Milestone 4: Model Analysis

Nosipho Precious Donkrag, Nontsikelelo Sharon Buhlungu, Tshepang Mogosi, Pitsi Pitsi

2024-10-21

```r
library(readr)
library(ggplot2)
library(gridExtra)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:gridExtra':
##
##     combine

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggcorrplot)
library(fastDummies)
library(corrplot)
```

```
## corrplot 0.94 loaded
```

```r
library(randomForest)
```

```
## randomForest 4.7-1.2

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

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

## The following object is masked from 'package:gridExtra':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(caret)
```

```
## Loading required package: lattice
```

```r
library(caret)
```

```r
setwd('C:/Users/nosip/Documents/third Year/BIN381/milestones')
```

Model Assessment: Random Forest Model

The following Milestone will be dedicated to the tuning of the already trained random forest model with the aim of improving the accuracy and precision of the model.

###Read the data Read in the data that has been prepared for the random forest (rf) model:

```r
data_df <- read.csv("final_data_form.csv",
                    header = TRUE)
```

```r
names(data_df)
```

```
##  [1] "marital_status"     "household_size"     "yrs_of_residence"
##  [4] "Annual_Salary"      "Months_Annual"      "FRS.Contribution"
##  [7] "Net_months"         "Gross_Salary"       "Gross_Months"
## [10] "Qualify"            "Education_Bach."    "Education_HS_grad"
## [13] "Education_Masters"  "Occupation_Cleric." "Occupation_Exec."
## [16] "Occupation_Prof."   "Occupation_Sales"   "age"
```

View the structure to ensure that the data types are as expected (only numeric).

```r
str(data_df)
```

```
## 'data.frame':    151133 obs. of  18 variables:
##  $ marital_status    : int  1 2 2 1 2 2 1 2 2 2 ...
##  $ household_size    : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ yrs_of_residence  : int  4 4 4 4 4 4 4 4 4 4 ...
##  $ Annual_Salary     : num  0.62 0.25 0.394 0.735 0.386 ...
##  $ Months_Annual     : int  2 3 4 1 6 3 18 2 4 8 ...
##  $ FRS.Contribution  : num  0.617 0.223 0.501 0.433 0.949 ...
##  $ Net_months        : num  -0.516 -0.211 -0.482 -0.958 2.335 ...
##  $ Gross_Salary      : num  0.0255 0.9321 0.9684 0.4696 0.8508 ...
##  $ Gross_Months      : num  -0.539 -0.192 -0.47 -0.956 2.31 ...
##  $ Qualify           : int  0 0 0 0 1 1 1 0 0 0 ...
##  $ Education_Bach.   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Education_HS_grad : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Education_Masters : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Occupation_Cleric.: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Occupation_Exec.  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Occupation_Prof.  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Occupation_Sales  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ age               : int  48 60 82 47 75 74 78 46 75 76 ...
```

The data types for the columns are as expected. The Gross Months and Net Months have been previously normalised and scaled; hence some values are negative; this is to be expected.

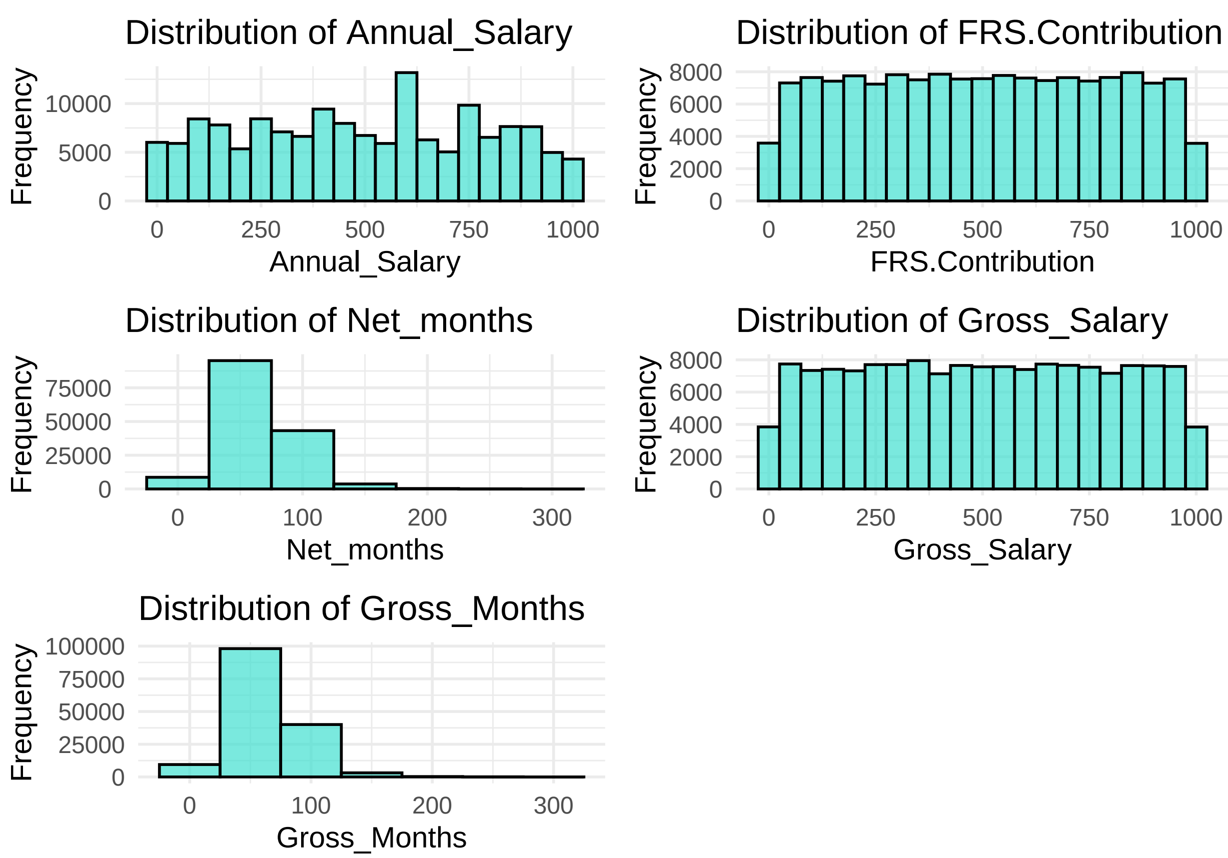
The histograms below show the columns before they were scaled:

```r
data_b4_scaling <- read.csv("data_for_ml.csv", header = TRUE)
```

```r
columns_to_distr <- c("Annual_Salary", "FRS.Contribution",
"Net_months", "Gross_Salary", "Gross_Months")

# folr loop to loop through the above columns:
plot_list <- list()
for (column in columns_to_distr) {
p <- ggplot(data_b4_scaling, aes_string(x = column)) +
geom_histogram(binwidth = 50, fill = "turquoise", color = "black", alpha = 0.7) +
labs(title = paste("Distribution of", column), x = column, y = "Frequency") +
theme_minimal()
plot_list[[column]] <- p
}
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
grid.arrange(grobs = plot_list, ncol = 2)
```



