import the needed lib

read file

rm(list = ls())   
WAUS <- read.csv("C:/Users/sjsa3/Desktop/Shared\_with\_Mac/year2\_sem1/FIT3152/Assignment 2/data.csv",  
 stringsAsFactors = T)  
WAUS <- WAUS %>% filter(CloudTomorrow != "NA") #remove the rows in which the value of CloudTmr is NA  
L <- as.data.frame(c(1:49))   
set.seed(31084222) # Your Student ID is the random seed   
L <- L[sample(nrow(L), 10, replace = FALSE),] # sample 10 locations ====  
WAUS <- WAUS[(WAUS$Location %in% L),]   
WAUS <- WAUS[sample(nrow(WAUS), 2000, replace = FALSE),] # sample 2000 rows   
WAUS$CloudTomorrow = as.factor(WAUS$CloudTomorrow)  
WAUS$Location = as.factor(WAUS$Location)

============================================Explore data Since we are predicting the cloudiness, there is no use of Location, Date, thus removing them

#date format  
WAUS$Date = ""  
WAUS$Day = as.character(WAUS$Day)  
WAUS$Month = as.character(WAUS$Month)  
WAUS$Year = as.character(WAUS$Year)  
WAUS$Date =paste(WAUS$Year,"-" ,WAUS$Month,"-", WAUS$Day)  
WAUS$Date = ymd(WAUS$Date)

## Warning: 31 failed to parse.

WAUS= WAUS %>% arrange(Date)  
WAUS <- WAUS %>% select(-Day,-Year,-Month,-Date)

summary <- skim(WAUS)  
summary[,c(1:9,15)]

Data summary

|  |  |
| --- | --- |
| Name | WAUS |
| Number of rows | 2000 |
| Number of columns | 20 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 6 |
| numeric | 14 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: factor**

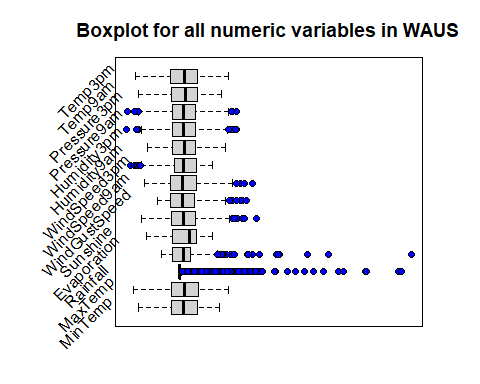
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| Location | 0 | 1.00 | FALSE | 10 | 14: 224, 40: 214, 29: 210, 10: 205 |
| WindGustDir | 100 | 0.95 | FALSE | 16 | ENE: 166, WNW: 164, E: 159, SSW: 133 |
| WindDir9am | 188 | 0.91 | FALSE | 16 | N: 144, E: 142, SE: 136, SW: 133 |
| WindDir3pm | 74 | 0.96 | FALSE | 16 | ENE: 177, WNW: 145, WSW: 137, NW: 131 |
| RainToday | 73 | 0.96 | FALSE | 2 | No: 1515, Yes: 412 |
| CloudTomorrow | 0 | 1.00 | FALSE | 2 | 0: 1387, 1: 613 |

**Variable type: numeric**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | mean | sd | hist |
| MinTemp | 33 | 0.98 | 13.39 | 7.30 | ▁▅▇▇▃ |
| MaxTemp | 24 | 0.99 | 25.20 | 6.96 | ▁▅▇▇▁ |
| Rainfall | 78 | 0.96 | 3.03 | 10.80 | ▇▁▁▁▁ |
| Evaporation | 853 | 0.57 | 5.88 | 4.29 | ▇▁▁▁▁ |
| Sunshine | 787 | 0.61 | 8.13 | 3.63 | ▂▃▃▇▃ |
| WindGustSpeed | 91 | 0.95 | 39.77 | 11.86 | ▁▇▅▁▁ |
| WindSpeed9am | 72 | 0.96 | 13.94 | 8.56 | ▇▇▅▂▁ |
| WindSpeed3pm | 64 | 0.97 | 19.69 | 8.34 | ▂▇▆▁▁ |
| Humidity9am | 44 | 0.98 | 68.43 | 18.75 | ▁▂▆▇▆ |
| Humidity3pm | 43 | 0.98 | 49.57 | 20.49 | ▃▆▇▅▂ |
| Pressure9am | 244 | 0.88 | 1017.26 | 6.64 | ▁▃▇▅▁ |
| Pressure3pm | 237 | 0.88 | 1014.56 | 6.70 | ▁▃▇▅▁ |
| Temp9am | 26 | 0.99 | 18.74 | 7.00 | ▁▆▇▇▃ |
| Temp3pm | 30 | 0.98 | 23.73 | 6.84 | ▁▆▇▇▁ |

kable = knitr::kable(summary[,c(1:9,15)])

par(mar=c(3,6,3,3))  
numScaleWAUS =as.data.frame(scale(select\_if(WAUS,is.numeric)))  
lablist.x<-as.vector(names(numScaleWAUS))  
boxplot(numScaleWAUS  
 , horizontal=T,yaxt = "n",xaxt = "n",frame=T,outbg = "blue",outpch = 21# Outliers symbol  
,main= "Boxplot for all numeric variables in WAUS" )  
#stripchart(numScaleWAUS, add = TRUE, col = "blue")  
text(y = seq(1, 14, by=1), par("usr")[1] -2, labels = lablist.x, srt = 45, pos = 1, xpd = TRUE)



