IE425
SPRING 2023

HOMEWORK 2
Random Forest & Gradient Boosting Machine & Cross Validation

FATMANUR YAMAN - 2019402204
MURAT TUTAR - 2020402264

Question 1

a. Partition the dataset using the caTools package into training and test sets where 80% of the observations go into the training set and 20% goes into the test set.

```
#1a. Train Test Split
set.seed(425)
split<-sample.split(raw_data$y, SplitRatio = 0.8)
train<-subset(raw_data, split==TRUE)
test<-subset(raw_data, split==FALSE)
```

b. Determine the best random forest (based on the random forest package) by using 10-fold cross validation five times with the caret package on the training set by playing with the mtry and ntree parameters. What are the best values of these two parameters?

The Code:

```
#1b. Best Random Forest with 10-Fold CV
library(randomForest)
library(Metrics)
mtry_tuning_seq <- seq(3,6,1)
ntree_tuning_seq <- seq(100,400,100)

best_accuracy <- 1

for (i in mtry_tuning_seq){
  for (j in ntree_tuning_seq){
    best_parameters <- list()
    set.seed(425)
    ctrl <- trainControl(method='repeatedcv', number=10, repeats=5)
    rf_parameters <- randomForest(y~., data=train, mtry=i, ntree=j, trControl=ctrl,importance=TRUE)
    rf_perdictions <- predict(rf_parameters, newdata=test)
    accuracy <- accuracy(actual=test$y, predicted=rf_perdictions)
    if (accuracy <= best_accuracy){
      best_accuracy <- accuracy
      new_parameters <- append(best_parameters, c(i,j))
    }
  }
}

best_parameters_int = as.integer(new_parameters)
new_parameters
```

The Output:

```
> cat("The best mtry: ",new_parameters[[1]], "and the best ntree: ", new_parameters[[2]])
The best mtry:  3 and the best ntree:  400
```

Using 10-fold CV five times the best parameters are found like mtry : 3 and ntree : 400.

c. What is the out-of-bag accuracy? Comment on which input attributes are important in making predictions.

The Code:

```
#Out-of-Bag Accuracy
set.seed(425)
ctrl <- trainControl(method='repeatedcv', number=10, repeats=5)
rf_final_model <- randomForest(y~., data=train, mtry=as.integer(new_parameters[[1]]),
                               ntree=as.integer(new_parameters[[2]]), trControl=ctrl,
                               importance=TRUE)
rf_perdictions_final <- predict(rf_final_model, newdata=test)
oob_error <- rf_final_model$err.rate[length(rf_final_model),1]

cat("The Out-Of-Bag Error is: ", oob_error)
```

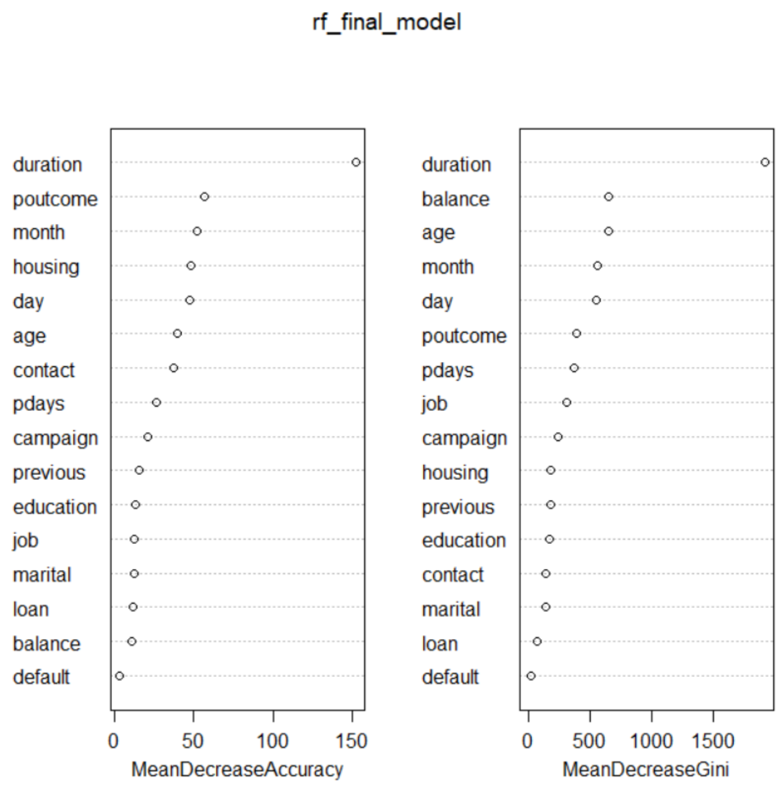
The Output:

```
> cat("The Out-Of-Bag Error is: ", oob_error)
The Out-Of-Bag Error is:  0.1040785
```

