Untitled

John Forward

2023-12-30

```
#Importing the database from SQL
library(RMySQL)

## Loading required package: DBI

USER <- 'root'
PASSWORD <- 'ENEZEjohn23@'
HOST <- 'localhost'
DBNAME <- 'world'

db <- dbConnect(MySQL(), user = USER, password = PASSWORD,
                host = HOST, dbname = DBNAME, port=3306)

World_Telcochurn <- dbGetQuery(db, statement = "select * from
world.telco_customerchurn")

dbDisconnect(db)

## [1] TRUE

#Viewing the data contents
head(World_Telcochurn)

##    CUSTOMERID COLLEGE INCOME OVERAGE LEFTOVER  HOUSE HANDSET_PRICE
## 1 BTLC-007761    zero  89318       0        0 162233           266
## 2 BTLC-007682     one 142814     187       17 346690           716
## 3 BTLC-002228    zero  55675       0       32 792662           257
## 4 BTLC-011752     one  39559       0        0 416439           165
## 5 BTLC-015958    zero 145081       0        0 341108           583
## 6 BTLC-013969     one 120631      66       17 467811           884
##   OVER_15MINS_CALLS_PER_MONTH AVERAGE_CALL_DURATION REPORTED_SATISFACTION
## 1                           1                    12                 unsat
## 2                          24                     4                 unsat
## 3                           1                     1            very_unsat
## 4                           0                    15              very_sat
## 5                           0                     9                   avg
## 6                           4                     6                   sat
##   REPORTED_USAGE_LEVEL CONSIDERING_CHANGE_OF_PLAN LEAVE
## 1          very_little                considering  STAY
## 2                 high                considering LEAVE
## 3          very_little              never_thought  STAY
## 4                 high                considering  STAY
```

```
## 5                   avg                          no LEAVE
## 6             very_high                 considering LEAVE
```

```
summary(World_Telcochurn)
```

```
##   CUSTOMERID          COLLEGE              INCOME          OVERAGE
##  Length:20000       Length:20000       Min.   : 20007   Min.   : -2.00
##  Class :character   Class :character   1st Qu.: 42217   1st Qu.:  0.00
##  Mode  :character   Mode  :character   Median : 75367   Median : 59.00
##                                        Mean   : 80281   Mean   : 85.98
##                                        3rd Qu.:115882   3rd Qu.:179.00
##                                        Max.   :159983   Max.   :335.00
##     LEFTOVER        HOUSE        HANDSET_PRICE
OVER_15MINS_CALLS_PER_MONTH
##  Min.   : 0.0   Min.   :150002   Min.   :130.0   Min.   : 0.000
##  1st Qu.: 0.0   1st Qu.:263714   1st Qu.:219.0   1st Qu.: 1.000
##  Median :14.0   Median :452260   Median :326.0   Median : 4.000
##  Mean   :23.9   Mean   :493155   Mean   :389.6   Mean   : 8.001
##  3rd Qu.:41.0   3rd Qu.:702378   3rd Qu.:533.2   3rd Qu.:15.000
##  Max.   :89.0   Max.   :999996   Max.   :899.0   Max.   :29.000
##  AVERAGE_CALL_DURATION REPORTED_SATISFACTION REPORTED_USAGE_LEVEL
##  Min.   : 1.000        Length:20000          Length:20000
##  1st Qu.: 2.000        Class :character      Class :character
##  Median : 5.000        Mode  :character      Mode  :character
##  Mean   : 6.002
##  3rd Qu.:10.000
##  Max.   :15.000
##  CONSIDERING_CHANGE_OF_PLAN    LEAVE
##  Length:20000               Length:20000
##  Class :character           Class :character
##  Mode  :character           Mode  :character
##
##
##
```

```
#Checking for missing values
World_Telcochurn <- na.omit(World_Telcochurn)
World_Telcochurn <- World_Telcochurn[, colSums(is.na(World_Telcochurn)) == 0]
```

```
#Loading the necessary Libraries
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.4.4     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.0
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────
```

```
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors

library(dplyr)
library(rpart)
library(plyr)

## ------------------------------------------------------------------------
----
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first,
then dplyr:
## library(plyr); library(dplyr)
## ------------------------------------------------------------------------
----
##
## Attaching package: 'plyr'
##
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
##
## The following object is masked from 'package:purrr':
##
##     compact

library(corrplot)

## corrplot 0.92 loaded

library(ggplot2)
library(gridExtra)

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

library(ggthemes)
library(caret)

## Loading required package: lattice
##
## Attaching package: 'caret'
##
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(randomForest)
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:gridExtra':
##
##     combine
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(party)
```

```
## Loading required package: grid
## Loading required package: mvtnorm
## Loading required package: modeltools
## Loading required package: stats4
##
## Attaching package: 'modeltools'
##
## The following object is masked from 'package:plyr':
##
##     empty
##
## Loading required package: strucchange
## Loading required package: zoo
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
```

```
##
##     as.Date, as.Date.numeric
##
## Loading required package: sandwich
##
## Attaching package: 'strucchange'
##
## The following object is masked from 'package:stringr':
##
##     boundary
##
##
## Attaching package: 'party'
##
## The following object is masked from 'package:dplyr':
##
##     where
```

```
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
##
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(glmnet)
```