============================================ preprocessing clean the data

impute the data

#drop all categorical values becuase they cannot be found  
WAUS = WAUS %>% drop\_na()

set-seed

set.seed(31084222) #Student ID as random seed   
 train.row = sample(1:nrow(WAUS), 0.7\*nrow(WAUS))   
 WAUS.train = WAUS[train.row,]   
 WAUS.test = WAUS[-train.row,]

data pre-processing ============================================ Naive bases

#set.seed(31084222)  
NB.Model=naiveBayes(CloudTomorrow~.,data=WAUS.train)  
NB.predict=predict(NB.Model,WAUS.test, type = "class")  
#computing confusion matrix  
neededPerfomanceIndex = c(1,2,5,6,12)  
  
cm\_NB = confusionMatrix(table(observed = WAUS.test$CloudTomorrow , predicted = NB.predict))  
pf.NB =as.data.frame( cm\_NB$byClass[neededPerfomanceIndex])%>% rbind(cm\_NB$overall[1])  
pf.NB = pf.NB %>% rownames\_to\_column()  
cm\_NB = as.data.frame(cm\_NB$table)  
cm\_NB$model = "NB"

============================================ Decision tree

train.tree <- tree(CloudTomorrow ~ ., data=WAUS.train)  
  
  
#To evalaute the accuracy of the tree we can look at the confusion matrix for the Training data.   
#obtaining class predictions  
tree.classTrain <- predict(train.tree, WAUS.test, type="class")  
#confusion matrix   
cm\_dt = confusionMatrix(table(observed =WAUS.test$CloudTomorrow , predicted = tree.classTrain))  
pf.dt =as.data.frame( cm\_dt$byClass[neededPerfomanceIndex])%>% rbind(cm\_dt$overall[1])  
pf.dt = pf.dt %>% rownames\_to\_column()  
cm\_dt = as.data.frame(cm\_dt$table)   
cm\_dt$model = "DT"

============================================ Bagging

bag.train= bagging(CloudTomorrow ~. ,data = WAUS.train, mfinal = 6)  
#test  
bagging.predict = predict.bagging(bag.train, newdata = WAUS.test,  
 type = "class")  
  
# confusion matrix  
#computing confusion matrix  
cm\_bag = confusionMatrix(table(observed =WAUS.test$CloudTomorrow , predicted = bagging.predict$class))  
pf.bag =as.data.frame( cm\_bag$byClass[neededPerfomanceIndex])%>% rbind(cm\_bag$overall[1])  
pf.bag = pf.bag %>% rownames\_to\_column()  
cm\_bag = as.data.frame(cm\_bag$table)  
cm\_bag$model = "Bagging"  
pf.bag

## rowname cm\_bag$byClass[neededPerfomanceIndex]  
## 1 Sensitivity 0.6967742  
## 2 Specificity 0.6933333  
## 3 Precision 0.8244275  
## 4 Recall 0.6967742  
## 5 <NA> NA  
## 6 6 0.6956522

============================================ Boosting

boosting.train= boosting(CloudTomorrow ~. ,data = WAUS.train, mfinal = 7)  
  
#test  
boosting.predict = predict.boosting(boosting.train, newdata = WAUS.test, type = "class")  
  
# confusion matrix  
cm\_boosting =confusionMatrix(table(observed =WAUS.test$CloudTomorrow , predicted = boosting.predict$class))  
pf.boosting =as.data.frame( cm\_boosting$byClass[neededPerfomanceIndex])%>% rbind(cm\_boosting$overall[1])  
pf.boosting = pf.boosting %>% rownames\_to\_column()  
cm\_boosting = as.data.frame(cm\_boosting$table)  
cm\_boosting$model = "Boosting"  
  
pf.boosting

## rowname cm\_boosting$byClass[neededPerfomanceIndex]  
## 1 Sensitivity 0.7265625  
## 2 Specificity 0.6274510  
## 3 Precision 0.7099237  
## 4 Recall 0.7265625  
## 5 <NA> NA  
## 6 6 0.6826087

============================================ RF

rf.train= randomForest(CloudTomorrow ~. ,data = WAUS.train)  
#test  
rf.predict = predict(rf.train, newdata = WAUS.test, type = "class")  
#computing confusion matrix  
cm\_rf = confusionMatrix(table(observed =WAUS.test$CloudTomorrow , predicted = rf.predict))  
pf.rf =as.data.frame( cm\_rf$byClass[neededPerfomanceIndex])%>% rbind(cm\_rf$overall[1])  
pf.rf = pf.rf %>% rownames\_to\_column()  
cm\_rf = as.data.frame(cm\_rf$table)  
cm\_rf$model = "RF"

