DTI 5126[EG]: Fundamentals for Applied Data Science

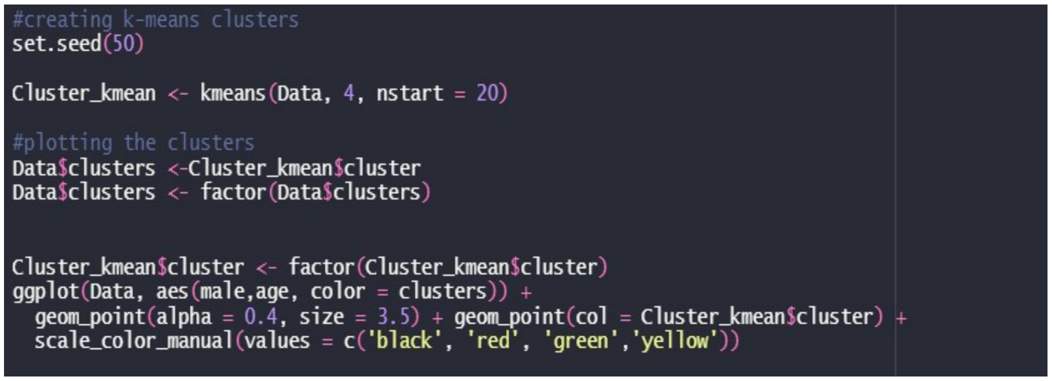
Summer 2021
Assignment 3 Clustering and model evaluation Using R

By
Omar Sorour

Part A

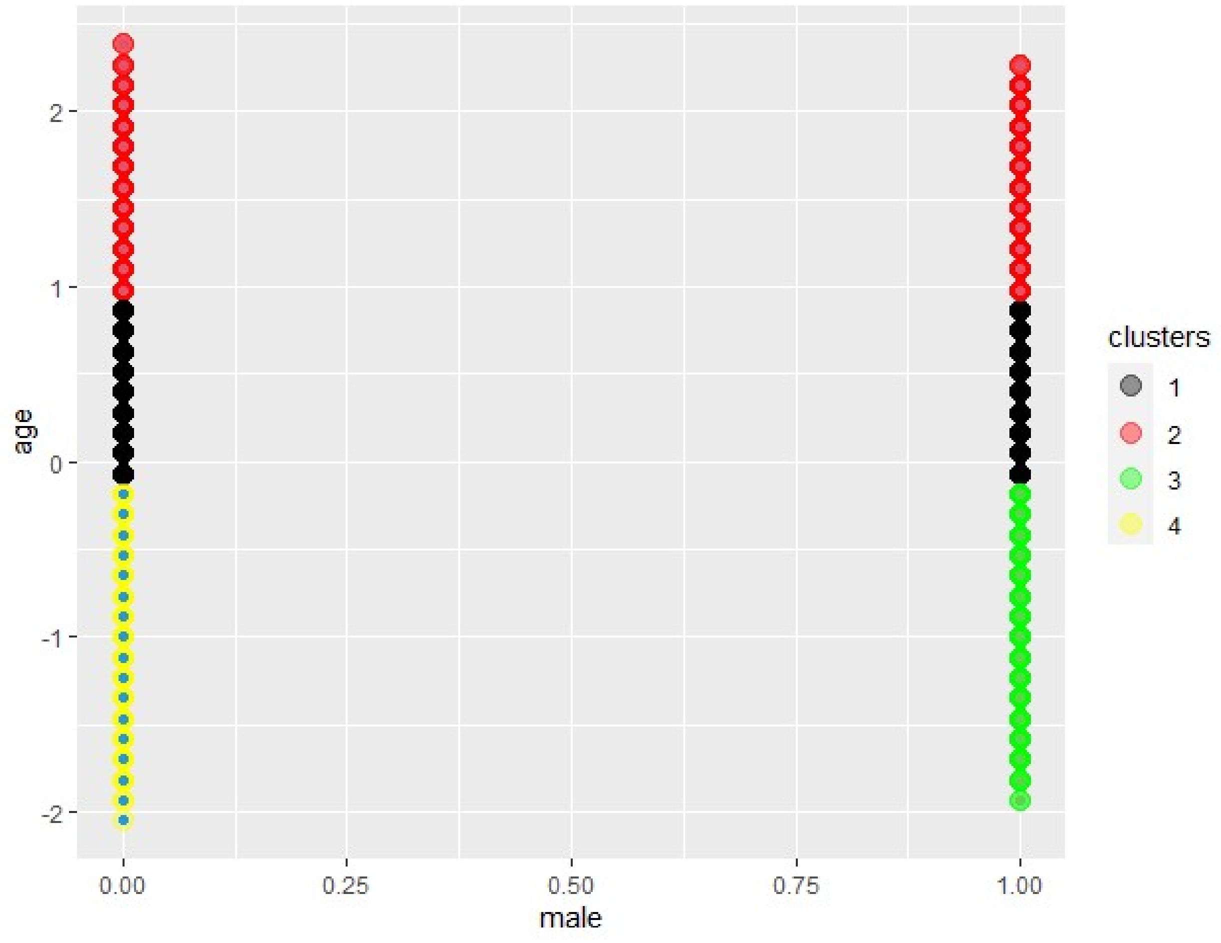
K-Means Clustering

A) Perform k-means clustering on the selected attributes, specifying k = 4 clusters and plot.

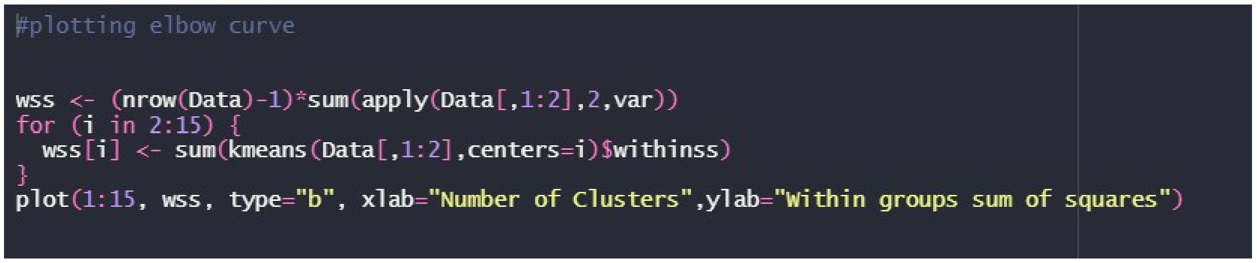


```
#creating k-means clusters
set.seed(50)

Cluster_kmean <- kmeans(Data, 4, nstart = 20)

#plotting the clusters
Data$clusters <-Cluster_kmean$cluster
Data$clusters <- factor(Data$clusters)


Cluster_kmean$cluster <- factor(Cluster_kmean$cluster)
ggplot(Data, aes(male,age, color = clusters)) +
  geom_point(alpha = 0.4, size = 3.5) + geom_point(col = Cluster_kmean$cluster) +
  scale_color_manual(values = c('black', 'red', 'green','yellow'))
```