However in milestone 3, the data was scaled appropriately.

Split the data

The data will be split as following: - Training - Testing - Validation

```r
#split the data
set.seed(123)
total_rows <- nrow(data_df)
#split data 70-30
train_indices <- sample(1:total_rows, 0.7 * total_rows)
train_data <- data_df[train_indices, ]
remaining_indices <- setdiff(1:total_rows, train_indices)
#testing and validation will each make up 15%
validation_indices <- sample(remaining_indices, 0.5 * length(remaining_indices))
test_indices <- setdiff(remaining_indices, validation_indices)
validation_data <- data_df[validation_indices, ]
test_data <- data_df[test_indices, ]
```

```r
cat("Training data size:", nrow(train_data), "\n")
```

```
## Training data size: 105793
```

```r
cat("Validation data size:", nrow(validation_data), "\n")
```

```
## Validation data size: 22670
```

```r
cat("Testing data size:", nrow(test_data), "\n")
```

```
## Testing data size: 22670
```

The rf Model (Max Depth Tuning)

The first parameter that will be tuned is the max-depth parameter of the tree; this will restrict how deep the trees go and this parameter will help with reducing over fitting.

```r
formula <- Qualify ~ marital_status + household_size + yrs_of_residence +
Annual_Salary + Months_Annual + FRS.Contribution + Net_months + Gross_Salary + Gross_Months +
Education_Bach. + Education_HS_grad + Education_Masters +
Occupation_Cleric. + Occupation_Exec. + Occupation_Prof. +
Occupation_Sales + age
```

```r
#metrices
depth_values <- c(90, 95)
accuracy_scores <- c()
f1_scores <- c()
```

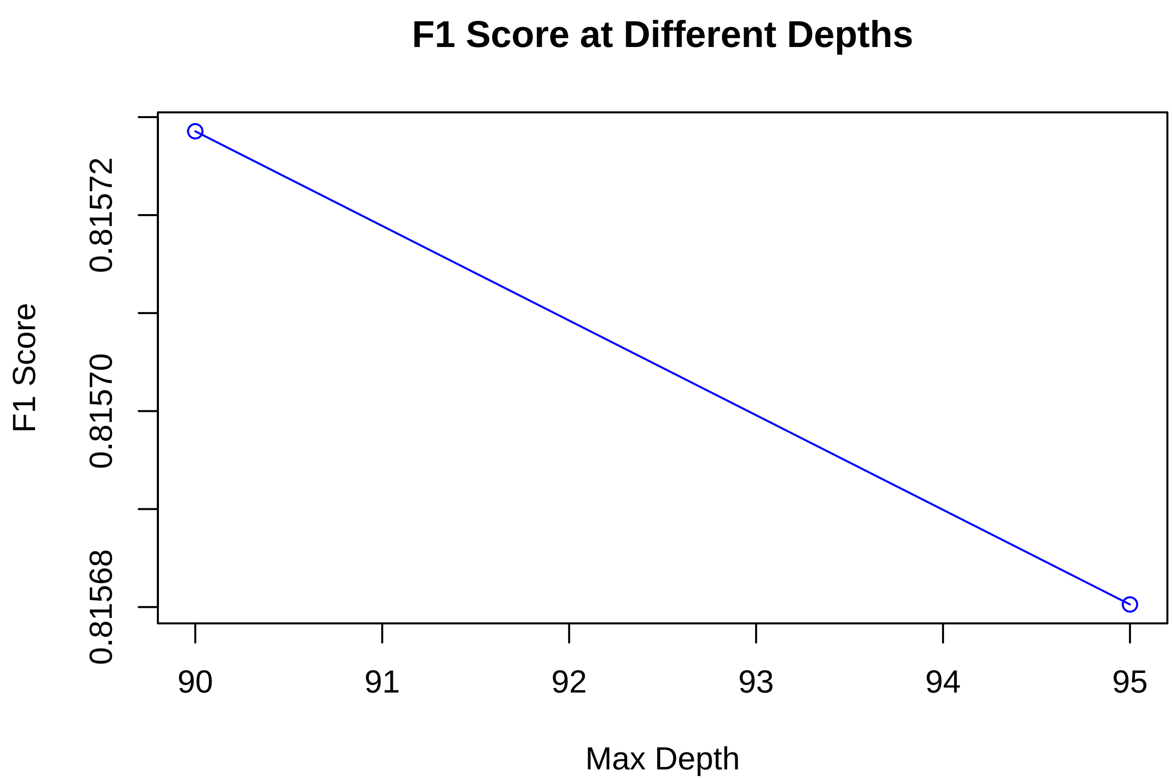
convert Qualify to factors; because in its numeric form the model assumes a regression nature.

```r
train_data$Qualify <- as.factor(train_data$Qualify)
validation_data$Qualify <- as.factor(validation_data$Qualify)
test_data$Qualify <- as.factor(test_data$Qualify)
```

