FIT3152 Assignment2

Looi Teck San 32439180

Question 1

Question 1 analysis is done before omitting the NA value

Number of data I'm going to analyse = 2000

Number of days where it is more humid than previous day:

```
> # number of row in more humid than previous day
> nrow(yes_humid)
[1] 984
```

984

Number of days where it is **not** more humid than previous day:

```
> # number of row in less humid than previous day
> nrow(no_humid)
[1] 951
```

951

The proportion of days when it is more humid than the previous day compared to those when it is less humid:

```
> # propotion of days when it is more humid than the previous
> # day compared to those where it is less humid
> nrow(yes_humid) / nrow(no_humid)
[1] 1.0347
```

From the result above we can tell that there are more humid than the previous day compared to those where it is less humid.

Using summary(WAUS) we can see that variables like Evaporation, Sunshine, Cloud9am and Cloud3pm consists of around 1000 of NA, which is roughly half of the datasets. Therefore I had decided to use standard deviation to see whether it make sense for me to replace the missing value to the mean value.

```
> #summary
> summary(WAUS)
      Year         Location        MinTemp          MaxTemp          Rainfall        Evaporation        Sunshine       WindGustDir        WindGustSpeed
 Min.   :2008   Min.   : 8.00   Min.   :-6.60   Min.   : 8.90   Min.   :  0.000   Min.   : 0.000   Min.   : 0.000   Length:2000        Min.   : 7.00
 1st Qu.:2011   1st Qu.:18.00   1st Qu.: 9.50   1st Qu.:19.30   1st Qu.:  0.000   1st Qu.: 3.200   1st Qu.: 5.575   Class :character   1st Qu.:30.00
 Median :2014   Median :31.00   Median :13.70   Median :23.30   Median :  0.000   Median : 5.000   Median : 9.100   Mode  :character   Median :37.00
 Mean   :2014   Mean   :29.95   Mean   :13.46   Mean   :23.51   Mean   :  2.867   Mean   : 5.318   Mean   : 8.034                      Mean   :39.12
 3rd Qu.:2017   3rd Qu.:40.00   3rd Qu.:17.90   3rd Qu.:27.27   3rd Qu.:  1.000   3rd Qu.: 7.000   3rd Qu.:10.800                      3rd Qu.:46.00
 Max.   :2019   Max.   :48.00   Max.   :28.20   Max.   :44.30   Max.   :164.600   Max.   :20.600   Max.   :13.700                      Max.   :96.00
 NA's   :23                     NA's   :21      NA's   :30      NA's   :63        NA's   :1232     NA's   :1132                        NA's   :136
  WindDir9am          WindDir3pm         WindSpeed9am    WindSpeed3pm  Pressure9am      Pressure3pm        Cloud9am        Cloud3pm         Temp9am
 Length:2000        Length:2000        Min.   : 0.00   Min.   : 0   Min.   : 988.9   Min.   : 989.1   Min.   :0.000   Min.   :0.000   Min.   :-2.20
 Class :character   Class :character   1st Qu.: 6.00   1st Qu.:13   1st Qu.:1013.2   1st Qu.:1011.0   1st Qu.:2.000   1st Qu.:2.000   1st Qu.:14.20
 Mode  :character   Mode  :character   Median :13.00   Median :19   Median :1017.8   Median :1015.2   Median :6.000   Median :5.000   Median :18.40
                                       Mean   :13.27   Mean   :19   Mean   :1017.4   Mean   :1015.1   Mean   :4.828   Mean   :4.518   Mean   :18.24
                                       3rd Qu.:19.00   3rd Qu.:24   3rd Qu.:1021.9   3rd Qu.:1019.5   3rd Qu.:7.000   3rd Qu.:7.000   3rd Qu.:22.40
                                       Max.   :69.00   Max.   :69   Max.   :1038.2   Max.   :1036.4   Max.   :8.000   Max.   :8.000   Max.   :33.90
                                       NA's   :97      NA's   :92   NA's   :366      NA's   :360      NA's   :923     NA's   :928     NA's   :31
    Temp3pm       RainToday            RISK_MM             MHT
 Min.   : 6.00   Length:2000        Min.   :  0.000   Min.   :0.0000
 1st Qu.:17.80   Class :character   1st Qu.:  0.000   1st Qu.:0.0000
 Median :21.70   Mode  :character   Median :  0.000   Median :1.0000
 Mean   :21.88                      Mean   :  2.845   Mean   :0.5085
 3rd Qu.:25.50                      3rd Qu.:  0.800   3rd Qu.:1.0000
 Max.   :44.10                      Max.   :371.000   Max.   :1.0000
 NA's   :29                         NA's   :76        NA's   :65
```

Evaporation

```
> # standard deviation for Evaporation
> sd(WAUS$Evaporation, na.rm=TRUE)
[1] 2.766871
> # coefficient of variation for Evaporation variable
> #
> sd(WAUS$Evaporation, na.rm=TRUE) / mean(WAUS$Evaporation, na.rm = TRUE)
[1] 0.5202999
```

Sunshine

```
> # standard deviation for Sunshine
> sd(WAUS$Sunshine, na.rm=TRUE)
[1] 3.706001
> # coefficient of variation(CV) for Sunshine variable
> sd(WAUS$Sunshine, na.rm=TRUE) / mean(WAUS$Sunshine, na.rm = TRUE)
[1] 0.4613036
```

Cloud9am

```
> # standard deviation for Cloud9am
> sd(WAUS$Cloud9am, na.rm=TRUE)
[1] 2.773886
> # coefficient of variation(CV) for Cloud9am variable
> sd(WAUS$Cloud9am, na.rm=TRUE) / mean(WAUS$Cloud9am, na.rm = TRUE)
[1] 0.5745145
```

Cloud3pm

```
> # standard deviation for Cloud3pm
> sd(WAUS$Cloud3pm, na.rm=TRUE)
[1] 2.773367
> # coefficient of variation(CV) for Cloud3pm variable
> sd(WAUS$Cloud3pm, na.rm=TRUE) / mean(WAUS$Cloud3pm, na.rm = TRUE)
[1] 0.6138858
```

So by using the coefficient of variation (CV) we can see the results for Evaporation, Sunshine, Cloud9am and Cloud3pm are low and they are lower than 1, so the standard deviation is low and the standard deviation suggested the data are clustered around the mean. Hence later in question 2 I am going to replace the NA value with mean value since the standard deviation suggest the data are clustered around the mean

Also, based on summary(WAUS) in my opinion, variable like Rainfall, RISK_MM can be omitted as their 1st quantile is equal to 0, meaning to say they have more than 25% of value 0, because if a variable contains a large number of zeros, it may not provide much information for prediction or building a model. But there isn't much information provided whether these 2 variables are important, hence I won't omit them first in question 4, I will decide whether to omit them only after I built and analyse the models in question 4 and question 8.

Question 2

First, I will convert all the character(String) data to factor because character(string) data cannot be directly used as input variables in tree-based classifiers.

```
> #convert character(string) data to factor
> WAUS[, c(8, 10, 11, 20)] <- lapply(WAUS[, c(8, 10, 11, 20)], factor)
```

Also, I will be converting the target variable from numerical data to factor because classifier are designed to predict categorical or discrete classes and the given target variable are in numerical type (MHT), so by converting the target variable to factor we could explicitly define the distinct classes or categories that the classifier tree should predict. Another reason is that when building the random forest, it required the target variable to be a factor.

```
> # convert target variable into factor
> WAUS[,22]<-as.factor(WAUS[,22])
```

Result:

```
> str(WAUS)
'data.frame':   2000 obs. of  22 variables:
 $ Year         : int  2017 2015 2009 2011 2017 2014 2015 2018 2012 2014 ...
 $ Location     : int  30 30 48 40 31 28 40 44 48 30 ...
 $ MinTemp      : num  19.7 12.7 15.5 24 17.3 14.9 25.6 18.2 17.6 2.4 ...
 $ MaxTemp      : num  35 32.2 19.4 29.7 25.1 21.8 32.5 27.6 NA 15.7 ...
 $ Rainfall     : num  0 0 2.2 0 1 0 0 9.6 16 0.2 ...
 $ Evaporation  : num  NA NA NA 4 NA 7 10.8 NA NA NA ...
 $ Sunshine     : num  11.7 12.4 NA 2.9 NA 12.8 11.2 NA NA 6.6 ...
 $ WindGustDir  : Factor w/ 16 levels "E","ENE","ESE",..: 1 1 3 1 2 11 2 5 11 3 ...
 $ WindGustSpeed: int  41 46 69 43 28 39 39 33 26 28 ...
 $ WindDir9am   : Factor w/ 16 levels "E","ENE","ESE",..: 1 11 3 10 6 3 2 5 12 NA ...
 $ WindDir3pm   : Factor w/ 16 levels "E","ENE","ESE",..: 11 13 3 1 5 10 2 13 3 3 ...
 $ WindSpeed9am : int  17 17 44 22 6 17 22 9 13 0 ...
 $ WindSpeed3pm : int  17 20 39 19 7 20 30 11 19 17 ...
 $ Pressure9am  : num  1010 1020 1026 1012 NA ...
 $ Pressure3pm  : num  1008 1015 1024 1010 NA ...
 $ Cloud9am     : int  NA 0 8 6 NA 1 6 NA 5 2 ...
 $ Cloud3pm     : int  NA 3 5 7 NA 1 2 NA 4 7 ...
 $ Temp9am      : num  26.2 23.4 18.2 28.2 20.2 20.6 29.9 20.3 20.4 6.8 ...
 $ Temp3pm      : num  33.9 30.6 18.4 28.3 24.1 20.5 30.6 25.8 23 14.6 ...
 $ RainToday    : Factor w/ 2 levels "No","Yes": 1 1 2 1 1 1 1 2 2 1 ...
 $ RISK_MM      : num  0 0 0 0 1 0 0 0.2 4.4 0 ...
 $ MHT          : Factor w/ 2 levels "0","1": 2 1 2 2 1 1 2 NA 1 1 ...
```