The resulted clusters were:

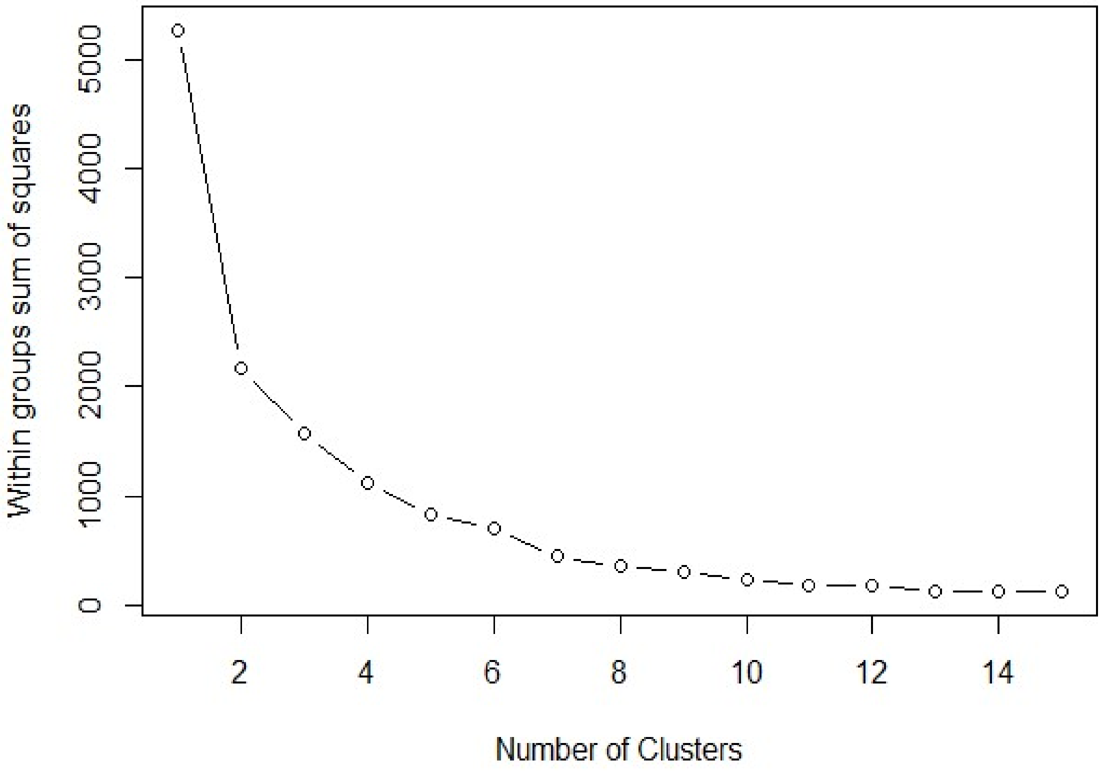


B) Apply the elbow method to determine the best k and plot.

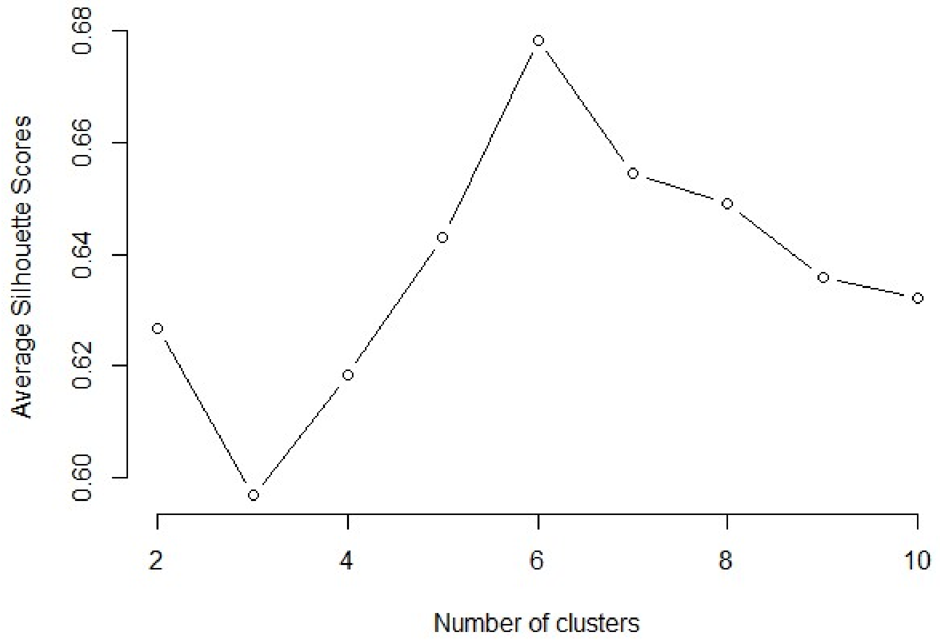


```
#plotting elbow curve


wss <- (nrow(Data)-1)*sum(apply(Data[,1:2],2,var))
for (i in 2:15) {
  wss[i] <- sum(kmeans(Data[,1:2],centers=i)$withinss)
}
plot(1:15, wss, type="b", xlab="Number of Clusters",ylab="Within groups sum of squares")
```

The results were like the following:



Since it’s little bit hard to determine the optimal K, I also plotted the silhouette score curve to make it easy for determining the optimal number of clusters.

