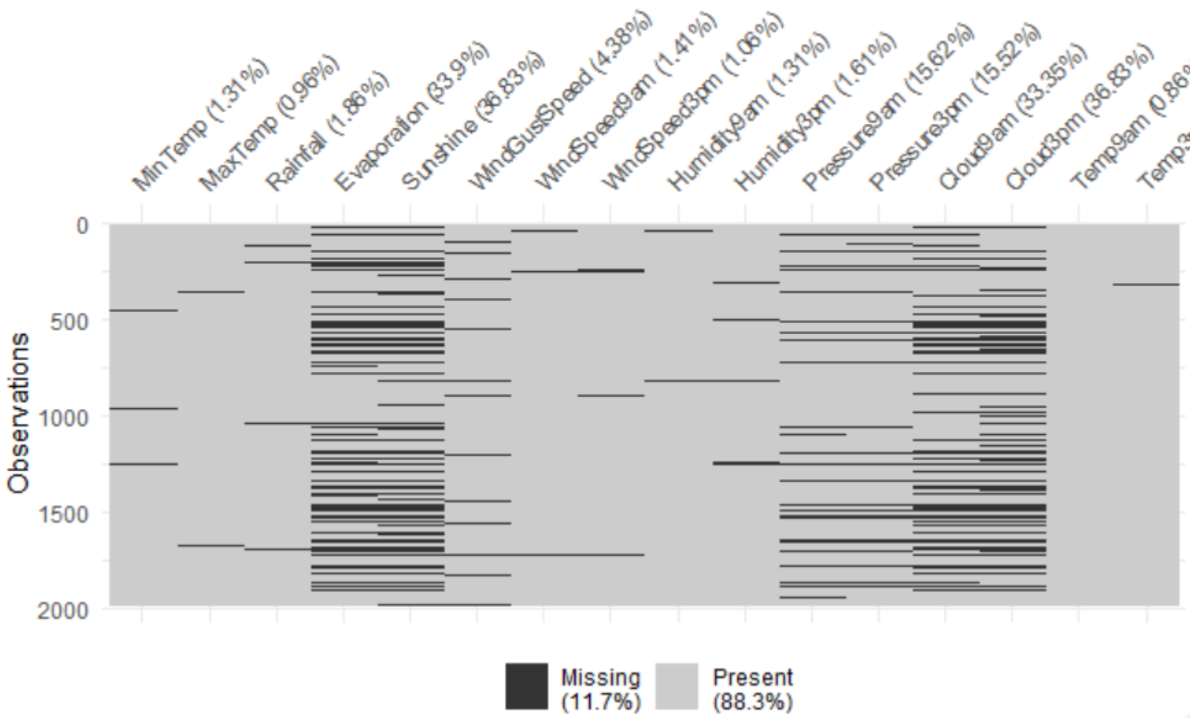
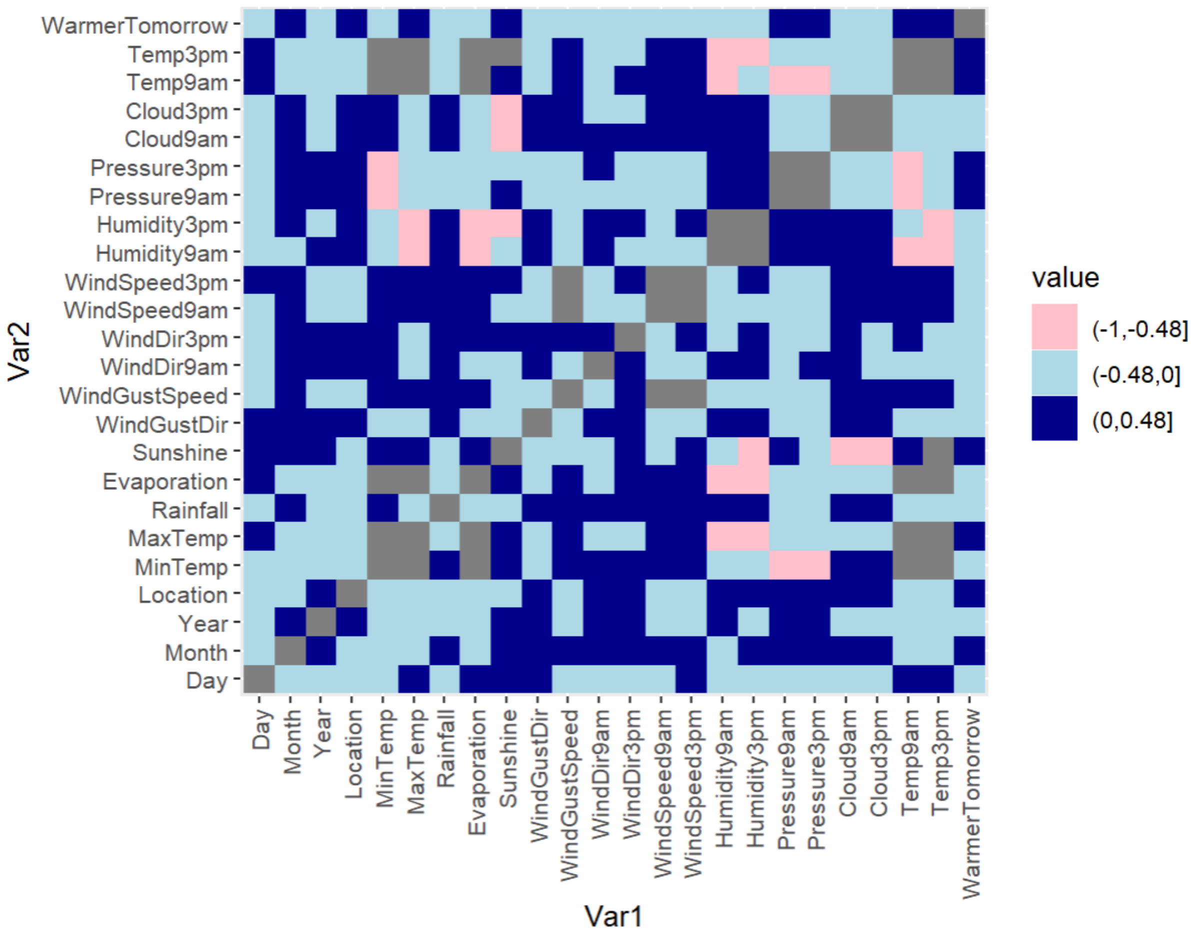
```
    MinTemp          MaxTemp          Rainfall          Evaporation       Sunshine        WindGustSpeed
 Min.   :-3.10   Min.   : 9.00   Min.   : 0.000   Min.   : 0.00   Min.   : 0.000   Min.   : 7.00
 1st Qu.: 8.10   1st Qu.:18.30   1st Qu.: 0.000   1st Qu.: 2.80   1st Qu.: 4.525   1st Qu.:31.00
 Median :12.60   Median :22.20   Median : 0.000   Median : 4.60   Median : 8.200   Median :39.00
 Mean   :12.32   Mean   :23.03   Mean   : 2.121   Mean   : 5.57   Mean   : 7.474   Mean   :39.98
 3rd Qu.:16.80   3rd Qu.:26.73   3rd Qu.: 0.600   3rd Qu.: 7.40   3rd Qu.:10.400   3rd Qu.:48.00
 Max.   :29.70   Max.   :45.10   Max.   :89.000   Max.   :81.60   Max.   :13.900   Max.   :94.00
 NA's   :26      NA's   :20      NA's   :37       NA's   :676     NA's   :734      NA's   :89
  WindSpeed9am    WindSpeed3pm     Humidity9am       Humidity3pm       Pressure9am     Pressure3pm
 Min.   : 0.00   Min.   : 0.00   Min.   :  5.00   Min.   :  2.00   Min.   : 991    Min.   : 985.5
 1st Qu.: 7.00   1st Qu.:13.00   1st Qu.: 57.00   1st Qu.: 36.00   1st Qu.:1013    1st Qu.:1011.1
 Median :13.00   Median :17.00   Median : 69.00   Median : 51.00   Median :1018    Median :1016.1
 Mean   :13.57   Mean   :18.14   Mean   : 67.82   Mean   : 50.54   Mean   :1018    Mean   :1015.9
 3rd Qu.:19.00   3rd Qu.:24.00   3rd Qu.: 82.00   3rd Qu.: 65.25   3rd Qu.:1023    3rd Qu.:1020.8
 Max.   :54.00   Max.   :54.00   Max.   :100.00   Max.   :100.00   Max.   :1039    Max.   :1036.2
 NA's   :29      NA's   :22      NA's   :26       NA's   :32       NA's   :310     NA's   :308
    Cloud9am         Cloud3pm        Temp9am          Temp3pm
 Min.   :0.000   Min.   :0.00   Min.   :-0.40   Min.   : 7.00
 1st Qu.:1.500   1st Qu.:2.00   1st Qu.:12.40   1st Qu.:17.20
 Median :5.000   Median :5.00   Median :16.80   Median :20.75
 Mean   :4.521   Mean   :4.52   Mean   :16.82   Mean   :21.58
 3rd Qu.:7.000   3rd Qu.:7.00   3rd Qu.:20.80   3rd Qu.:25.00
 Max.   :8.000   Max.   :8.00   Max.   :37.80   Max.   :43.60
 NA's   :665     NA's   :734    NA's   :17      NA's   :20
```