Moreover, based on question 1, the standard deviation for Evaporation, Sunshine, Cloud9am and Cloud3pm are low, hence it indicates that the data are clustered around the mean and also these 4 variables consists of around 50% missing value, so I will replace all the NA value in these 4 variables to their mean.

```
> #Evaporation
> humid.data$Evaporation[is.na(humid.data$Evaporation)] <- mean(humid.data$Evaporation, na.rm=TRUE)
> #Sunshine
> humid.data$Sunshine[is.na(humid.data$Sunshine)] <- mean(humid.data$Sunshine, na.rm=TRUE)
> #Cloud9am
> humid.data$Cloud9am[is.na(humid.data$Cloud9am)] <- mean(humid.data$Cloud9am, na.rm=TRUE)
> #Cloud3pm
> humid.data$Cloud3pm[is.na(humid.data$Cloud3pm)] <- mean(humid.data$Cloud3pm, na.rm=TRUE)
```

Lastly, the pre-process step I would consider is removing the rows consists of NA value as some of the classifer like boosting will not work with NA value. Also, by removing NA value the classifiers are able to minimize bias and incorrect prediction, overall, by removing the NA value, it helps to improve the overall performance for the classifier.

```
> #omit NA value
> humid.data <- WAUS[complete.cases(WAUS),]
```

Question 3

```
#question 3
set.seed(32439180) #Student ID as random seed
train.row = sample(1:nrow(humid.data), 0.7*nrow(humid.data))
humid.train <- humid.data[train.row,]
humid.test <- humid.data[-train.row,]
str(humid.data)
```

Result:

humid.test	149 obs. of 22 variables
humid.train	346 obs. of 22 variables

Question 4

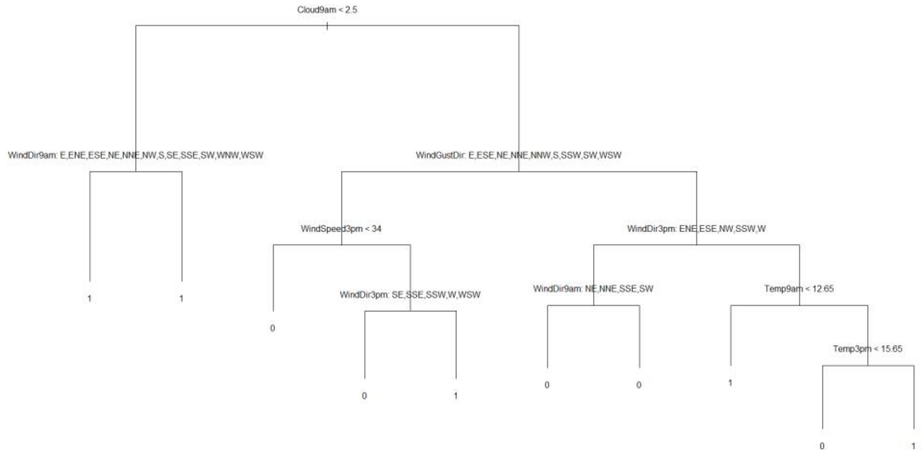
Decision Tree

```
# decision tree
set.seed(32439180)
humid.tree = tree(MHT~., data = humid.train)

summary(humid.tree)

par(mar = c(8, 5, 2, 3))
plot(humid.tree)
text(humid.tree, pretty=0)
```

Decision Tree visualization:



As I did not omit any of the variables and I did not perform cross validation and pruning, so we can see that I'm getting a rather complex, imbalanced tree and overfitted tree.

Naïve Bayes

```
# Naive Bayes
set.seed(32439180)
humid.bayes = naiveBayes(MHT~., data=humid.train)
```

Bagging

```
# Bagging
set.seed(32439180)
humid.bag <- bagging(MHT~., data = humid.train)
```

Boosting

```
# Boosting
set.seed(32439180)
humid.boost <- boosting(MHT~., data = humid.train)
```

I did not specify mfinal for both bagging and boosting is because according to the question we need to use their default settings.

Random Forest

```
# Random forest
set.seed(32439180)
humid.rf = randomForest(MHT~., data = humid.train, na.action = na.exclude)
```

Question 5

Decision Tree

```
> print(humid.tree.tab)
               Actual_Class
Predicted_Class   0   1
              0 115 106
              1  65  89
> # accuracy
> # replace NA value
> (115+89)/(115+106+65+89)
[1] 0.544
> #precision
> (89)/(89+65)
[1] 0.5779221
> #recall
> (89)/(89+106)
[1] 0.4564103
```

as we can see **the accuracy is 0.544**, it is slightly better than random guessing and the accuracy score is very poor. Also, according to the confusion matrix, the classifier is better at predicting it will be less humid tomorrow than today than it will be more humid tomorrow than today.

Naïve Bayes

```
> print(humid.bayes.tab)
               Actual_Class
Predicted_Class   0   1
              0  51  40
              1 129 155
> # replace NA value
> (51+155)/(51+40+129+155)
[1] 0.5493333
> #precision
> (155)/(155+129)
[1] 0.5457746
> #recall
> (155)/(155+40)
[1] 0.7948718
```

For Naïve Bayes, **The accuracy is 0.549**, which is roughly 55%. The accuracy is still poor as it is slightly better than random guessing, so it is not good enough. Also, according to the confusion matrix, the classifier is better at predicting it will be more humid tomorrow than today than it will be less humid tomorrow than today.

Bagging

```
> print(humidpred.bag$confusion)
               Observed Class
Predicted Class   0   1
              0  89  73
              1  91 122
> # replace NA value
> (89+122) / (89+73+91+122)
[1] 0.5626667
> #precision
> (122)/(122+91)
[1] 0.57277
> #recall
> (122)/(122+73)
[1] 0.625641
```

For Bagging, **the accuracy is 0.56**. It is slightly better than random guessing, but the accuracy score is still poor. Also, according to the confusion matrix, the classifier is better at predicting it will be more humid tomorrow than today than it will be less humid tomorrow than today.

Boosting

```
> print(humidpred.boost$confusion)
               Observed Class
Predicted Class   0   1
              0 103  82
              1  77 113
> # replaced NA value
> (103+113)/(103+82+77+113)
[1] 0.576
> #precision
> (113)/(113+77)
[1] 0.5947368
> #recall
> (113)/(113+82)
[1] 0.5794872
```

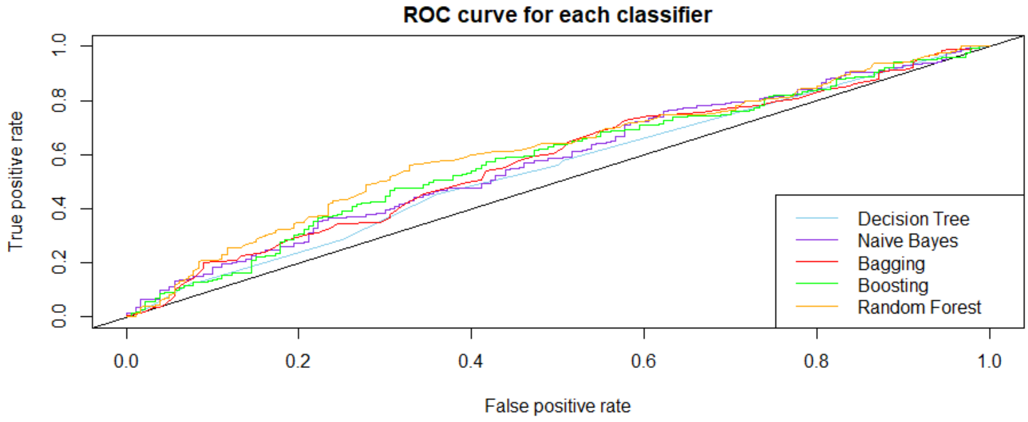
For Boosting **the accuracy is 0.576.** It is slightly better than random guessing, but the accuracy score is still poor. Also, according to the confusion matrix, the classifier is slightly better (10 cases more) at predicting it will be more humid tomorrow than today than it will be less humid tomorrow than today.

Random Forest

```
> print(humid.rf.tab)
               Actual_Class
Predicted_Class   0   1
              0  69  49
              1 111 146
> # replaced NA value
> (69+146)/(69+49+111+146)
[1] 0.5733333
> #precision
> (146)/(146+111)
[1] 0.5680934
> #recall
> (146)/(146+49)
[1] 0.7487179
```

As for Random forest, **the accuracy is 0.573**. It is slightly better than random guessing, but the accuracy score is still poor. Also, according to the confusion matrix, the classifier is better at predicting it will be more humid tomorrow than today than it will be less humid tomorrow than today.