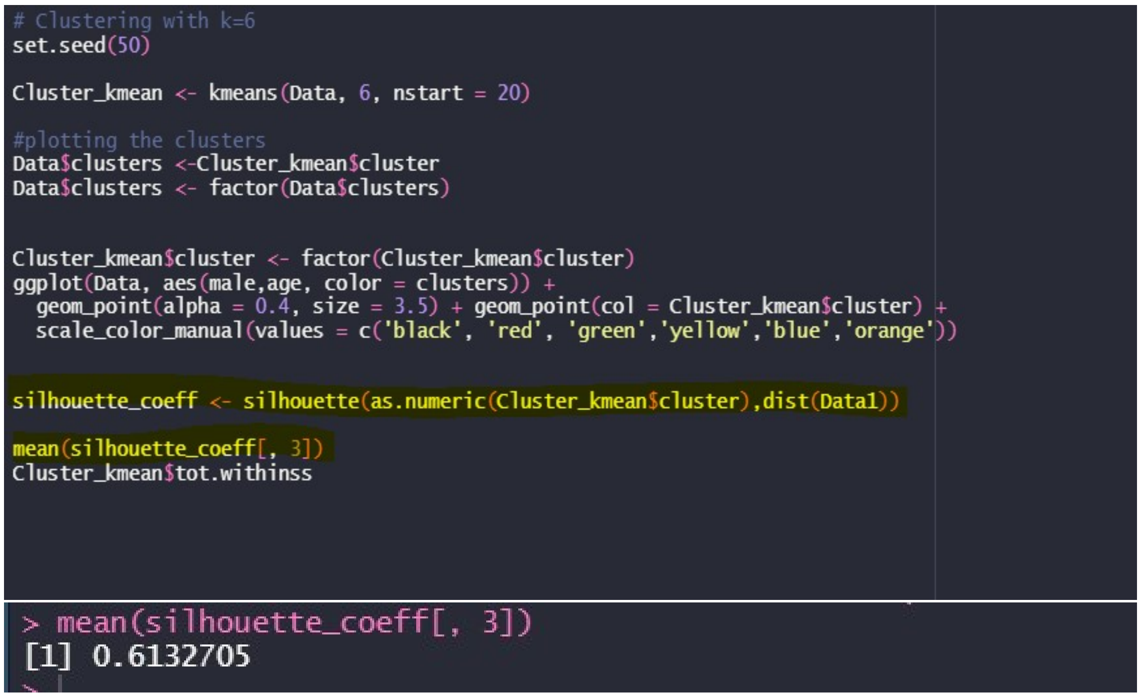


I also made a comparison between all k in range [4:6] as the following:

	Silhouette	Inertia
K=4	0.4707748	1901.969
K=5	0.5823827	796.1527
K=6	0.6132705	572.4491

From the previous we can say that the best number of clusters is **6.**

C) Evaluate the quality of the clusters using the Silhouette Coefficient method.



```r
# Clustering with k=6
set.seed(50)

Cluster_kmean <- kmeans(Data, 6, nstart = 20)

#plotting the clusters
Data$clusters <-Cluster_kmean$cluster
Data$clusters <- factor(Data$clusters)


Cluster_kmean$cluster <- factor(Cluster_kmean$cluster)
ggplot(Data, aes(male,age, color = clusters)) +
  geom_point(alpha = 0.4, size = 3.5) + geom_point(col = Cluster_kmean$cluster) +
  scale_color_manual(values = c('black', 'red', 'green','yellow','blue','orange'))

silhouette_coeff <- silhouette(as.numeric(Cluster_kmean$cluster),dist(Data1))

mean(silhouette_coeff[, 3])
Cluster_kmean$tot.withinss
```

```
> mean(silhouette_coeff[, 3])
[1] 0.6132705
```

2) Hierarchical clustering

A) Use hierarchical agglomerative clustering with single linkage.

P1=10, P2=20, P3=40, P4=80, P5=85, P6=121, P7=160, P8=168, P9=195.

First, I created the distance matrix:

	P1	P2	P3	P4	P5	P6	P7	P8	P9
P1	0								
P2	10	0							
P3	30	20	0						
P4	70	60	40	0					
P5	75	65	45	5	0				
P6	111	101	81	41	36	0			
P7	150	140	120	80	75	39	0		
P8	158	148	128	88	83	47	8	0	
P9	185	175	155	115	110	74	35	27	0

Then, I grouped the points that had the lowest distance together in one cluster, and calculated the minimum distance between that cluster and all other points out this cluster.

	P1	P2	P3	P6	P7	P8	P9	C1(P4,P5)
P1	0							
P2	10	0						
P3	30	20	0					
P6	111	101	81	0				
P7	150	140	120	39	0			
P8	158	148	128	47	8	0		
P9	185	175	155	74	35	27	0	
C1(P4, P5)	70	60	40	36	75	83	110	0

Then I repeated these two steps multiple times till I ended up with only one cluster.

	P1	P2	P3	P6	P9	C1(P4,P5)	C2(P6,P7)
P1	0						
P2	10	0					
P3	30	20	0				
P6	111	101	81	0			
P9	185	175	155	74	0		
C1(P4, P5)	70	60	45	36	110	0	
C2(P7, P8)	150	140	120	39	27	75	0

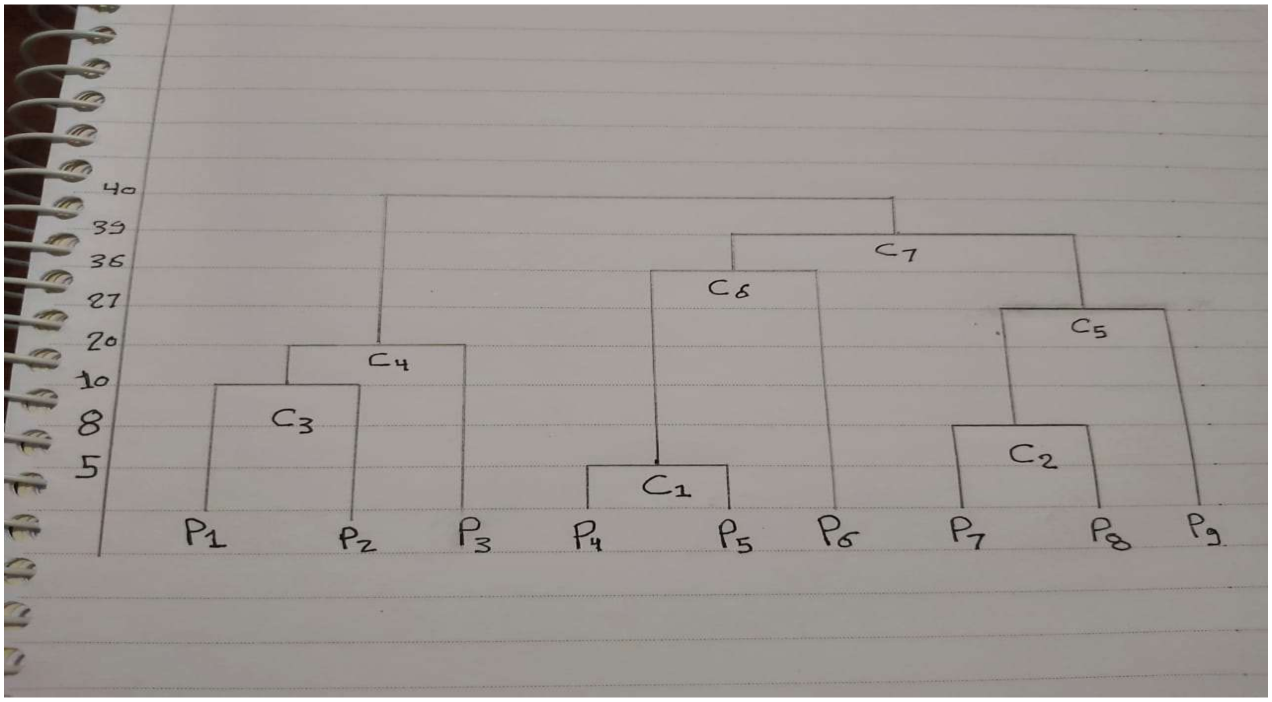
	P3	P6	P9	C1(P4,P5)	C2(P6,P7)	C3(P1,P2)
P3	0					
P6	81	0				
P9	155	74	0			
C1(P4,P5)	45	36	110	0		
C2(P7,P8)	120	39	27	75	0	
C3(P1,P2)	20	101	175	60	140	0