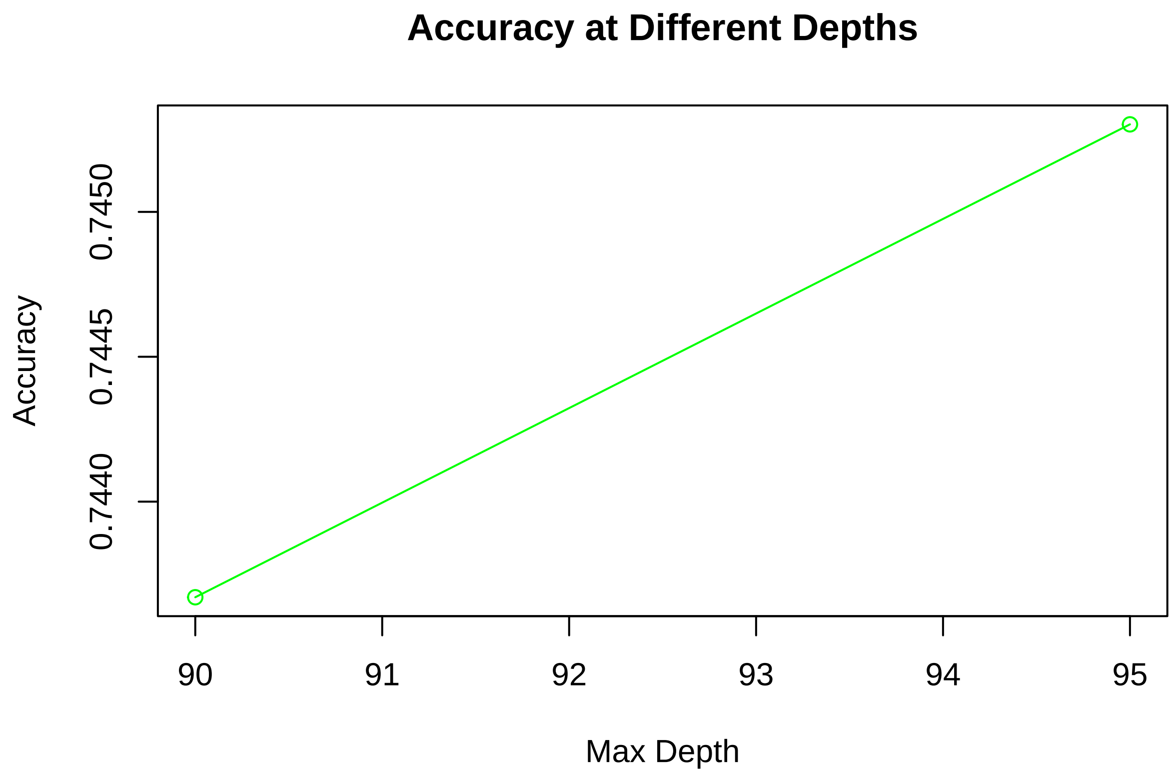
```r
for (depth in depth_values) {
  # Train the random forest model with the current depth
  model <- randomForest(formula, data = train_data, maxnodes = depth)

  # Make predictions on the validation dataset
  predictions <- predict(model, newdata = validation_data)

  # Convert predictions to factors to match the validation data's Qualify column
  predictions <- as.factor(predictions)

  # Calculate the confusion matrix
```

```r
  confusion_matrix <- confusionMatrix(predictions, validation_data$Qualify)

  # Extract Accuracy from confusion matrix
  accuracy_scores <- c(accuracy_scores, confusion_matrix$overall['Accuracy'])

  # Extract F1 score from confusion matrix (using 'F1' method from caret)
  f1_scores <- c(f1_scores, confusion_matrix$byClass['F1'])
}
```

```r
# Plot F1 Score vs Max Depth
plot(depth_values, f1_scores, type = "o", col = "blue",
     xlab = "Max Depth", ylab = "F1 Score",
     main = "F1 Score at Different Depths")
```




```r
# Plot Accuracy vs Max Depth
plot(depth_values, accuracy_scores, type = "o", col = "green",
     xlab = "Max Depth", ylab = "Accuracy",
     main = "Accuracy at Different Depths")
```



The tree depth was cycled from 0 to 95. As the depth increases; it increases the accuracy score and F1 score. However, this is a slow increase; hence the next attempt will be to tune the features from the previously generated Gini Feature importance from Milestone 3.

Feature Tuning (Gini Importance)

```
names(data_df)
```

```
##  [1] "marital_status"     "household_size"     "yrs_of_residence"
##  [4] "Annual_Salary"      "Months_Annual"      "FRS.Contribution"
##  [7] "Net_months"         "Gross_Salary"       "Gross_Months"
## [10] "Qualify"            "Education_Bach."    "Education_HS_grad"
## [13] "Education_Masters"  "Occupation_Cleric." "Occupation_Exec."
## [16] "Occupation_Prof."   "Occupation_Sales"   "age"
```

from the above Gini importance the following columns appear to have little impact on the model; hence they will be dropped: - marital_status - yrs_of_residence - Occupation_Sales - Education_Bach. - Education_Masters - Occupation_Prof. - household_size - Education_HS_grad - Occupation_Exec. - Occupation_Exec.