Question 6



AUC for each classifier:

Decision Tree

Calculate the confidence

```
#Decision Tree
tree.pre.humid = predict(humid.tree, humid.test, type="vector")
tree.pred.humid <- prediction(tree.pre.humid[,2], humid.test$MHT)
tree.pref.humid <- performance(tree.pred.humid,"tpr","fpr")
```

AUC

```
> print(as.numeric(tree.auc @y.values))
[1] 0.5487464
```

AUC for Decision Tree: 0.5487

Naïve Bayes

Calculate the confidence

```
# Naïve Bayes
bayes.pre.humid = predict(humid.bayes, humid.test, type = 'raw')
bayes.pred.humid <- prediction( bayes.pre.humid[,2], humid.test$MHT)
bayes.pref.humid <- performance(bayes.pred.humid,"tpr","fpr")
```

AUC

```
> print(as.numeric(bayes.auc @y.values))
[1] 0.5757265
```

AUC for Naïve Bayes: 0.5757

Bagging
Calculate the confidence

```
# Bagging
bag.pre.humid = predict.bagging(humid.bag, humid.test)
bag.pred.humid <- prediction(bag.pre.humid$prob[,2], humid.test$MHT)
bag.pref.humid <- performance(bag.pred.humid,"tpr","fpr")
```

AUC

```
> print(as.numeric(bag.auc @y.values))
[1] 0.5751425
```

AUC for Bagging: 0.5751

Boosting
Calculate the confidence

```
# Boosting
boost.pre.humid = predict.boosting(humid.boost, humid.test)
boost.pred.humid <- prediction(boost.pre.humid$prob[,2], humid.test$MHT)
boost.pref.humid <- performance(boost.pred.humid,"tpr","fpr")
```

AUC

```
> print(as.numeric(boost.auc @y.values))
[1] 0.5815385
```

AUC for Boosting: 0.5815

Random Forest
Calculate the confidence

```
# Random forest
rf.pre.humid = predict(humid.rf, humid.test, type="prob")
rf.pred.humid <- prediction(rf.pre.humid[,2], humid.test$MHT)
rf.pref.humid <- performance(rf.pred.humid,"tpr","fpr")
```

AUC

```
> print(as.numeric(rf.auc @y.values))
[1] 0.6110114
```

AUC for Random Forest: 0.6110

Question 7

Classifier	**Accuracy**	**Area Under Curve (AUC)**	**Precision**	**Recall**
Decision Tree	0.544	0.5487	0.5779	0.4564
Naïve Bayes	0.549	0.5757	0.5458	0.7949
Bagging	0.563	0.5751	0.5728	0.6256
Boosting	0.576	0.5815	0.5947	0.5795
Random Forest	0.573	0.6110	0.5681	0.7487

We can clearly see from the table there is a single best classifier which is Random Forest as it had the highest Area Under Curve. Even though Boosting's accuracy slightly outperform Random Forest accuracy but since AUC is considered more informative than accuracy and also AUC is a more accurate for model's performance, therefore the best classifier would be Random Forest. Also, the recall is better than precision so it is suggested that this Random Forest classifier is better at predicting actual positive instances(true positives) compared to accurately predicting only positive instances among all predicted positives and it has a lower chances of producing false negative errors.

Question 8

As by looking at the score only would be hard to determine whether a variable is important, hence I decided to make it to the ratio with the most important variable.

Decision Tree

```
> # Decision Tree
> summary(humid.tree)

Classification tree:
tree(formula = MHT ~ ., data = humid.train)
Variables actually used in tree construction:
[1] "Cloud9am"     "WindDir9am"    "WindGustDir"   "WindSpeed3pm" "WindDir3pm"    "Temp9am"
[7] "Temp3pm"
Number of terminal nodes:  10
Residual mean deviance:  1.222 = 1055 / 863
Misclassification error rate: 0.3528 = 308 / 873
```

For Decision Tree the most important variable would be Cloud9am. Since Decision tree will automatically select those important variables to build the decision tree, hence there are still some other important variables like WindDir9am, WindGustDir, WindSpeed3pm, WindDir3pm, Temp9am and Temp3pm.

Besides variables that are not used in this tree can be omitted, for example variables like Pressure9am, Pressure3pm, MinTemp, Location, Evaporation, Sunshine, Rainfall, Cloud3pm, MaxTemp, WindSpeed9am, Year, Risk_MM, WindGustSpeed and RainToday.

Bagging

```
> # Bagging
> sor.impo.bag <- sort((humid.bag$importance), decreasing = TRUE)
> print(sor.impo.bag)
   WindDir9am    WindDir3pm   WindGustDir      Cloud9am   Pressure9am   Pressure3pm  WindSpeed3pm
    22.039974     17.341908     16.345193      8.510735      3.876821      3.139414      3.106075
      MinTemp       Temp9am      Location   Evaporation      Sunshine      Rainfall      Cloud3pm
     2.683660      2.373281      2.086508      2.036501      2.029902      1.960071      1.892842
      Temp3pm       MaxTemp  WindSpeed9am          Year       RISK_MM WindGustSpeed     RainToday
     1.885080      1.840198      1.833860      1.768470      1.700750      1.548757      0.000000
> # ratio with the most important variable
> (sor.impo.bag /max(sor.impo.bag ))
   WindDir9am    WindDir3pm   WindGustDir      Cloud9am   Pressure9am   Pressure3pm  WindSpeed3pm
   1.00000000    0.78683885    0.74161580    0.38614996    0.17589953    0.14244184    0.14092918
      MinTemp       Temp9am      Location   Evaporation      Sunshine      Rainfall      Cloud3pm
   0.12176332    0.10768074    0.09466926    0.09240035    0.09210093    0.08893256    0.08588224
      Temp3pm       MaxTemp  WindSpeed9am          Year       RISK_MM WindGustSpeed     RainToday
   0.08553005    0.08349364    0.08320609    0.08023919    0.07716660    0.07027034    0.00000000
```

For Bagging, the most important variable will be WindDir9am as it has the highest score among all the other variables. Besides, there are still a few variables with a quite good scores such as WindDir3pm and WindGustDir, this indicates that they are also some of the important variables.

Since variable RainToday has a 0 score, means it does not contribute to the model at all hence this variable can be omitted from the data with very little effect on performance. Also, variables with a ratio with the most important variable lesser than 0.1 for example Location, Evaporation, Sunshine, Rainfall, Cloud3pm, Temp3pm, MaxTemp, WindSpeed9am, Year, RISK_MM and WindGustSpeed might also indicate if I omit these variables it may cause very little effect on performance, which might lower the performance a little because according to the result, it still contribute but just not as much, but in return we get a simpler classifier.

Boosting

```
> # Boosting
> sor.impo.boost <-sort((humid.boost$importance), decreasing =TRUE)
> print(sor.impo.boost)
  WindDir9am   WindDir3pm  WindGustDir  Pressure9am      MinTemp      Temp9am  Pressure3pm
   15.616534    14.798036    13.931430     5.227870     5.066395     4.699523     4.621898
     Temp3pm WindSpeed3pm WindGustSpeed WindSpeed9am     MaxTemp     Sunshine     Cloud9am
    3.899317     3.805499     3.516697     3.511858     2.983020     2.837438     2.755291
 Evaporation      RISK_MM         Year     Rainfall     Cloud3pm     Location    RainToday
    2.529417     2.338475     2.336155     1.988168     1.920673     1.616302     0.000000
> # ratio with the most important variable
> (sor.impo.boost/max(sor.impo.boost))
  WindDir9am   WindDir3pm  WindGustDir  Pressure9am      MinTemp      Temp9am  Pressure3pm
   1.0000000    0.9475877    0.8920949    0.3347651    0.3244250    0.3009325    0.2959618
     Temp3pm WindSpeed3pm WindGustSpeed WindSpeed9am     MaxTemp     Sunshine     Cloud9am
   0.2496916    0.2436840    0.2251906    0.2248808    0.1910168    0.1816945    0.1764342
 Evaporation      RISK_MM         Year     Rainfall     Cloud3pm     Location    RainToday
   0.1619704    0.1497435    0.1495950    0.1273118    0.1229897    0.1034994    0.0000000
```

For Boosting classifier, the most important variable will be WindDir9am as it has the highest score among all the other variables. Besides, variables like WindDir3pm and WindGustDir have a quite good score as well, this indicates that they are also some of the important variables for Boosting classifier.

Furthermore, from the result, there is one variable with a score of 0, RainToday, hence this variable could be omitted from the data with very little effect on performance.