	P6	P9	C1(P4,P5)	C2(P6,P7)	C4(C3,P3)
P6	0				
P9	74	0			
C1(P4,P5)	36	110	0		
C2(P7,P8)	39	27	75	0	
C4(C3,P3)	81	155	45	120	0

	P6	C1(P4,P5)	C4(C3,P3)	C5(C2,P9)
P6	0			
C1(P4,P5)	36	0		
C4(C3,P3)	81	45	0	
C5(C2,P9)	39	75	120	0

	C4(C2,P9)	C7(C5,C9)
C4(C3,P3)	0	
C7(C5,C6)	39	0

C8(C4,C7) in the final cluster



As we can see points (1,2,3) had been clustered together in C4 and Points (4,5,6) in C6 and (7,8,9) in C5. Moreover, since C6 is closer to C5 then C4 they have been cluster together in this approach (single linkage).

B) hierarchical agglomerative clustering with complete linkage.

I did the same steps except that I calculated the maximum distance between any new cluster and all the points. That resulted in the same results of single linkage **EXCEPT that C4 and C6 were clustered together not C6 and C5 as in single linkage this because we took the max distance not the min.**

	P1	P2	P3	P4	P5	P6	P7	P8	P9
P1	0								
P2	10	0							
P3	30	20	0						
P4	70	60	40	0					
P5	75	65	45	5	0				
P6	111	101	81	41	36	0			
P7	150	140	120	80	75	39	0		
P8	158	148	128	88	83	47	8	0	
P9	185	175	155	115	110	74	35	27	0

	P1	P2	P3	P6	P7	P8	P9	C1(P4,P5)
P1	0							
P2	10	0						
P3	30	20	0					
P6	111	101	81	0				
P7	150	140	120	39	0			
P8	158	148	128	47	8	0		
P9	185	175	155	74	35	27	0	
C1(P4,P5)	75	65	45	41	80	88	115	0

	P1	P2	P3	P6	P9	C1(P4,P5)	C2(P7,P8)
P1	0						
P2	10	0					
P3	30	20	0				
P6	111	101	81	0			
P9	185	175	155	74	0		
C1(P4,P5)	75	65	45	41	115	0	
C2(P7,P8)	158	148	128	47	35	88	0

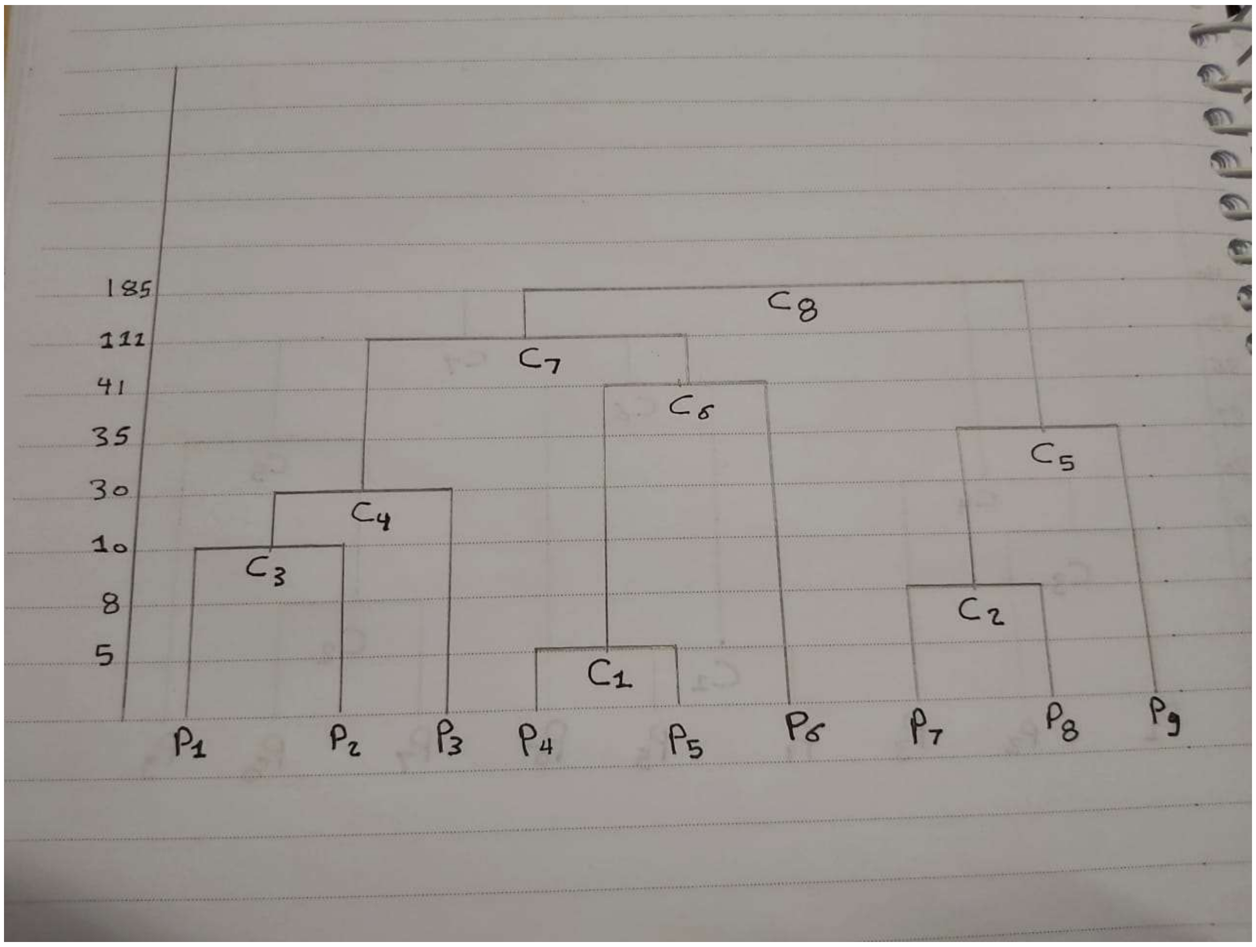
	P3	P6	P9	C1(P4,P5)	C2(P7,P8)	C3(P1,P2)
P3	0					
P6	81	0				
P9	155	74	0			
C1(P4,P5)	45	41	115	0		
C2(P7,P8)	128	47	35	88	0	
C3(P1,P2)	30	111	185	75	158	0