The Out-Of-Bag Error is 0.1040785.

The Importance table is:

```
> round(importance(rf_final_model),2)
              no    yes MeanDecreaseAccuracy MeanDecreaseGini
age        35.38  17.20                39.97           646.24
job         9.83   6.56                12.64           307.18
marital     6.88  11.64                12.57           137.66
education   9.64   8.93                12.98           169.40
default     2.68   1.78                 3.32            12.75
balance     7.10   7.55                11.09           653.88
housing    38.19  24.60                47.85           179.29
loan        1.73  15.06                11.24            65.08
contact    34.03  17.65                37.18           140.98
day        44.94   8.60                47.19           549.28
month      49.90  20.31                51.96           563.13
duration  101.79 169.53               152.18          1922.92
campaign   17.87   9.60                20.62           242.07
pdays      24.10  22.86                26.08           367.27
previous   14.91  14.40                15.59           176.54
poutcome   47.69   1.94                56.43           391.89
```



The most important attributes are duration, balance, age, month, and day according to both the mean decrease in the accuracy and the gini index.

d. Provide the Confusion Matrix along with sensitivity, specificity, precision, recall, and the F measure on the test set obtained by the best random forest. Does the out-of-bag accuracy provide a good estimate for the accuracy on the test set?

```
Confusion Matrix and Statistics

          Reference
Prediction   no  yes
       no  7790  691
       yes  194  367

               Accuracy : 0.9021
                 95% CI : (0.8958, 0.9082)
    No Information Rate : 0.883
    P-Value [Acc > NIR] : 3.671e-09

                  Kappa : 0.4051

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.34688
            Specificity : 0.97570
         Pos Pred Value : 0.65419
         Neg Pred Value : 0.91852
             Prevalence : 0.11701
         Detection Rate : 0.04059
   Detection Prevalence : 0.06204
      Balanced Accuracy : 0.66129

       'Positive' Class : yes
```

The out-of-bag accuracy provides a good estimate since the accuracy is high and p value is small. The sensitivity is lower than the specificity which indicates that the model is better in predicting the negative values. But, sensitivity is not enough for the satisfaction. There should be better models that will make predictions better.

e. Repeat part b with the gradient boosting machine using the caret and gbm packages by playing with the interaction.depth, n.trees, shrinkage, and n.minobsinnode parameters. What are the best values of these four parameters?

The Code:

```
#1e. Gradient Boosting Machine Parameter Tuning
install.packages("gbm")
library(gbm)


gbmGrid=expand.grid(interaction.depth = c(3,4,5),
                    n.trees = (5:10)*10,
                    shrinkage = (1:3)*0.1,
                    n.minobsinnode = 20)
set.seed(425)
library(caret)
ctrl1 <- trainControl(method="cv",number=10)
gbm_model <- train(y~., data=train, method="gbm", metric="Accuracy",verbose = FALSE,
                   trControl = ctrl1,tuneGrid = gbmGrid)
max_accuracy <- which.max(gbm_model$results$Accuracy)

cat("The best boosting tree is: shrinkage: ", gbm_model$results$shrinkage[max_accuracy],
    " intearction.depth  :", gbm_model$results$interaction.depth[max_accuracy],
    " n.minobsinnode: ", gbm_model$results$n.minobsinnode[max_accuracy],
    " n.trees: ", gbm_model$results$n.trees[max_accuracy])
```