Random Forest

```
> # Random Forest
> sor.impo.rf <- humid.rf$importance[order(humid.rf$importance, decreasing =TRUE),]
> print(sor.impo.rf)
 WindDir9am  WindGustDir   WindDir3pm  Pressure9am  Pressure3pm      MinTemp      Temp9am      Temp3pm
  50.479370    43.518555    43.383201    22.962529    22.722007    22.677546    22.323464    20.872803
    MaxTemp WindSpeed3pm     Cloud9am WindGustSpeed WindSpeed9am    Sunshine         Year  Evaporation
  20.235953    19.163631    18.473276    18.130732    16.563603    15.558974    15.050797    13.556142
   Cloud3pm     Rainfall     Location      RISK_MM    RainToday
  13.194591    12.052251    11.028188    11.006137     1.899196
> #since it is hard to determine whether it is the most important variable or not hence:
> print(sor.impo.rf/max(sor.impo.rf))
 WindDir9am  WindGustDir   WindDir3pm  Pressure9am  Pressure3pm      MinTemp      Temp9am      Temp3pm
  1.0000000    0.8621057    0.8594244    0.4548894    0.4501246    0.4492438    0.4422295    0.4134917
    MaxTemp WindSpeed3pm     Cloud9am WindGustSpeed WindSpeed9am    Sunshine         Year  Evaporation
  0.4008757    0.3796329    0.3659569    0.3591711    0.3281262    0.3082244    0.2981574    0.2685482
   Cloud3pm     Rainfall     Location      RISK_MM    RainToday
  0.2613858    0.2387560    0.2184692    0.2180324    0.0376232
```

For Random Forest classifier, the most important variable will be WindDir9am as it has the highest score among all the other variables. Besides, variables like WindDir3pm and WindGustDir have a quite good score as well, this indicates that they are also some of the important variables for Random Forest classifier.

Moreover, RainToday is the variable with the lowest score, this indicates that it might not be so important for the classifier and will have little effect on performance if it is omitted from the data and so we can omit this variable from the classifier.

Conclusion

As for Naïve Bayes, since we can’t visualize the predictors used hence it won’t be included in this analysis.

We can conclude that the most important variable across all 3 different classifiers will be WindDir9am as it is the most important variable for all 3 different classifiers except Decision Tree where it is Cloud9am. Also, the least important variable will be RainToday as it is the least important variable for all 4 different classifiers, therefore it can be omitted from the data and will have very little effect on the performance.

Question 9

Important factors

The factors in my decision on which classifier to use included interpretability and handling discrete and continuous variables. Decision trees are highly transparent, where we can easily see what variables is used to build the tree, number of terminal nodes, most importantly we are able to plot the decision tree out because the tree consists of nodes and branches that can be easily understood and some simple trees are simple enough for a person to classify by hand.

Attributes used

So for my approach, since I got the most important variables for decision tree in question 8, hence I used all those important variables and built a decision tree.
Firstly, I pre-processed the data again as I'm only selecting a subset of variables for this question hence by pre-processing the data again, I'm able to get a larger datasets as compared to the datasets I pre-processed in question 2. Lastly, I took the same pre-process steps in question 2.

Cross validation

Original tree tends to overfit, hence I decided to use cross validation to determine the best tree size

```
> cv.humid.tree.9
$size
[1] 5 4 3 1

$dev
[1] 354 354 360 537
```

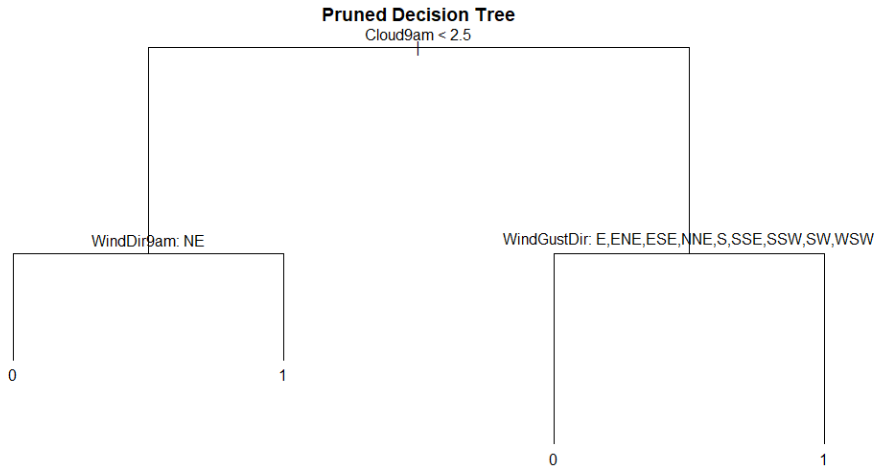
According to the result of cross validation, the best size with smallest misclassification rate would be size 4

Pruning

```
#prune using size 4 considering lowest misclassification rate
pruned.cv.tree.9 = prune.misclass(humid.s.tree, best = 4)
```

Result

As we can see in the graph, it is simple enough simple enough for a person to be able to classify whether it will be more humid tomorrow or not by hand.



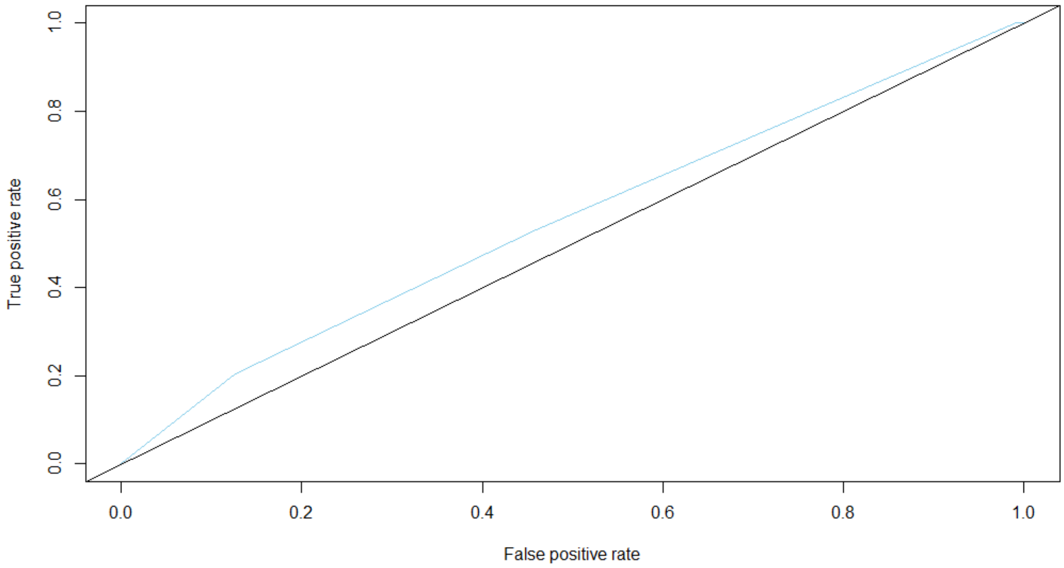
Performance

Accuracy

```
> table(actual = humid.test.9$MHT, predicted = sim.tree.pred)
      predicted
actual   0   1
     0 189 161
     1 184 208
> #Accuracy
> (189+208)/(189+161+184+208)
[1] 0.5350404
```

AUC

```
> #AUC
> s.tree.auc = performance(s.tree.pred.humid, "auc")
> print(as.numeric(s.tree.auc @y.values))
[1] 0.5509038
```



So for the performance, this simpler decision tree's accuracy is slightly worse than the decision tree built with every single variables in the datasets in question 4, but the difference is not significance and the AUC is better.

Also, I am able to get a simpler decision tree which is simple enough for a person to be able to classify by hand. Furthermore, this also proved that in question 8 I decided to discard all the variables were not used in that tree will caused very little effect on the performance.

Question 10

Attributes used

Based on my analysis on question 8, to be fair for all models I am going to use the same datasets among all classifiers and I will only remove the variable RainToday because it is the least important variables among all classifiers. I won't remove the other variables because they each have a different ratio to the most important variable respectively, for example, MaxTemp in Bagging classifier only have a ratio of 0.08 to the most important variable in Bagging classifier. But the same variable in Boosting and Random Forest, it has a ratio of 0.19 and 0.40 to the most important variable respectively. Therefore, to be fair in selecting the best classifier, im only going to remove only RainToday variable and use the same dataset across all models.

```
# preprocess the data
# only select columns i needed for this question
str(humid.data.10)
humid.data.10 <- subset(no.humid.data, select = c(1:19,21:22))
```

I had omitted variable RainToday using the code above.

How I created my improved model

First, I build each classifiers based on question 8 suggestions and I reselect a new group of data. After that I perform cross validation for Decision Tree, Naïve Bayes, Bagging, Boosting and Random Forest. And lastly, I calculate the accuracy and AUC to select the best classifier.

Train Control

Learned from https://topepo.github.io/caret/model-training-and-tuning.html

```
# train the model
# cv https://topepo.github.io/caret/model-training-and-tuning.html
set.seed(32439180)
tr.control <- trainControl(method = "cv", number=10)
```

Decision Tree

```
# Decion Tree
set.seed(32439180)
tree.cv.10 = tree(MHT~., data= humid.train.10)
```

First, I built a Decision Tree and did not perform cross validation and pruning. Since original tree usually over fitted hence I will perform cross validation and pruning.

Cross validation

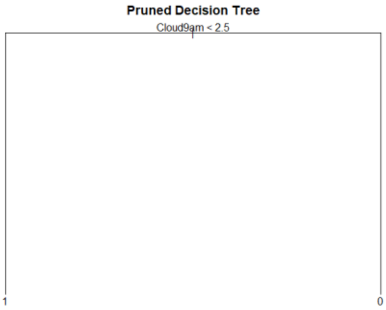
```
> cv.humid.tree
$size
[1] 4 2 1

$dev
[1] 482 482 607
```

Based on the result of cross validation, prune using size 2 considering lowest misclassification rate.