Drop columns with low Gini importance

```
# Drop the specified columns
data_df <- data_df %>%
  select(-marital_status, -yrs_of_residence, -Occupation_Sales,
         -Education_Bach., -Education_Masters, -Occupation_Prof.,
         -household_size, -Education_HS_grad, -Occupation_Exec.)
```

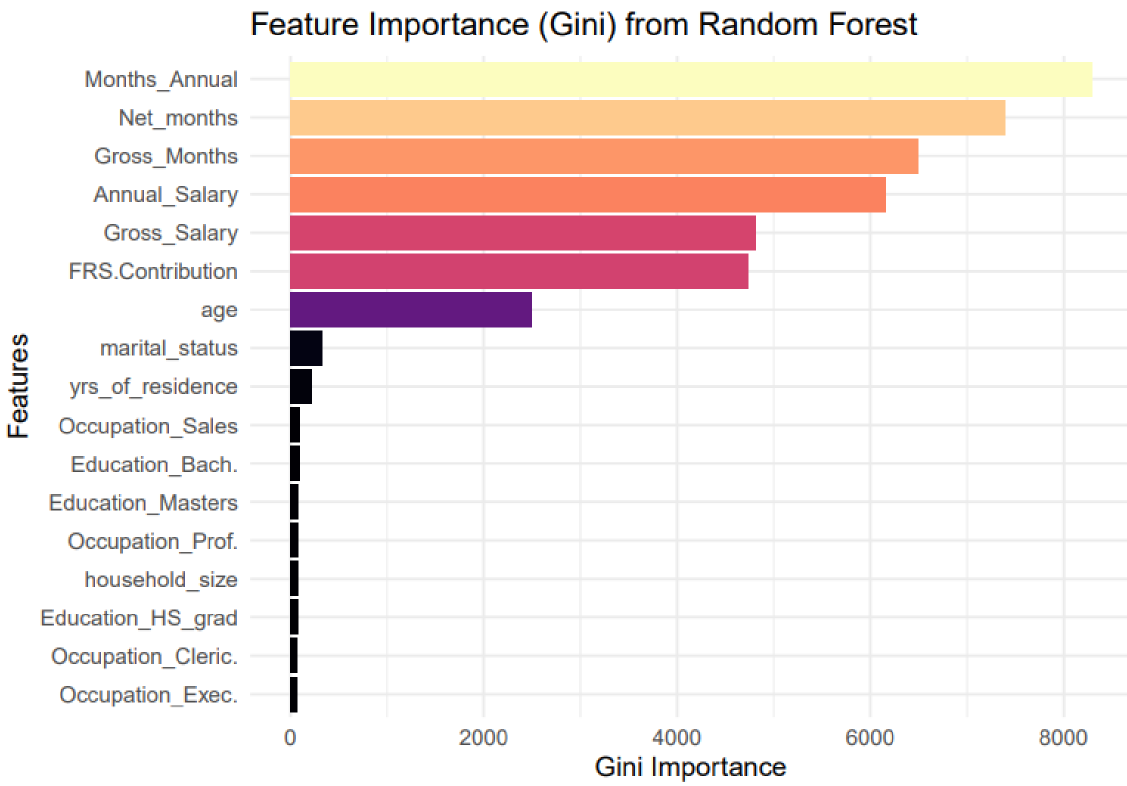


Figure 1: Ft importance

Split Data

```
#split the data
set.seed(123)
total_rows <- nrow(data_df)
#split data 70-30
train_indices <- sample(1:total_rows, 0.7 * total_rows)
train_data <- data_df[train_indices, ]
remaining_indices <- setdiff(1:total_rows, train_indices)
#testing and validation will each make up 15%
validation_indices <- sample(remaining_indices, 0.5 * length(remaining_indices))
test_indices <- setdiff(remaining_indices, validation_indices)
validation_data <- data_df[validation_indices, ]
test_data <- data_df[test_indices, ]
```

define formula without the columns:

```
# Updated formula
formula <- Qualify ~ Annual_Salary + Months_Annual + FRS.Contribution + Net_months + Gross_Salary + Gros
Occupation_Cleric. + age
```

```
#metrices
depth_values <- c(40, 45)
accuracy_scores <- c()
f1_scores <- c()
```

convert Qualify to factors; because in its numeric form the model assumes a regression nature.

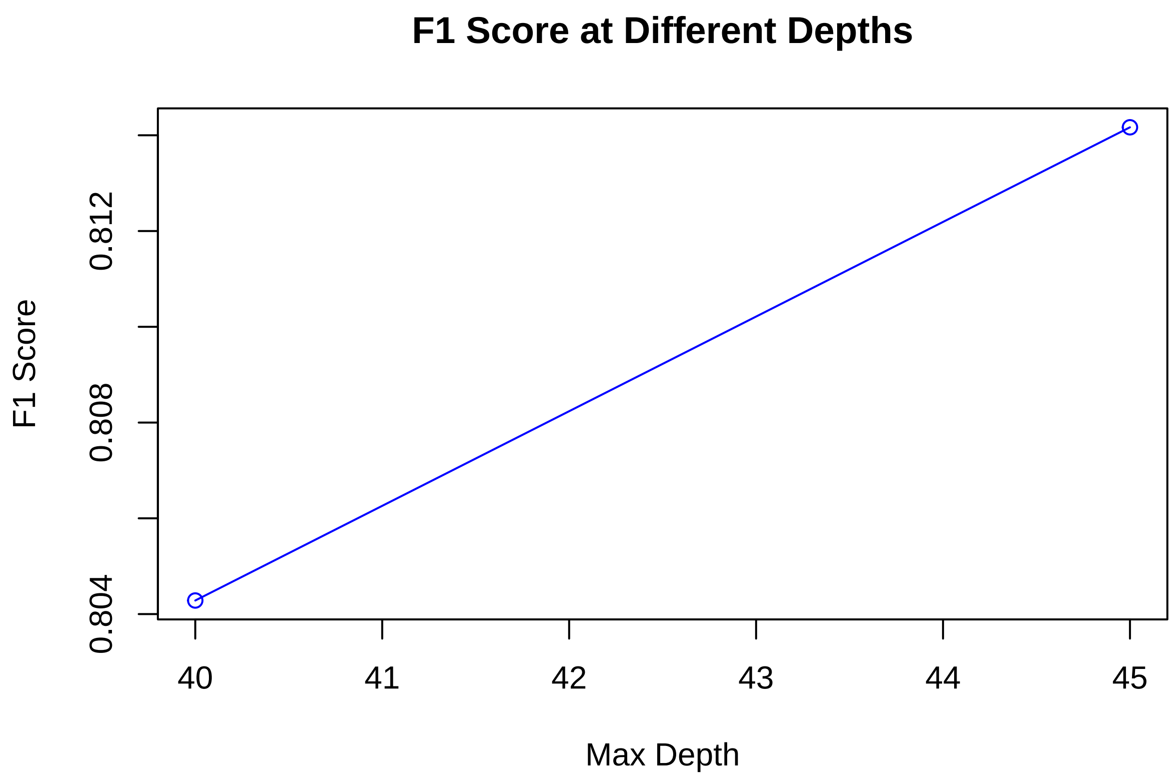
```
train_data$Qualify <- as.factor(train_data$Qualify)
validation_data$Qualify <- as.factor(validation_data$Qualify)
test_data$Qualify <- as.factor(test_data$Qualify)
```