Graph 1.0



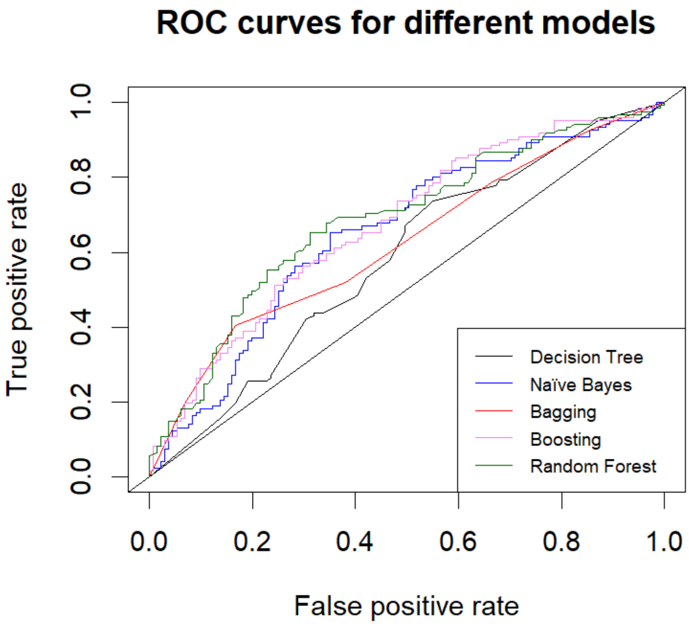
Graph 1.1

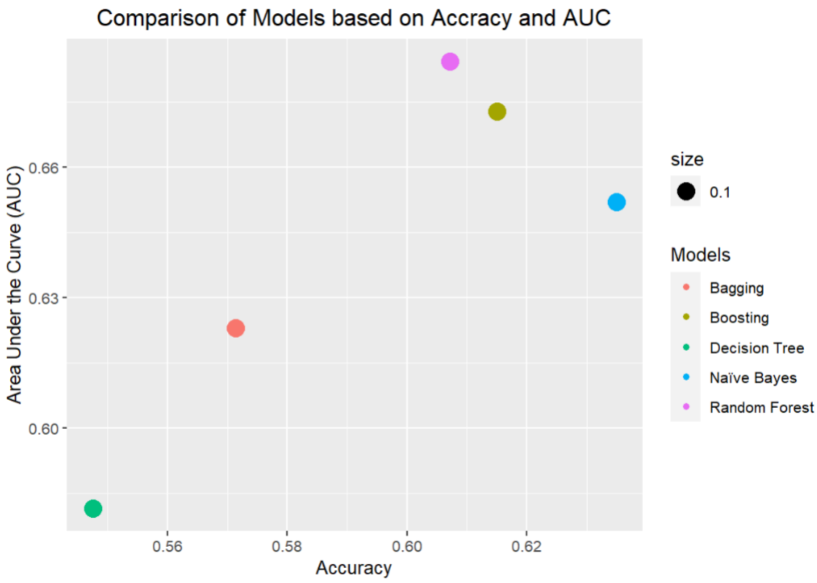
There exists a lot of null values in the data especially in evaporation, Sunshine, cloud9am and cloud3pm columns with more than a third of values missing (Graph 1.1). The columns with highest number of missing values cannot be ignored as they will affect the credibility of our prediction models. The missing values cannot be imputed as four aforementioned columns have more than 30 percent missing values each. Hence, we eliminate the rows with missing values and hope that the dataset is large enough for a well-performing classifier to be developed. As time is a human construct and has no connection to the natural world, we can avoid using time variables like Day, Month in the development of our classifiers. Moreover, we avoid using location as climate change has affected weather patterns and make analysis without it more concrete and valuable. The ratio of warmer to cooler days is 1.132116 which shows that data is balanced and does not affect the performance of certain classifiers.



Graph 1.2

The linear relationship between variables and target variable (i.e., WarmerTomorrow) is not especially strong (Graph 1.2). This means that classifiers like Bagging, Boosting and ANN that are better equipped to handle non-linear relationships in the data should be developed and expected to perform better.

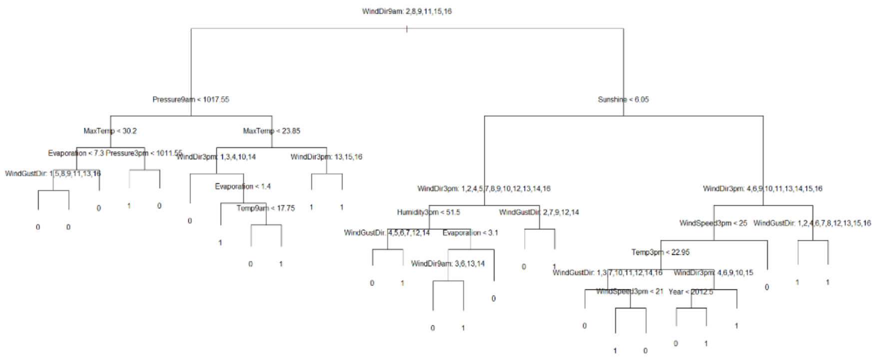




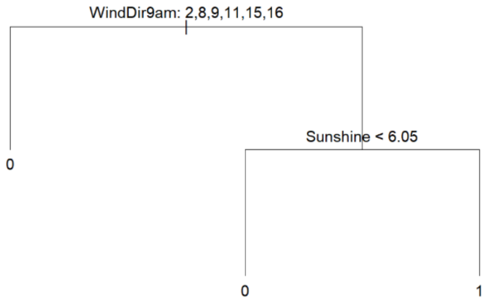
Graph 1.3

Graph 1.4.0

Random Forrest is the best performing classifier followed by boosting as close second based on AUC (Graph 1.3 and 1.4.0). This superior performance can be attributed to the fact that Random Forest uses a subset of data points and subset of attributes from the original dataset as there were large number of attributes present in data set. Overall, the performances of these models are subpar and further improvements are required. One of the many options available is to filter out weak learners as they dilute the effects of stronger predictors and make models unnecessarily more complicated. Hence, we will try to find the most important predictors in the models developed.

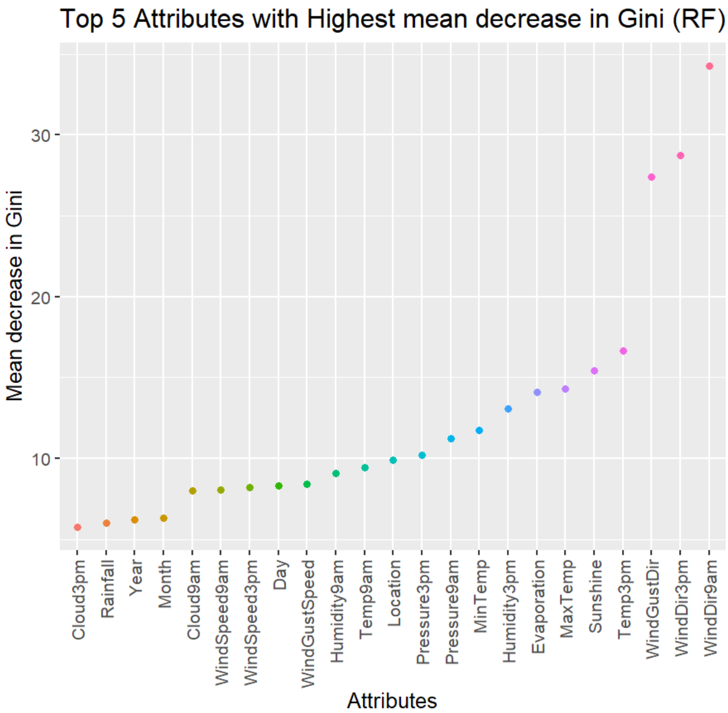


Graph 1.4.1

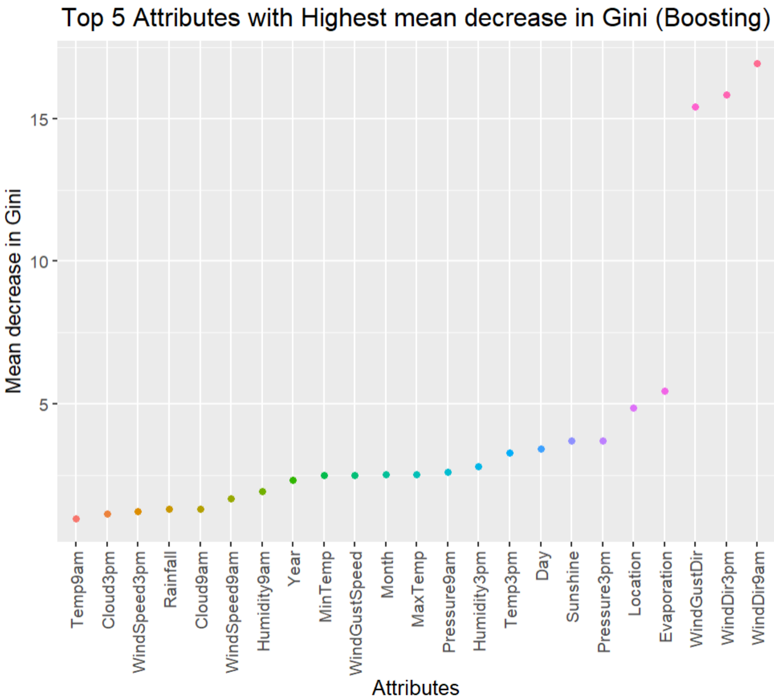


Graph 1.5

The number of variables used in original decision tree were 13 with 27 terminal nodes (Graph 1.4.1). After pruning the decision tree, the number of variables reduced to two (WindDir9am and Sunshine) and three terminal nodes (Graph 1.5). The accuracy and AUC increased from 0.547619 and 0.581446 to 0.6349 and 0.6231153 respectively. This shows that more variables do not guarantee better performance.