============================================ Comparision

confMatrix =cm\_dt %>%   
 rbind( cm\_rf,cm\_boosting, cm\_bag, cm\_NB)

pf.dt = pf.dt %>%   
 rename(Performance = `cm\_dt$byClass[neededPerfomanceIndex]` ) %>%   
 mutate(model = "DT" )  
pf.dt[6,1] = "Accuracy"  
#make FPR  
pf.dt[7,2] = 1- pf.dt[2,2]   
pf.dt[7,3] = "DT"  
pf.dt[7,1] = "FPR"  
  
  
pf.NB = pf.NB %>%   
 rename(Performance = `cm\_NB$byClass[neededPerfomanceIndex]` ) %>%   
 mutate(model = "NB")  
pf.NB[6,1] = "Accuracy"  
pf.NB [7,2] = 1- pf.NB [2,2]   
pf.NB [7,3] = "NB"  
pf.NB [7,1] = "FPR"  
  
pf.bag = pf.bag %>%   
 rename(Performance = `cm\_bag$byClass[neededPerfomanceIndex]` ) %>%   
 mutate(model = "Bag")  
pf.bag[6,1] = "Accuracy"  
pf.bag [7,2] = 1- pf.bag [2,2]   
pf.bag [7,3] = "Bag"  
pf.bag [7,1] = "FPR"  
  
pf.boosting = pf.boosting %>%   
 rename(Performance = `cm\_boosting$byClass[neededPerfomanceIndex]` ) %>%   
 mutate(model = "Boosting")  
pf.boosting[6,1] = "Accuracy"  
pf.boosting[7,2] = 1- pf.boosting[2,2]   
pf.boosting[7,3] = "Boosting"  
pf.boosting[7,1] = "FPR"  
  
pf.rf = pf.rf %>%   
 rename(Performance = `cm\_rf$byClass[neededPerfomanceIndex]` ) %>%   
 mutate(model = "RF")  
pf.rf[6,1] = "Accuracy"  
pf.rf [7,2] = 1- pf.rf [2,2]   
pf.rf [7,3] = "RF"  
pf.rf [7,1] = "FPR"  
  
  
pf.All = pf.dt %>% rbind(pf.NB,pf.bag,pf.boosting,pf.rf) %>%   
 filter( Performance != is.na(Performance))  
  
pf.All = pf.All %>% pivot\_wider(names\_from =rowname, values\_from =Performance )   
  
pf.All = pf.All[,1] %>% cbind(round(pf.All[,-1],digits=3)) %>% select(-Sensitivity)

dt <- predict(train.tree, WAUS.test, type = "vector")  
nb <- predict(NB.Model, WAUS.test, type = "raw")  
bag<- predict(bag.train, WAUS.test, type = "vector")  
boosting <- predict(boosting.train, WAUS.test, type = "vector")  
rf <- predict(rf.train, WAUS.test, type = "prob")  
  
dt.roc <- roc(WAUS.test$CloudTomorrow,dt[,2])

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

nb.roc <-roc(WAUS.test$CloudTomorrow,nb[,2])

## Setting levels: control = 0, case = 1  
## Setting direction: controls < cases

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

## Setting levels: control = 0, case = 1  
## Setting direction: controls < cases

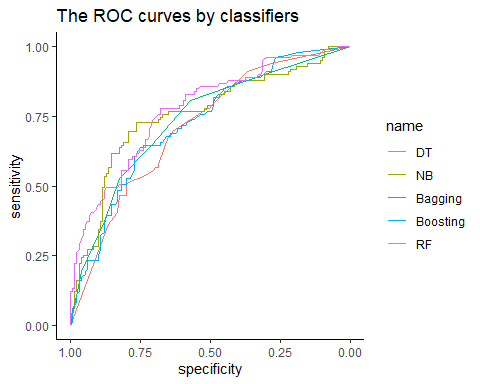
boosting.roc <-roc(WAUS.test$CloudTomorrow,boosting$prob[,2])

## Setting levels: control = 0, case = 1  
## Setting direction: controls < cases

rf.roc <-roc(WAUS.test$CloudTomorrow,rf[,2])

## Setting levels: control = 0, case = 1  
## Setting direction: controls < cases

rocs <- list()  
rocs[["DT"]] <- dt.roc  
rocs[["NB"]] <- nb.roc  
rocs[["Bagging"]] <- bag.roc  
rocs[["Boosting"]] <- boosting.roc  
rocs[["RF"]] <- rf.roc  
  
color = c("red","#0000ff","#4cd7d0","green","black")  
  
ggroc(rocs)+theme\_classic()+ggtitle("The ROC curves by classifiers")



