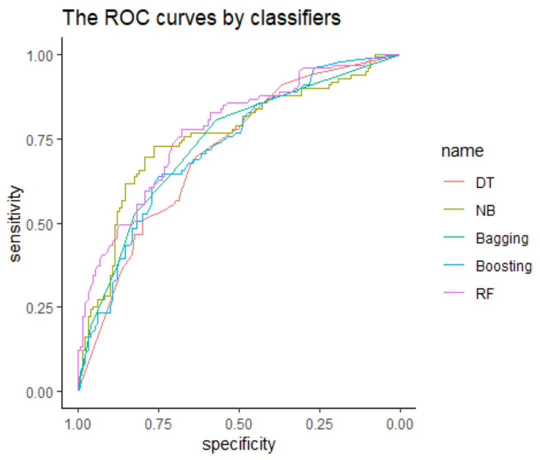


Figure 3.2. The ROC curves by classifiers

Secondly, AUC is where the best model was differentiated. Even though the accuracy of Bagging and RF is the same, the AUCs are different. That is, the dataset is biased towards “Cloudy” or “Not Cloudy”; Bagging might discriminate well on either side but RF tends to perform well overall. AUC is the proof of this case since it is an evaluation of the classifiers as thresholds vary over all possible values, of which RF rates comparatively higher than others. The cause might be that by partitioning the sample and randomly choosing the variables to model subtrees, the samples are more enriched, thus avoiding biases.

4. The variable importance (Q8)

Only Humidity3pm, MinTemp, Sunshine, WinDir3pm, WinDir9am and WinGustDir could be selected from the data because they are of higher effect on performance. The way the features are selected is based on the Gini index, which is a measure of variance; the higher the variance signifies the more misclassification exists, thus driving the values of Gini Index higher. In this scenario, ensemble methods like Bagging, Boosting and, Random Forrest are used to determine the variable importance.

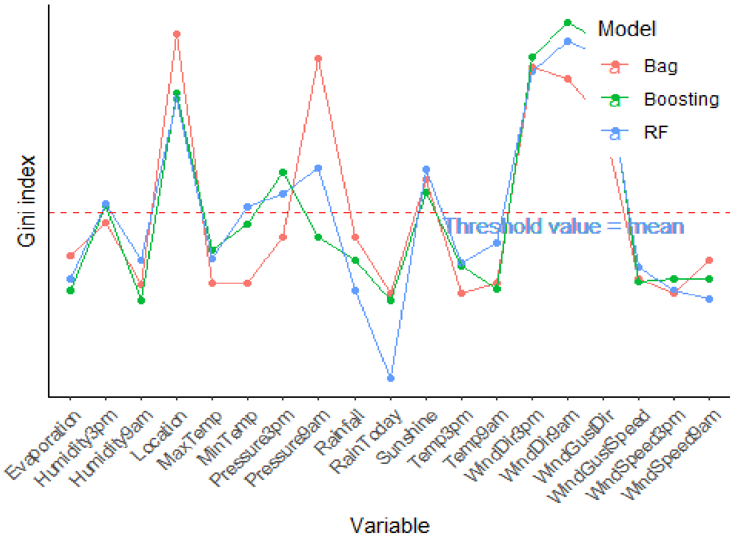


Figure 4.1. The importance of each variable by classifiers

The reason I do not use Naïve bayse and other classifiers to select features is that their variables are of equal weighting. Whereas methods like Boosting are more suitable as their task is to reduce the variance error; they analyse different subsets of data and generate a collective output at the end. Since Boosting weighs its variables from the bags, it is particularly useful. Therefore, variable importance is based on ensemble methods.

Three models follow similar patterns on weighing its variables. Hence, I set the threshold value to the mean (Figure 4.3). Only features higher than the mean, could they be selected, which yields better classification.

5.1. Evaluation of the improved model based on simplicity (Q9)

A decision tree is a simple classifier for a person to classify if it will be cloudy or not tomorrow.

5.1.1. Feature selection:

The conditions of being windy, cold, sunny, and high pressure are the four main factors affecting the decision of the class, all of which are based on the mentioned important variables. For the sake of simplicity in modelling, all predictor variables are categorical. Therefore, threshold values are set to determine the level of those conditions, which are listed below:

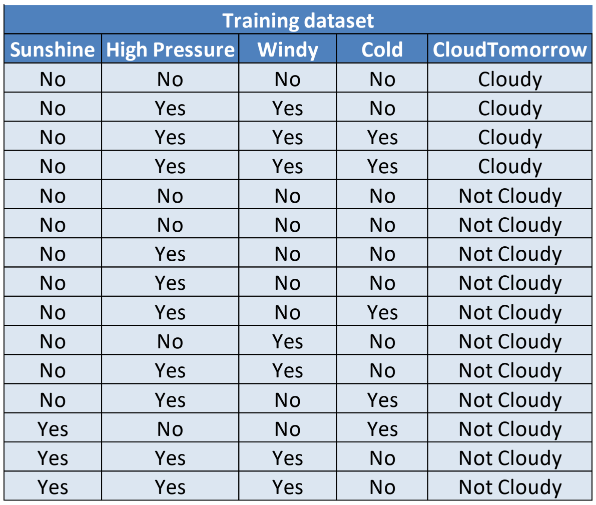
1. Pressure is high if "Pressure3pm" and "Pressure9am" > 1013. (Standard average pressure)

2. It is cold when "MinTemp" < 10.

3. It is sunny if the hours of "Sunshine" > 12.

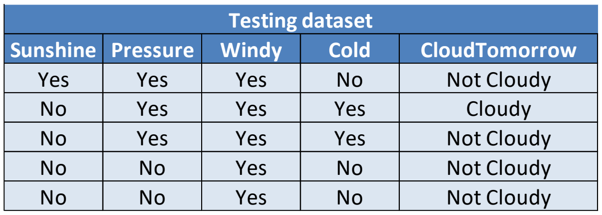
4. It is windy if the "WindSpeed3pm" and "WindSpeed9am" > 15

5.1.2. The process of modelling:



Training dataset				
Sunshine	High Pressure	Windy	Cold	CloudTomorrow
No	No	No	No	Cloudy
No	Yes	Yes	No	Cloudy
No	Yes	Yes	Yes	Cloudy
No	Yes	Yes	Yes	Cloudy
No	No	No	No	Not Cloudy
No	No	No	No	Not Cloudy
No	Yes	No	No	Not Cloudy
No	Yes	No	No	Not Cloudy
No	Yes	No	Yes	Not Cloudy
No	No	Yes	No	Not Cloudy
No	Yes	Yes	No	Not Cloudy
No	Yes	No	Yes	Not Cloudy
Yes	No	No	Yes	Not Cloudy
Yes	Yes	Yes	No	Not Cloudy
Yes	Yes	Yes	No	Not Cloudy

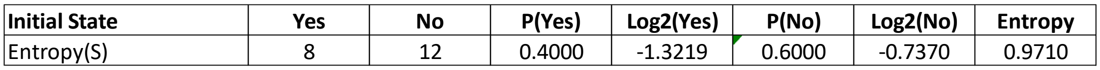
1.1. 15 Rows were sampled from the WAUS.Train for training the model.