Pruning

```
#prune using size 2 considering lowest misclassification rate
pruned.cv.tree.10 = prune.misclass(tree.cv.10, best = 2)
```



Performance

Accuracy

```
> table( predicted = pruned.PD.predict ,actual = humid.test.10$MHT)
         actual
predicted   0   1
        0 162 143
        1  28  45
>  #Accuracy
> (162+45)/(162+28+143+45)
[1] 0.547619
> #precision
> (45)/(45+28)
[1] 0.6164384
> #recall
> (45)/(45+143)
[1] 0.2393617
```

AUC

```
> #AUC
> cv.tree.auc = performance(cv.pred.tree.10, "auc")
> print(as.numeric(cv.tree.auc @y.values))
[1] 0.5459966
```

The accuracy for this decision tree is 0.5476 and **AUC is 0.5460**, the performance is still poor as it is just slightly better than random guessing. After omitted RainToday and performed cross validation, the accuracy performed slightly better and AUC performed slightly worse as compared to question 4. Also, the precision is higher than recall so it is suggested that this Decision Tree classifier is better at predicting actual positive prediction(true positive) accurately compared to predicting all actual positive prediction (true positives + false negatives) and it has a lower chance of producing false positive errors.

Naïve Bayes

```
# naive bayes
set.seed(32439180)
bayes.cv.10 <- train(MHT~., data=humid.train.10, method='naive_bayes',trControl = tr.control)
```

As for naïve bayes, I performed cross validation as well using tr.control. After that I evaluate the performance by accuracy and AUC.

Performance

Accuracy

```
> #Accuracy
> print(bayes.cv.tab )
               Actual_Class
Predicted_Class   0   1
              0  59  34
              1 131 154
> (59+154)/(59+34+131+154)
[1] 0.5634921
> #precision
> (154)/(154+131)
[1] 0.5403509
> #recall
> (154)/(154+34)
[1] 0.8191489
```

AUC

```
> # AUC
> bayes.cv.auc = performance(bayes.cv.pred, "auc")
> print(as.numeric(bayes.cv.auc @y.values))
[1] 0.5952688
```

The accuracy for this Naïve Bayes is 0.5635 and **AUC is 0.5953**, the performance is still poor as it is just slightly better than random guessing. After omitted RainToday and performed cross validation, the accuracy and AUC showed an improvement of approximately 0.02 as compared to question 4. Also, the recall is better than precision so it is suggested that this Naïve Bayes classifier is better at predicting actual positive instances(true positives) compared to accurately predicting only positive instances among all predicted positives and it has a lower chances of producing false negative errors.

Bagging

```
#bagging with cv
set.seed(32439180)
bag.cv.10 <- train(MHT~., data=humid.train.10, method='AdaBag', trControl = tr.control)
```

As for Bagging, I performed cross validation as well using tr.control. After that I evaluate the performance by accuracy and AUC.

Performance

Accuracy

```
> print(bag.cv.tab )
               Actual_Class
Predicted_Class   0   1
              0  89  52
              1 101 136
> #Accuracy
> (89+136)/(89+52+101+136)
[1] 0.5952381
> #precision
> (136)/(136+101)
[1] 0.5738397
> #recall
> (136)/(136+52)
[1] 0.7234043
```

AUC

```
> # AUC
> bag.cv.auc = performance(bag.cv.pred, "auc")
> print(as.numeric(bag.cv.auc @y.values))
[1] 0.6163634
```

The accuracy for this Bagging classifier is 0.5952 and **AUC is 0.6164**, the performance is acceptable but it is still not good enough as it's only 10% better than random guessing. After omitted RainToday and performed cross validation, the accuracy and AUC showed an improvement of approximately 0.03 and 0.04 respectively as compared to question 4. Also, the recall is better than precision so it is suggested that this Bagging classifier is better at predicting actual positive instances(true positives) compared to accurately predicting only positive instances among all predicted positives and it has a lower chances of producing false negative errors.

Boosting

Performance
Accuracy

```
> print(boost.cv.tab)
               Actual_Class
Predicted_Class   0   1
              0  92  61
              1  98 127
> #Accuracy
> (92+127)/(92+61+98+127)
[1] 0.5793651
> #precision
> (127)/(127+98)
[1] 0.5644444
> #recall
> (127)/(127+61)
[1] 0.6755319
```

AUC

```
> # AUC
> boost.cv.auc = performance(boost.cv.pred, "auc")
> print(as.numeric(boost.cv.auc @y.values))
[1] 0.6093645
```

The accuracy for this Boosting classifier is 0.5794 and **AUC is 0.6094**, the performance is acceptable but it is still not good enough as in AUC it's only about 10% better than random guessing. After omitting RainToday and performed cross-validation, the accuracy and AUC showed an improvement of approximately 0.003 and 0.03 respectively as compared to question 4. Also, the recall is better than precision so it is suggested that this Boosting classifier is better at predicting actual positive instances(true positives) compared to accurately predicting only positive instances among all predicted positives and it has a lower chances of producing false negative errors.

Random Forest

Performance
Accuracy

```
> print(rf.cv.tab)
               Actual_Class
Predicted_Class   0   1
              0 103  67
              1  87 121
> #Accuracy
> (103+121)/(103+67+87+121)
[1] 0.5925926
> #precision
> (121)/(121+87)
[1] 0.5817308
> #recall
> (121)/(121+67)
[1] 0.643617
```

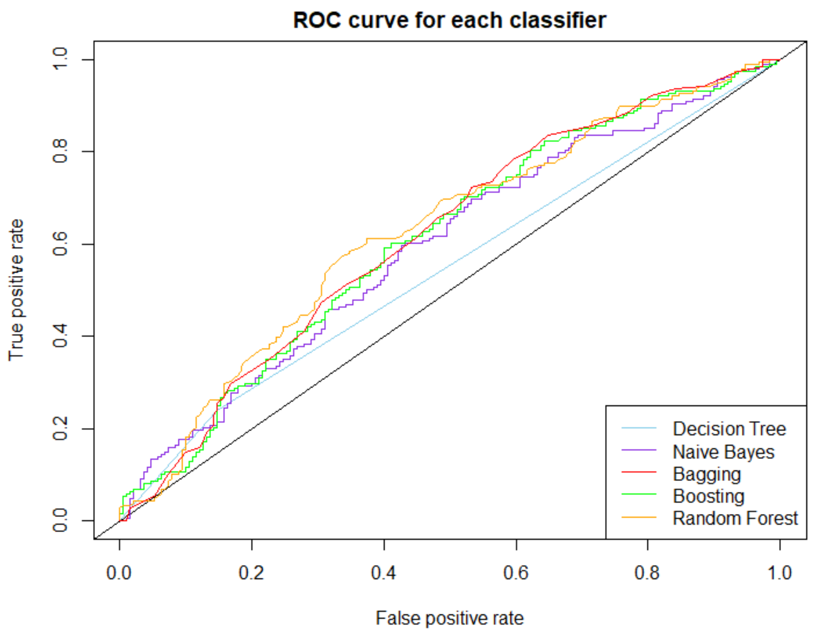
AUC

```
> # AUC
> rf.cv.auc = performance(rf.cv.pred, "auc")
> print(as.numeric(rf.cv.auc @y.values))
[1] 0.625126
```

The accuracy for this Random Forest classifier is 0.5926 and AUC is 0.6251, the performance is acceptable but it is still not good enough as it's only 10% better than random guessing for both accuracy and AUC. After omitted RainToday and performed cross validation, the accuracy and AUC showed an improvement of approximately 0.02 and 0.01 respectively as compared to question 4. Also, the recall is better than precision so it is suggested that this Random Forest classifier is better at predicting actual positive instances(true positives) compared to accurately predicting only positive instances among all predicted positives and it has a lower chances of producing false negative errors.

Performance for all classifiers

Classifier	Accuracy	AUC	Precision	Recall
Decision Tree	0.5476	0.5460	0.6164	0.2394
Naïve Bayes	0.5635	0.5953	0.5404	0.8191
Bagging	0.5820	0.6164	0.5738	0.7234
Boosting	0.5794	0.6094	0.5644	0.6755
Random Forest	0.5926	0.6251	0.5817	0.6436



Conclusion

According to question 7, we can clearly see that there's an improvement in terms of the accuracy and AUC result for all classifiers after omitting not important variable and performing cross validation.

The best tree-based classifier will be Random Forest. I am choosing this model and the factors affecting my decisions would be the model's AUC first then the model's accuracy because AUC is considered more informative than accuracy and also AUC is a more accurate for model's performance. Based on the table, Random Forest is the classifier with the highest accuracy and AUC hence it is the best tree-based classifier.

Question 11

Attributes used

I only included numeric and integer variables and excluded all character variables for building the ANN model because ANN are designed to handle either categorical or continuous data separately.

Data pre-process

First of all, I will only select columns with numeric or integer types of data. After that I will divide the data into a 70% training set and 30% testing set. Also, we can see from the data, variables like Year, Pressure9am and Pressure3pm have much larger range from other variables, hence I will perform normalization for the data before fitting into the ANN model.

```
#scale the data
humid.train.11[1:17] <- scale(humid.train.11[1:17])

humid.test.11[1:17] <- scale(humid.test.11[1:17])
```

Difference with other model

For other models in the assignment, they work for mixture of categorical and continuous data, but for ANN model, it only works well with either categorical or continuous data. Also, ANN model is a deep learning model while the other model is machine learning model. ANN model can have multiple hidden layer

ANN Classifier

```
set.seed(32439180)
humid.nn = neuralnet(MHT ~., humid.train.11, hidden=3,linear.output = FALSE)
```

For neuralnet, I had set the parameter linear.output = FALSE is because for this question I'm doing a classification problem.