	P6	P9	C1(P4,P5)	C2(P7,P8)	C4(C3,P3)
P6	0				
P9	74	0			
C1(P4,P5)	41	115	0		
C2(P7,P8)	47	35	88	0	
C4(C3,P3)	111	185	45	158	0

	P6	C1(P4,P5)	C4(C3,P3)	C5(C2,P9)
P6	0			
C1(P4,P5)	41	0		
C4(C3,P3)	111	45	0	
C5(C2,P9)	74	115	185	0

	C4(C3,P3)	C5(C2,P9)	C6(C1,P6)
C4(C3,P3)	0		
C5(C2,P9)	185	0	
C6(C1,P6)	111	115	0

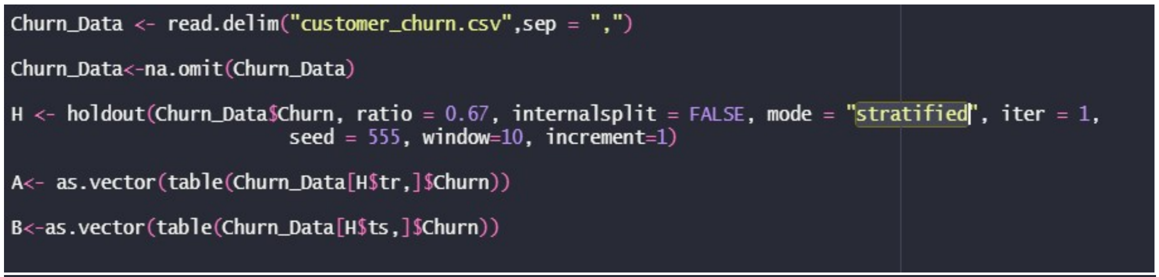
	C5(C2,P9)	C7(C4,C6)
C5(C2,P9)	0	
C7(C4,C6)	185	0



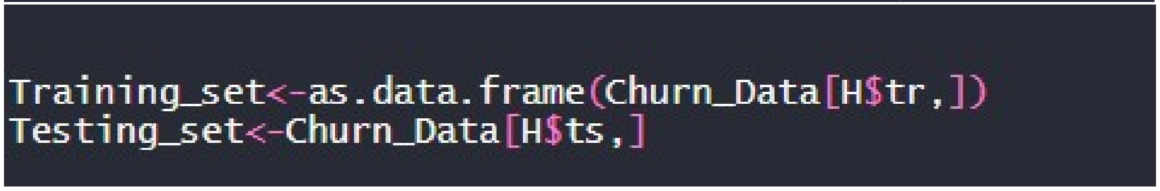
Part B

A) Partition the data set using the holdout method, so that 67% of the records are included in the training data set and 33% are included in the test data set. Use a bar graph to confirm your proportions.

I used the “holdout” function provided by “rminer” package for performing this sampling. I also set the mode of the sampling to “stratified” so that the representation of each class in the testing set and the training set remains the same as in the original data.

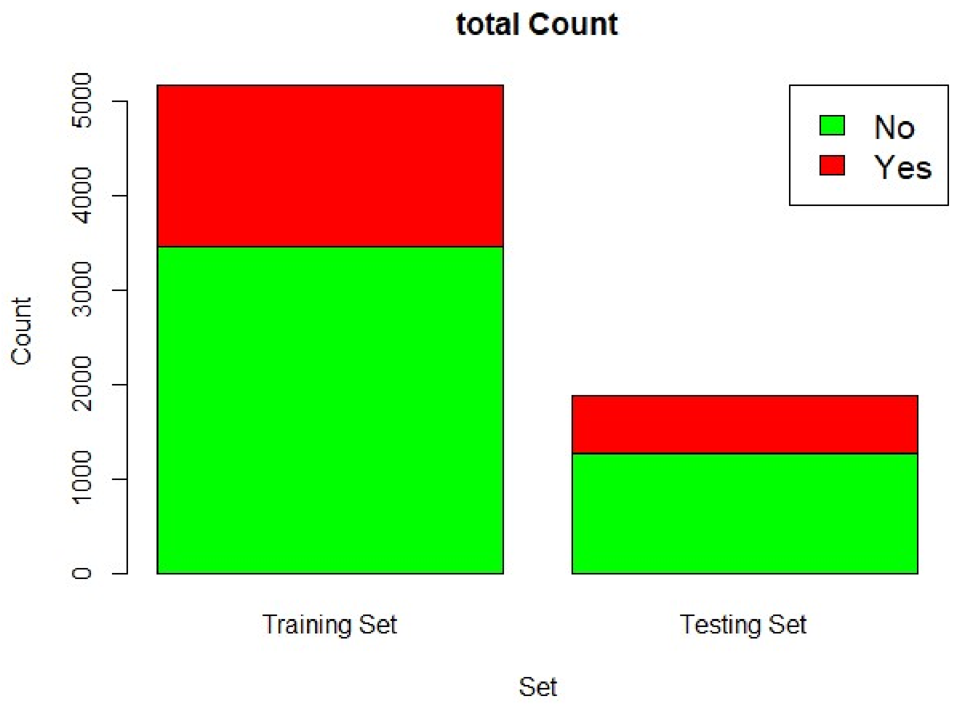


```r
Churn_Data <- read.delim("customer_churn.csv",sep = ",")

Churn_Data<-na.omit(Churn_Data)

H <- holdout(Churn_Data$Churn, ratio = 0.67, internalsplit = FALSE, mode = "stratified", iter = 1,
                  seed = 555, window=10, increment=1)

A<- as.vector(table(Churn_Data[H$tr,]$Churn))

B<-as.vector(table(Churn_Data[H$ts,]$Churn))
```



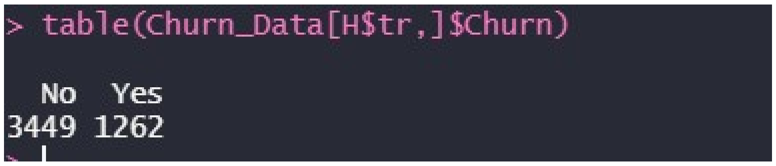
```r
Training_set<-as.data.frame(Churn_Data[H$tr,])
Testing_set<-Churn_Data[H$ts,]
```

The results were as illustrated in this bar graph



B) Identify the total number of records in the training data set and how many records in the training data set have a churn value of true (or 1). Calculate how many true churn records you need to resample in order to have 20% of the rebalanced data set have true churn values.