Testing dataset				
Sunshine	Pressure	Windy	Cold	CloudTomorrow
Yes	Yes	Yes	No	Not Cloudy
No	Yes	Yes	Yes	Cloudy
No	Yes	Yes	Yes	Not Cloudy
No	No	Yes	No	Not Cloudy
No	No	Yes	No	Not Cloudy

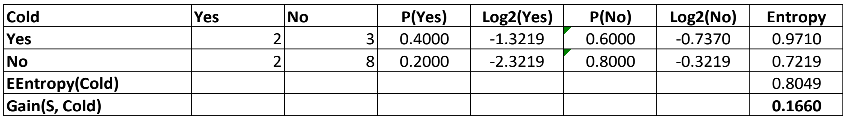
1.2. 5 Rows were sampled from the WAUS.Test for testing the model

2. Calculate the entropy of the Initial State.

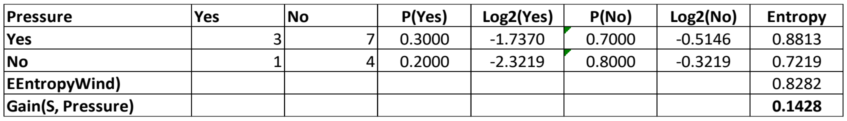


Initial State	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Entropy(S)	8	12	0.4000	-1.3219	0.6000	-0.7370	0.9710

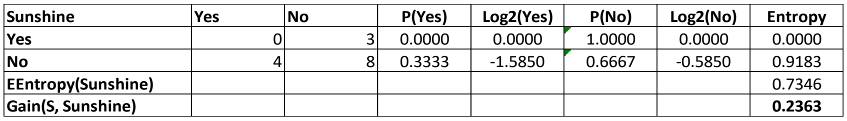
3. Calculate the Information Gain for each predictor variable.



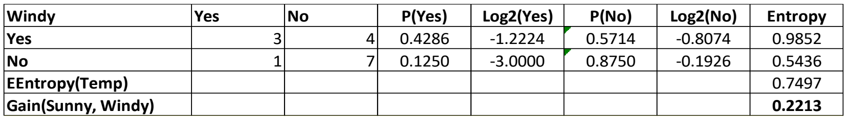
Cold	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Yes	2	3	0.4000	-1.3219	0.6000	-0.7370	0.9710
No	2	8	0.2000	-2.3219	0.8000	-0.3219	0.7219
EEntropy(Cold)							0.8049
Gain(S, Cold)							**0.1660**



Pressure	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Yes	3	7	0.3000	-1.7370	0.7000	-0.5146	0.8813
No	1	4	0.2000	-2.3219	0.8000	-0.3219	0.7219
EEntropyWind)							0.8282
Gain(S, Pressure)							**0.1428**



Sunshine	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Yes	0	3	0.0000	0.0000	1.0000	0.0000	0.0000
No	4	8	0.3333	-1.5850	0.6667	-0.5850	0.9183
EEntropy(Sunshine)							0.7346
Gain(S, Sunshine)							**0.2363**



Windy	Yes	No	P(Yes)	Log2(Yes)	P(No)	Log2(No)	Entropy
Yes	3	4	0.4286	-1.2224	0.5714	-0.8074	0.9852
No	1	7	0.1250	-3.0000	0.8750	-0.1926	0.5436
EEntropy(Temp)							0.7497
Gain(Sunny, Windy)							**0.2213**

From the tables above, “Sunshine” has the greatest information gain, thus becoming the root node.

4.1. Split the attributes for the next branches.

Note : Since the following parts of calculation conforms the same to the above steps, so from the next ones are skipped.

Gain(Not Sunny, Windy)=0.2212

Gain(Not Sunny, Cold)=0.0968

Gain(Not Sunny, Pressure)=0.0642

The above are the values of information gain for each variable given that the day is not sunny. “Windy” has the highest value, therefore it is the node when the day is not sunny.

Gain (Sunny, Windy)=0.9710

Gain (Sunny, Cold)=0.9710

Gain (Sunny, Pressure)=0.9710

Likewise, the above are the values of information gain given that the day is sunny. Since one can observe that all values are the same, therefore, the algorithm will select the feature randomly, in which “Pressure” is selected as a node when the day is sunny.

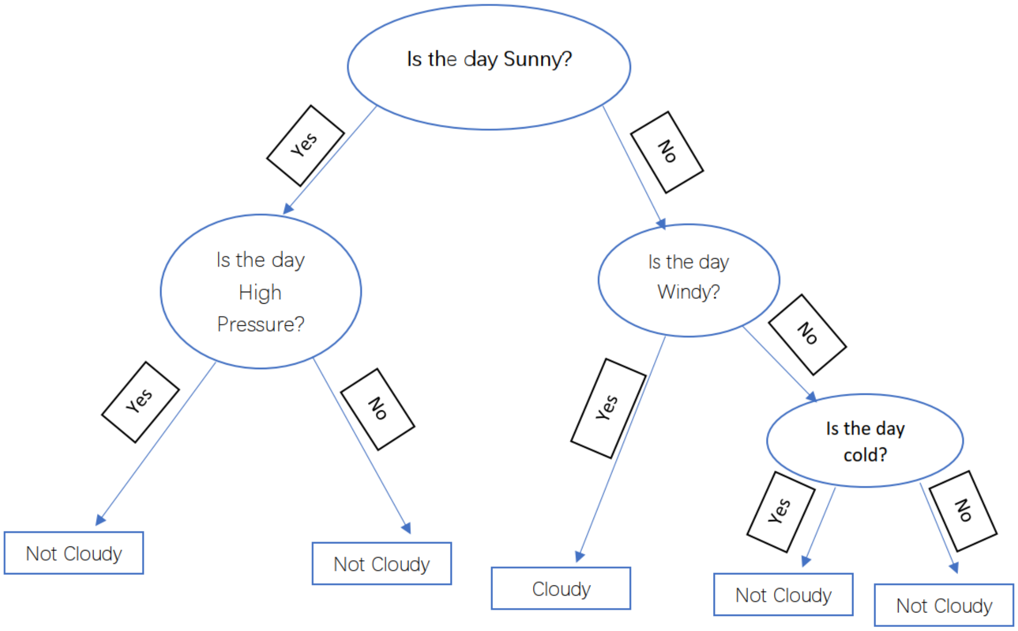
4.2. Split the attribute for the next branches.

Gain (Not Windy, Cold) = 0.4553

Gain (Windy, Cold) = 0.4200

Since the value for “Cold” given that it is not windy is larger, the node of cold will be selected when it is not sunny and not windy.

According the above steps, the model is generated as below :



The accuracy of the model = (1+2) / (5) = 0.6

The recall of the model = (1) / (2+1) = 0.3

Although the accuracy of this model is merely different than other classifiers, that does not necessarily indicate that this model is good. Given the limited number of the training dataset, this model can be highly biased. Perhaps it is the randomness of the testing set that drives this accuracy to 60 % in classifying the label. Likewise, the label of the training set is mostly “Not Cloudy” and “Not Sunny”. That is, this model produces many false negative result, given that the Recall rate and AUC could be very low. Hence, this model cannot perform better than other classifiers.

5.2. Evaluation of the improved model based on performance (Q10)

As Random Forrest is the best model, it is used to optimise so as to get the best performance. The accuracy went up from .69 to .72 whereas AUC remained the same.

It is improved by adjusting the optimal parameters using trial and error, with the code shown below.

```
rf.train = randomForest(CloudTomorrow ~. ,data = df.improved.train,
      ntree = 50000, importance = T, mtry = 2, nodesize = 42)
```

`Importance = T` was adjusted so that the importance of predictors could be assessed. Removing the low-important features could decrease the time complexity of the algorithm. Also, it increases the accuracy by reducing the variance.

`mtry = 2` was to tune the number of predictors sampled for splitting the trees from 4 to 2. Since RF's terminal nodes of each tree could be very small.

`ntree = 50000` was to increase the number of trees

`nodesize = 42` was to tune the minimum size of terminal nodes

Note : The reason why Cross-validation was not used was that it utilised too many computational resources to increase .1 in accuracy with lower AUC, which is not worthy.