auc.dt <- as.data.frame( auc(dt.roc)) %>%mutate(model = "DT") %>%   
 rename(AUC = `auc(dt.roc)`)  
 auc.nb <- as.data.frame(auc(nb.roc))%>%mutate(model = "NB") %>%   
 rename(AUC = `auc(nb.roc)`)   
 auc.bag <- as.data.frame(auc(bag.roc))%>%mutate(model = "Bag") %>%   
 rename(AUC =`auc(bag.roc)` )  
 auc.boosting <- as.data.frame(auc(boosting.roc))%>%mutate(model = "Boosting") %>%   
 rename(AUC =`auc(boosting.roc)` )  
 auc.rf <- as.data.frame( auc(rf.roc))%>%mutate(model = "RF") %>%   
 rename(AUC = `auc(rf.roc)`)  
  
table.all <- auc.dt %>% rbind( auc.nb,auc.bag,auc.boosting,auc.rf) %>% select(-model)  
 table.all <- pf.All %>% select(-Specificity) %>% cbind(table.all)  
 table.all = table.all[,1] %>% cbind( round(table.all[,-1],digits=3) ) %>% rename(Model = ".")  
  
knitr::kable(table.all)

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

library("viridis") # Load

## Loading required package: viridisLite

## Warning: package 'viridisLite' was built under R version 4.0.5

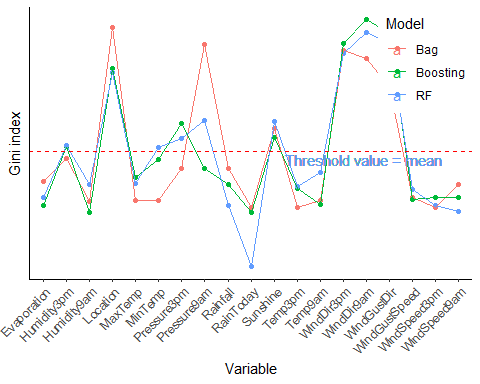
##   
## Attaching package: 'viridis'

## The following object is masked from 'package:scales':  
##   
## viridis\_pal

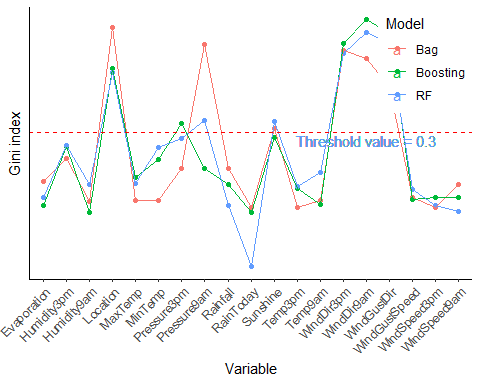
importance <- as.data.frame( bag.train$importance)  
importance <-importance %>% cbind( as.data.frame( boosting.train$importance)) %>% rownames\_to\_column()  
importance <-importance %>% full\_join(as.data.frame( rf.train$importance) %>% rownames\_to\_column() )

## Joining, by = "rowname"

importance <- importance [,1]%>% cbind( scale(importance[,-1]))  
importance <- as.data.frame(importance) %>% rename(Variable=".",   
 Bag ="bag.train$importance",  
 Boosting ="boosting.train$importance",  
 RF = "MeanDecreaseGini")  
  
importance <- importance %>% pivot\_longer(cols= -Variable,names\_to = "Model", values\_to = "Values")  
importance[,3] <- as.numeric(unlist( importance[,3]))   
importance [,3]<-round(importance[,3],digits = 3)  
   
  
  
ggplot(importance, aes(x = Variable, y =Values, colour = Model, group = Model)) +  
 geom\_point()+  
 geom\_line()+  
 theme\_classic()+  
 labs(y = "Gini index")+  
 theme(axis.text.y =element\_blank(),  
 axis.ticks.y = element\_blank(),  
 axis.text.x = element\_text(angle = 45, vjust =1, hjust=1),  
 legend.position=c(1,1),legend.justification = c(1,1))+  
 geom\_hline(yintercept = mean(importance$Values), linetype="dashed", color = "red")+  
 geom\_text(aes(15,0,label = "Threshold value = mean ", vjust = 1.2))



ggplot(importance, aes(x = Variable, y =Values, colour = Model, group = Model)) +  
 geom\_point()+  
 geom\_line()+  
 theme\_classic()+  
 labs(y = "Gini index")+  
 theme(axis.text.y =element\_blank(),  
 axis.ticks.y = element\_blank(),  
 axis.text.x = element\_text(angle = 45, vjust =1, hjust=1),  
 legend.position=c(1,1),legend.justification = c(1,1))+  
 geom\_hline(yintercept = 0.3, linetype="dashed", color = "red")+  
 geom\_text(aes(15,0.3,label = "Threshold value = 0.3", vjust = 1.2))



#columns for important variables  
x = c("MinTemp","Pressure9am","Pressure3pm","Sunshine","WindDir3pm","WindDir9am","WindGustDir","CloudTomorrow")

library(neuralnet)

## Warning: package 'neuralnet' was built under R version 4.0.5

##   
## Attaching package: 'neuralnet'

## The following object is masked from 'package:ROCR':  
##   
## prediction

## The following object is masked from 'package:dplyr':  
##   
## compute

library(dummies)

## dummies-1.5.6 provided by Decision Patterns

set.seed(999999)  
x = c("MinTemp","Pressure9am","Pressure3pm","Sunshine","WindDir3pm","WindDir9am","WindGustDir","CloudTomorrow")  
  
nn.train <-select(WAUS.train[,x], !is.numeric)

## Warning: Predicate functions must be wrapped in `where()`.  
##   
## # Bad  
## data %>% select(is.numeric)  
##   
## # Good  
## data %>% select(where(is.numeric))  
##   
## i Please update your code.  
## This message is displayed once per session.