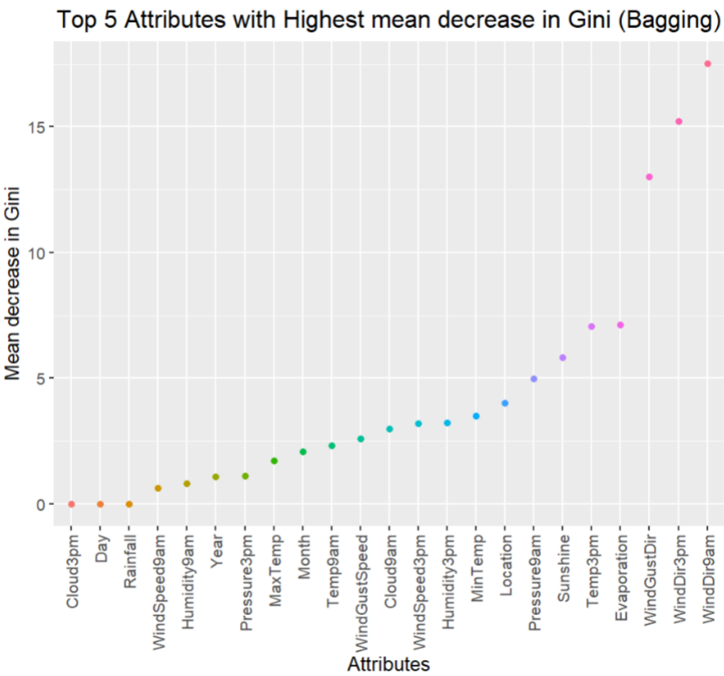


Graph 1.6

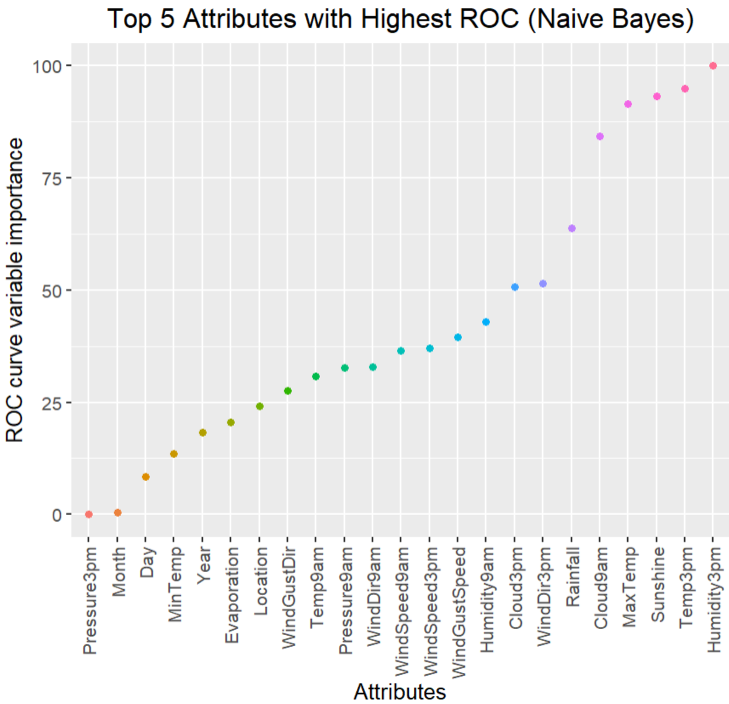


Graph 1.7

The mean decrease in Gini measures the homogeneity of leaves or nodes in ensemble methods like boosting, bagging and Random Forest. The importance of a variable is higher if its mean decrease in Gini is higher. The top five most important variables (i.e., The mean decrease in Gini > 15) chosen by the best performing classifier (RF) are WindDir9am, WindDir3pm, WindGustDir, Temp3pm and Sunshine (in decreasing order) (Graph 1.6). Boosting model also shares the top three predictors with RF which reaffirms the importance of these variables (Graph 1.7).

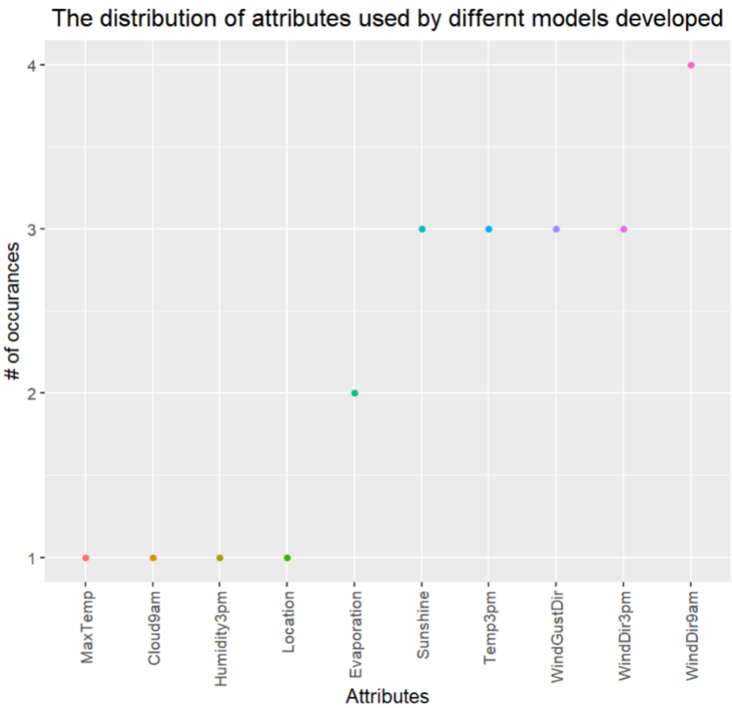


Graph 1.8



Graph 1.9

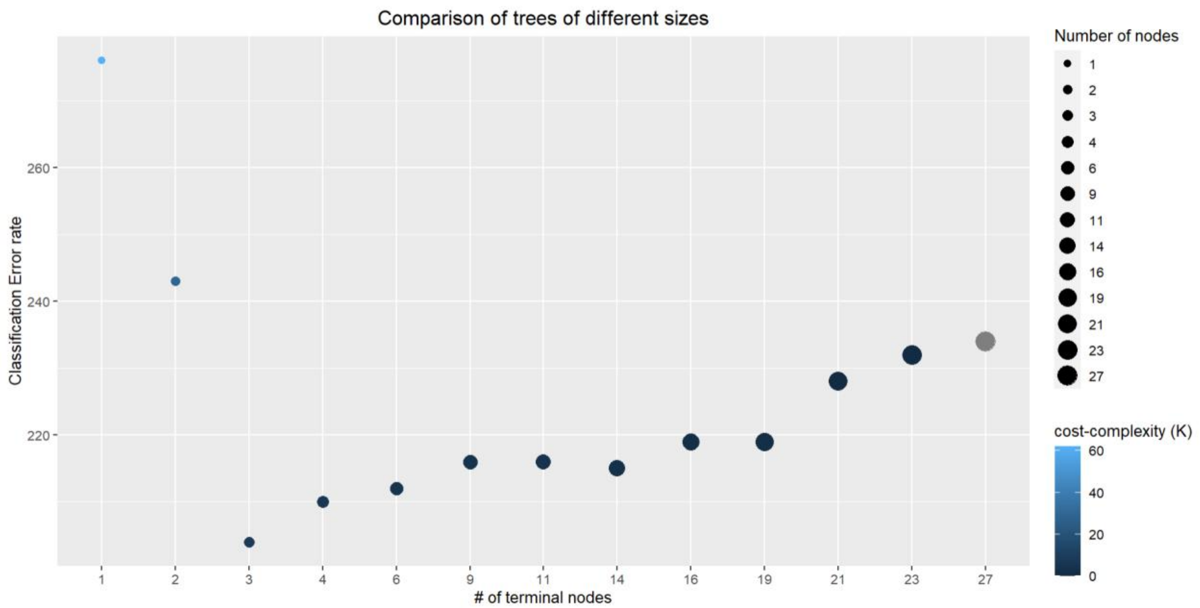
Interestingly, Bagging algorithm also shares the top four most important predictors with Boosting even though they are fundamentally two different algorithms and can partially be explained by the fact that the dataset is balanced. However, Naïve bayes picked Humidity3pm, Temp3pm, Sunshine, MaxTemp and Cloud9am as top five predictors (i.e., ROC > 75) which is somehow different to what other models picked. The area under ROC curve is used as the measure of variable importance for Naïve Bayes here.



Graph 2.0

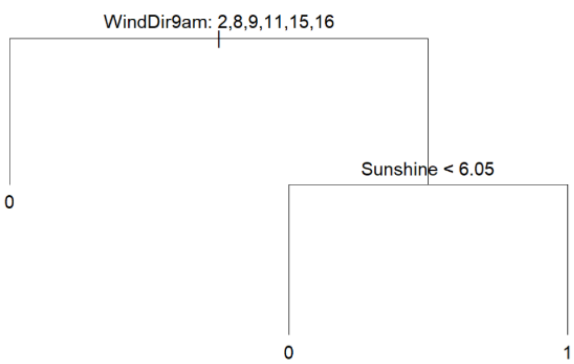
The frequency of attributes which were appeared in the top five of each classifier is graphed above (Graph 2.0). We use the top five attributes "Sunshine", "Temp3pm", "WindGustDir", "WindDir3pm", "WindDir9am" as part of our data science workflow. And if we are not satisfied with the performance metrics of models developed, we will use more variables in our models.

Since one of the classifiers that is easy for humans to classify new instances of data is moderately deep decision tree, we use the decision tree developed in task 4 ad try to reduce its complexity by reducing the number of its terminal nodes and attributes. It is true that original tree could have been used by a human, but the process is going to be monotonous which makes it error-prone to be used by humans due to relatively high number of attributes and some variables having large set of values.

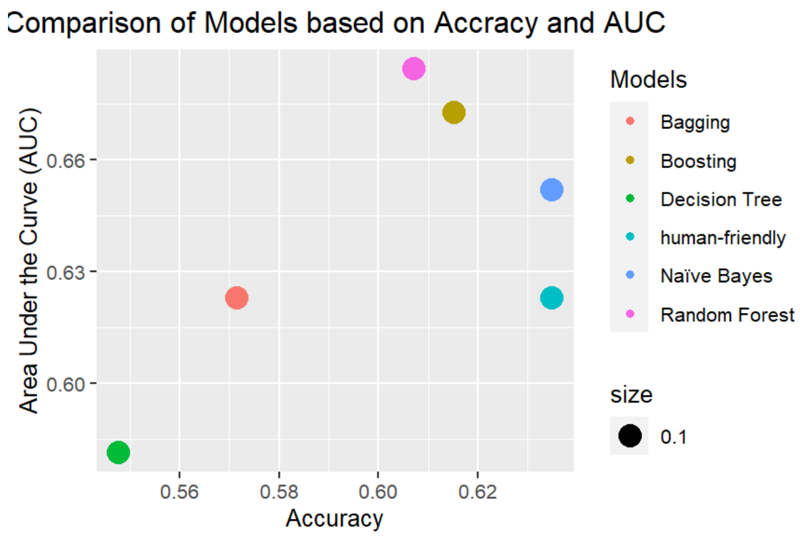


Graph 2.1

By using the function **cv.tree()** which cross-validates to determine the ideal tree complexity, the tree with three terminal nodes has the lowest classification error rate and low-cost complexity (K) (Graph 2.1).