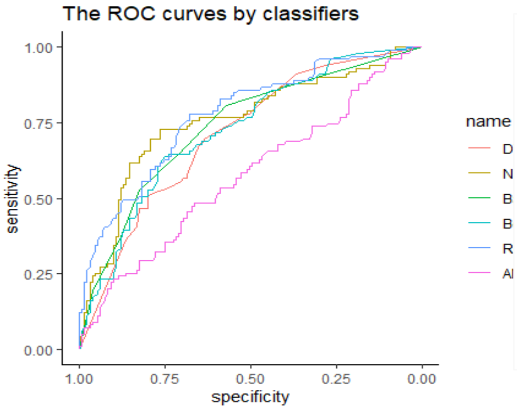
This model is found to be the best in terms of time complexity and performance. By setting up more trees produces higher accuracy at the expense of slower learning. When realising this trade-off, some parameters above are to balance the runtime, so that the time complexity would not perform beyond the one at default setting. For example, setting the node size larger makes trees smaller to be grown, thus taking less time. And waiving the low-important variables can also achieve this purpose. For these reasons, tunning the parameters allows the model to achieve higher accuracy meanwhile running at the similar efficiency to the RF at default setting.

6. Artificial Neural Network classifier (Q11)

The features are selected based on the measures done in part 4. Through trial and error for modelling ANN, it is concluded that the variables of the best model are all categorical, all of which are converted to binary columns. This means that scaling is not needed. In addition, as mentioned above, all the rows with missing values were already removed for model comparison; therefore, this step can be skipped.

```
nn <- neuralnet(CloudTomorrow0 + CloudTomorrow1 ~ . ,data = nn.train[,],
hidden = 3, linear.output = F, threshold=0.01)
```

6.1. This comparison between ANN and other classifiers



The performance of ANN does not result as great as other classifiers. Its accuracy is only .513 and AUC is .572. As shown in the ROC curve, with most of the points lying towards the centre "abline", ANN (Pink) does not perform as well as other classifiers, which indicates that lots of classes are labelled at random.

Generally, there are two reasons, in the aspects of dataset and fundamental.

The model is overfitting and has high variance: 2000 samples are not sufficient for a complex model like ANN to fit. ANN is considered a data-hungry model which entails millions of samples to form a big network in which its performance is strongly depended on the level of noise and richness of the dataset. However, in this case, if the samples were increased, then the trade-off is that it would be incomparable to other classifiers, since feeding different size of datasets to different models would draw an arbitrary conclusion.

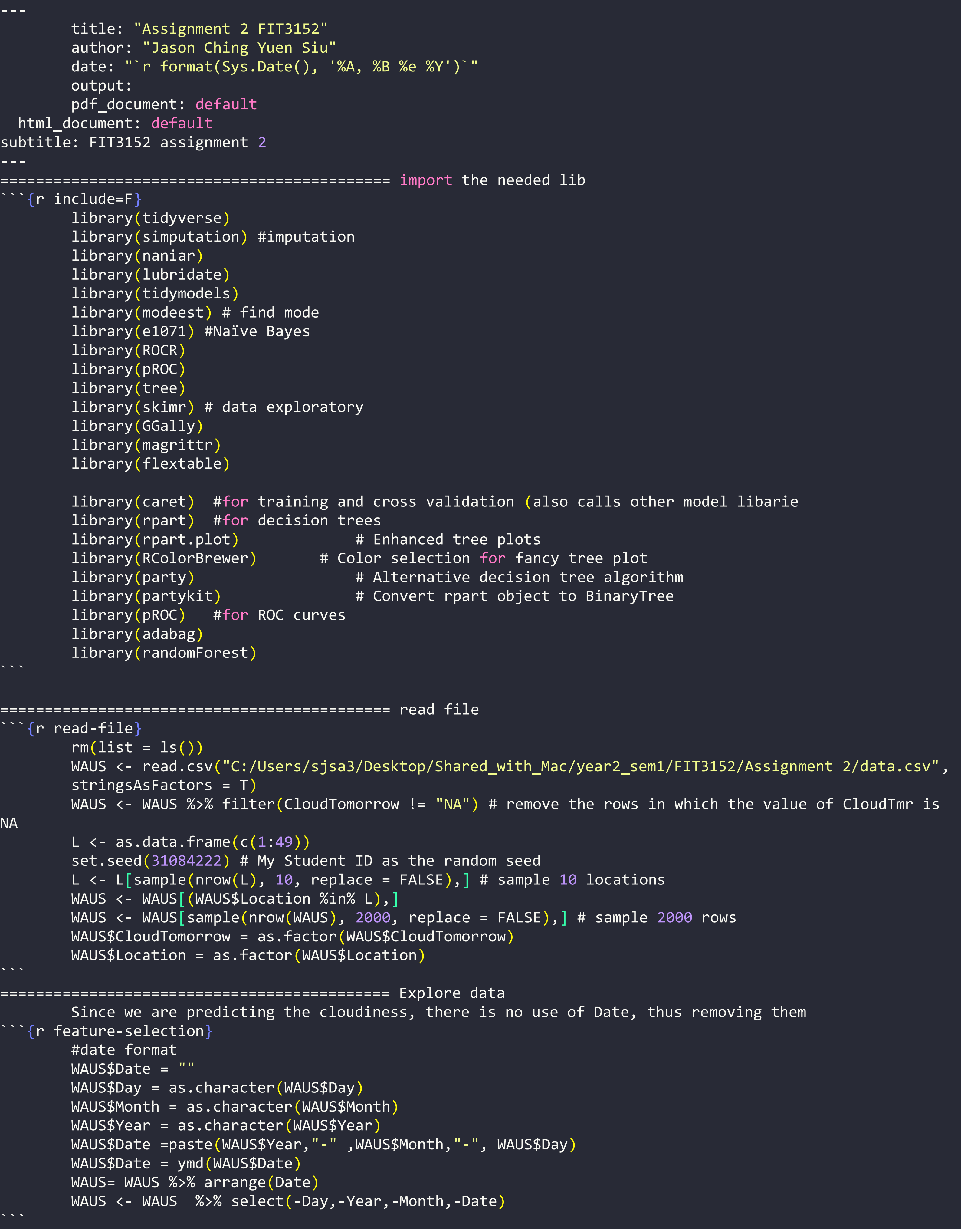
ANN model does not suit for solving this problem: Evidently, no matter how much learning rate and thresholds the ANN were altered in the parameter, after trial and error, the performance remained the same. That is, the model does not classify well if tomorrow is cloudy or not. The nature of problem defines which models to use. There is no the "perfect" classifier for solving all problems. As shown in the ROC curve, the only way is to implement every possible classifier that could work on the dataset, and then decide which model perform best for the problem. ANN is known to solve complex problems like image recognition but not the other ones. So, it is rational to infer that the model might not suit well due to the nature of problem.

Appendix

The code of this assignment is shown from the next page.