The Output:

```
> cat("The best boosting tree is: shrinkage: ", gbm_model$results$shrinkage[max_accuracy],
+     " intearction.depth  :", gbm_model$results$interaction.depth[max_accuracy],
+     " n.minobsinnode: ", gbm_model$results$n.minobsinnode[max_accuracy],
+     " n.trees: ", gbm_model$results$n.trees[max_accuracy])
The best boosting tree is: shrinkage:  0.3  intearction.depth  : 3  n.minobsinnode:  20  n.trees:  100
```

With the help of the maximum accuracy of the gbm models, the best boosting tree parameters are: shrinkage = 0.3, interaction.depth = 3, n.minobsinnode = 20, n.trees = 100.

f. Provide the Confusion Matrix along with sensitivity, specificity, precision, recall, and the F measure on the test set obtained by the best boosting tree.

The Code:

```
#1f. Confusion Matrix
set.seed(425)
ctrl1 <- trainControl(method="cv",number=10)
gbm_model_final <- train(y~., data=train, method="gbm", metric="Accuracy",verbose = FALSE,
                   trControl = ctrl1, tuneGrid=expand.grid(shrinkage=0.3,
                   interaction.depth=3,
                   n.minobsinnode=20,
                   n.trees=100))

gbm_perdictions <- predict(gbm_model, newdata=test)

confusionMatrix(gbm_perdictions,test$y,positive = "yes")
```

The Output:

```
Confusion Matrix and Statistics

          Reference
Prediction   no  yes
       no  7748  638
       yes  236  420

               Accuracy : 0.9033
                 95% CI : (0.8971, 0.9094)
    No Information Rate : 0.883
    P-Value [Acc > NIR] : 3.58e-10

                  Kappa : 0.4399

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.39698
            Specificity : 0.97044
         Pos Pred Value : 0.64024
         Neg Pred Value : 0.92392
             Prevalence : 0.11701
         Detection Rate : 0.04645
   Detection Prevalence : 0.07255
      Balanced Accuracy : 0.68371

       'Positive' Class : yes
```

Question 2

a. Partition the dataset into training and test sets where 80% of goes into the training set and 20% goes into the test set.

The Code:

```
#Train Test Split
set.seed(425)
split<-sample.split(raw_data$Rented.Bike.Count, SplitRatio = 0.8)
train<-subset(raw_data, split==TRUE)
test<-subset(raw_data, split==FALSE)
```

b. Determine the best random forest (based on the random forest package) by using 10-fold cross validation five times with the caret package on the training set by playing with the mtry and ntree parameters. What are the best values of these two parameters?

The Code:

```
#2b. Best Random Forest with 10-Fold CV
library(randomForest)
library(Metrics)
mtry_tuning_seq <- seq(3,6,1)
ntree_tuning_seq <- seq(100,400,100)

best_accuracy <- Inf

for (i in mtry_tuning_seq){
  for (j in ntree_tuning_seq){
    best_parameters <- list()
    set.seed(425)
    ctrl <- trainControl(method='repeatedcv', number=10, repeats=5)
    rf_parameters <- randomForest(Rented.Bike.Count~., data=train, mtry=i, ntree=j,
                                  trControl=ctrl,importance=TRUE)
    rf_perdictions <- predict(rf_parameters, newdata=test)
    rmse_tree <- rmse(actual=test$y, predicted=rf_perdictions)
    if (accuracy <= best_accuracy){
      best_rmse <- rmse_tree
      new_parameters <- append(best_parameters, c(i,j))
    }
  }
}

best_parameters_int = as.integer(new_parameters)
new_parameters

cat("The best mtry: ",new_parameters[[1]], "and the best ntree: ", new_parameters[[2]])
```

The Output:

```
> cat("The best mtry: ",new_parameters[[1]], "and the best ntree: ", new_parameters[[2]])
The best mtry:  6 and the best ntree:  400
```

Using 10-fold CV, the best parameters are found like mtry : 6 and ntree : 400.