After sampling the data using holdout function, I had 4711 record in the training set 26% of them represented the minority class “Yes”

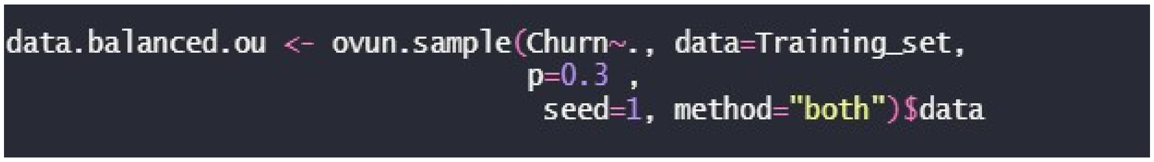


```
> table(Churn_Data[H$tr,]$Churn)

  No  Yes 
3449 1262 
> 
```

So as per Dr. Olubisi directions, we need to make the ratio of this class 30%, so this class should have 170 more record so that its ration in the 4711 record would become 30%

C) Perform the rebalancing described in (b) and confirm that 30% of the records in the rebalanced data set have true churn values.

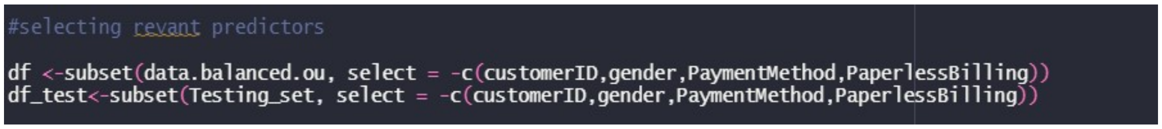
I did this by the prebuilt function “ovun.sample” provided by “ROSE” package.



```
data.balanced.ou <- ovun.sample(Churn~., data=Training_set,
                                p=0.3 ,
                                 seed=1, method="both")$data
```

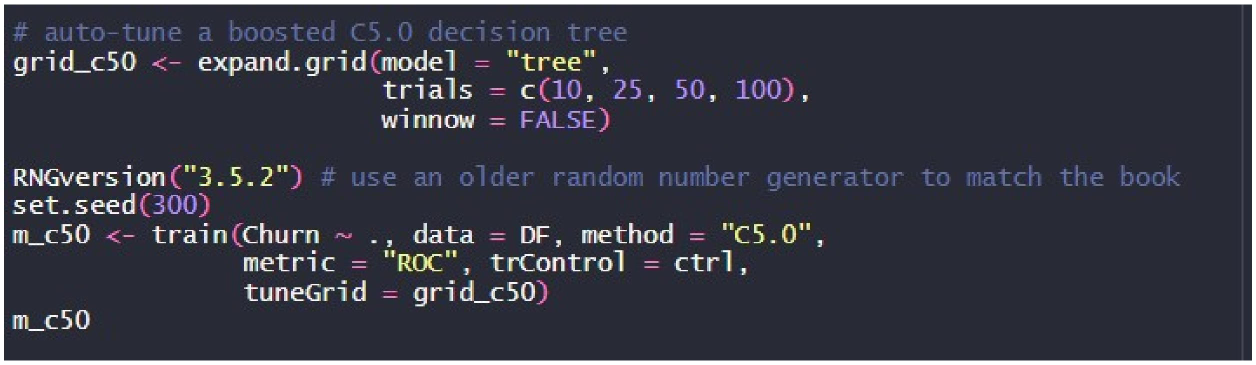
D) Create a decision tree model that can predict Churn using the data set given. Use predictors you think are appropriate and obtain the predicted value.

The attributes customerID, gender,PaymentMethod ,PaperlessBilling has no effect on whether the customer is going to churn or not, So I dropped them before training the model.



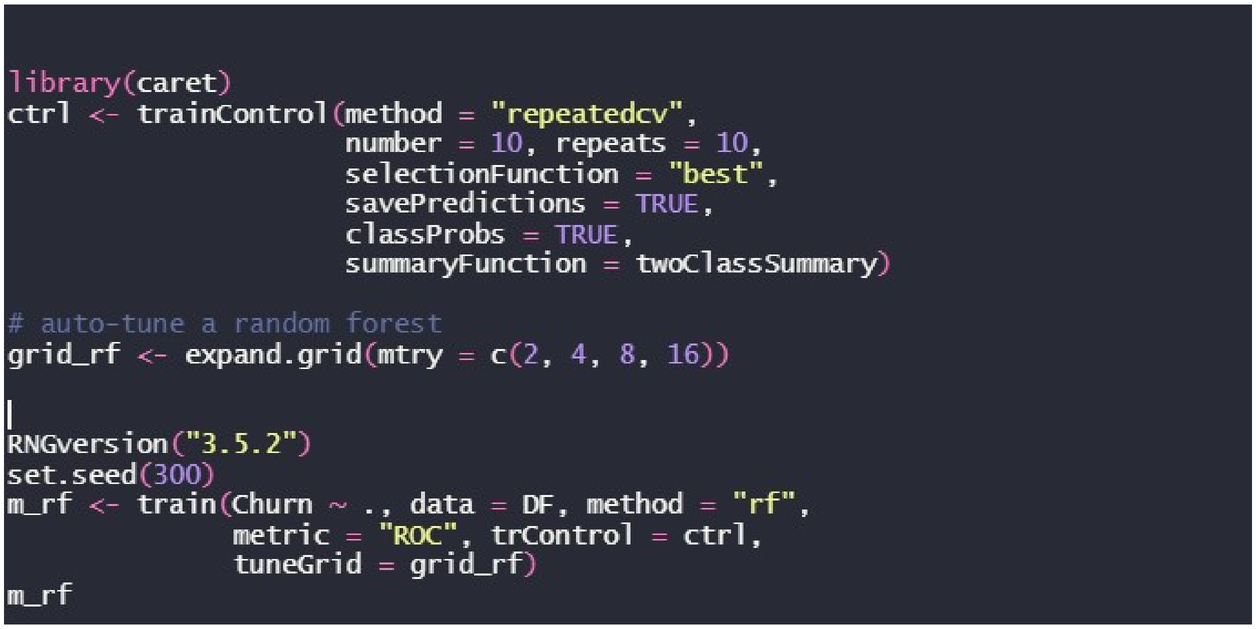
```
#selecting revant predictors

df <-subset(data.balanced.ou, select = -c(customerID,gender,PaymentMethod,PaperlessBilling))
df_test<-subset(Testing_set, select = -c(customerID,gender,PaymentMethod,PaperlessBilling))
```