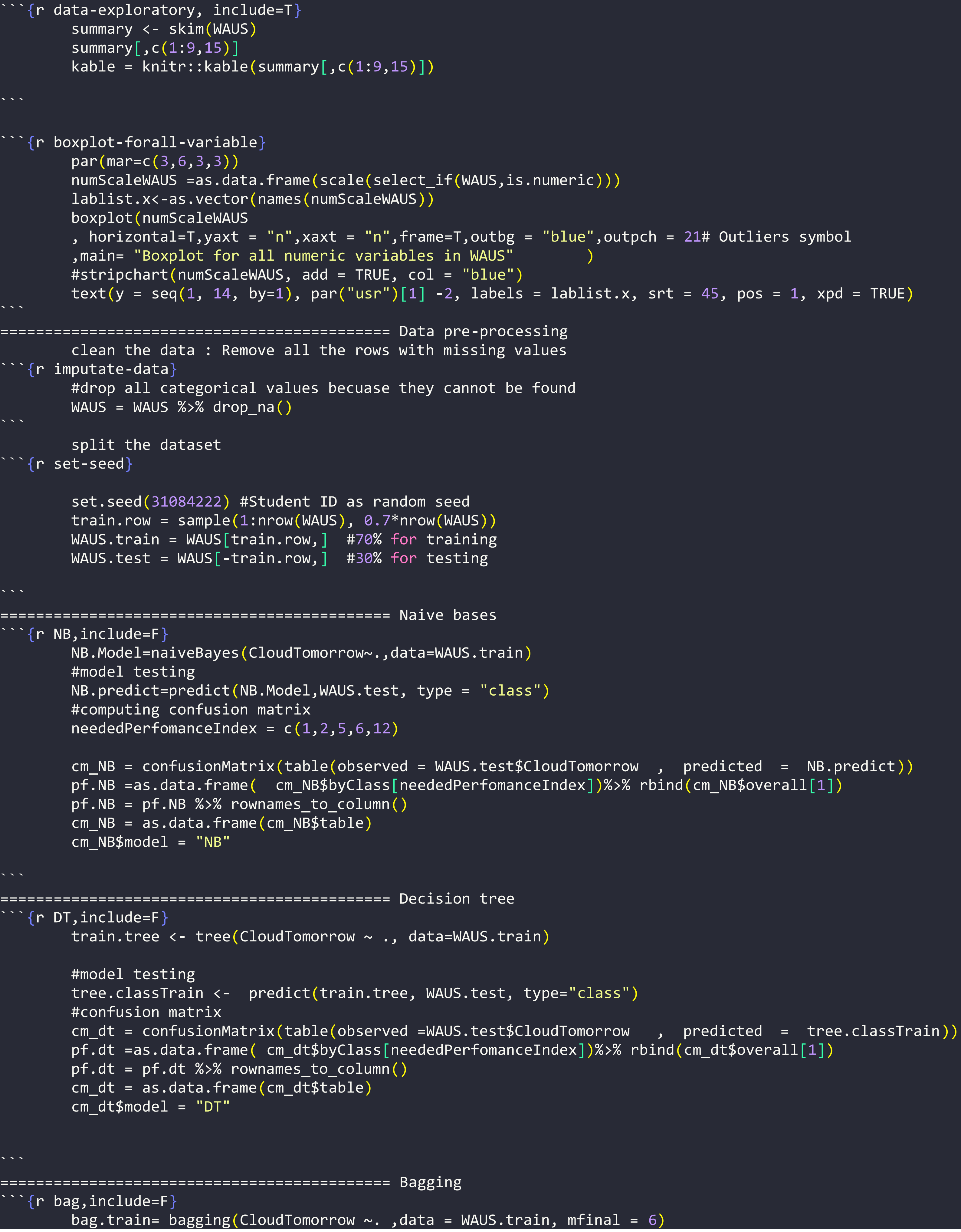
If codes from the next page is found to be invalid, please go to the link below for downloading the R markdown files:

https://github.com/JasonXsiu/FIT3152asm2



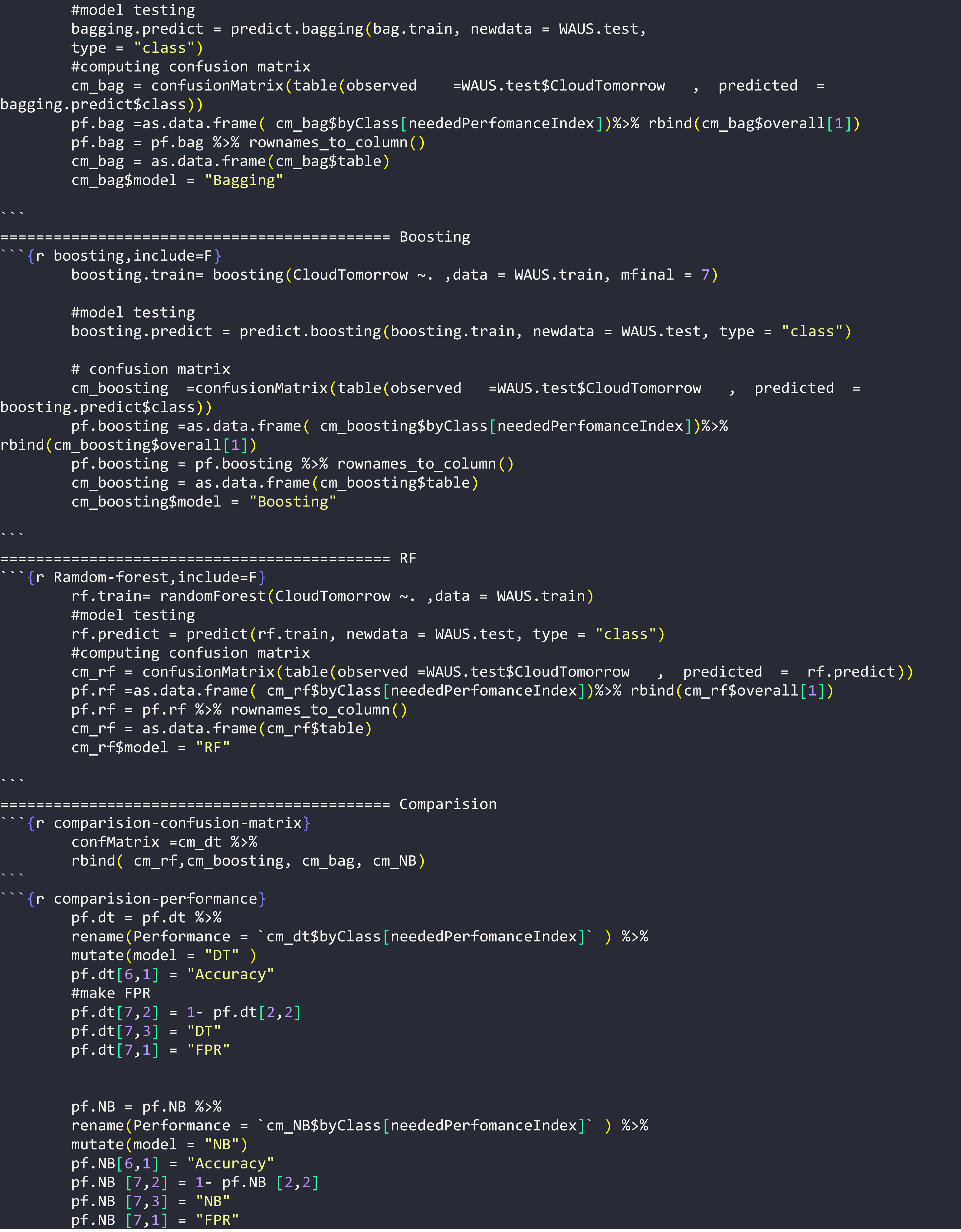
~~~
---
        title: "Assignment 2 FIT3152"
        author: "Jason Ching Yuen Siu"
        date: "`r format(Sys.Date(), '%A, %B %e %Y')`"
        output:
        pdf_document: default
  html_document: default
subtitle: FIT3152 assignment 2
---
============================================ import the needed lib
```{r include=F}
        library(tidyverse)
        library(simputation) #imputation
        library(naniar)
        library(lubridate)
        library(tidymodels)
        library(modeest) # find mode
        library(e1071) #Naïve Bayes
        library(ROCR)
        library(pROC)
        library(tree)
        library(skimr) # data exploratory
        library(GGally)
        library(magrittr)
        library(flextable)