Train the model again

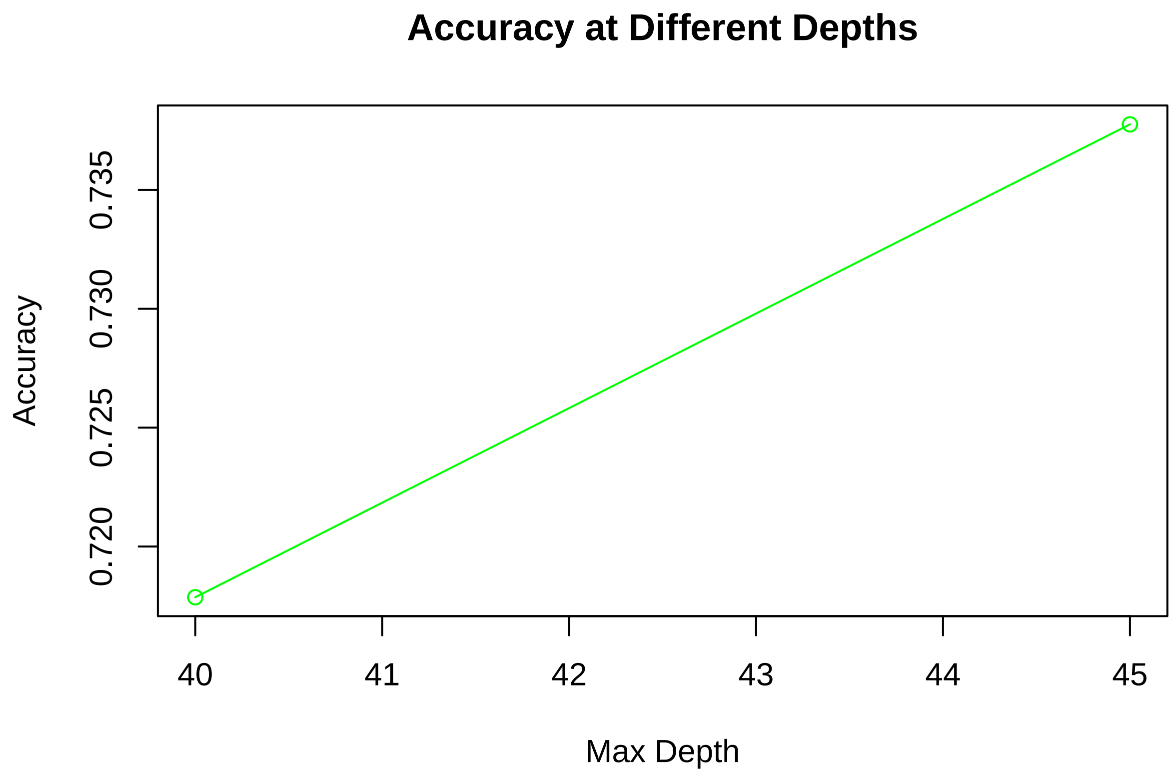
```
for (depth in depth_values) {
  # Train the random forest model with the current depth
  model <- randomForest(formula, data = train_data, maxnodes = depth)

  # Make predictions on the validation dataset
  predictions <- predict(model, newdata = validation_data)

  # Convert predictions to factors to match the validation data's Qualify column
  predictions <- as.factor(predictions)

  # Calculate the confusion matrix
  confusion_matrix <- confusionMatrix(predictions, validation_data$Qualify)

  # Extract Accuracy from confusion matrix
  accuracy_scores <- c(accuracy_scores, confusion_matrix$overall['Accuracy'])

  # Extract F1 score from confusion matrix (using 'F1' method from caret)
  f1_scores <- c(f1_scores, confusion_matrix$byClass['F1'])
}
```

```
# Plot F1 Score vs Max Depth
plot(depth_values, f1_scores, type = "o", col = "blue",
```

```r
     xlab = "Max Depth", ylab = "F1 Score",
     main = "F1 Score at Different Depths")
```




```r
# Plot Accuracy vs Max Depth
plot(depth_values, accuracy_scores, type = "o", col = "green",
     xlab = "Max Depth", ylab = "Accuracy",
     main = "Accuracy at Different Depths")
```



The F1 score and accuracy increased; up until a max depth of 40 (accuracy = 71%; f1-score = 80%); after this accuracy consistently decreased; while the f1-score increased. The Aim is to strike a balance between these two metrics.