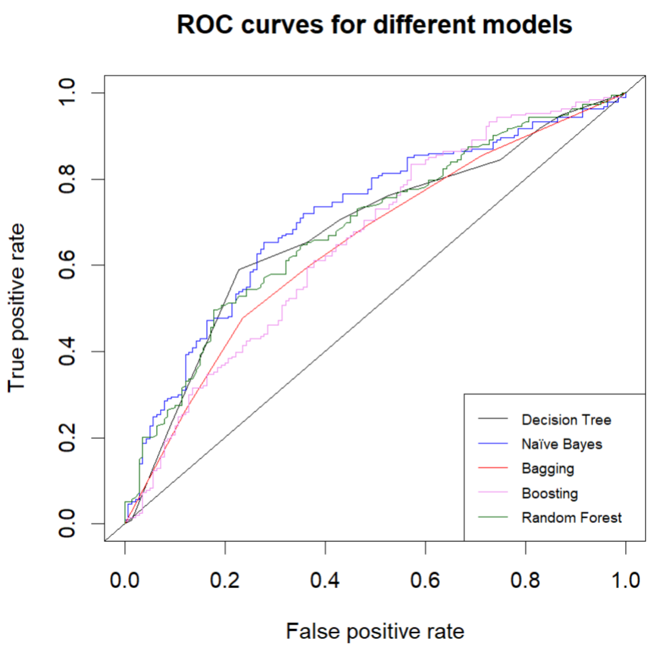
Graph 2.2



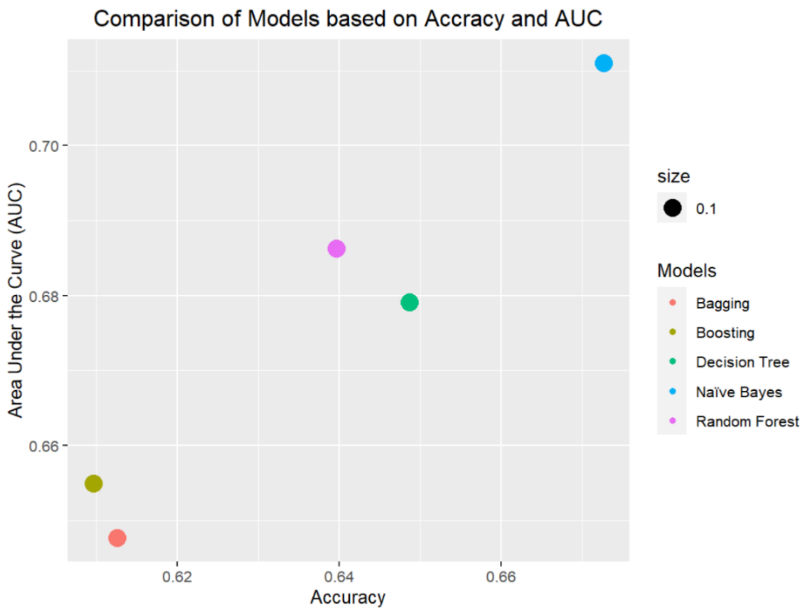
Graph 2.3

The developed tree (Graph 2.3) has accuracy of 0.63 which is as good as Naïve Bayes. However, its AUC which is a better metric to measure the performance of classifiers is 0.62. This simple model surprisingly performs even slightly better than bagging which is a more complex algorithm and not possible by humans to implement. In fact, it performs quite well in comparison to more sophisticated algorithms like Boosting and Random Forest. However, the models developed do not perform extremely well like the AUC for RF is only 0.6844679 (~ 70). We now try to develop a better tree-based classifier by utilising different techniques.

Performance of models by using top 5 Variables



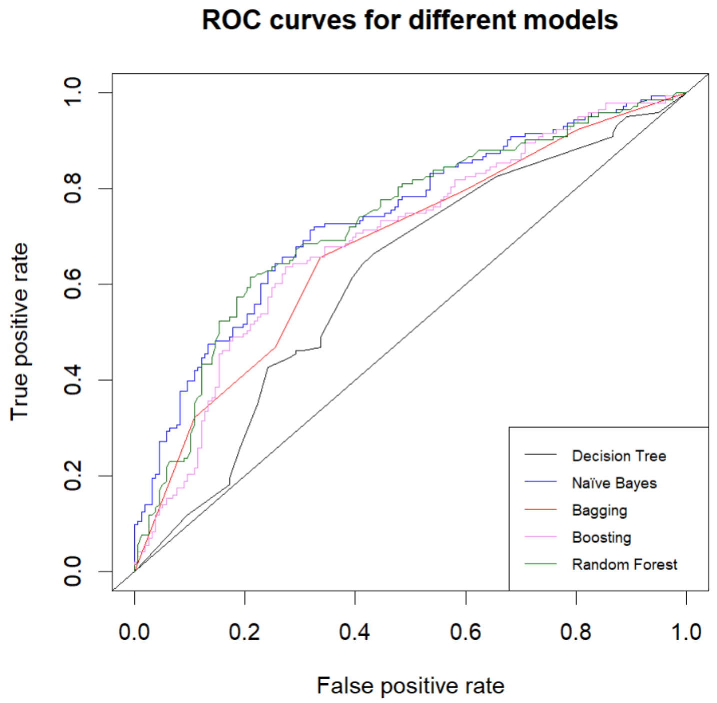
Graph 2.4



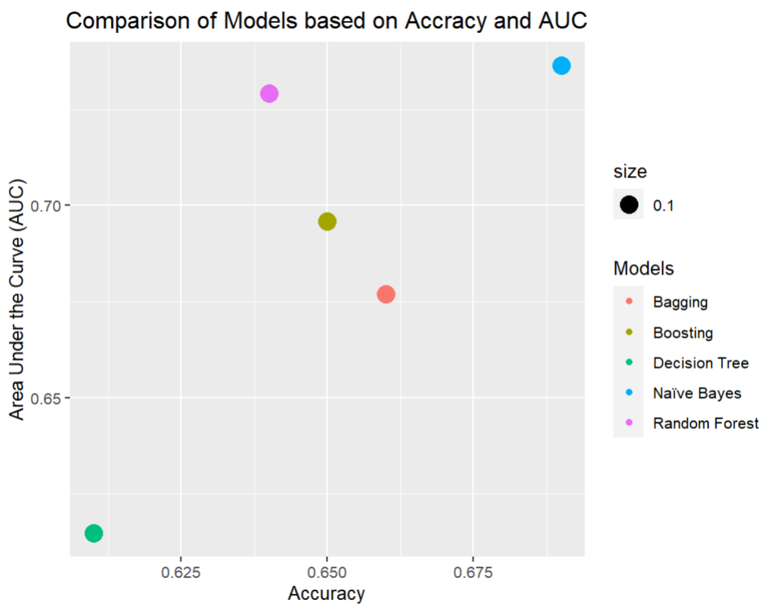
Graph 2.5

By using only attributes "Sunshine", "Temp3pm", "WindGustDir", "WindDir3pm", "WindDir9am" and omitting Null values, we will have a bigger dataset and witness better performance especially for Naïve Bayes and Decision tree (Graph 2.5). As improvements are minor, we now expand the selection of our attributes to top 9 most important attributes (as previously discussed (Graph 2.1)).

Performance of models by using top 9 Variables



Graph 2.6

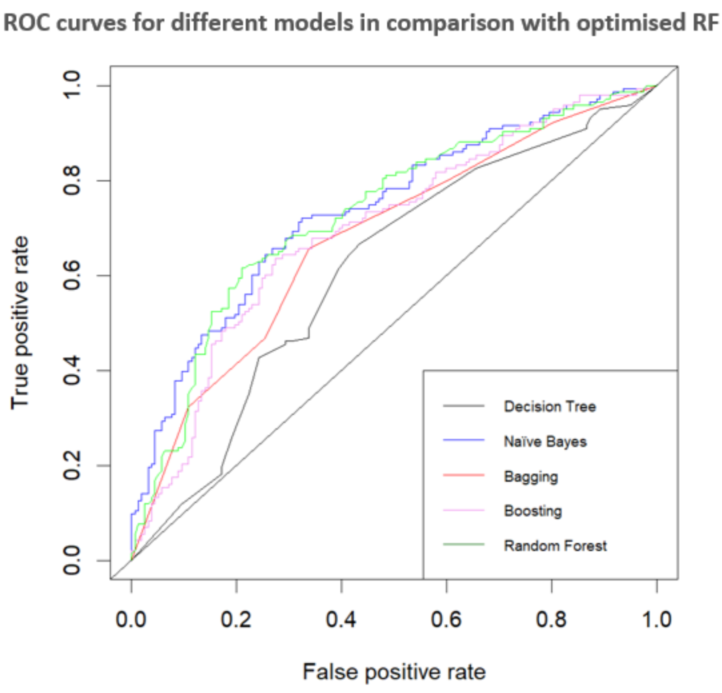


Graph 2.7

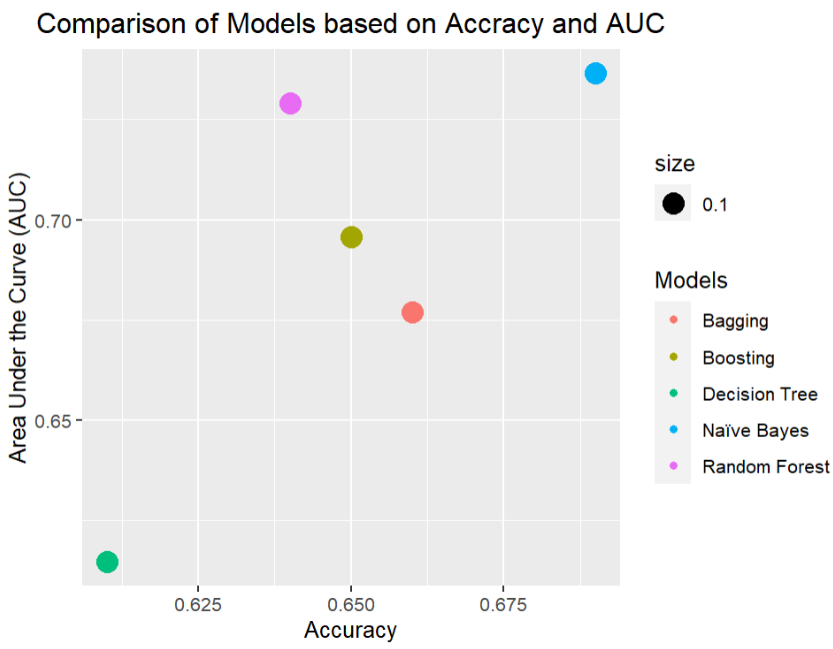
We notice that Naïve Bayes and RF outperform other models by a bigger margin as they are more equipped to handle weak learners (Graph 2.6 & 2.7). Now, we try to optimise parameters of random forest to achieve the best tree-based model.