Performance

Accuracy

```
> # plot confusion matrix
> table(predicted = humid.nn.pred$V1, actual = humid.test.11$MHT)
         actual
predicted   0   1
        0  71  47
        1 138 148
> # accuracy
> (71+148)/(71+138+47+148)
[1] 0.5420792
> #precision
> (148)/(148+138)
[1] 0.5174825
> #recall
> (148)/(148+47)
[1] 0.7589744
```

AUC

```
> # AUC
> ann.auc = performance(ann.pred, "auc")
> print(as.numeric(ann.auc @y.values))
[1] 0.5920869
```

So the accuracy for this ANN classifier would be 0.5421 and the AUC would be 0.5921. The performance for this model is poor because the accuracy is poor as it is just slightly better than random guessing and also AUC slightly better but it is still lower than 0.6 so it is not performing well also. As compared to the best classifiers in question 10, it is only performing better than the Decision Tree based on AUC and it is performing worse than the rest. Also, the recall is better than precision so it is suggested that this ANN classifier is better at predicting actual positive instances(true positives) compared to accurately predicting only positive instances among all predicted positives and it has a lower chances of producing false negative errors.

<u>Question 12</u>
learned from: FIT2086 week 9 studio 9
Package details: [kknn: Weighted k-Nearest Neighbors (r-project.org)]

I will be using KNN classifier and kknn package for this question

Description
For this question, I'm using K-Nearest Neighbours (KNN) model. KNN is a supervised classifier. It works by first decide the number of neighbours (k) after that it will decide each particular data point and calculate the distance to look for the nearest neighbour (k) to that data point and the data point would be classified to the nearest neighbour (k). The benefits of using this method is that continuous and categorical variables can be easily handle and it is efficient on learning in high dimensions. Also, disadvantages for using is would be there is no interpretability unlike decision tree, sometimes it is hard to determine how many neighbours(k) and also when selecting which variables to use can be challenging.

Variables used
In this question, I decided not to use RainToday is because based on question 8, it is not a very important variables. Besides that, I will include each and every variables.

Performance

Accuracy

```
> # the confusion matrix
> table(predicted=pred.nn, actual = humid.test.12$MHT)
         actual
predicted   0   1
        0  88  59
        1 102 129
> #Accuracy
> (88+129)/(88+59+102+129)
[1] 0.5740741
> #precision
> (129)/(129+102)
[1] 0.5584416
> #recall
> (129)/(129+59)
[1] 0.6861702
```

AUC

```
> # AUC
> knn.auc = performance(knn.auc.pred, "auc")
> print(as.numeric(knn.auc @y.values))
[1] 0.5925252
```

The accuracy for this KNN classifier would be 0.5741 and the AUC would be 0.5925. The accuracy is performing better than the Random Forest classifier(question 4) before doing the cross validation but it is not performing better than the best classifier in question 10, the Random Forest classifier after cross validation. But still, the performance for my model is poor because both the accuracy and AUC is poor as it is just slightly better than random guessing. Also, the recall is better than precision so it is suggested that this KNN classifier is better at predicting actual positive instances(true positives) compared to accurately predicting only positive instances among all predicted positives and it has a lower chances of producing false negative errors.