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

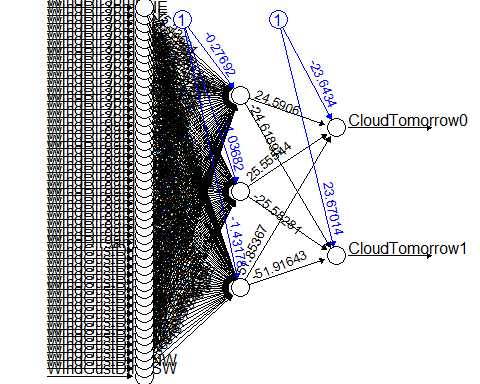
## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts = FALSE):  
## non-list contrasts argument ignored

## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts = FALSE):  
## non-list contrasts argument ignored  
  
## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts = FALSE):  
## non-list contrasts argument ignored  
  
## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts = FALSE):  
## non-list contrasts argument ignored

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

## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts = FALSE):  
## non-list contrasts argument ignored  
  
## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts = FALSE):  
## non-list contrasts argument ignored  
  
## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts = FALSE):  
## non-list contrasts argument ignored  
  
## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts = FALSE):  
## non-list contrasts argument ignored

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



nn$result.matrix

## [,1]  
## error 5.949690e+01  
## reached.threshold 9.626709e-03  
## steps 1.352500e+04  
## Intercept.to.1layhid1 -2.769248e-01  
## WindDir3pmE.to.1layhid1 1.533621e+01  
## WindDir3pmENE.to.1layhid1 1.012686e+00  
## WindDir3pmESE.to.1layhid1 9.935850e+00  
## WindDir3pmN.to.1layhid1 4.222823e+00  
## WindDir3pmNE.to.1layhid1 1.402632e+00  
## WindDir3pmNNE.to.1layhid1 -6.167997e+00  
## WindDir3pmNNW.to.1layhid1 2.592571e+00  
## WindDir3pmNW.to.1layhid1 -1.746252e+01  
## WindDir3pmS.to.1layhid1 -3.110530e+01  
## WindDir3pmSE.to.1layhid1 -4.284290e+01  
## WindDir3pmSSE.to.1layhid1 -9.723981e+00  
## WindDir3pmSSW.to.1layhid1 -7.631614e+02  
## WindDir3pmSW.to.1layhid1 -7.538305e+02  
## WindDir3pmW.to.1layhid1 -3.437313e+01  
## WindDir3pmWNW.to.1layhid1 2.806137e+00  
## WindDir3pmWSW.to.1layhid1 -3.516274e+01  
## WindDir9amE.to.1layhid1 6.395414e+00  
## WindDir9amENE.to.1layhid1 1.193018e+01  
## WindDir9amESE.to.1layhid1 -8.020254e+00  
## WindDir9amN.to.1layhid1 -6.539115e+00  
## WindDir9amNE.to.1layhid1 -2.066651e+00  
## WindDir9amNNE.to.1layhid1 1.201233e+01  
## WindDir9amNNW.to.1layhid1 1.013356e+01  
## WindDir9amNW.to.1layhid1 -1.576576e+01  
## WindDir9amS.to.1layhid1 3.772110e+01  
## WindDir9amSE.to.1layhid1 9.027727e+00  
## WindDir9amSSE.to.1layhid1 7.820905e+00  
## WindDir9amSSW.to.1layhid1 3.541685e+00  
## WindDir9amSW.to.1layhid1 -7.925428e+00  
## WindDir9amW.to.1layhid1 -7.546073e+02  
## WindDir9amWNW.to.1layhid1 1.023037e+01  
## WindDir9amWSW.to.1layhid1 4.691855e-01  
## WindGustDirE.to.1layhid1 6.750530e+00  
## WindGustDirENE.to.1layhid1 -3.929189e+01  
## WindGustDirESE.to.1layhid1 -5.164589e+00  
## WindGustDirN.to.1layhid1 -6.900155e+00  
## WindGustDirNE.to.1layhid1 -2.333398e+00  
## WindGustDirNNE.to.1layhid1 -1.177818e+01  
## WindGustDirNNW.to.1layhid1 4.012770e+00  
## WindGustDirNW.to.1layhid1 1.233505e+01  
## WindGustDirS.to.1layhid1 1.586441e+01  
## WindGustDirSE.to.1layhid1 -3.150542e+00  
## WindGustDirSSE.to.1layhid1 3.635863e+01  
## WindGustDirSSW.to.1layhid1 -7.483638e+02  
## WindGustDirSW.to.1layhid1 -5.906606e+00  
## WindGustDirW.to.1layhid1 2.187041e+01  
## WindGustDirWNW.to.1layhid1 -1.026115e+01  
## WindGustDirWSW.to.1layhid1 9.985001e+00  
## Intercept.to.1layhid2 -1.036818e+00  
## WindDir3pmE.to.1layhid2 -7.096391e+00  
## WindDir3pmENE.to.1layhid2 8.460749e+00  
## WindDir3pmESE.to.1layhid2 -7.459505e+00  
## WindDir3pmN.to.1layhid2 -7.259226e+02  
## WindDir3pmNE.to.1layhid2 -1.778466e+00  
## WindDir3pmNNE.to.1layhid2 -2.759447e+00  
## WindDir3pmNNW.to.1layhid2 3.692635e+00  
## WindDir3pmNW.to.1layhid2 4.075324e+00  
## WindDir3pmS.to.1layhid2 8.317516e+00  
## WindDir3pmSE.to.1layhid2 -8.672842e+00  
## WindDir3pmSSE.to.1layhid2 -2.563972e+00  
## WindDir3pmSSW.to.1layhid2 -7.224656e+02  
## WindDir3pmSW.to.1layhid2 1.446051e+00  
## WindDir3pmW.to.1layhid2 -1.040301e+01  