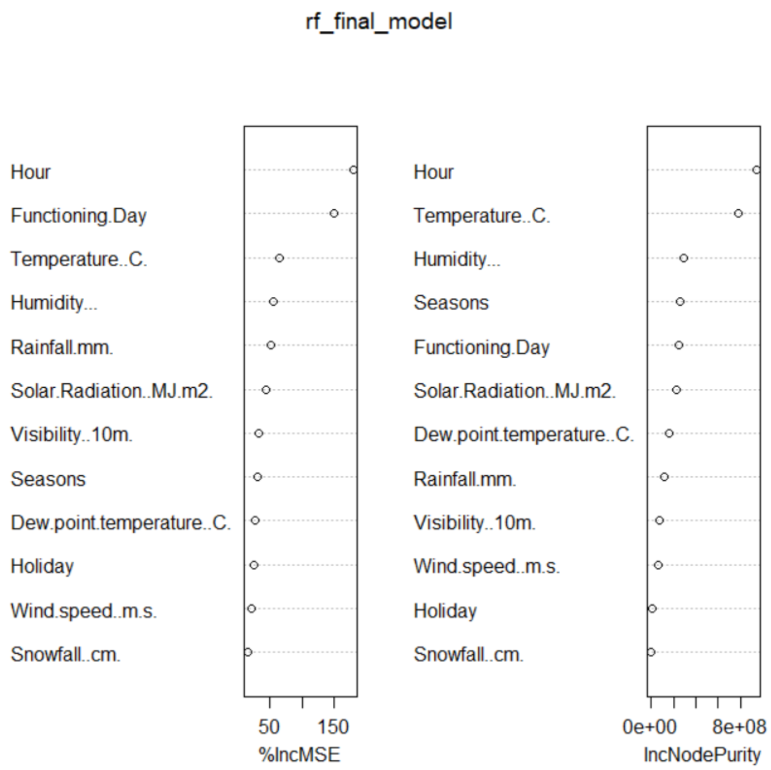
c. Comment on which input attributes are important in making predictions.

The Code:

```
#2c. Attribute Importance
set.seed(425)
ctrl <- trainControl(method='repeatedcv', number=10, repeats=5)
rf_final_model <- randomForest(Rented.Bike.Count~., data=train, mtry=as.integer(new_parameters[[1]]))
rf_perdictions_final <- predict(rf_final_model, newdata=test)
round(importance(rf_final_model),2)
varImpPlot(rf_final_model)
```

The Output:

```
                          %IncMSE IncNodePurity
Hour                       180.51     941862698
Temperature..C.             64.50     783317264
Humidity...                 55.97     293773509
Wind.speed..m.s.            21.38      61399726
Visibility..10m.            31.67      77213434
Dew.point.temperature..C.   27.07     164419595
Solar.Radiation..MJ.m2.     43.59     221691314
Rainfall.mm.                51.21     114308752
Snowfall..cm.               15.38       2591965
Seasons                     30.29     256241602
Holiday                     24.30       8829203
Functioning.Day            150.23     248777716
```



The most important attributes are Hour, Temperature, Humidity, Functioning Day, and Solar Radiation according to both increase in mean squared error and node purity.

d. Make predictions in the test set and report the root mean square error rate and mean absolute error.