Appendix

```
#fit3152 assignment 2
setwd("D:/users/user2/dekstop/monash/THIRDYEAR/fit3152/ass2")
library(tree)
#install.packages("e1071")
library(e1071)
#install.packages(("ROCR"))
library(ROCR)
#install.packages("randomForest")
library(randomForest)
#install.packages("adabag")
library(adabag)
#install.packages("rpart")
library(rpart)

library(gridExtra)

library(caret)


#install.packages("neuralnet")


#creating individual data
rm(list = ls())
WAUS <- read.csv("HumidPredict2023D.csv")
L <- as.data.frame(c(1:49))
set.seed(32439180) # Your Student ID is the random seed
L <- L[sample(nrow(L), 10, replace = FALSE),] # sample 10 locations
WAUS <- WAUS[(WAUS$Location %in% L),]
WAUS <- WAUS[sample(nrow(WAUS), 2000, replace = FALSE),] # sample 2000 rows

#Question 1

#extracting when it is more humid than the previous day compared to those where it is less humid
str(WAUS)

yes_humid <- WAUS[which(WAUS$MHT == 1),]
no_humid <- WAUS[which(WAUS$MHT == 0),]

# number of row in more humid than previous day
nrow(yes_humid)

# propotion
nrow(yes_humid) / nrow(WAUS)

# number of row in less humid than previous day
nrow(no_humid)

# propotion
nrow(no_humid) / nrow(WAUS)

# propotion of days when it is more humid than the previous
# day compared to those where it is less humid
nrow(yes_humid) / nrow(no_humid)

# description of the data
description(WAUS)
```

```
#summary
summary(WAUS)


# standard deviation for Evaporation
sd(WAUS$Evaporation, na.rm=TRUE)

# coefficient of variation(CV) for Evaporation variable
sd(WAUS$Evaporation, na.rm=TRUE) / mean(WAUS$Evaporation, na.rm = TRUE)

# as we can see the result is lower than 1, so the standard deviation is low which suggest the data are
clustered around the mean
# and hence later in question 2 it make sense for me to replace the NA value with mean value since the
standard deviation suggest the data
# are clustered around the mean


# standard deviation for Sunshine
sd(WAUS$Sunshine, na.rm=TRUE)

# coefficient of variation(CV) for Sunshine variable
sd(WAUS$Sunshine, na.rm=TRUE) / mean(WAUS$Sunshine, na.rm = TRUE)

# as we can see the result is lower than 1, so the standard deviation is low which suggest the data are
clustered around the mean
# and hence later in question 2 it make sense for me to replace the NA value with mean value since the
standard deviation suggest the data
# are clustered around the mean

# standard deviation for Cloud9am
sd(WAUS$Cloud9am, na.rm=TRUE)

# coefficient of variation(CV) for Cloud9am variable
sd(WAUS$Cloud9am, na.rm=TRUE) / mean(WAUS$Cloud9am, na.rm = TRUE)

# as we can see the result is lower than 1, so the standard deviation is low which suggest the data are
clustered around the mean
# and hence later in question 2 it make sense for me to replace the NA value with mean value since the
standard deviation suggest the data
# are clustered around the mean

# standard deviation for Cloud3pm
sd(WAUS$Cloud3pm, na.rm=TRUE)

# coefficient of variation(CV) for Cloud3pm variable
sd(WAUS$Cloud3pm, na.rm=TRUE) / mean(WAUS$Cloud3pm, na.rm = TRUE)

# as we can see the result is lower than 1, so the standard deviation is low which suggest the data are
clustered around the mean
# and hence later in question 2 it make sense for me to replace the NA value with mean value since the
standard deviation suggest the data
# are clustered around the mean


#question 2 preprocessing
head(WAUS)
summary(WAUS)

str(WAUS)
```

```
humid.data <- WAUS

# converting NA to their mean value for variable with rouhgly 50% of missing value

#Evaporation
humid.data$Evaporation[is.na(humid.data$Evaporation)] <- mean(humid.data$Evaporation,
na.rm=TRUE)

#Sunshine
humid.data$Sunshine[is.na(humid.data$Sunshine)] <- mean(humid.data$Sunshine, na.rm=TRUE)

#Cloud9am
humid.data$Cloud9am[is.na(humid.data$Cloud9am)] <- mean(humid.data$Cloud9am, na.rm=TRUE)

#Cloud3pm
humid.data$Cloud3pm[is.na(humid.data$Cloud3pm)] <- mean(humid.data$Cloud3pm, na.rm=TRUE)

# data without omitting NA value
no.humid.data <- humid.data

#convert character(string) data to factor
humid.data[, c(8, 10, 11, 20)] <- lapply(humid.data[, c(8, 10, 11, 20)], factor)

# convert target variable into factor
humid.data[,22]<-as.factor(humid.data[,22])

#omit NA value
humid.data <- humid.data[complete.cases(humid.data),]

#question 3
set.seed(32439180) #Student ID as random seed
train.row = sample(1:nrow(humid.data), 0.7*nrow(humid.data))
humid.train <- humid.data[train.row,]
humid.test <- humid.data[-train.row,]


#question 4

# decision tree
set.seed(32439180)
humid.tree = tree(MHT~., data = humid.train)

par(mar = c(8, 5, 2, 3))
plot(humid.tree)
text(humid.tree, pretty=0)

# Naïve Bayes
set.seed(32439180)
humid.bayes = naiveBayes(MHT~., data=humid.train)

# Bagging
set.seed(32439180)
humid.bag <- bagging(MHT~., data = humid.train)

# Boosting
set.seed(32439180)
humid.boost <- boosting(MHT~., data = humid.train)

# Random forest
set.seed(32439180)
```

```
humid.rf = randomForest(MHT~., data = humid.train, na.action = na.exclude)

#question 5
#decision tree
humid.predtree = predict(humid.tree, humid.test, type="class")
humid.tree.tab = table(Predicted_Class = humid.predtree, Actual_Class = humid.test$MHT)
print(humid.tree.tab)


# accuracy
# replace NA value
(115+89)/(115+106+65+89)

#precision
(89)/(89+65)

#recall
(89)/(89+106)

# Naïve Bayes
humid.predbayes = predict(humid.bayes, humid.test)
humid.bayes.tab = table(Predicted_Class = humid.predbayes, Actual_Class = humid.test$MHT)
print(humid.bayes.tab)

# accuracy
# replace NA value
(51+155)/(51+40+129+155)

#precision
(155)/(155+129)

#recall
(155)/(155+40)

# Bagging
humidpred.bag = predict.bagging(humid.bag, humid.test)
print(humidpred.bag$confusion)

# accuracy
# replace NA value
(89+122) / (89+73+91+122)

#precision
(122)/(122+91)

#recall
(122)/(122+73)

# Boosting
humidpred.boost = predict.boosting(humid.boost, humid.test)
print(humidpred.boost$confusion)

# accuracy
# replaced NA value
(103+113)/(103+82+77+113)

#precision
(113)/(113+77)

#recall
```

```
(113)/(113+82)

# Random forest
humid.predrf = predict(humid.rf, humid.test)
humid.rf.tab = table(Predicted_Class = humid.predrf, Actual_Class = humid.test$MHT)
print(humid.rf.tab)

#accuracy
# replaced NA value
(69+146)/(69+49+111+146)

#precision
(146)/(146+111)

#recall
(146)/(146+49)

#question 6
#Decision Tree
tree.pre.humid = predict(humid.tree, humid.test, type="vector")
tree.pred.humid <- prediction(tree.pre.humid[,2], humid.test$MHT)
tree.pref.humid <- performance(tree.pred.humid,"tpr","fpr")

# only 11, less than 1% so our roc cruve is almost equal to 1
par(mar = c(8, 5, 2, 3))
plot(tree.pref.humid, ylab="True positive rate", col = "skyblue")
abline(0,1)

#AUC
tree.auc = performance(tree.pred.humid, "auc")
print(as.numeric(tree.auc @y.values))

# Naïve Bayes
bayes.pre.humid = predict(humid.bayes, humid.test, type = 'raw')
bayes.pred.humid <- prediction( bayes.pre.humid[,2], humid.test$MHT)
bayes.pref.humid <- performance(bayes.pred.humid,"tpr","fpr")
plot(bayes.pref.humid , add=TRUE, col = "blueviolet")

#AUC
bayes.auc = performance(bayes.pred.humid, "auc")
print(as.numeric(bayes.auc @y.values))

# Bagging
bag.pre.humid = predict.bagging(humid.bag, humid.test)
bag.pred.humid <- prediction(bag.pre.humid$prob[,2], humid.test$MHT)
bag.pref.humid <- performance(bag.pred.humid,"tpr","fpr")
plot(bag.pref.humid, add=TRUE, col="red")

# AUC
bag.auc = performance(bag.pred.humid, "auc")
print(as.numeric(bag.auc @y.values))

# Boosting
boost.pre.humid = predict.boosting(humid.boost, humid.test)
boost.pred.humid <- prediction(boost.pre.humid$prob[,2], humid.test$MHT)
boost.pref.humid <- performance(boost.pred.humid,"tpr","fpr")
plot(boost.pref.humid, add=TRUE, col="green")

# AUC
boost.auc = performance(boost.pred.humid, "auc")
```

```
print(as.numeric(boost.auc @y.values))

# Random forest
rf.pre.humid = predict(humid.rf, humid.test, type="prob")
rf.pred.humid <- prediction(rf.pre.humid[,2], humid.test$MHT)
rf.pref.humid <- performance(rf.pred.humid,"tpr","fpr")
plot(rf.pref.humid, add=TRUE, col="orange")

# AUC
rf.auc = performance(rf.pred.humid, "auc")
print(as.numeric(rf.auc @y.values))

#add in title and legend
title("ROC curve for each classifier")
legend("bottomright", legend = c("Decision Tree", "Naive Bayes", "Bagging", "Boosting", "Random
Forest"), lty = 1, col = c("skyblue","blueviolet","red", "green", "orange"))

#question 7
# in word

# question 8
#atrr importance

# Decision Tree
summary(humid.tree)

# Bagging
sor.impo.bag <- sort((humid.bag$importance), decreasing = TRUE)
print(sor.impo.bag)

# ratio with the most important variable
(sor.impo.bag /max(sor.impo.bag ))

# Boosting
sor.impo.boost <-sort((humid.boost$importance), decreasing =TRUE)
print(sor.impo.boost)

# ratio with the most important variable
(sor.impo.boost/max(sor.impo.boost))

# Random Forest
sor.impo.rf <- humid.rf$importance[order(humid.rf$importance, decreasing =TRUE),]
print(sor.impo.rf)

#since it is hard to determine whether it is the most important variable or not hence:
print(sor.impo.rf/max(sor.impo.rf))

#question 9
# Select important variables based on question 8
# hence i got 7 variables Cloud9am, WindDir9am, WindGustDir, WindSpeed3pm, WindDir3pm,
Temp9am and Temp3pm

# preprocess the data
# only select columns i needed for this question
str(humid.data.9)
humid.data.9 <- subset(no.humid.data, select = c(Cloud9am, WindDir9am, WindGustDir,
WindSpeed3pm, WindDir3pm, Temp9am, Temp3pm,MHT))

#omit NA value
humid.data.9 <- humid.data.9[complete.cases(humid.data.9),]
```

```
#convert character(string) data  and target variable(MHT) to factor
humid.data.9[, c(2, 3, 5, 8)] <- lapply(humid.data.9[, c(2, 3, 5, 8)], factor)

# reselect the training and testing set
set.seed(32439180) #Student ID as random seed
train.row.9 = sample(1:nrow(humid.data.9), 0.7*nrow(humid.data.9))
humid.train.9 <- humid.data.9[train.row,]
humid.test.9 <- humid.data.9[-train.row,]

set.seed(32439180)
humid.s.tree =