## WindDir3pmWNW.to.1layhid2 -1.037559e+01  
## WindDir3pmWSW.to.1layhid2 -8.248760e-01  
## WindDir9amE.to.1layhid2 -5.234863e+00  
## WindDir9amENE.to.1layhid2 -3.660340e+00  
## WindDir9amESE.to.1layhid2 -1.069059e+01  
## WindDir9amN.to.1layhid2 2.605217e-01  
## WindDir9amNE.to.1layhid2 1.163337e+01  
## WindDir9amNNE.to.1layhid2 2.869687e+00  
## WindDir9amNNW.to.1layhid2 1.095204e+01  
## WindDir9amNW.to.1layhid2 2.022367e+00  
## WindDir9amS.to.1layhid2 4.087638e-01  
## WindDir9amSE.to.1layhid2 -2.944471e+00  
## WindDir9amSSE.to.1layhid2 -7.248475e+02  
## WindDir9amSSW.to.1layhid2 5.198301e+00  
## WindDir9amSW.to.1layhid2 2.304601e+00  
## WindDir9amW.to.1layhid2 5.656830e+00  
## WindDir9amWNW.to.1layhid2 7.511396e+00  
## WindDir9amWSW.to.1layhid2 -6.191616e+00  
## WindGustDirE.to.1layhid2 3.546488e+00  
## WindGustDirENE.to.1layhid2 -1.389452e+00  
## WindGustDirESE.to.1layhid2 -7.228127e+02  
## WindGustDirN.to.1layhid2 1.089006e+00  
## WindGustDirNE.to.1layhid2 -1.684519e+01  
## WindGustDirNNE.to.1layhid2 -7.225429e+00  
## WindGustDirNNW.to.1layhid2 -1.582979e+01  
## WindGustDirNW.to.1layhid2 -2.067963e+00  
## WindGustDirS.to.1layhid2 2.789527e+00  
## WindGustDirSE.to.1layhid2 8.881057e+00  
## WindGustDirSSE.to.1layhid2 4.796884e-01  
## WindGustDirSSW.to.1layhid2 7.341531e+00  
## WindGustDirSW.to.1layhid2 2.104875e-01  
## WindGustDirW.to.1layhid2 5.709235e+00  
## WindGustDirWNW.to.1layhid2 -7.019525e+00  
## WindGustDirWSW.to.1layhid2 -3.097799e+00  
## Intercept.to.1layhid3 -1.431779e+00  
## WindDir3pmE.to.1layhid3 -6.426862e+00  
## WindDir3pmENE.to.1layhid3 -7.801509e+00  
## WindDir3pmESE.to.1layhid3 -3.603315e+00  
## WindDir3pmN.to.1layhid3 5.245864e-01  
## WindDir3pmNE.to.1layhid3 -3.524949e+00  
## WindDir3pmNNE.to.1layhid3 1.410772e+00  
## WindDir3pmNNW.to.1layhid3 -5.835309e+00  
## WindDir3pmNW.to.1layhid3 -3.294675e+00  
## WindDir3pmS.to.1layhid3 -2.219529e+00  
## WindDir3pmSE.to.1layhid3 2.734951e+00  
## WindDir3pmSSE.to.1layhid3 1.939818e+00  
## WindDir3pmSSW.to.1layhid3 1.418617e+01  
## WindDir3pmSW.to.1layhid3 1.685164e+00  
## WindDir3pmW.to.1layhid3 1.796615e+00  
## WindDir3pmWNW.to.1layhid3 -2.237300e+00  
## WindDir3pmWSW.to.1layhid3 1.704011e+00  
## WindDir9amE.to.1layhid3 1.942077e+00  
## WindDir9amENE.to.1layhid3 -1.434291e+00  
## WindDir9amESE.to.1layhid3 4.901959e+00  
## WindDir9amN.to.1layhid3 1.484177e+00  
## WindDir9amNE.to.1layhid3 -4.909735e+00  
## WindDir9amNNE.to.1layhid3 -7.058440e+00  
## WindDir9amNNW.to.1layhid3 -7.080679e+02  
## WindDir9amNW.to.1layhid3 4.362121e+00  
## WindDir9amS.to.1layhid3 -2.955643e+00  
## WindDir9amSE.to.1layhid3 -7.157187e+02  
## WindDir9amSSE.to.1layhid3 -1.135849e+00  
## WindDir9amSSW.to.1layhid3 -2.142475e+00  
## WindDir9amSW.to.1layhid3 -1.615586e+00  
## WindDir9amW.to.1layhid3 -7.104427e+02  
## WindDir9amWNW.to.1layhid3 -1.397994e+01  
## WindDir9amWSW.to.1layhid3 -7.709970e-01  
## WindGustDirE.to.1layhid3 -5.288757e+00  
## WindGustDirENE.to.1layhid3 5.382320e+00  
## WindGustDirESE.to.1layhid3 -9.848562e-02  
## WindGustDirN.to.1layhid3 -8.075926e-01  
## WindGustDirNE.to.1layhid3 -5.404397e+00  
## WindGustDirNNE.to.1layhid3 6.843197e+00  
## WindGustDirNNW.to.1layhid3 4.721649e+00  
## WindGustDirNW.to.1layhid3 -7.080558e+02  
## WindGustDirS.to.1layhid3 -1.105423e+01  
## WindGustDirSE.to.1layhid3 -3.980206e-01  
## WindGustDirSSE.to.1layhid3 -1.242576e+01  
## WindGustDirSSW.to.1layhid3 -1.426479e+00  
## WindGustDirSW.to.1layhid3 4.066502e-01  
## WindGustDirW.to.1layhid3 -4.728274e+00  
## WindGustDirWNW.to.1layhid3 2.264606e+00  
## WindGustDirWSW.to.1layhid3 3.651775e-01  
## Intercept.to.CloudTomorrow0 -2.364340e+01  
## 1layhid1.to.CloudTomorrow0 2.459060e+01  
## 1layhid2.to.CloudTomorrow0 2.555544e+01  
## 1layhid3.to.CloudTomorrow0 5.185367e+01  
## Intercept.to.CloudTomorrow1 2.367014e+01  
## 1layhid1.to.CloudTomorrow1 -2.461891e+01  
## 1layhid2.to.CloudTomorrow1 -2.558281e+01  
## 1layhid3.to.CloudTomorrow1 -5.191643e+01