In this process, we will try to tune two parameters namely *mtry* and *ntree* which have the biggest effect on final accuracy of RF model. Mtry is number of variables chosen at random for each split, this is significant as tree algorithms are greedy and do not always find the best order of variables. On the other hand, ntree is used to find the optimal number of trees as to increase performance as well as avoid overfitting.

There are few methods for finding optimal parameter values but here we use grid search and random search. Still both methods provide values 0.67 and 0.73 for accuracy and AUC respectively.



Graph 2.8

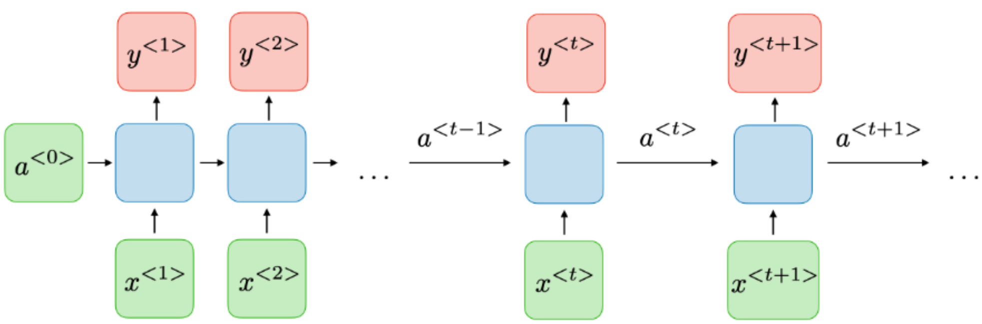


Graph 2.9

Tuning these two parameters did not result in noticeable changes in performance of RF model which is evident from comparing graph 2.7 and 2.9. The development of different classification models in conjunction with parameter optimisation and variable/feature selection has produced some moderately capable classifiers. As we do not have more data, we may try to develop an ANN to see if we achieve better performance. Since ANNs are considered as state-of-the-art classifiers to find complex and non-linear patterns in data, we now strive to develop an ANN model to predict whether tomorrow will be warmer or cooler than today.

An ANN model was developed with 8 hidden layers with numbers of nodes in each layer being 62, 50, 40, 30, 20, 10, 5, and 2 from left to right. The activation function "logistic" was used with learning rate 0.001. Majority of attributes were used except the unimportant ones which we have already mentioned at the beginning of report. Then, the factor variables were represented as one-hot vectors and all the remaining numeric variables were normalised from the original dataset. However, the developed model did not perform better than our latest RF model which can be attributed to rudimentary structure of the ANN which was bounded by project's time constraint.

A Recurrent neural network (RNN) should be developed as weather conditions develop over longer periods of time than a day. RNN's are specifically designed to handle time series data like weather forecasts.



Graph 3.0

(Amidi & Amidi, 2022)

Weather variables for every single day will be passed as inputs (e.g., $x^{<t>}$) and a prediction (i.e., $y^{<t+1>}$) will be made for day t+1 based on the inputs $x^{<1>}$ to $x^{<t+1>}$ and weights $a^{<0>}$ to $a^{<t>}$ (Graph 3.0).

Bibliography:

1. Amidi, A., & Amidi, S. (2022). *CS 230 - Recurrent Neural Networks Cheatsheet*. Stanford.edu. Retrieved 30 May 2022, from https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks.

Appendix:

```
rm(list = ls())

WAUS <- read.csv("WarmerTomorrow2022.csv")

L <- as.data.frame(c(1:49))

set.seed(2916)

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


# store sample in new file

write.csv(WAUS, "WAUS2.csv")

WAUS = read.csv("WAUS2.csv")


head(WAUS)

# Graph 1.0
#get rid of time variables
WAUS = WAUS[,-c(1:4)]

# check for NA's in WarmerTomorrow column
summary(WAUS$WarmerTomorrow)
```

```
# select the numeric columns
library("dplyr")
WAUS_num = select_if(WAUS, is.numeric)


# basic Statistics of variables
summary(WAUS_num[,c(6:21)])


# the proportion of warmer days to cooler days


# get rid of null variables in WarmerTomorrow column
WAUS = WAUS[complete.cases(WAUS$WarmerTomorrow),]

WAUS1 = WAUS

# we place warmer day's records in WAUS1_warmer dataframe
WAUS1_warmer = WAUS1[WAUS1$WarmerTomorrow == 1,]


# we place cool day's records in WAUS1_cool dataframe
WAUS1_cool = WAUS1[WAUS1$WarmerTomorrow == 0,]

# now we calculate the proportion
warm_to_cool = nrow(WAUS1_warmer)/nrow(WAUS1_cool)

warm_to_cool

# Graph 1.1

# install.packages("naniar")

library(naniar)
```

```
vis_miss(WAUS_num[,c(6:21)])

# too many Null values

WAUS2 = WAUS

# get rid of X column
WAUS2$X = NULL

# get rid of rows with NULL values
WAUS2 = WAUS2[complete.cases(WAUS2[,1:24]),]

nrow(WAUS) # 1985

nrow(WAUS2) # 838

data_loss = (nrow(WAUS2)/nrow(WAUS))* 100

data_loss # 42.21662 % of data lost

# we place warmer day's records in WAUS2_warmer dataframe
WAUS2_warmer = WAUS2[WAUS2$WarmerTomorrow == 1,]



# we place cool day's records in WAUS2_cool dataframe
WAUS2_cool = WAUS2[WAUS2$WarmerTomorrow == 0,]

# now we calculate the proportion
warm_to_cool = nrow(WAUS2_warmer)/nrow(WAUS2_cool)

warm_to_cool # 1.084577

# correlation matrix (Graph 1.2)
```

```r
library(tidyverse)


# change character columns to factor

WAUS2$WindDir9am = recode(WAUS2$WindDir9am , N="1", S ="2", ESE ="3" , E ="4" , NNE = "5" , NNW = "6" , NE = "7" , W = "8" , SSE ="9" , ENE ="10", SW= "11" ,NW = "12", SE = "13", WNW ="14", WSW = "15", SSW = "16")

WAUS2$WindDir3pm = recode(WAUS2$WindDir3pm , N="1", S ="2", ESE ="3" , E ="4" , NNE = "5" , NNW = "6" , NE = "7" , W = "8" , SSE ="9" , ENE ="10", SW= "11" ,NW = "12", SE = "13", WNW ="14", WSW = "15", SSW = "16")

WAUS2$WindGustDir = recode(WAUS2$WindGustDir , N="1", S ="2", ESE ="3" , E ="4" , NNE = "5" , NNW = "6" , NE = "7" , W = "8" , SSE ="9" , ENE ="10", SW= "11" ,NW = "12", SE = "13", WNW ="14", WSW = "15", SSW = "16")




# change the values to numeric

WAUS2[, c(1:24)]<- sapply( WAUS2[, c(1:24)], as.numeric)


library(reshape2)


corr = melt(round(cor(WAUS2[complete.cases(WAUS2[,1:24]),]), digits = 3))


corr$Y1 <- cut(corr$value, breaks = c(-Inf, -1, -0.48, 0, 0.48, 1, Inf))


g = ggplot(data = corr, aes(x=Var1, y=Var2, fill=value))

g = g + geom_tile(aes(fill = Y1))+ scale_fill_manual(breaks=c("(-Inf,-1]", "(-1,-0.48]", "(-0.48,0]", "(0,0.48]", "(0.48, 1]", "(1,Inf)"), values = c("red", "pink", "lightblue", "darkblue", "orange", "white")) + theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

g




# Q2

#Make WarmerTomorrow a factor (otherwise model fits a regression tree). also, change

#some other variables to factor.

WAUS2$WindGustDir = as.factor(WAUS2$WindGustDir)

WAUS2$WindDir9am = as.factor(WAUS2$WindDir9am)

WAUS2$WindDir3pm = as.factor(WAUS2$WindDir3pm)
```

```
WAUS2$Location = as.factor(WAUS2$Location)
WAUS2$WarmerTomorrow = as.factor(WAUS2$WarmerTomorrow)

str(WAUS2)

#Q3
# divide data into 70% and 30% training and testing dataset
set.seed(29620716)
train.row = sample(1:nrow(WAUS2), 0.7*nrow(WAUS2))
WAUS.train = WAUS2[train.row,]
WAUS.test = WAUS2[-train.row,]

nrow(WAUS.train) # 586

nrow(WAUS.test) # 252

#Q4
# install.packages("tree")
library(tree)
# install.packages("e1071")
library(e1071)
# install.packages(("ROCR"))
library(ROCR)
# install.packages("rpart")
library(rpart)
library(ggplot2)
library(lattice)
library(caret)



#Q4(graph 1.2)

# Decision tree
tree.fit=tree(WarmerTomorrow~., data=WAUS.train)
```

```
summary(tree.fit)

# now we plot the tree
plot(tree.fit)
text(tree.fit, pretty=1)

#make predictions with model
tpredict = predict(tree.fit, WAUS.test, type = "class")
t1 = table(actual = WAUS.test$WarmerTomorrow , predicted = tpredict)
cat("\n#Decsion Tree Confusion\n")
print(t1)

confusionMatrix(t1)

# Accuracy of Decision Tree
decision.accracy = (t1[1,1]+t1[2,2])/sum(t1)

decision.accracy # 0.547619



# do predictions as probabilities and draw ROC
WAUS.pred.tree = predict(tree.fit , WAUS.test, type = "vector")
WAUSDpred <- prediction( WAUS.pred.tree[,2], WAUS.test$WarmerTomorrow)
WAUSDperf <- performance(WAUSDpred,"tpr","fpr")

# plot the ROC curve
plot(WAUSDperf)
abline(0,1)

# AUC of the curve
tree.auc = performance(WAUSDpred, "auc")
print(as.numeric(tree.auc@y.values)) # 0.581446

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