The Code:

```
#2d. RMSE & MAE
rmse(actual = test$Rented.Bike.Count,predicted = rf_perdictions_final)
mae(actual = test$Rented.Bike.Count,predicted = rf_perdictions_final)
```

The Output:

```
> rmse(actual = test$Rented.Bike.Count,predicted = rf_perdictions_final)
[1] 193.8868
> mae(actual = test$Rented.Bike.Count,predicted = rf_perdictions_final)
[1] 120.3629
```

The root mean square error is 193.8868 while the mean absolute error is 120.3629.

e. Repeat part b with the gradient boosting using the caret and gbm packages by playing with the interaction.depth, n.trees, shrinkage, and n.minobsinnode parameters. What are the best values of these four parameters?

The Code:

```
#2e. Gradient Boosting Machine Parameter Tuning
install.packages("gbm")
library(gbm)

gbmGrid=expand.grid(interaction.depth = c(3,4,5),
                    n.trees = (5:10)*10,
                    shrinkage = (1:3)*0.1,
                    n.minobsinnode = 20)
set.seed(425)
library(caret)
ctrl1 <- trainControl(method="cv",number=10)
gbm_model <- train(Rented.Bike.Count~., data=train, method="gbm", metric="RMSE",
                   verbose = FALSE, trControl = ctrl1,tuneGrid = gbmGrid)
#Finding the min rmse index.
min_rmse <- which.min(gbm_model$results$RMSE)
cat("The best boosting tree is: shrinkage: ", gbm_model$results$shrinkage[min_rmse],
    " intearction.depth  :", gbm_model$results$interaction.depth[min_rmse],
    " n.minobsinnode: ", gbm_model$results$n.minobsinnode[min_rmse],
    " n.trees: ", gbm_model$results$n.trees[min_rmse])
```

The Output:

```
> #Finding the min rmse index.
> min_rmse <- which.min(gbm_model$results$RMSE)
> cat("The best boosting tree is: shrinkage: ", gbm_model$results$shrinkage[min_rmse],
+     " intearction.depth  :", gbm_model$results$interaction.depth[min_rmse],
+     " n.minobsinnode: ", gbm_model$results$n.minobsinnode[min_rmse],
+     " n.trees: ", gbm_model$results$n.trees[min_rmse])
The best boosting tree is: shrinkage:  0.3  intearction.depth  : 5  n.minobsinnode:  20  n.trees:  100
```

With the help of the minimum value of RMSE among the gbm models, the best boosting tree parameters are: shrinkage = 0.3, interaction.depth = 5, n.minobsinnode = 20, n.trees = 100.

f. Make predictions in the test set and report the root mean square error rate and mean absolute error.

The Code:

```
#Finding the min rmse index.
min_rmse <- which.min(gbm_model$results$RMSE)
cat("The best boosting tree is: shrinkage: ", gbm_model$results$shrinkage[max_accuracy],
    " intearction.depth  :", gbm_model$results$interaction.depth[max_accuracy],
    " n.minobsinnode: ", gbm_model$results$n.minobsinnode[max_accuracy],
    " n.trees: ", gbm_model$results$n.trees[max_accuracy])

#2f. Final Prediction
set.seed(425)
ctrl1 <- trainControl(method="cv",number=10)
gbm_model_final <- train(Rented.Bike.Count~., data=train, method="gbm", metric="RMSE",
                         verbose = FALSE, trControl = ctrl1,
                         tuneGrid=expand.grid(shrinkage=0.3
                                              interaction.depth=5,
                                              n.minobsinnode=20,
                                              n.trees=100))

gbm_perdictions <- predict(gbm_model_final, newdata=test)

rmse(actual = test$Rented.Bike.Count,predicted = gbm_perdictions)
mae(actual = test$Rented.Bike.Count,predicted = gbm_perdictions)
```

The Output:

```
> rmse(actual = test$Rented.Bike.Count,predicted = gbm_perdictions)
[1] 216.5296
> mae(actual = test$Rented.Bike.Count,predicted = gbm_perdictions)
[1] 143.2459
```

The root mean square error is 216.5296 while the mean absolute error is 143.2459.