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

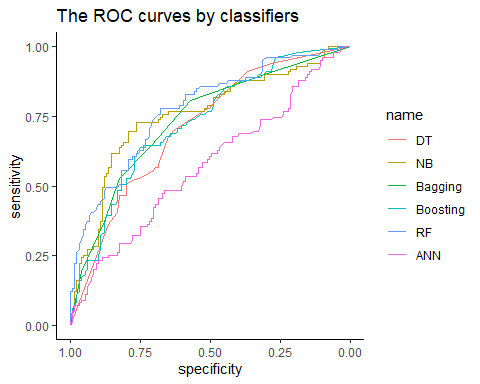
## Confusion Matrix and Statistics  
##   
## predicted  
## observed 0 1  
## 0 50 81  
## 1 31 68  
##   
## Accuracy : 0.513   
## 95% CI : (0.4465, 0.5793)  
## No Information Rate : 0.6478   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.0646   
##   
## Mcnemar's Test P-Value : 3.656e-06   
##   
## Sensitivity : 0.6173   
## Specificity : 0.4564   
## Pos Pred Value : 0.3817   
## Neg Pred Value : 0.6869   
## Prevalence : 0.3522   
## Detection Rate : 0.2174   
## Detection Prevalence : 0.5696   
## Balanced Accuracy : 0.5368   
##   
## 'Positive' Class : 0   
##

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

## Setting levels: control = 0, case = 1

## Setting direction: controls > cases

rocs <- list()  
rocs[["DT"]] <- dt.roc  
rocs[["NB"]] <- nb.roc  
rocs[["Bagging"]] <- bag.roc  
rocs[["Boosting"]] <- boosting.roc  
rocs[["RF"]] <- rf.roc  
rocs[["ANN"]] <- ann.roc  
  
color = c("red","#0000ff","#4cd7d0","green","black")  
  
ggroc(rocs)+theme\_classic()+ggtitle("The ROC curves by classifiers")