Training Decision tree



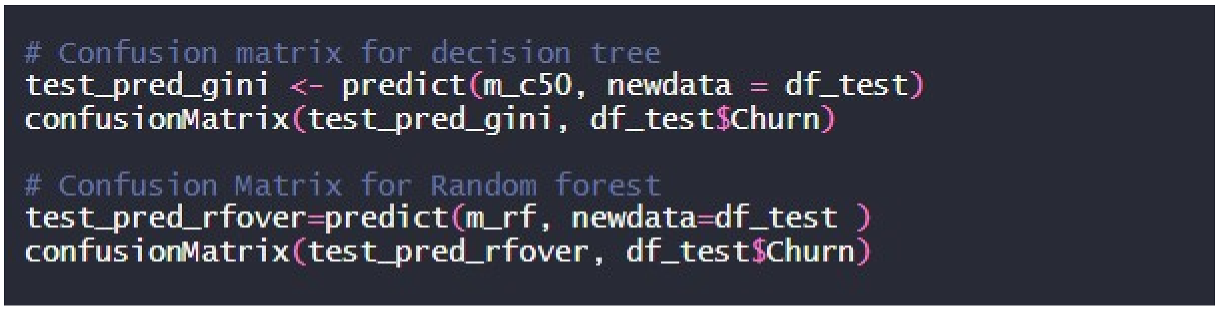
```r
# auto-tune a boosted C5.0 decision tree
grid_c50 <- expand.grid(model = "tree",
                        trials = c(10, 25, 50, 100),
                        winnow = FALSE)

RNGversion("3.5.2") # use an older random number generator to match the book
set.seed(300)
m_c50 <- train(Churn ~ ., data = DF, method = "C5.0",
               metric = "ROC", trControl = ctrl,
               tuneGrid = grid_c50)
m_c50
```

E) Use an ensemble method (e.g., Random Forest, Adaboost) to obtain the predicted value of Churn. Tune the hyper-parameters (e.g., node size, max depth, max terminal nodes, etc.) of the ensemble model and compare against the initial model.



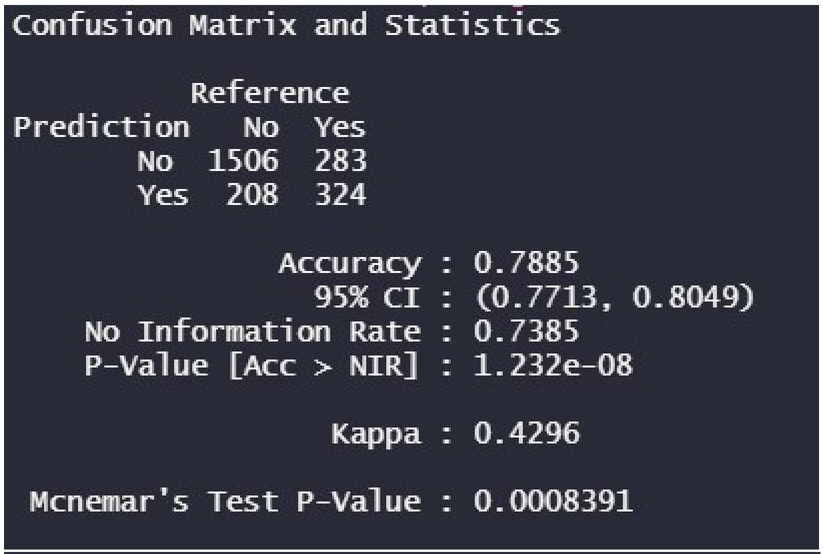
```r
library(caret)
ctrl <- trainControl(method = "repeatedcv",
                     number = 10, repeats = 10,
                     selectionFunction = "best",
                     savePredictions = TRUE,
                     classProbs = TRUE,
                     summaryFunction = twoClassSummary)

# auto-tune a random forest
grid_rf <- expand.grid(mtry = c(2, 4, 8, 16))


RNGversion("3.5.2")
set.seed(300)
m_rf <- train(Churn ~ ., data = DF, method = "rf",
              metric = "ROC", trControl = ctrl,
              tuneGrid = grid_rf)
m_rf
```

F) Using a confusion matrix, compare the evaluation measures from the ensemble method with the decision tree model based on the following criteria: Accuracy, Sensitivity and Specificity. Identify the model that performed best and worst according to each criterion.

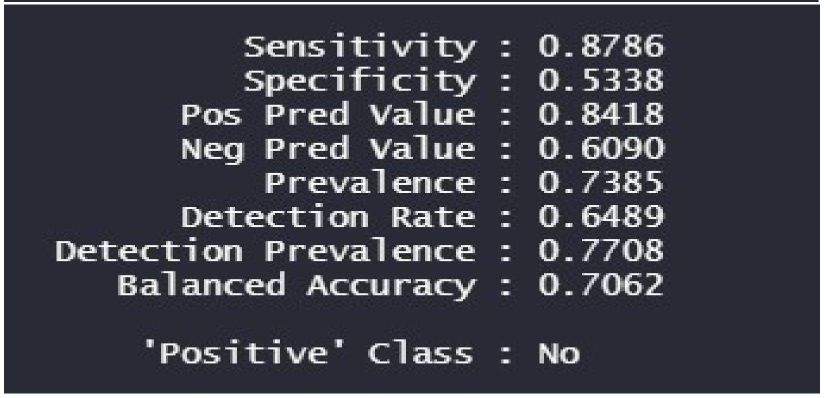


```r
# Confusion matrix for decision tree
test_pred_gini <- predict(m_c50, newdata = df_test)
confusionMatrix(test_pred_gini, df_test$Churn)

# Confusion Matrix for Random forest
test_pred_rfover=predict(m_rf, newdata=df_test )
confusionMatrix(test_pred_rfover, df_test$Churn)
```

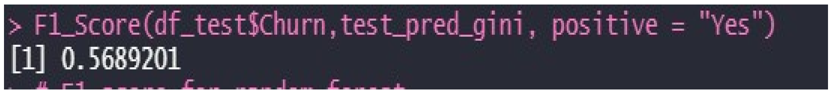
The confusion matrix for the decision tree model was:



```
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  1506  283
       Yes  208  324

               Accuracy : 0.7885
                 95% CI : (0.7713, 0.8049)
    No Information Rate : 0.7385
    P-Value [Acc > NIR] : 1.232e-08

                  Kappa : 0.4296

 Mcnemar's Test P-Value : 0.0008391
```