        library(caret)  #for training and cross validation (also calls other model libarie
        library(rpart)  #for decision trees
        library(rpart.plot)             # Enhanced tree plots
        library(RColorBrewer)       # Color selection for fancy tree plot
        library(party)                  # Alternative decision tree algorithm
        library(partykit)               # Convert rpart object to BinaryTree
        library(pROC)   #for ROC curves
        library(adabag)
        library(randomForest)
~~~

== read file
```{r read-file}
        rm(list = ls())
        WAUS <- read.csv("C:/Users/sjsa3/Desktop/Shared_with_Mac/year2_sem1/FIT3152/Assignment 2/data.csv",
        stringsAsFactors = T)
        WAUS <- WAUS %>% filter(CloudTomorrow != "NA") # remove the rows in which the value of CloudTmr is
NA
        L <- as.data.frame(c(1:49))
        set.seed(31084222) # My Student ID as the random seed
        L <- L[sample(nrow(L), 10, replace = FALSE),] # sample 10 locations
        WAUS <- WAUS[(WAUS$Location %in% L),]
        WAUS <- WAUS[sample(nrow(WAUS), 2000, replace = FALSE),] # sample 2000 rows
        WAUS$CloudTomorrow = as.factor(WAUS$CloudTomorrow)
        WAUS$Location = as.factor(WAUS$Location)
```
== Explore data
        Since we are predicting the cloudiness, there is no use of Date, thus removing them
```{r feature-selection}
        #date format
        WAUS$Date = ""
        WAUS$Day = as.character(WAUS$Day)
        WAUS$Month = as.character(WAUS$Month)
        WAUS$Year = as.character(WAUS$Year)
        WAUS$Date =paste(WAUS$Year,"-" ,WAUS$Month,"-", WAUS$Day)
        WAUS$Date = ymd(WAUS$Date)
        WAUS= WAUS %>% arrange(Date)
        WAUS <- WAUS  %>% select(-Day,-Year,-Month,-Date)
```
```
```

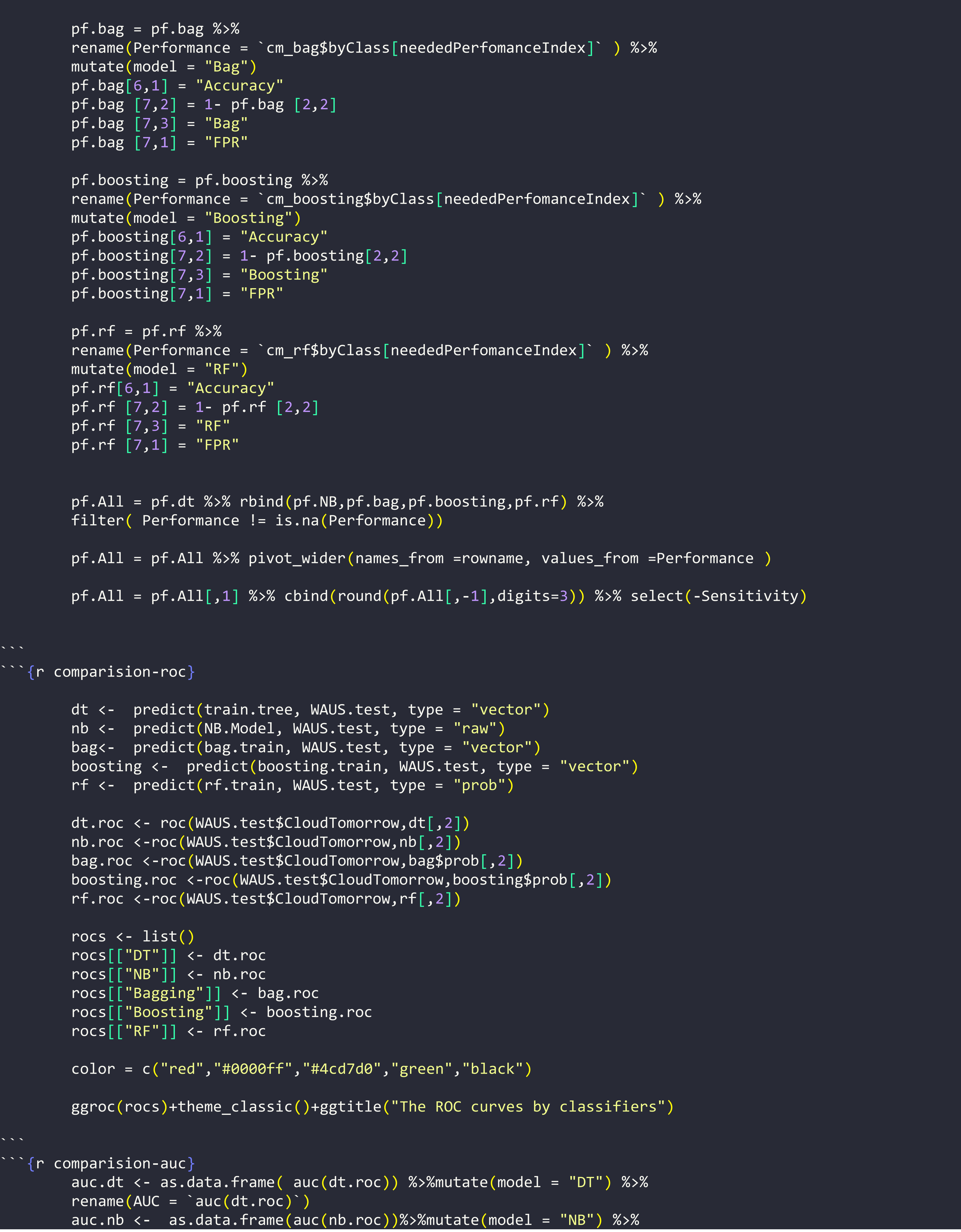


~~~
```{r data-exploratory, include=T}
        summary <- skim(WAUS)
        summary[,c(1:9,15)]
        kable = knitr::kable(summary[,c(1:9,15)])

~~~

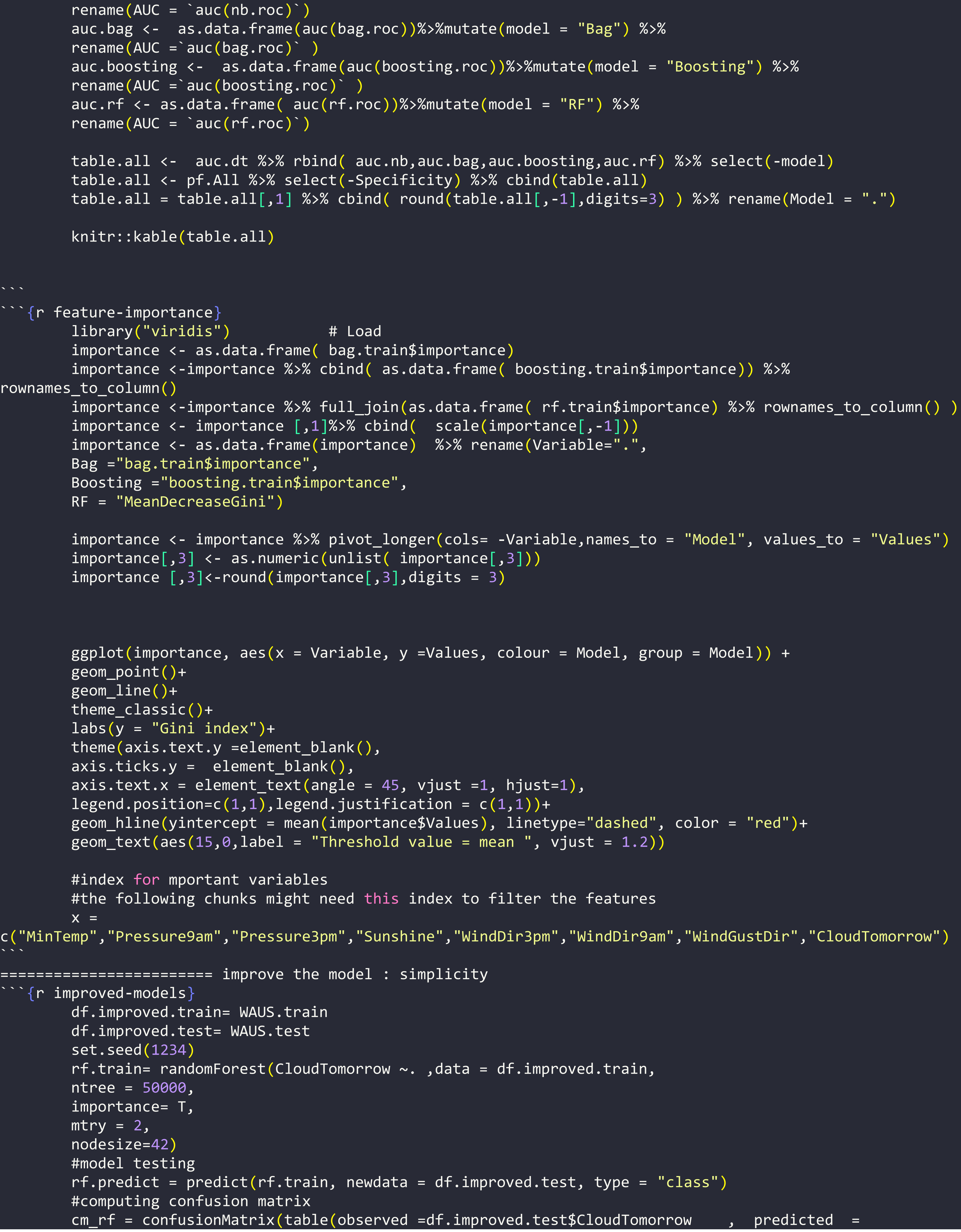
```{r boxplot-forall-variable}
        par(mar=c(3,6,3,3))
        numScaleWAUS =as.data.frame(scale(select_if(WAUS,is.numeric)))
        lablist.x<-as.vector(names(numScaleWAUS))
        boxplot(numScaleWAUS
        , horizontal=T,yaxt = "n",xaxt = "n",frame=T,outbg = "blue",outpch = 21# Outliers symbol
        ,main= "Boxplot for all numeric variables in WAUS"        )
        #stripchart(numScaleWAUS, add = TRUE, col = "blue")
        text(y = seq(1, 14, by=1), par("usr")[1] -2, labels = lablist.x, srt = 45, pos = 1, xpd = TRUE)
```
== Data pre-processing
        clean the data : Remove all the rows with missing values
```{r imputate-data}
        #drop all categorical values becuase they cannot be found
        WAUS = WAUS %>% drop_na()
```
        split the dataset
```{r set-seed}

        set.seed(31084222) #Student ID as random seed
        train.row = sample(1:nrow(WAUS), 0.7*nrow(WAUS))
        WAUS.train = WAUS[train.row,]  #70% for training
        WAUS.test = WAUS[-train.row,]  #30% for testing