tree(MHT~Cloud9am+WindDir9am+WindGustDir+WindSpeed3pm+WindDir3pm+Temp9am+Temp3
pm, data= humid.train.9)
summary(humid.s.tree)

plot(humid.s.tree)
text(humid.s.tree, pretty=0)

set.seed(32439180)
cv.humid.tree.9= cv.tree(humid.s.tree, FUN = prune.misclass, K=500)
cv.humid.tree.9

#prune using size 4 considering lowest misclassification rate
pruned.cv.tree.9 = prune.misclass(humid.s.tree, best = 4)
summary(pruned.cv.tree.9)
plot(pruned.cv.tree.9)
text(pruned.cv.tree.9, pretty = 0)
title("Pruned Decision Tree")

# check accuracy using the simple tree
sim.tree.pred = predict(pruned.cv.tree.9, humid.test.9, type = "class")
table(actual = humid.test.9$MHT, predicted = sim.tree.pred)

#Accuracy
(189+208)/(189+161+184+208)

#AUC
s.tree.pre.humid = predict(pruned.cv.tree.9, humid.test.9, type="vector")
s.tree.pred.humid <- prediction(s.tree.pre.humid[,2], humid.test.9$MHT)
s.tree.pref.humid <- performance(s.tree.pred.humid,"tpr","fpr")

# only 11, less than 1% so our roc cruve is almost equal to 1
par(mar = c(8, 5, 2, 3))
plot(s.tree.pref.humid, ylab="True positive rate", col = "skyblue")
abline(0,1)

#AUC
s.tree.auc = performance(s.tree.pred.humid, "auc")
print(as.numeric(s.tree.auc @y.values))



# Question 10

# preprocess the data
# only select columns i needed for this question
humid.data.10 <- subset(no.humid.data, select = c(1:19,21:22))

#omit NA value
```

```
humid.data.10 <- humid.data.10[complete.cases(humid.data.10),]

#factorise the categorical variables
humid.data.10[, c(8, 10, 11)] <- lapply(humid.data.10[, c(8, 10, 11)], factor)

#factorise the target variable MHT
humid.data.10[,21]<-as.factor(humid.data.10[,21])

# reselect the training and testing set
set.seed(32439180) #Student ID as random seed
train.row.10  = sample(1:nrow(humid.data.10 ), 0.7*nrow(humid.data.10 ))
humid.train.10  <- humid.data.10[train.row.10,]
humid.test.10  <- humid.data.10[-train.row.10,]

# train the model
# cv https://topepo.github.io/caret/model-training-and-tuning.html
set.seed(32439180)
tr.control <- trainControl(method = "cv", number=10)

# Decision Tree
set.seed(32439180)
tree.cv.10 = tree(MHT~., data= humid.train.10)
summary(tree.cv.10)

plot(tree.cv.10)
text(tree.cv.10, pretty=0)
title("Decision Tree")

# since orginal trees tend to overfit
# i willperform cross validation to select the optimal size tree
set.seed(32439180)
cv.humid.tree= cv.tree(tree.cv.10, FUN = prune.misclass, K=500)
cv.humid.tree

#prune using size 2 considering lowest misclassification rate
pruned.cv.tree.10 = prune.misclass(tree.cv.10, best = 2)
summary(pruned.cv.tree.10)
plot(pruned.cv.tree.10)
text(pruned.cv.tree.10, pretty = 0)
title("Pruned Decision Tree")

# check accuracy using the pruned tree
pruned.PD.predict = predict(pruned.cv.tree.10, humid.test.10, type = "class")
table(predicted = pruned.PD.predict ,actual = humid.test.10$MHT)

 #Accuracy
(162+45)/(162+28+143+45)

#precision
(45)/(45+28)

#recall
(45)/(45+143)

#AUC
cv.pre.tree.10 = predict(pruned.cv.tree.10, humid.test.10, type="vector")
cv.pred.tree.10 <- prediction(cv.pre.tree.10[,2], humid.test.10$MHT)
cv.perf.tree.10 <- performance(cv.pred.tree.10,"tpr","fpr")

# only 11, less than 1% so our roc cruve is almost equal to 1
```

```
par(mar = c(8, 5, 2, 3))
plot(cv.perf.tree.10, ylab="True positive rate", col = "skyblue")
abline(0,1)

#AUC
cv.tree.auc = performance(cv.pred.tree.10, "auc")
print(as.numeric(cv.tree.auc @y.values))

# naive bayes
set.seed(32439180)
bayes.cv.10 <- train(MHT~., data=humid.train.10, method='naive_bayes',trControl = tr.control)
bayes.cv.pred <- predict(bayes.cv.10, newdata = humid.test.10)
bayes.cv.tab = table(Predicted_Class = bayes.cv.pred, Actual_Class = humid.test.10$MHT)

#Accuracy
print(bayes.cv.tab )
(59+154)/(59+34+131+154)

#precision
(154)/(154+131)

#recall
(154)/(154+34)

bayes.cv.pro <- predict(bayes.cv.10, newdata=humid.test.10,type="prob")
bayes.cv.pred <- prediction(bayes.cv.pro[,2], humid.test.10$MHT)
bayes.cv.perf <- performance(bayes.cv.pred,"tpr","fpr")
plot(bayes.cv.perf, add=TRUE, col="blueviolet")

# AUC
bayes.cv.auc = performance(bayes.cv.pred, "auc")
print(as.numeric(bayes.cv.auc @y.values))

#bagging with cv
set.seed(32439180)
bag.cv.10 <- train(MHT~., data=humid.train.10, method='AdaBag', trControl = tr.control)

bag.cv.pred <- predict(bag.cv.10, newdata=humid.test.10)
bag.cv.tab = table(Predicted_Class = bag.cv.pred, Actual_Class = humid.test.10$MHT)
print(bag.cv.tab )

#Accuracy
(89+136)/(89+52+101+136)

#precision
(136)/(136+101)

#recall
(136)/(136+52)

bag.cv.pro <- predict(bag.cv.10, newdata=humid.test.10,type="prob")
bag.cv.pred <- prediction(bag.cv.pro[,2], humid.test.10$MHT)
bag.cv.perf <- performance(bag.cv.pred,"tpr","fpr")
plot(bag.cv.perf, add=TRUE, col="red")

# AUC
bag.cv.auc = performance(bag.cv.pred, "auc")
print(as.numeric(bag.cv.auc @y.values))

#boosting with cross validation
```

```
set.seed(32439180)
boost.cv.10 <- train(MHT~., data=humid.train.10, method='AdaBoost.M1', trControl = tr.control)

boost.cv.pred <- predict(boost.cv.10, newdata=humid.test.10)
boost.cv.tab = table(Predicted_Class = boost.cv.pred, Actual_Class = humid.test.10$MHT)
print(boost.cv.tab)

#Accuracy
(98+119)/(98+69+92+119)

#precision
(119)/(119+92)

#recall
(119)/(119+69)

boost.cv.pro <- predict(boost.cv.10, newdata=humid.test.10,type="prob")
boost.cv.pred <- prediction(boost.cv.pro[,2], humid.test.10$MHT)
boost.cv.perf <- performance(boost.cv.pred,"tpr","fpr")
plot(boost.cv.perf, add=TRUE, col="green")

# AUC
boost.cv.auc = performance(boost.cv.pred, "auc")
print(as.numeric(boost.cv.auc @y.values))

# Random Forest with cv
set.seed(32439180)
rf.cv.10 <- train(MHT~., data=humid.train.10, method='rf', trControl = tr.control)

rf.cv.pred <- predict(rf.cv.10, newdata=humid.test.10)
rf.cv.tab = table(Predicted_Class = rf.cv.pred, Actual_Class = humid.test.10$MHT)
print(rf.cv.tab)

#Accuracy
(103+121)/(103+67+87+121)

#precision
(121)/(121+87)

#recall
(121)/(121+67)

rf.cv.pro <- predict(rf.cv.10, newdata=humid.test.10,type="prob")
rf.cv.pred <- prediction(rf.cv.pro[,2], humid.test.10$MHT)
rf.cv.perf <- performance(rf.cv.pred,"tpr","fpr")
plot(rf.cv.perf, add=TRUE, col="orange")

# AUC
rf.cv.auc = performance(rf.cv.pred, "auc")
print(as.numeric(rf.cv.auc @y.values))

#add in title and legend
title("ROC curve for each classifier")
legend("bottomright", legend = c("Decision Tree", "Naive Bayes", "Bagging", "Boosting", "Random
Forest"), lty = 1, col = c("skyblue","blueviolet","red", "green", "orange"))

# Question 11
library(neuralnet)

# preprocess
```

```
# preprocess the data
# only select columns i needed for this question
humid.data.11 <- subset(no.humid.data, select = c(1:7,9,12:19,21:22))
str(humid.data.11)

#omit NA value
humid.data.11 <- humid.data.11[complete.cases(humid.data.11),]

# reselect the training and testing set
set.seed(32439180) #Student ID as random seed
train.row.11  = sample(1:nrow(humid.data.11 ), 0.7*nrow(humid.data.11 ))
humid.train.11  <- humid.data.11[train.row.11,]
humid.test.11  <- humid.data.11[-train.row.11,]

#scale the data
humid.train.11[1:17] <- scale(humid.train.11[1:17])
humid.test.11[1:17] <- scale(humid.test.11[1:17])

# build the model
set.seed(32439180)
humid.nn = neuralnet(MHT ~., humid.train.11, hidden=3,linear.output = FALSE)

# make prediction
humid.nn.pred = compute(humid.nn, humid.test.11)

# now round these down to integers
humid.nn.pred = as.data.frame(round(humid.nn.pred$net.result,0))

# plot confusion matrix
table(predicted = humid.nn.pred$V1, actual = humid.test.11$MHT)

# accuracy
(71+148)/(71+138+47+148)

#precision
(148)/(148+138)

#recall
(148)/(148+47)

#AUC

# unload the library as neuralnet package will cause problem to prediction()
detach("package:neuralnet", unload=TRUE)
ann.pre = predict(humid.nn, humid.test.11, type="prob")
ann.pred <- prediction(ann.pre, humid.test.11$MHT)

# AUC
ann.auc = performance(ann.pred, "auc")
print(as.numeric(ann.auc @y.values))

#question 12
# learned from FIT2086 week 9 studio 9
library(kknn)

# data pre-process
humid.data.12 <- subset(no.humid.data, select = c(1:19,21:22))

#omit NA value
humid.data.12 <- humid.data.12[complete.cases(humid.data.12),]
```

```
#factorise the categorical variables
humid.data.12[, c(8, 10, 11)] <- lapply(humid.data.12[, c(8, 10, 11)], factor)


# reselect the training and testing set
set.seed(32439180) #Student ID as random seed
train.row.12  = sample(1:nrow(humid.data.12 ), 0.7*nrow(humid.data.12 ))
humid.train.12  <- humid.data.12[train.row.12,]
humid.test.12  <- humid.data.12[-train.row.12,]

# Normalise the data
humid.train.12[1:7] <- scale(humid.train.12[1:7])
humid.train.12[9] <- scale(humid.train.12[9])
humid.train.12[12:20] <- scale(humid.train.12[12:20])

humid.test.12[1:7] <- scale(humid.test.12[1:7])
humid.test.12[9] <- scale(humid.test.12[9])
humid.test.12[12:20] <- scale(humid.test.12[12:20])

# create a list for each different kernels
kernels = c("rectangular","triangular","epanechnikov","gaussian","rank","optimal")

# Use tran.kknn() to try diffrent combination of diffrent k values and kernel and select the best
# nominated by cross validation
knn = train.kknn(MHT ~ ., data = humid.train.12, kmax=25, kernel=kernels)

# after that make a prediction based on knn suggestion
knn.pred = fitted( kknn(MHT ~ ., humid.train.12 , humid.test.12, kernel = knn$best.parameters$kernel,
k = knn$best.parameters$k) )

#AUC
knn.auc.pred <- prediction(knn.pred, humid.test.12$MHT)

knn.auc = performance(knn.auc.pred, "auc")
print(as.numeric(knn.auc @y.values))

# because knn.pred only return each probabiliy of the prediction = 1 or =0, hence i set a threshold value
of 0.5 so if the value is gretaer than 0.5
# the prediction would be 1 else 0 so that i can make the confusion matrix
pred.nn <- ifelse(knn.pred>0.5, 1, 0)

# the confusion matrix
table(predicted=pred.nn, actual = humid.test.12$MHT)

#Accuracy
(88+129)/(88+59+102+129)

#precision
(129)/(129+102)

#recall
(129)/(129+59)
```