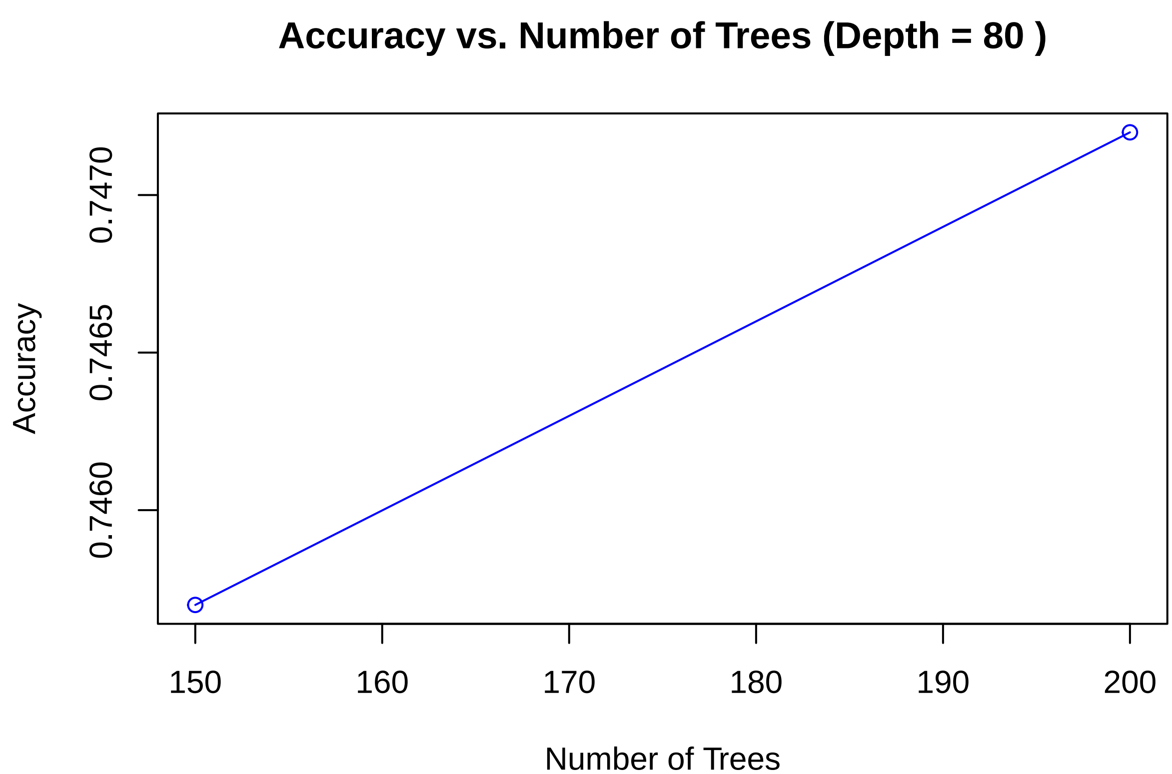
Number of Trees Parameter

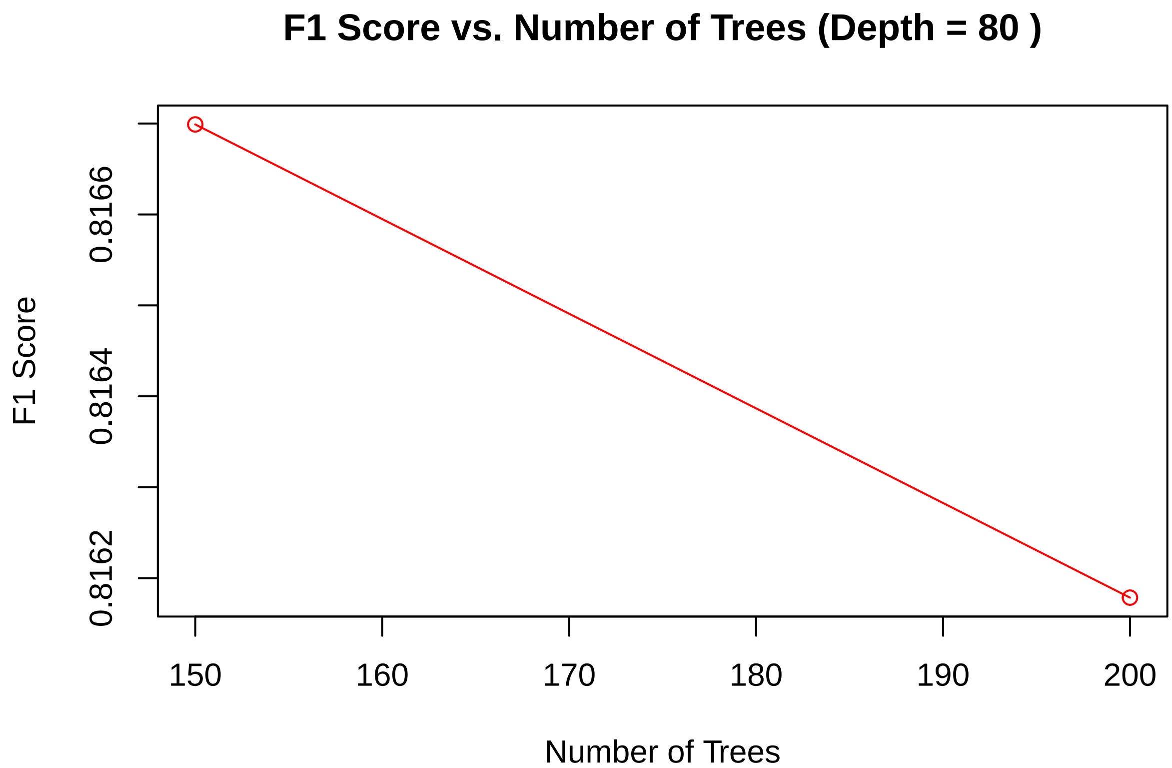
The next parameter to tune is the Number of trees parameter;along with the depth.