```
== Naive bases
```{r NB,include=F}
        NB.Model=naiveBayes(CloudTomorrow~.,data=WAUS.train)
        #model testing
        NB.predict=predict(NB.Model,WAUS.test, type = "class")
        #computing confusion matrix
        neededPerfomanceIndex = c(1,2,5,6,12)

        cm_NB = confusionMatrix(table(observed = WAUS.test$CloudTomorrow  ,  predicted  =  NB.predict))
        pf.NB =as.data.frame(  cm_NB$byClass[neededPerfomanceIndex])%>% rbind(cm_NB$overall[1])
        pf.NB = pf.NB %>% rownames_to_column()
        cm_NB = as.data.frame(cm_NB$table)
        cm_NB$model = "NB"

```
== Decision tree
```{r DT,include=F}
        train.tree <- tree(CloudTomorrow ~ ., data=WAUS.train)

        #model testing
        tree.classTrain <-  predict(train.tree, WAUS.test, type="class")
        #confusion matrix
        cm_dt = confusionMatrix(table(observed =WAUS.test$CloudTomorrow   ,  predicted  =  tree.classTrain))
        pf.dt =as.data.frame( cm_dt$byClass[neededPerfomanceIndex])%>% rbind(cm_dt$overall[1])
        pf.dt = pf.dt %>% rownames_to_column()
        cm_dt = as.data.frame(cm_dt$table)
        cm_dt$model = "DT"


```
== Bagging
```{r bag,include=F}
        bag.train= bagging(CloudTomorrow ~. ,data = WAUS.train, mfinal = 6)
```



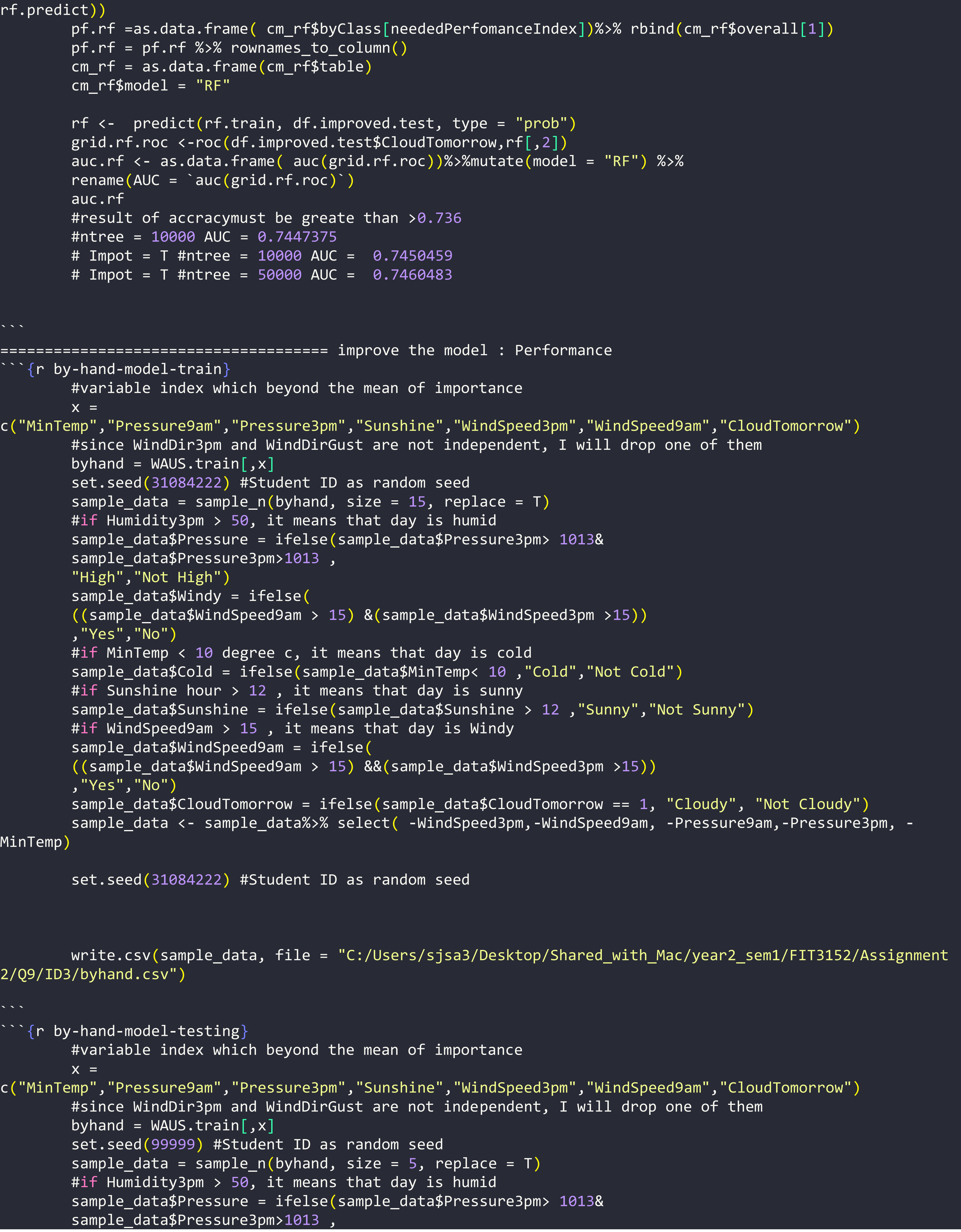
~~~
        #model testing
        bagging.predict = predict.bagging(bag.train, newdata = WAUS.test,
        type = "class")
        #computing confusion matrix
        cm_bag = confusionMatrix(table(observed    =WAUS.test$CloudTomorrow   ,  predicted  =
bagging.predict$class))
        pf.bag =as.data.frame( cm_bag$byClass[neededPerfomanceIndex])%>% rbind(cm_bag$overall[1])
        pf.bag = pf.bag %>% rownames_to_column()
        cm_bag = as.data.frame(cm_bag$table)
        cm_bag$model = "Bagging"

