

```

title: "Assignment 1 ETC1010 - 5510"
author: "Jason Ching Yuen Siu"
date: "`r format(Sys.Date(), '%A, %B %e %Y')`"
output:
pdf_document: default
html_document: default
subtitle: FIT3152 assignment 2
---
===== import the needed lib
```{r include=F}
 library(tidyverse)
 library(simputation) #imputation
 library(naniar)
 library(lubridate)
 library(tidymodels)
 library(modeest) # find mode
 library(e1071) #Naïve Bayes
 library(ROCR)
 library(pROC)
 library(tree)
 library(skimr) # data exploratory
 library(GGally)
 library(magrittr)
 library(flextable)

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

===== read file
```{r read-file}
  rm(list = ls())
  WAUS <- read.csv("C:/Users/sjsa3/Desktop/Shared_with_Mac/year2_sem1/FIT3152/Assignment 2/data.csv",
stringsAsFactors = T)
  WAUS <- WAUS %>% filter(CloudTomorrow != "NA") # remove the rows in which the value of CloudTmr is
NA

  L <- as.data.frame(c(1:49))
  set.seed(31084222) # My Student ID as the random seed
  L <- L[sample(nrow(L), 10, replace = FALSE),] # sample 10 locations
  WAUS <- WAUS[(WAUS$Location %in% L),]
  WAUS <- WAUS[sample(nrow(WAUS), 2000, replace = FALSE),] # sample 2000 rows
  WAUS$CloudTomorrow = as.factor(WAUS$CloudTomorrow)
  WAUS$Location = as.factor(WAUS$Location)
...

===== Explore data
  Since we are predicting the cloudiness, there is no use of Date, thus removing them
```{r feature-selection}
 #date format
 WAUS$Date = ""
 WAUS$Day = as.character(WAUS$Day)
 WAUS$Month = as.character(WAUS$Month)
 WAUS$Year = as.character(WAUS$Year)
 WAUS$Date =paste(WAUS$Year,"-", WAUS$Month,"-", WAUS$Day)
 WAUS$Date = ymd(WAUS$Date)
 WAUS= WAUS %>% arrange(Date)
 WAUS <- WAUS %>% select(-Day,-Year,-Month,-Date)
...

```

```

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

...

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

...

===== Data pre-processing
 clean the data : Remove all the rows with missing values
{r impute-data}
 #drop all categorical values becuae they cannot be found
 WAUS = WAUS %>% drop_na()

...

 split the dataset
{r set-seed}

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

...

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

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

...

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

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

...

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

```

```

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

...

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

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

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

...

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

...

===== Comparision
```{r comparision-confusion-matrix}
    confMatrix =cm_dt %>%
    rbind( cm_rf,cm_boosting, cm_bag, cm_NB)

...

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

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

```

```

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

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

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

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

```

```

```

```

```

```{r comparision-roc}

```

```

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

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

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

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

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

```

```

```

```

```

```{r comparision-auc}

```

```

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

```

```

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

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

knitr::kable(table.all)

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

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

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

#index for mportant variables
#the following chunks might need this index to filter the features
x =
c("MinTemp","Pressure9am","Pressure3pm","Sunshine","WindDir3pm","WindDir9am","WindGustDir","CloudTomorrow")
```

===== improve the model : simplicity
```{r improved-models}
df.improved.train= WAUS.train
df.improved.test= WAUS.test
set.seed(1234)
rf.train= randomForest(CloudTomorrow ~. ,data = df.improved.train,
ntree = 50000,
importance= T,
mtry = 2,
nodesize=42)
#model testing
rf.predict = predict(rf.train, newdata = df.improved.test, type = "class")
#computing confusion matrix
cm_rf = confusionMatrix(table(observed =df.improved.test$CloudTomorrow , predicted =

```

```

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

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

...

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

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

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

...
```{r by-hand-model-testing}
#variable index which beyond the mean of importance
x =
c("MinTemp", "Pressure9am", "Pressure3pm", "Sunshine", "WindSpeed3pm", "WindSpeed9am", "CloudTomorrow")
#since WindDir3pm and WindDirGust are not independent, I will drop one of them
byhand = WAUS.train[,x]
set.seed(99999) #Student ID as random seed
sample_data = sample_n(byhand, size = 5, replace = T)
#if Humidity3pm > 50, it means that day is humid
sample_data$Pressure = ifelse(sample_data$Pressure3pm > 1013 &
sample_data$Pressure3pm > 1013 ,

```

```

"High", "Not High")
sample_data$Windy = ifelse(
  ((sample_data$WindSpeed9am > 15) &&(sample_data$WindSpeed3pm >15))
, "Yes", "No")
#if MinTemp < 10 degree c, it means that day is cold
sample_data$Cold = ifelse(sample_data$MinTemp< 10 , "Cold", "Not Cold")
#if Sunshine hour > 12 , it means that day is sunny
sample_data$Sunshine = ifelse(sample_data$Sunshine > 12 , "Sunny", "Not Sunny")
#if WindSpeed9am > 15 , it means that day is Windy
sample_data$WindSpeed9am = ifelse(
  ((sample_data$WindSpeed9am > 15) &&(sample_data$WindSpeed3pm >15))
, "Yes", "No")
sample_data$CloudTomorrow = ifelse(sample_data$CloudTomorrow == 1, "Cloudy", "Not Cloudy")
sample_data <- sample_data%>% select( -WindSpeed3pm, -WindSpeed9am, -Pressure9am, -Pressure3pm, -
MinTemp)

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

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

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

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

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

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

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

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

 cm

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

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

```

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

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

...