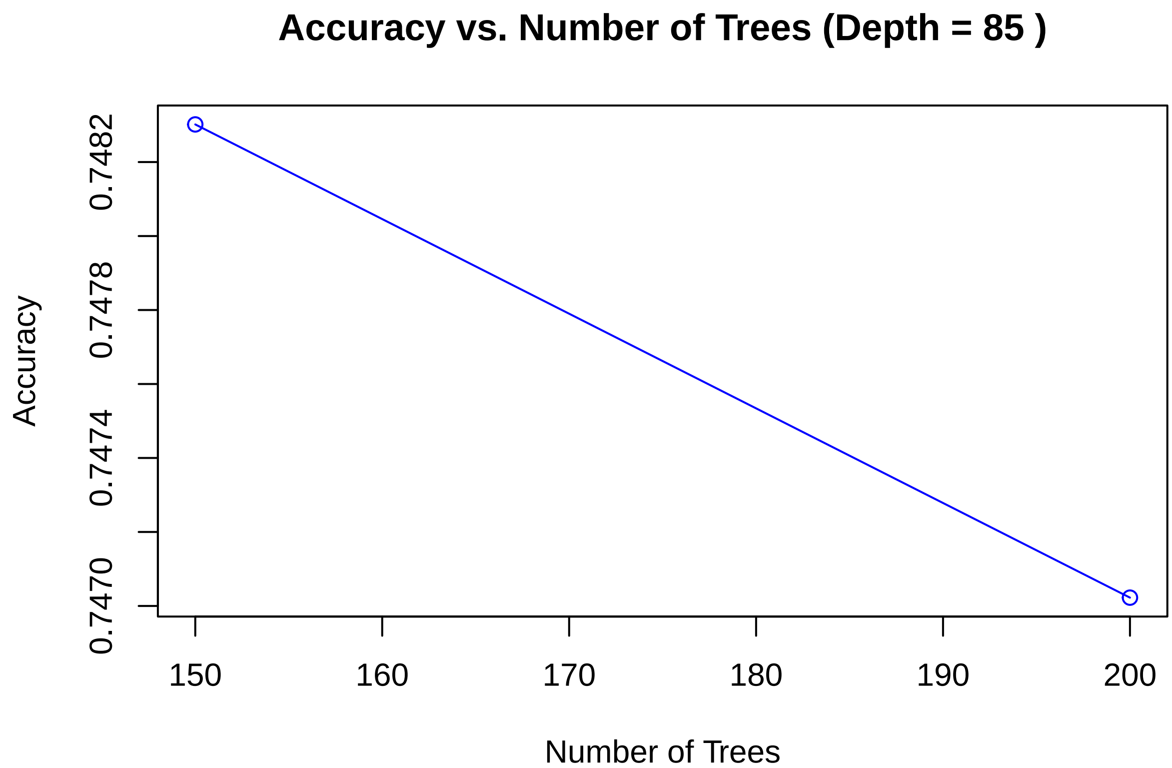
```
depth_values <- c(80, 85)
trees_values <- c(150, 200)

accuracy_scores <- list()
f1_scores <- list()
```

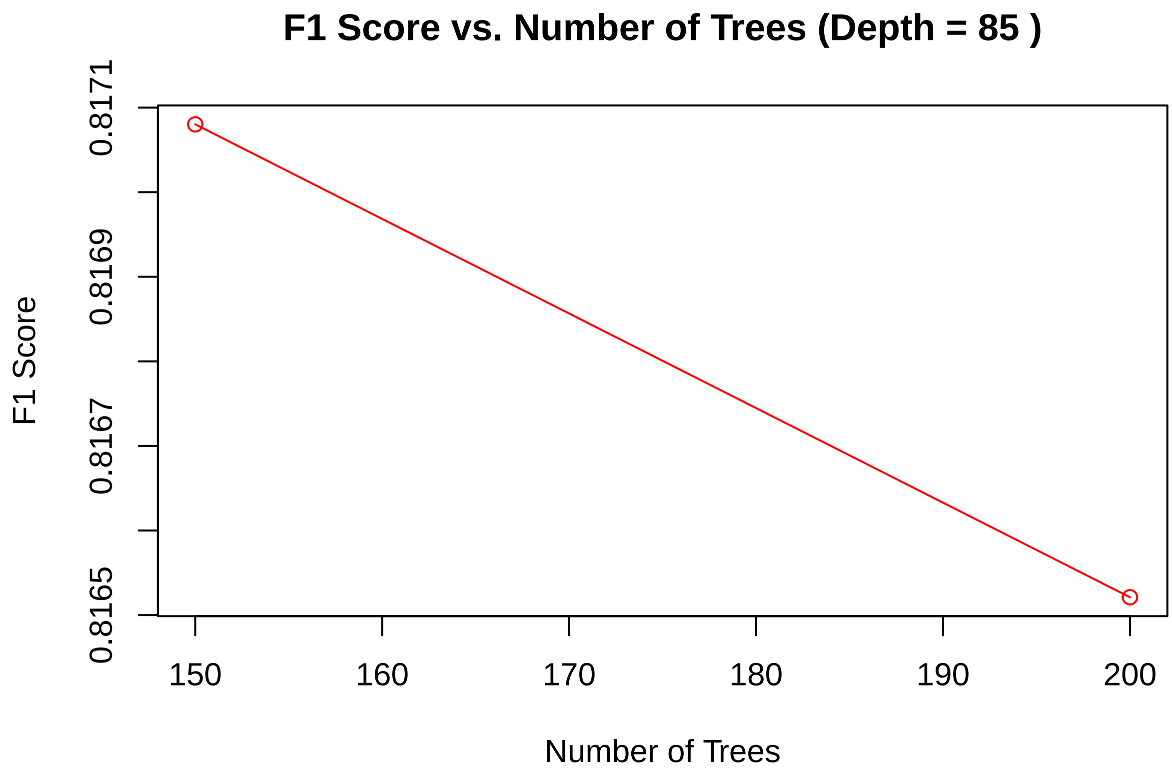
```
# Loop through each depth
for (depth in depth_values) {

  # Store metrics for each depth
  accuracy_depth <- c()
  f1_depth <- c()

  # Loop through each number of trees
  for (num_trees in trees_values) {

    # Train the Random Forest model with current depth and number of trees
    model <- randomForest(formula, data = train_data, maxnodes = depth, ntree = num_trees)

    # Make predictions on the validation dataset
```

```r
    predictions <- predict(model, newdata = validation_data)

    # Calculate confusion matrix
    confusion_matrix <- confusionMatrix(predictions, validation_data$Qualify)

    # Extract accuracy and F1 score
    accuracy_depth <- c(accuracy_depth, confusion_matrix$overall['Accuracy'])
    f1_depth <- c(f1_depth, confusion_matrix$byClass['F1'])
  }

  # Store the accuracy and F1 scores for this depth
  accuracy_scores[[as.character(depth)]] <- accuracy_depth
  f1_scores[[as.character(depth)]] <- f1_depth

  # Plot accuracy vs. number of trees for this depth
  plot(trees_values, accuracy_depth, type = "o", col = "blue",
       xlab = "Number of Trees", ylab = "Accuracy",
       main = paste("Accuracy vs. Number of Trees (Depth =", depth, ")"))

  # Plot F1 score vs. number of trees for this depth
  plot(trees_values, f1_depth, type = "o", col = "red",
       xlab = "Number of Trees", ylab = "F1 Score",
       main = paste("F1 Score vs. Number of Trees (Depth =", depth, ")"))
}
```