```
# Naïve Bayes
#fit model
NV.model=naiveBayes(WarmerTomorrow~., data=WAUS.train)

#make predictions with model
NV.predict = predict(NV.model, WAUS.test, type = "class")
NV_table =table(predicted = NV.predict , actual = WAUS.test$WarmerTomorrow)
NV_table

# accuracy of model
confusionMatrix(NV_table)
naive_accuracy = (NV_table[1,1]+NV_table[2,2])/sum(NV_table)

naive_accuracy # 0.6349206

# do predictions as probabilities and draw ROC
WAUS.pred.NV = predict(NV.model , WAUS.test, type = "raw")
WAUSNVpred <- prediction( WAUS.pred.NV[,2], WAUS.test$WarmerTomorrow)

# plot the ROC curve
WAUSNVperf <- performance(WAUSNVpred,"tpr","fpr")
plot(WAUSNVperf, add=TRUE, col = "blue")
abline(0,1)

# AUC of the curve
Naive.auc = performance(WAUSNVpred, "auc")
print(as.numeric(Naive.auc@y.values)) # 0.6520093

#################################################################################

#Bagging
#install.packages("adabag")
library(adabag)
```

```
library(rpart)
# develop the bagging model
WAUS.bag <- bagging(WarmerTomorrow~., data=WAUS.train, mfinal=5)

# make predictions
WAUSpred.bag <- predict.bagging(WAUS.bag, WAUS.test)

# plot the ROC curve
WAUSBagpred <- prediction( WAUSpred.bag$prob[,2], WAUS.test$WarmerTomorrow)
WAUSBagperf <- performance(WAUSBagpred,"tpr","fpr")
plot(WAUSBagperf, add=TRUE, col = "red")
cat("\n#Bagging Confusion\n")
bag.table = WAUSpred.bag$confusion
print(bag.table)

# model accuracy
bag.accuracy = (bag.table[1,1]+bag.table[2,2])/sum(bag.table)

bag.accuracy # 0.5714286

#calculate AUC of the curve
bag.auc = performance(WAUSBagpred, "auc")
print(as.numeric(bag.auc@y.values)) # 0.6229575

##############################################################################

#Boosting
# develop the model
WAUS.Boost <- boosting(WarmerTomorrow ~. , data = WAUS.train, WAUSfinal=10)

#make predictions using the model
WAUSpred.boost <- predict.boosting(WAUS.Boost, newdata=WAUS.test)

# plot the ROC curve
```

```
WAUSBoostpred <- prediction( WAUSpred.boost$prob[,2], WAUS.test$WarmerTomorrow)
WAUSBoostperf <- performance(WAUSBoostpred,"tpr","fpr")
plot(WAUSBoostperf, add=TRUE, col = "violet")

# confusion matrix
cat("\n#Boosting Confusion\n")
boost.table = WAUSpred.boost$confusion
print(boost.table)

# accuracy of model
boosting.accuracy = (boost.table[1,1]+boost.table[2,2])/sum(boost.table)

boosting.accuracy # 0.6150794

#calculate AUC of the curve
boosting.auc = performance(WAUSBoostpred, "auc")
print(as.numeric(boosting.auc@y.values)) # 0.6728913

##############################################################################

# Random Forrest
library(randomForest)

# develop the model
WAUS.rf <- randomForest(WarmerTomorrow~. , data = WAUS.train, na.action = na.exclude)

# make prediction with the model
WAUSpredrf <- predict(WAUS.rf, WAUS.test)

# confusion matrix
t3=table(Predicted_Class = WAUSpredrf, Actual_Class = WAUS.test$WarmerTomorrow)
cat("\n#Random Forest Confusion\n")
print(t3)
```

```
# make predictions with model
WAUSpred.rf <- predict(WAUS.rf, WAUS.test, type="prob")

summary(WAUSpred.rf)

#plot the curve
WAUSFpred <- prediction( WAUSpred.rf[,2], WAUS.test$WarmerTomorrow)
WAUSFperf <- performance(WAUSFpred,"tpr","fpr")
plot(WAUSFperf, add=TRUE, col = "darkgreen")

# accuracy of model
rf.accuracy = (t3[1,1]+t3[2,2])/sum(t3)

rf.accuracy # 0.6071429

#calculate AUC of the curve
rf.auc = performance(WAUSFpred, "auc")
print(as.numeric(rf.auc@y.values)) # 0.6844679



# Add title and legend to graph
legend(x= "bottomright", y=0.9, legend=c("Decision Tree", "Naïve Bayes", "Bagging" , "Boosting" , "Random Forest"),
    col=c("black", "blue", "red", "violet", "darkgreen"), lty=1, cex=0.65)
title(main = "ROC curves for different models")
##########################################################################

#Q7 (graph 1.3)
# put Accuracy and AUC of each model into a table
Accuracy = c(decision.accracy,naive_accuracy,bag.accuracy, boosting.accuracy, rf.accuracy)
AUC = c(as.numeric(tree.auc@y.values), as.numeric(Naive.auc@y.values), as.numeric(bag.auc@y.values), as.numeric(boosting.auc@y.values), as.numeric(rf.auc@y.values) )
Models = c("Decision Tree", "Naïve Bayes", "Bagging" , "Boosting" , "Random Forest")

# combine the columns
```

```
com_table = cbind(Models, Accuracy, AUC)

com_table = as.data.frame(com_table)

com_table$Accuracy = as.numeric(com_table$Accuracy)

com_table$AUC = as.numeric(com_table$AUC)

str(com_table)


# now we plot

qplot(Accuracy, AUC,data = com_table, xlab = "Accuracy", ylab="Area Under the Curve (AUC)", main = "Comparison of Models based on Accracy and AUC", color = Models, cex=0.1) + theme(plot.title = element_text(hjust = 0.5))


##########################################################################################




# Q8

#cross validation and pruning of Decision tree


# finding the best number of splits or terminal nodes

test.tree_fit=cv.tree(tree.fit, FUN=prune.misclass)

print(test.tree_fit)


# prune the tree to 3 terminal nodes


prune.tree_fit = prune.misclass(tree.fit, best=3)

print(summary(prune.tree_fit))

plot(prune.tree_fit)

text(prune.tree_fit, pretty=0)


#test accuracy after pruning

tp.predict = predict(prune.tree_fit, WAUS.test, type = "class")

t2=table(predicted = tp.predict, actual = WAUS.test$WarmerTomorrow)

print(t2)


# accuracy of pruned decision tree
```

```
confusionMatrix(t2)



# compute predictions and calculate the AUC


WAUS.pred.tree2 = predict(prune.tree_fit , WAUS.test, type = "vector")
WAUSDpred2 <- prediction( WAUS.pred.tree2[,2], WAUS.test$WarmerTomorrow)
WAUSDperf <- performance(WAUSDpred2,"tpr","fpr")


# AUC of the curve
tree.auc2 = performance(WAUSDpred2, "auc")
print(as.numeric(tree.auc2@y.values))





# Bagging most important variables
cat("\n#Baging Attribute Importance\n")


# sort variables based on Mean Decrease in Gini
print(sort(WAUS.bag$importance), ascending = TRUE)
# put it into a dataframe
var_bag = sort(WAUS.bag$importance)
var_bag = as.data.frame(var_bag)


#Add a column with rownames(i.e., Attribute names)
var_bag$predictors= row.names(var_bag)
# change to factor to save order data points appaer in dataframe
var_bag$predictors <- factor(var_bag$predictors, levels = var_bag$predictors[order(var_bag$var_bag )])


# plot the var_bag
qplot(predictors,var_bag, data = var_bag, xlab = "Attributes", ylab="Mean decrease in Gini", main = "Top 5 Attributes with Highest mean decrease in Gini (Bagging)", color = predictors) + theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+ theme(plot.title = element_text(hjust = 0.5))+guides(colour=guide_legend(nrow=10))+ theme(legend.position = "none")
```

```
# top 5 most important variables for Boosting

top_5_bag = var_bag$predictors[19:23]

top_5_bag


# Boosting most important variables

cat("\n# Boosting Attribute Importance\n")

# sort variables based on Mean Decrease in Gini

print(sort(WAUS.Boost$importance), ascending = TRUE)


# put it into a dataframe

var_Boost = sort(WAUS.Boost$importance)

var_Boost = as.data.frame(var_Boost)


#Add a column with rownames(i.e., Attribute names)

var_Boost$predictors= row.names(var_Boost)

# change to factor to save order data points appear in dataframe

var_Boost$predictors <- factor(var_Boost$predictors, levels = var_Boost$predictors[order(var_Boost$var_Boost )])



# plot the var_Boost

qplot(predictors,var_Boost, data = var_Boost, xlab = "Attributes", ylab="Mean decrease in Gini", main = "Top 5 Attributes with Highest mean decrease in Gini (Boosting)", color = predictors) + theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+ theme(plot.title = element_text(hjust = 0.5))+ theme(legend.position = "none")


# top 5 most important variables for Boosting

top_5_boost = var_Boost$predictors[19:23]

top_5_boost


# Random Forrest most important variables

cat("\n#Random Forrest Attribute Importance\n")


#Random Forrest most important variables

cat("\n#Random Forrest Attribute Importance\n")
```

```
library(data.table)

rf.importance = as.data.frame(WAUS.rf$importance)

setDT(rf.importance, keep.rownames = TRUE)[]

colnames(rf.importance) = c("predictors","MeanDecreaseGini")


# order based on importance

rf.importance = rf.importance[order(rf.importance$MeanDecreaseGini), ]

rf.importance


# change to factor to save order data points appaer in dataframe

rf.importance$predictors <- factor(rf.importance$predictors, levels = rf.importance$predictors[order(rf.importance$MeanDecreaseGini )])