```
============================================ Boosting
```{r boosting,include=F}
        boosting.train= boosting(CloudTomorrow ~. ,data = WAUS.train, mfinal = 7)

        #model testing
        boosting.predict = predict.boosting(boosting.train, newdata = WAUS.test, type = "class")

        # confusion matrix
        cm_boosting  =confusionMatrix(table(observed   =WAUS.test$CloudTomorrow   ,  predicted  =
boosting.predict$class))
        pf.boosting =as.data.frame( cm_boosting$byClass[neededPerfomanceIndex])%>%
rbind(cm_boosting$overall[1])
        pf.boosting = pf.boosting %>% rownames_to_column()
        cm_boosting = as.data.frame(cm_boosting$table)
        cm_boosting$model = "Boosting"

```
============================================ RF
```{r Ramdom-forest,include=F}
        rf.train= randomForest(CloudTomorrow ~. ,data = WAUS.train)
        #model testing
        rf.predict = predict(rf.train, newdata = WAUS.test, type = "class")
        #computing confusion matrix
        cm_rf = confusionMatrix(table(observed =WAUS.test$CloudTomorrow   ,  predicted  =  rf.predict))
        pf.rf =as.data.frame( cm_rf$byClass[neededPerfomanceIndex])%>% rbind(cm_rf$overall[1])
        pf.rf = pf.rf %>% rownames_to_column()
        cm_rf = as.data.frame(cm_rf$table)
        cm_rf$model = "RF"

```
============================================ Comparision
```{r comparision-confusion-matrix}
        confMatrix =cm_dt %>%
        rbind( cm_rf,cm_boosting, cm_bag, cm_NB)
```
```{r comparision-performance}
        pf.dt = pf.dt %>%
        rename(Performance = `cm_dt$byClass[neededPerfomanceIndex]` ) %>%
        mutate(model = "DT" )
        pf.dt[6,1] = "Accuracy"
        #make FPR
        pf.dt[7,2] = 1- pf.dt[2,2]
        pf.dt[7,3] = "DT"
        pf.dt[7,1] = "FPR"


        pf.NB = pf.NB %>%
        rename(Performance = `cm_NB$byClass[neededPerfomanceIndex]` ) %>%
        mutate(model = "NB")
        pf.NB[6,1] = "Accuracy"
        pf.NB [7,2] = 1- pf.NB [2,2]
        pf.NB [7,3] = "NB"
        pf.NB [7,1] = "FPR"
~~~



~~~
        pf.bag = pf.bag %>%
        rename(Performance = `cm_bag$byClass[neededPerfomanceIndex]` ) %>%
        mutate(model = "Bag")
        pf.bag[6,1] = "Accuracy"
        pf.bag [7,2] = 1- pf.bag [2,2]
        pf.bag [7,3] = "Bag"
        pf.bag [7,1] = "FPR"

        pf.boosting = pf.boosting %>%
        rename(Performance = `cm_boosting$byClass[neededPerfomanceIndex]` ) %>%
        mutate(model = "Boosting")
        pf.boosting[6,1] = "Accuracy"
        pf.boosting[7,2] = 1- pf.boosting[2,2]
        pf.boosting[7,3] = "Boosting"
        pf.boosting[7,1] = "FPR"

        pf.rf = pf.rf %>%
        rename(Performance = `cm_rf$byClass[neededPerfomanceIndex]` ) %>%
        mutate(model = "RF")
        pf.rf[6,1] = "Accuracy"
        pf.rf [7,2] = 1- pf.rf [2,2]
        pf.rf [7,3] = "RF"
        pf.rf [7,1] = "FPR"


        pf.All = pf.dt %>% rbind(pf.NB,pf.bag,pf.boosting,pf.rf) %>%
        filter( Performance != is.na(Performance))

        pf.All = pf.All %>% pivot_wider(names_from =rowname, values_from =Performance )

        pf.All = pf.All[,1] %>% cbind(round(pf.All[,-1],digits=3)) %>% select(-Sensitivity)


```
```{r comparision-roc}

        dt <-  predict(train.tree, WAUS.test, type = "vector")
        nb <-  predict(NB.Model, WAUS.test, type = "raw")
        bag<-  predict(bag.train, WAUS.test, type = "vector")
        boosting <-  predict(boosting.train, WAUS.test, type = "vector")
        rf <-  predict(rf.train, WAUS.test, type = "prob")

        dt.roc <- roc(WAUS.test$CloudTomorrow,dt[,2])
        nb.roc <-roc(WAUS.test$CloudTomorrow,nb[,2])
        bag.roc <-roc(WAUS.test$CloudTomorrow,bag$prob[,2])
        boosting.roc <-roc(WAUS.test$CloudTomorrow,boosting$prob[,2])
        rf.roc <-roc(WAUS.test$CloudTomorrow,rf[,2])

        rocs <- list()
        rocs[["DT"]] <- dt.roc
        rocs[["NB"]] <- nb.roc
        rocs[["Bagging"]] <- bag.roc
        rocs[["Boosting"]] <- boosting.roc
        rocs[["RF"]] <- rf.roc

        color = c("red","#0000ff","#4cd7d0","green","black")

        ggroc(rocs)+theme_classic()+ggtitle("The ROC curves by classifiers")

```
```{r comparision-auc}
        auc.dt <- as.data.frame( auc(dt.roc)) %>%mutate(model = "DT") %>%
        rename(AUC = `auc(dt.roc)`)
        auc.nb <-  as.data.frame(auc(nb.roc))%>%mutate(model = "NB") %>%
~~~



```
        rename(AUC = `auc(nb.roc)`)
        auc.bag <-  as.data.frame(auc(bag.roc))%>%mutate(model = "Bag") %>%
        rename(AUC =`auc(bag.roc)` )
        auc.boosting <-  as.data.frame(auc(boosting.roc))%>%mutate(model = "Boosting") %>%
        rename(AUC =`auc(boosting.roc)` )
        auc.rf <- as.data.frame( auc(rf.roc))%>%mutate(model = "RF") %>%
        rename(AUC = `auc(rf.roc)`)

        table.all <-  auc.dt %>% rbind( auc.nb,auc.bag,auc.boosting,auc.rf) %>% select(-model)
        table.all <- pf.All %>% select(-Specificity) %>% cbind(table.all)
        table.all = table.all[,1] %>% cbind( round(table.all[,-1],digits=3) ) %>% rename(Model = ".")

        knitr::kable(table.all)