F1 Score vs. Number of Trees (Depth = 80)
F1 Score
0.8162
0.8164
0.8166
150
160
170
180
190
200
Number of Trees


Accuracy vs. Number of Trees (Depth = 85)
Accuracy
0.7470
0.7474
0.7478
0.7482
150
160
170
180
190
200
Number of Trees



The highest accuracy score is 74% at a depth of 85; and a number of trees of 150.This also correlated with the highest f1 score of 82%. However; these do not beat the original untuned model that will be given below.

Original untuned Model

```
model_retrained <- randomForest(Qualify ~ ., data = train_data)
```

```
# Validate the model again using the validation set
validation_predictions_retrained <- predict(model_retrained, newdata = validation_data)
```

```
validation_confusion_matrix_retrained <- table(Actual = validation_data$Qualify, Predicted = validation_
```

performance metrics:

```
accuracy_retrained <- sum(diag(validation_confusion_matrix_retrained)) / sum(validation_confusion_matri
sensitivity_retrained <- validation_confusion_matrix_retrained[2, 2] / sum(validation_confusion_matrix_
specificity_retrained <- validation_confusion_matrix_retrained[1, 1] / sum(validation_confusion_matrix_
precision_retrained <- validation_confusion_matrix_retrained[2, 2] / sum(validation_confusion_matrix_re
recall_retrained <- sensitivity_retrained
f1_score_retrained <- 2 * (precision_retrained * recall_retrained) / (precision_retrained + recall_retr
```

```
cat("Accuracy:", round(accuracy_retrained * 100, 2), "%\n")
```

```
## Accuracy: 96.9 %
```

```
cat("Sensitivity (Recall):", round(sensitivity_retrained * 100, 2), "%\n")
```

```
## Sensitivity (Recall): 95.88 %
```

```r
cat("Specificity:", round(specificity_retrained * 100, 2), "%\n")
```

```
## Specificity: 97.59 %
```

```r
cat("Precision:", round(precision_retrained * 100, 2), "%\n")
```

```
## Precision: 96.37 %
```

```r
cat("F1 Score:", round(f1_score_retrained * 100, 2), "%\n")
```

```
## F1 Score: 96.12 %
```

Metrics Interpretation

Accuracy (88.04%): - The model correctly predicted 87% of the total records in the validation set. The classes are slightly imbalanced but not significantly as to highly affect the accuracy score.

- Recall (Sensitivity) (81.5%):
  - The model's ability to correctly classify positive (1) records. The model correctly identified 81% of True positives.
  - Which means 18.5% of true positives we predicted to be negative.
  - The model correctly identified 81.5% of customers who qualify for the service.
- Specificity (92.25%):
  - The models ability to classify true negatives.
  - the model successfully identified 95% of applicants who do not qualify for the service.
- Precision (87.76%):
  - This is the measure of the accuracy of positive predictions.
- F1 Score (84.51%): This is the true measure of the performance of the model. it is the harmonic mean of precision and sensitivity. An 84% F1-Score indicates a solid model performance in predicting customer who qualify for the service and those who do not.

```r
print(names(data_df))
```

The original model performs better than the tuned model. hence we will be continuing with the original model.

```
## [1] "Annual_Salary"       "Months_Annual"       "FRS.Contribution"
## [4] "Net_months"          "Gross_Salary"        "Gross_Months"
## [7] "Qualify"             "Occupation_Cleric." "age"
```

Save updated model

```r
saveRDS(model_retrained, "retrained_rf_model2.rds")
```

Conclusion

The tuning of the Random Forest model through adjusting both depth and number of trees yielded the highest accuracy of 74% at a depth of 85 with 150 trees. This also correlated with the highest F1 score of 82%. These improvements indicate that careful tuning can lead to better performance of the model.

However; the original un-tuned model; with the newly dropped column that have a low feature importance score; still outperformed the tuned model. Hence, the original model will be carried on to the next milestones.