# plot the rf.importance

qplot(predictors,MeanDecreaseGini, data = rf.importance, xlab = "Attributes", ylab="Mean decrease in Gini", main = "Top 5 Attributes with Highest mean decrease in Gini (RF)", color = predictors) + theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+ theme(plot.title = element_text(hjust = 0.5))+ theme(legend.position = "none")


# top 5 most important variables for RF

top_5_RF = rf.importance$predictors[19:23]

top_5_RF


# Naive Bayes most important variables

cat("\n#Naive Bayes Attribute Importance\n")


Grid = data.frame(usekernel=TRUE,laplace = 0,adjust=1)


Naive_Bayes_2 = train(WarmerTomorrow ~ .,data=WAUS.train,method="naive_bayes",

            trControl=trainControl(method="none"),

            tuneGrid=Grid)


# put important variables into a dataframe

imp_naive = as.data.frame(varImp(Naive_Bayes_2)$importance)
```

```
# X1 column is duplicate of X0 (omit it)

imp_naive$X1 = NULL


# add rownames as a column

imp_naive$names <- rownames(imp_naive)


#change column names

colnames(imp_naive) = c("var_Naive","predictors")


# sort variables based on Mean Decrease in Gini

print(sort(imp_naive$predictors), ascending = TRUE)


# change to factor to save order data points appear in dataframe

imp_naive$predictors <- factor(imp_naive$predictors, levels = imp_naive$predictors[order(imp_naive$var_Naive)])


# plot the imp_naive

qplot(predictors,var_Naive, data = imp_naive, xlab = "Attributes", ylab="ROC curve variable importance", main = "Top 5 Attributes with Highest ROC (Naive Bayes)", color = predictors) + theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+ theme(plot.title = element_text(hjust = 0.5))+guides(colour=guide_legend(nrow=10))+ theme(legend.position = "none")


# top 5 most important variables for NV

top_5_NV = sort(imp_naive$predictors, ascending = TRUE)[19:23]

top_5_NV




# top 2 most important variables for pruned decision tree

top_2_dt = c("WindDir9am", "Sunshine")


# we combine all the top attributes

dis_var = c(top_2_dt, top_5_bag, top_5_boost, top_5_NV, top_5_RF)


dist_impvariables = as.data.frame(table(c(top_5_bag, top_5_boost, top_5_NV, top_5_RF, as.factor(top_2_dt))))
```

```
dist_impvariables = dist_impvariables[dist_impvariables$Freq > 0,]

# change column names

colnames(dist_impvariables) = c("Predictor", "Occurences")

dist_impvariables

# change to factor to save order data points appear in dataframe
dist_impvariables$Predictor <- factor(dist_impvariables$Predictor, levels = dist_impvariables$Predictor[order(dist_impvariables$Occurences)])

# plot the dist_impvariables
qplot(Predictor,Occurences, data = dist_impvariables, xlab = "Attributes", ylab="# of occurances", main = "The distribution of attributes used by differnt models developed", color = Predictor) + theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+ theme(plot.title = element_text(hjust = 0.5))+ theme(legend.position = "none")

#Q9
# Decision tree simple enough for humans
# we use the most important variables fro previous question
indices <- c(4)
top_9 <- as.character(dist_impvariables$Predictor[-indices])
print(top_9)

top_9_var_train = WAUS.train[,c(top_9)]
str(top_9_var_train)

# Q9 (we use the pruned decision tree already developed in q8)

plot(prune.tree_fit)
text(prune.tree_fit, pretty=0)
```

```
#test accuracy after pruning
tp.predict = predict(prune.tree_fit, WAUS.test, type = "class")
t2 = table(predicted = tp.predict, actual = WAUS.test$WarmerTomorrow)
print(t2)

# accuracy of pruned decision tree

confusionMatrix(t2) # Accuracy : 0.6349



# compute predictions and calculate the AUC

WAUS.pred.tree2 = predict(prune.tree_fit , WAUS.test, type = "vector")
WAUSDpred2 <- prediction( WAUS.pred.tree2[,2], WAUS.test$WarmerTomorrow)
WAUSDperf <- performance(WAUSDpred2,"tpr","fpr")

# AUC of the curve
tree.auc2 = performance(WAUSDpred2, "auc")
print(as.numeric(tree.auc2@y.values)) # 0.6231153

# we add the row (Models: human-friendly, Accuracy:0.6349, AUC: 0.6231153 to com_table in q7

com_table[nrow(com_table) + 1,] = c("human-friendly", 0.6349, 0.6231153)

str(com_table)

# change Accuracy and AUC columns to numeric
com_table$Accuracy = as.numeric(com_table$Accuracy)
com_table$AUC = as.numeric(com_table$AUC)

# we plot again for comparison
qplot(Accuracy, AUC, data = com_table, xlab = "Accuracy", ylab="Area Under the Curve (AUC)", main =
"Comparison of Models based on Accracy and AUC", color = Models, cex=0.1) + theme(plot.title =
element_text(hjust = 0.5))
```

```
prune_sizes = test.tree_fit$size

prune_misclas = test.tree_fit$dev

prune_cost = test.tree_fit$k


qplot(prune_sizes, prune_misclas, size = prune_cost)




# now we plot

qplot(as.factor(prune_sizes), prune_misclas, xlab = "# of terminal nodes", ylab="Classification Error rate", main = "Comparison of trees of different sizes", size = as.factor(prune_sizes), color = prune_cost, cex=0.1) + theme(plot.title = element_text(hjust = 0.5))  + labs(size="Number of nodes", color="cost-complexity (K)")



# Create model with default parameters

control <- trainControl(method="repeatedcv", number=10, repeats=3)

seed <- 7

metric <- "Accuracy"

set.seed(seed)

mtry <- sqrt(ncol(WAUS.train.3)-1)

tunegrid <- expand.grid(.mtry=mtry)

rf_default <- train(WarmerTomorrow ~., data=WAUS.train.3, method="rf", metric=metric, tuneGrid=tunegrid, trControl=control)

print(rf_default)


# Random Search

control <- trainControl(method="repeatedcv", number=10, repeats=3, search="random")

set.seed(seed)


rf_random <- train(WarmerTomorrow ~., data=WAUS.train.3, method="rf", metric=metric, tuneLength=15, trControl=control)

print(rf_random)

plot(rf_random)
```

```
# grid search
control <- trainControl(method="repeatedcv", number=10, repeats=3, search="grid")
set.seed(seed)
tunegrid <- expand.grid(.mtry=c(1:15))
rf_gridsearch <- train(WarmerTomorrow ~., data=WAUS.train.3, method="rf", metric=metric, tuneGrid=tunegrid, trControl=control)
print(rf_gridsearch)
plot(rf_gridsearch)

# make predictions
rf.prediction <-predict(rf_random, WAUS.test.3)

#Accuracy
confusionMatrix(rf.prediction, WAUS.test.3$WarmerTomorrow)

#plot the curve
WAUSFpred <- prediction( WAUSpred.rf[,2], WAUS.test.3$WarmerTomorrow)
WAUSFperf <- performance(WAUSFpred,"tpr","fpr")
plot(WAUSFperf, add = TRUE, col = "green")
abline(0,1)

#calculate AUC of the curve
rf.auc = performance(WAUSFpred, "auc")
print(as.numeric(rf.auc@y.values))

# Add title
title(main = "ROC curve for final RF")

# now we try to add to our training data by using only the

# MaxTemp    Cloud9am    Humidity3pm    Sunshine
# Temp3pm    Evaporation WindGustDir WindDir3pm  WindDir9am
```

```
WAUS3 = WAUS

# get rid of X column
WAUS3$X = NULL

WAUS3 = WAUS3[, c("MaxTemp", "Cloud9am", "Humidity3pm", "Sunshine", "Temp3pm", "Evaporation", "WindGustDir", "WindDir3pm", "WindDir9am", "WarmerTomorrow")]

# number of data points before removing NULL values

nrow(WAUS3)

# get rid of rows with NULL values
WAUS3 = WAUS3[complete.cases(WAUS3[,]),]

# compare the change in number of rows
nrow(WAUS3)
nrow(WAUS2)

# recode factor variables
WAUS2$WindDir9am = recode(WAUS2$WindDir9am , N="1", S ="2", ESE ="3" , E ="4" , NNE = "5" , NNW = "6" , NE = "7" , W = "8" , SSE ="9" , ENE ="10", SW= "11" ,NW = "12", SE = "13", WNW ="14", WSW = "15", SSW = "16")
WAUS2$WindDir3pm = recode(WAUS2$WindDir3pm , N="1", S ="2", ESE ="3" , E ="4" , NNE = "5" , NNW = "6" , NE = "7" , W = "8" , SSE ="9" , ENE ="10", SW= "11" ,NW = "12", SE = "13", WNW ="14", WSW = "15", SSW = "16")
WAUS2$WindGustDir = recode(WAUS2$WindGustDir , N="1", S ="2", ESE ="3" , E ="4" , NNE = "5" , NNW = "6" , NE = "7" , W = "8" , SSE ="9" , ENE ="10", SW= "11" ,NW = "12", SE = "13", WNW ="14", WSW = "15", SSW = "16")

#some other variables to factor.
WAUS3$WindGustDir = as.factor(WAUS3$WindGustDir)
WAUS3$WindDir9am = as.factor(WAUS3$WindDir9am)
WAUS3$WindDir3pm = as.factor(WAUS3$WindDir3pm)
```

```
WAUS3$WarmerTomorrow = as.factor(WAUS3$WarmerTomorrow)

str(WAUS3)

# convert Cloud9am and Humidity3pm
# WAUS3$Cloud9am = as.numeric(WAUS3$Cloud9am)

WAUS3$Humidity3pm = as.numeric(WAUS3$Humidity3pm)



# install.packages("tree")
library(tree)
# install.packages("e1071")
library(e1071)
# install.packages(("ROCR"))
library(ROCR)
# install.packages("rpart")
library(rpart)
library(ggplot2)
library(lattice)
library(caret)