```
```{r feature-importance}
        library("viridis")           # Load
        importance <- as.data.frame( bag.train$importance)
        importance <-importance %>% cbind( as.data.frame( boosting.train$importance)) %>%
rownames_to_column()
        importance <-importance %>% full_join(as.data.frame( rf.train$importance) %>% rownames_to_column() )
        importance <- importance [,1]%>% cbind(  scale(importance[,-1]))
        importance <- as.data.frame(importance)  %>% rename(Variable=".",
        Bag ="bag.train$importance",
        Boosting ="boosting.train$importance",
        RF = "MeanDecreaseGini")

        importance <- importance %>% pivot_longer(cols= -Variable,names_to = "Model", values_to = "Values")
        importance[,3] <- as.numeric(unlist( importance[,3]))
        importance [,3]<-round(importance[,3],digits = 3)



        ggplot(importance, aes(x = Variable, y =Values, colour = Model, group = Model)) +
        geom_point()+
        geom_line()+
        theme_classic()+
        labs(y = "Gini index")+
        theme(axis.text.y =element_blank(),
        axis.ticks.y =  element_blank(),
        axis.text.x = element_text(angle = 45, vjust =1, hjust=1),
        legend.position=c(1,1),legend.justification = c(1,1))+
        geom_hline(yintercept = mean(importance$Values), linetype="dashed", color = "red")+
        geom_text(aes(15,0,label = "Threshold value = mean ", vjust = 1.2))

        #index for mportant variables
        #the following chunks might need this index to filter the features
        x =
c("MinTemp","Pressure9am","Pressure3pm","Sunshine","WindDir3pm","WindDir9am","WindGustDir","CloudTomorrow")
```
======================== improve the model : simplicity
```{r improved-models}
        df.improved.train= WAUS.train
        df.improved.test= WAUS.test
        set.seed(1234)
        rf.train= randomForest(CloudTomorrow ~. ,data = df.improved.train,
        ntree = 50000,
        importance= T,
        mtry = 2,
        nodesize=42)
        #model testing
        rf.predict = predict(rf.train, newdata = df.improved.test, type = "class")
        #computing confusion matrix
        cm_rf = confusionMatrix(table(observed =df.improved.test$CloudTomorrow    ,  predicted  =
```



```
rf.predict))
        pf.rf =as.data.frame( cm_rf$byClass[neededPerfomanceIndex])%>% rbind(cm_rf$overall[1])
        pf.rf = pf.rf %>% rownames_to_column()
        cm_rf = as.data.frame(cm_rf$table)
        cm_rf$model = "RF"

        rf <-  predict(rf.train, df.improved.test, type = "prob")
        grid.rf.roc <-roc(df.improved.test$CloudTomorrow,rf[,2])
        auc.rf <- as.data.frame( auc(grid.rf.roc))%>%mutate(model = "RF") %>%
        rename(AUC = `auc(grid.rf.roc)`)
        auc.rf
        #result of accracymust be greate than >0.736
        #ntree = 10000 AUC = 0.7447375
        # Impot = T #ntree = 10000 AUC =  0.7450459
        # Impot = T #ntree = 50000 AUC =  0.7460483


```
===================================== improve the model : Performance
```{r by-hand-model-train}
        #variable index which beyond the mean of importance
        x =
c("MinTemp","Pressure9am","Pressure3pm","Sunshine","WindSpeed3pm","WindSpeed9am","CloudTomorrow")
        #since WindDir3pm and WindDirGust are not independent, I will drop one of them
        byhand = WAUS.train[,x]
        set.seed(31084222) #Student ID as random seed
        sample_data = sample_n(byhand, size = 15, replace = T)
        #if Humidity3pm > 50, it means that day is humid
        sample_data$Pressure = ifelse(sample_data$Pressure3pm> 1013&
        sample_data$Pressure3pm>1013 ,
        "High","Not High")
        sample_data$Windy = ifelse(
        ((sample_data$WindSpeed9am > 15) &(sample_data$WindSpeed3pm >15))
        ,"Yes","No")
        #if MinTemp < 10 degree c, it means that day is cold
        sample_data$Cold = ifelse(sample_data$MinTemp< 10 ,"Cold","Not Cold")
        #if Sunshine hour > 12 , it means that day is sunny
        sample_data$Sunshine = ifelse(sample_data$Sunshine > 12 ,"Sunny","Not Sunny")
        #if WindSpeed9am > 15 , it means that day is Windy
        sample_data$WindSpeed9am = ifelse(
        ((sample_data$WindSpeed9am > 15) &&(sample_data$WindSpeed3pm >15))
        ,"Yes","No")
        sample_data$CloudTomorrow = ifelse(sample_data$CloudTomorrow == 1, "Cloudy", "Not Cloudy")
        sample_data <- sample_data%>% select( -WindSpeed3pm,-WindSpeed9am, -Pressure9am,-Pressure3pm, -
MinTemp)

        set.seed(31084222) #Student ID as random seed



        write.csv(sample_data, file = "C:/Users/sjsa3/Desktop/Shared_with_Mac/year2_sem1/FIT3152/Assignment
2/Q9/ID3/byhand.csv")

```
```{r by-hand-model-testing}
        #variable index which beyond the mean of importance
        x =
c("MinTemp","Pressure9am","Pressure3pm","Sunshine","WindSpeed3pm","WindSpeed9am","CloudTomorrow")
        #since WindDir3pm and WindDirGust are not independent, I will drop one of them
        byhand = WAUS.train[,x]
        set.seed(99999) #Student ID as random seed
        sample_data = sample_n(byhand, size = 5, replace = T)
        #if Humidity3pm > 50, it means that day is humid
        sample_data$Pressure = ifelse(sample_data$Pressure3pm> 1013&
        sample_data$Pressure3pm>1013 ,
        "High","Not High")
        sample_data$Windy = ifelse(
        ((sample_data$WindSpeed9am > 15) &&(sample_data$WindSpeed3pm >15))
        ,"Yes","No")
        #if MinTemp < 10 degree c, it means that day is cold
        sample_data$Cold = ifelse(sample_data$MinTemp< 10 ,"Cold","Not Cold")
        #if Sunshine hour > 12 , it means that day is sunny
        sample_data$Sunshine = ifelse(sample_data$Sunshine > 12 ,"Sunny","Not Sunny")
        #if WindSpeed9am > 15 , it means that day is Windy
        sample_data$WindSpeed9am = ifelse(
        ((sample_data$WindSpeed9am > 15) &&(sample_data$WindSpeed3pm >15))
        ,"Yes","No")
        sample_data$CloudTomorrow = ifelse(sample_data$CloudTomorrow == 1, "Cloudy", "Not Cloudy")
        sample_data <- sample_data%>% select( -WindSpeed3pm,-WindSpeed9am, -Pressure9am,-Pressure3pm, -
MinTemp)

        set.seed(31084222) #Student ID as random seed

        write.csv(sample_data, file = "C:/Users/sjsa3/Desktop/Shared_with_Mac/year2_sem1/FIT3152/Assignment
2/Q9/ID3/testing.csv")

```
====================================== ANN
```{r ANN-final }
        library(neuralnet)
        library(dummies)
        set.seed(999999)
        x =
c("MinTemp","Pressure9am","Pressure3pm","Sunshine","WindDir3pm","WindDir9am","WindGustDir","CloudTomorrow")

        nn.train <-select(WAUS.train[,x], !is.numeric)
        #make all factoed variable as dummy variables
        nn.train <- dummy.data.frame(nn.train)

        nn.test <-select(WAUS.test[,x], !is.numeric)

        #make all factoed variable as dummy variables
        nn.test <- dummy.data.frame(nn.test)

        nn <- neuralnet(    CloudTomorrow0 + CloudTomorrow1 ~ . ,
        data = nn.train[,],
        hidden=3,
        linear.output = F,
        threshold=0.01
        )
        plot(nn, rep="best")

        #model testing
        nn.pred = compute(nn,nn.test[,-c(53,52)])
        #================ predict
        #   round   the     predictions     to 0   or 1
        nn.predr = ifelse(nn.pred$net.result > 0.5, 1,0)
        #   make    data    frame   of A, B, C, classified 0   or 1
        cm = table(observed = WAUS.test$CloudTomorrow, predicted = nn.predr[,1])  #    remove rows
classified 0  - leave    only   classified 1
        cm = confusionMatrix(cm)

        cm

        ann <-  predict(nn, nn.test, type = "vector")
        ann.roc <- roc(WAUS.test$CloudTomorrow,ann[,2])

        rocs <- list()
        rocs[["DT"]] <- dt.roc
        rocs[["NB"]] <- nb.roc
```

~~~
        rocs[["Bagging"]] <- bag.roc
        rocs[["Boosting"]] <- boosting.roc
        rocs[["RF"]] <- rf.roc
        rocs[["ANN"]] <- ann.roc

        color = c("red","#0000ff","#4cd7d0","green","black")
        ggroc(rocs)+theme_classic()+ggtitle("The ROC curves by classifiers")
        auc(ann.roc)

```
~~~