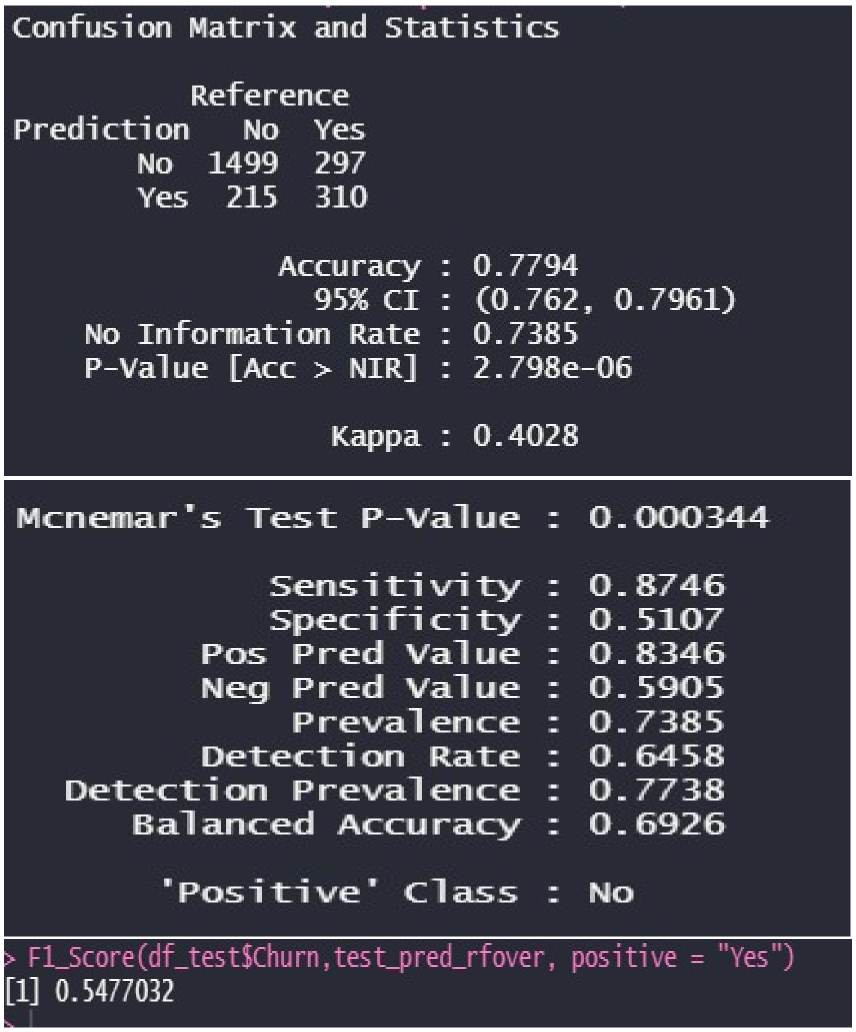


```
            Sensitivity : 0.8786
            Specificity : 0.5338
         Pos Pred Value : 0.8418
         Neg Pred Value : 0.6090
             Prevalence : 0.7385
         Detection Rate : 0.6489
   Detection Prevalence : 0.7708
      Balanced Accuracy : 0.7062

       'Positive' Class : No
```



```
> F1_Score(df_test$Churn,test_pred_gini, positive = "Yes")
[1] 0.5689201
```

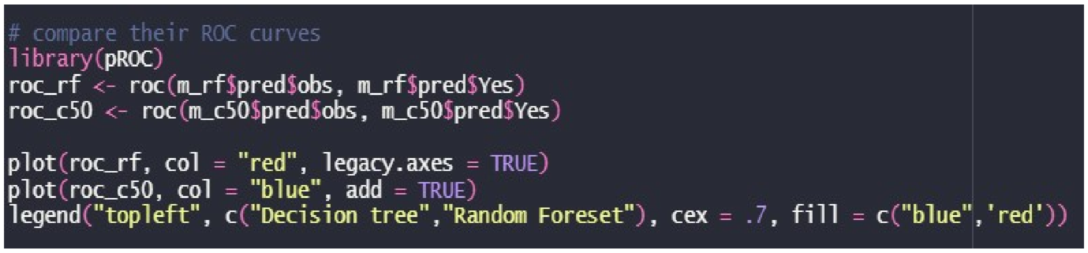
For the Random Forest model:



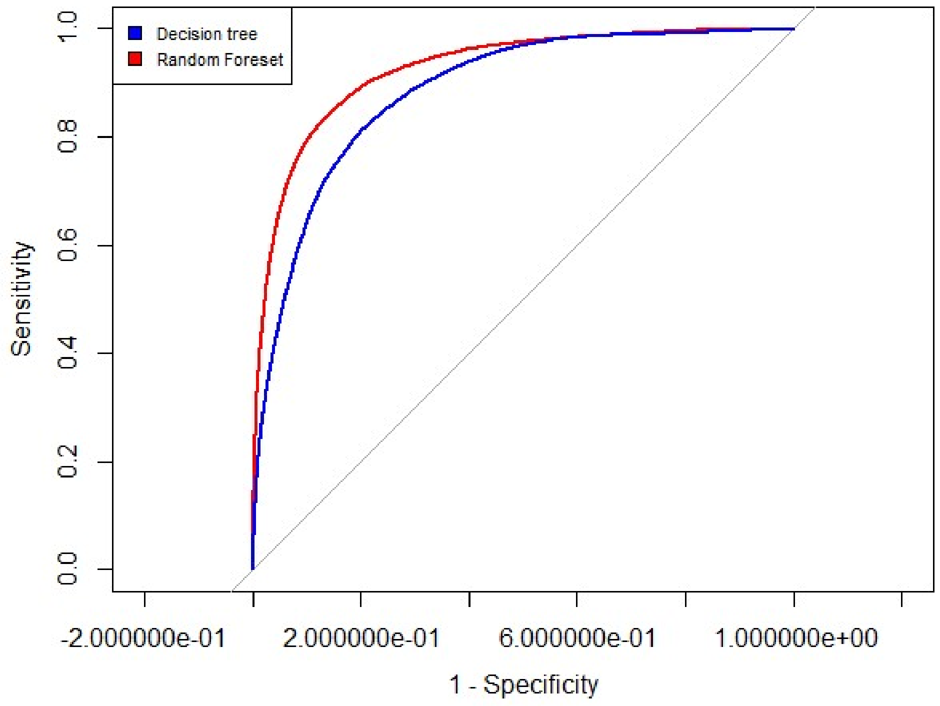
```
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  1499  297
       Yes  215  310

               Accuracy : 0.7794
                 95% CI : (0.762, 0.7961)
    No Information Rate : 0.7385
    P-Value [Acc > NIR] : 2.798e-06

                  Kappa : 0.4028

 Mcnemar's Test P-Value : 0.000344

            Sensitivity : 0.8746
            Specificity : 0.5107
         Pos Pred Value : 0.8346
         Neg Pred Value : 0.5905
             Prevalence : 0.7385
         Detection Rate : 0.6458
   Detection Prevalence : 0.7738
      Balanced Accuracy : 0.6926

       'Positive' Class : No

> F1_Score(df_test$Churn,test_pred_rfover, positive = "Yes")
[1] 0.5477032
```

From the previous we conclude that the decision performed a little bit better than the decision tree in accuracy, sensitivity, and F1 score.

G) Carry out a ROC analysis to compare the performance of the ensemble method with the decision tree technique. Plot the ROC graph of the models.



```
# compare their ROC curves
library(pROC)
roc_rf <- roc(m_rf$pred$obs, m_rf$pred$Yes)
roc_c50 <- roc(m_c50$pred$obs, m_c50$pred$Yes)

plot(roc_rf, col = "red", legacy.axes = TRUE)
plot(roc_c50, col = "blue", add = TRUE)
legend("topleft", c("Decision tree","Random Foreset"), cex = .7, fill = c("blue",'red'))
```



The curve of the random forest is slightly better than the decision tree curve.