======================== improve

df.improved.train= WAUS.train  
df.improved.test= WAUS.test  
  
set.seed(1234)  
rf.train= randomForest(CloudTomorrow ~. ,data = df.improved.train,  
 ntree = 50000,   
 importance= T,  
 mtry = 2,  
 nodesize=42)  
#test  
rf.predict = predict(rf.train, newdata = df.improved.test, type = "class")  
#computing confusion matrix  
cm\_rf = confusionMatrix(table(observed =df.improved.test$CloudTomorrow , predicted = rf.predict))  
pf.rf =as.data.frame( cm\_rf$byClass[neededPerfomanceIndex])%>% rbind(cm\_rf$overall[1])  
pf.rf = pf.rf %>% rownames\_to\_column()  
cm\_rf = as.data.frame(cm\_rf$table)  
cm\_rf$model = "RF"  
  
rf <- predict(rf.train, df.improved.test, type = "prob")  
grid.rf.roc <-roc(df.improved.test$CloudTomorrow,rf[,2])

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

auc.rf <- as.data.frame( auc(grid.rf.roc))%>%mutate(model = "RF") %>%   
 rename(AUC = `auc(grid.rf.roc)`)  
 auc.rf

## AUC model  
## 1 0.7629347 RF

#>0.736  
 #ntree = 10000 AUC = 0.7447375   
 # Impot = T #ntree = 10000 AUC = 0.7450459   
 # Impot = T #ntree = 50000 AUC = 0.7460483

===================================== Q9

#variable index which beyond the mean of importance  
 x = c("MinTemp","Pressure9am","Pressure3pm","Sunshine","WindSpeed3pm","WindSpeed9am","CloudTomorrow")  
#since WindDir3pm and WindDirGust are not independent, I will drop one of them  
 byhand = WAUS.train[,x]   
 set.seed(31084222) #Student ID as random seed   
sample\_data = sample\_n(byhand, size = 15, replace = T)  
#if Humidity3pm > 50, it means that day is humid  
sample\_data$Pressure = ifelse(sample\_data$Pressure3pm> 1013&  
 sample\_data$Pressure3pm>1013 ,  
 "High","Not High")  
sample\_data$Windy = ifelse(  
 ((sample\_data$WindSpeed9am > 15) &(sample\_data$WindSpeed3pm >15))  
 ,"Yes","No")  
#if MinTemp < 10 degree c, it means that day is cold  
sample\_data$Cold = ifelse(sample\_data$MinTemp< 10 ,"Cold","Not Cold")  
#if Sunshine hour > 12 , it means that day is sunny  
sample\_data$Sunshine = ifelse(sample\_data$Sunshine > 12 ,"Sunny","Not Sunny")  
#if WindSpeed9am > 15 , it means that day is Windy  
sample\_data$WindSpeed9am = ifelse(  
 ((sample\_data$WindSpeed9am > 15) &&(sample\_data$WindSpeed3pm >15))  
 ,"Yes","No")  
sample\_data$CloudTomorrow = ifelse(sample\_data$CloudTomorrow == 1, "Cloudy", "Not Cloudy")  
sample\_data <- sample\_data%>% select( -WindSpeed3pm,-WindSpeed9am, -Pressure9am,-Pressure3pm, -MinTemp)  
  
 set.seed(31084222) #Student ID as random seed   
   
  
   
write.csv(sample\_data, file = "C:/Users/sjsa3/Desktop/Shared\_with\_Mac/year2\_sem1/FIT3152/Assignment 2/Q9/ID3/byhand.csv")

#variable index which beyond the mean of importance  
 x = c("MinTemp","Pressure9am","Pressure3pm","Sunshine","WindSpeed3pm","WindSpeed9am","CloudTomorrow")  
#since WindDir3pm and WindDirGust are not independent, I will drop one of them  
 byhand = WAUS.train[,x]   
 set.seed(99999) #Student ID as random seed   
sample\_data = sample\_n(byhand, size = 5, replace = T)  
#if Humidity3pm > 50, it means that day is humid  
sample\_data$Pressure = ifelse(sample\_data$Pressure3pm> 1013&  
 sample\_data$Pressure3pm>1013 ,  
 "High","Not High")  
sample\_data$Windy = ifelse(  
 ((sample\_data$WindSpeed9am > 15) &&(sample\_data$WindSpeed3pm >15))  
 ,"Yes","No")  
#if MinTemp < 10 degree c, it means that day is cold  
sample\_data$Cold = ifelse(sample\_data$MinTemp< 10 ,"Cold","Not Cold")  
#if Sunshine hour > 12 , it means that day is sunny  
sample\_data$Sunshine = ifelse(sample\_data$Sunshine > 12 ,"Sunny","Not Sunny")  
#if WindSpeed9am > 15 , it means that day is Windy  
sample\_data$WindSpeed9am = ifelse(  
 ((sample\_data$WindSpeed9am > 15) &&(sample\_data$WindSpeed3pm >15))  
 ,"Yes","No")  
sample\_data$CloudTomorrow = ifelse(sample\_data$CloudTomorrow == 1, "Cloudy", "Not Cloudy")  
sample\_data <- sample\_data%>% select( -WindSpeed3pm,-WindSpeed9am, -Pressure9am,-Pressure3pm, -MinTemp)  
  
 set.seed(31084222) #Student ID as random seed   
   
write.csv(sample\_data, file = "C:/Users/sjsa3/Desktop/Shared\_with\_Mac/year2\_sem1/FIT3152/Assignment 2/Q9/ID3/testing.csv")