# divide data into 70% and 30% training and testing dataset
set.seed(29620716)
train.row.3 = sample(1:nrow(WAUS3), 0.7*nrow(WAUS3))
WAUS.train.3 = WAUS3[train.row.3,]
WAUS.test.3 = WAUS3[-train.row.3,]

nrow(WAUS.train.3) # 774

nrow(WAUS.test.3) # 260
```

```
# Decision tree
tree.fit=tree(WarmerTomorrow~., data=WAUS.train.3)
summary(tree.fit)

# now we plot the tree
plot(tree.fit)
text(tree.fit, pretty=1)

#make predictions with model
tpredict = predict(tree.fit, WAUS.test.3, type = "class")
t1 = table(actual = WAUS.test.3$WarmerTomorrow , predicted = tpredict)
cat("\n#Decsion Tree Confusion\n")
print(t1)

confusionMatrix(t1)

# Accuracy of Decision Tree
decision.accracy = (t1[1,1]+t1[2,2])/sum(t1)

decision.accracy # 0.547619



# do predictions as probabilities and draw ROC
WAUS.pred.tree = predict(tree.fit , WAUS.test.3, type = "vector")
WAUSDpred <- prediction( WAUS.pred.tree[,2], WAUS.test.3$WarmerTomorrow)
WAUSDperf <- performance(WAUSDpred,"tpr","fpr")

# plot the ROC curve
plot(WAUSDperf)
abline(0,1)

# AUC of the curve
tree.auc = performance(WAUSDpred, "auc")
print(as.numeric(tree.auc@y.values)) # 0.581446
```

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

# Naïve Bayes
#fit model
NV.model=naiveBayes(WarmerTomorrow~., data=WAUS.train.3)

#make predictions with model
NV.predict = predict(NV.model, WAUS.test.3, type = "class")
NV_table =table(predicted = NV.predict , actual = WAUS.test.3$WarmerTomorrow)
NV_table

# accuracy of model
confusionMatrix(NV_table)
naive_accuracy = (NV_table[1,1]+NV_table[2,2])/sum(NV_table)

naive_accuracy # 0.6349206

# do predictions as probabilities and draw ROC
WAUS.pred.NV = predict(NV.model , WAUS.test.3, type = "raw")
WAUSNVpred <- prediction( WAUS.pred.NV[,2], WAUS.test.3$WarmerTomorrow)

# plot the ROC curve
WAUSNVperf <- performance(WAUSNVpred,"tpr","fpr")
plot(WAUSNVperf, add=TRUE, col = "blue")
abline(0,1)

# AUC of the curve
Naive.auc = performance(WAUSNVpred, "auc")
print(as.numeric(Naive.auc@y.values)) # 0.6520093

##############################################################################

#Bagging
```

```
#install.packages("adabag")
library(adabag)
library(rpart)
# develop the bagging model
WAUS.bag <- bagging(WarmerTomorrow~., data=WAUS.train.3, mfinal=5)

# make predictions
WAUSpred.bag <- predict.bagging(WAUS.bag, WAUS.test.3)

# plot the ROC curve
WAUSBagpred <- prediction( WAUSpred.bag$prob[,2], WAUS.test.3$WarmerTomorrow)
WAUSBagperf <- performance(WAUSBagpred,"tpr","fpr")
plot(WAUSBagperf, add=TRUE, col = "red")
cat("\n#Bagging Confusion\n")
bag.table = WAUSpred.bag$confusion
print(bag.table)

# model accuracy
bag.accuracy = (bag.table[1,1]+bag.table[2,2])/sum(bag.table)

bag.accuracy # 0.5714286

#calculate AUC of the curve
bag.auc = performance(WAUSBagpred, "auc")
print(as.numeric(bag.auc@y.values)) # 0.6229575

################################################################################

#Boosting
# develop the model
WAUS.Boost <- boosting(WarmerTomorrow ~. , data = WAUS.train.3, WAUSfinal=10)

#make predictions using the model
WAUSpred.boost <- predict.boosting(WAUS.Boost, newdata=WAUS.test.3)
```

```
# plot the ROC curve
WAUSBoostpred <- prediction( WAUSpred.boost$prob[,2], WAUS.test.3$WarmerTomorrow)
WAUSBoostperf <- performance(WAUSBoostpred,"tpr","fpr")
plot(WAUSBoostperf, add=TRUE, col = "violet")

# confusion matrix
cat("\n#Boosting Confusion\n")
boost.table = WAUSpred.boost$confusion
print(boost.table)

# accuracy of model
boosting.accuracy = (boost.table[1,1]+boost.table[2,2])/sum(boost.table)

boosting.accuracy # 0.6150794

#calculate AUC of the curve
boosting.auc = performance(WAUSBoostpred, "auc")
print(as.numeric(boosting.auc@y.values)) # 0.6728913




# Add title and legend to graph
legend(x= "bottomright", y=0.9, legend=c("Decision Tree", "Naïve Bayes", "Bagging" , "Boosting" , "Random
Forest"),
       col=c("black", "blue", "red", "violet", "darkgreen"), lty=1, cex=0.65)
title(main = "ROC curves for different models /n in comparison with optimised RF")



# put Accuracy and AUC of each model into a table
Accuracy = c(decision.accracy,naive_accuracy,bag.accuracy, boosting.accuracy, rf.accuracy)
AUC  =  c(as.numeric(tree.auc@y.values),  as.numeric(Naive.auc@y.values),  as.numeric(bag.auc@y.values),
as.numeric(boosting.auc@y.values), as.numeric(rf.auc@y.values) )
Models = c("Decision Tree", "Naïve Bayes", "Bagging" , "Boosting" , "Random Forest")
```

```
# combine the columns

com_table = cbind(Models, Accuracy, AUC)

com_table = as.data.frame(com_table)

com_table$Accuracy = as.numeric(com_table$Accuracy)

com_table$AUC = as.numeric(com_table$AUC)

str(com_table)


# now we plot

qplot(Accuracy, AUC,data = com_table, xlab = "Accuracy", ylab="Area Under the Curve (AUC)", main = "Comparison of Models based on Accracy and AUC", color = Models, cex=0.1) + theme(plot.title = element_text(hjust = 0.5))



# Q11


# trying to use all the variables in development of neuralnet

# finding numeric and factor columns

WAUS2_num = select_if(WAUS2, is.numeric)

str(WAUS2_num)



WAUS2_fac = select_if(WAUS2, is.factor)

str(WAUS2_fac)


# make dummy variables from factor variables


WAUS.mm = model.matrix(~WindGustDir+WindDir9am+WindDir3pm+WarmerTomorrow, data=WAUS2)

WAUS.mm[,]<- sapply( WAUS.mm[, ], as.character )

str(WAUS.mm[,2])


WAUS.mm[,]<- sapply( WAUS.mm[, ], as.character )

str(WAUS.mm[,2])



WAUS.mm = as.data.frame(WAUS.mm)
```

```
WAUS.mm[,]<- sapply( WAUS.mm[, ], as.numeric )


# make a copy
WAUS4 = WAUS2


# get rid of factor variables and unimportant variables
WAUS4 = WAUS4[ ,c(-1:-4)]


# names of factor columns
fact_col_names = names(Filter(is.factor, WAUS4))

WAUS4[ ,c(as.character(fact_col_names))] = list(NULL)




#noramlise the columns
min_max_norm <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}


#apply Min-Max normalization to first fifteen columns in dataset
WAUS4 <- as.data.frame(sapply(WAUS4[1:16], min_max_norm))



# we combine numeric and dummy columns
# WAUS4$WarmerTomorrow = WAUS2$WarmerTomorrow
WAUS.combined = cbind(WAUS4,WAUS.mm)


# convert all columns to numeric
WAUS.combined[,]<- sapply( WAUS.combined[, ], as.numeric )


# check dataset
str(WAUS.combined)
```

```
# creat test and training data
train.row = sample(1:nrow(WAUS.combined), 0.8*nrow(WAUS.combined))
nn.train = WAUS.combined[train.row,]
nn.test = WAUS.combined[-train.row,]

# install.packages("neuralnet")
library(neuralnet)

# formula for ANN
var.nn <- as.formula(paste0('WarmerTomorrow1 ~ ',
                     paste(names(nn.train[!names(nn.train) %in% c('WarmerTomorrow1','(Intercept)')]),
                       collapse = ' + ')))

# develop ANN model
WAUS.nn = neuralnet(var.nn, nn.train, hidden = c(62, 50, 40, 30, 20, 10, 5, 2, 1),
              act.fct = "logistic", linear.output = FALSE,
              lifesign = "full", algorithm = 'backprop', learningrate=0.001, rep = 1)




# plot ANN
plot(WAUS.nn)



# evaluate the performance
WAUSnn.pred = compute(WAUS.nn, nn.test[1:62])

# now round these down to integers
WAUSnn.pred = as.data.frame(round(WAUSnn.pred $net.result,0))
# confusion matrix
p = nn.test$WarmerTomorrow1
# p = p-1
T.nn = table(observed = p , predicted = WAUSnn.pred$V1)
```

```
T.nn

# accuracy of model
decision.accracy = (T.nn[1,1]+T.nn[2,2])/sum(T.nn)
decision.accracy

# Cross-validation of neuralnet
# # Set seed for reproducibility purposes
set.seed(80)
# 10 fold cross validation
k <- 10
# Results from cv
outs <- NULL
# Train test split proportions
proportion <- 0.95 # Set to 0.995 for LOOCV

# Crossvalidate, go!
for(i in 1:k)
{
 index <- sample(1:nrow(WAUS.combined), round(proportion*nrow(WAUS.combined)))
 train_cv <- WAUS.combined[index, ]
 test_cv <- WAUS.combined[-index, ]
 nn_cv <- neuralnet(var.nn,
            data = train_cv,
            hidden = c(13, 10, 3),
            act.fct = "logistic",
            linear.output = FALSE)

 WAUSnn.pred = compute(nn_cv, test_cv[1:52])


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

```
  nrow(WAUSnn.pred)


  # confusion matrix
  p = test_cv$WarmerTomorrow
  # p = p-1
  T.nn = table(observed = p , predicted = WAUSnn.pred$V1)


  T.nn


  # accuracy of model
  decision.accracy = (T.nn[1,1]+T.nn[2,2])/sum(T.nn)



  outs[i] <- decision.accracy
}

mean(outs)
```