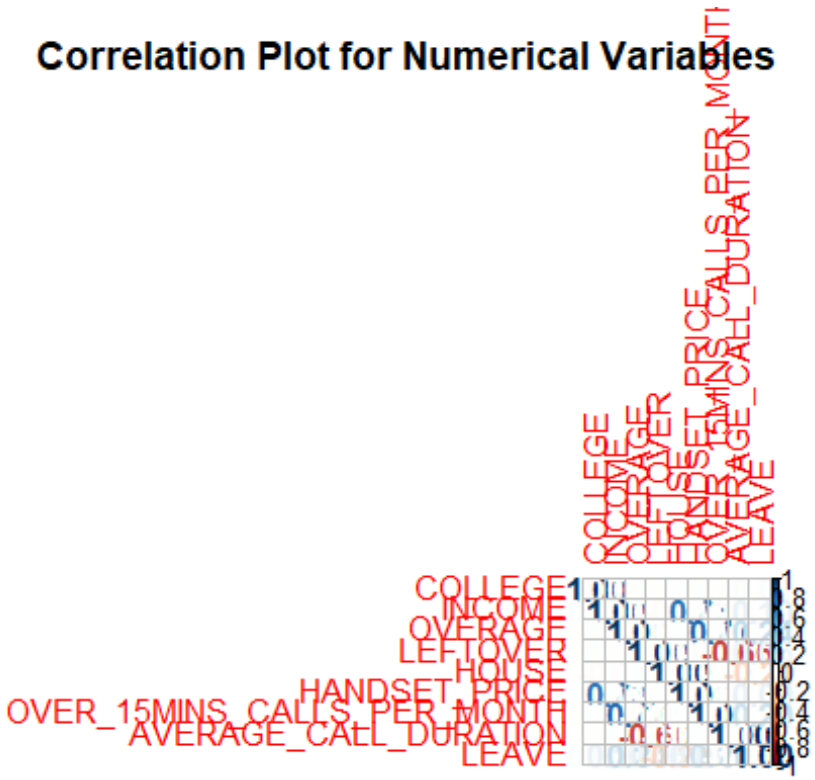
```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-8
```

```
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at
https://goo.gl/ve3WBa

library(pheatmap)
library(class)
library(caTools)
library(rpart.plot)
library(dendextend)

##
## ---------------------
## Welcome to dendextend version 1.17.1
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at:
https://github.com/talgalili/dendextend/issues
## You may ask questions at stackoverflow, use the r and dendextend tags:
##   https://stackoverflow.com/questions/tagged/dendextend
##
##  To suppress this message use:
suppressPackageStartupMessages(library(dendextend))
## ---------------------
##
##
## Attaching package: 'dendextend'
##
## The following object is masked from 'package:rpart':
##
##     prune
##
## The following object is masked from 'package:stats':
##
##     cutree

library(colorspace)

##
## Attaching package: 'colorspace'
##
## The following object is masked from 'package:pROC':
##
##     coords

library(circlize)

## ========================================
## circlize version 0.4.15
## CRAN page: https://cran.r-project.org/package=circlize
```

```
## Github page: https://github.com/jokergoo/circlize
## Documentation: https://jokergoo.github.io/circlize_book/book/
##
## If you use it in published research, please cite:
## Gu, Z. circlize implements and enhances circular visualization
##   in R. Bioinformatics 2014.
##
## This message can be suppressed by:
##   suppressPackageStartupMessages(library(circlize))
## ========================================

library(cluster)
library(data.table)

##
## Attaching package: 'data.table'
##
## The following object is masked from 'package:dendextend':
##
##     set
##
## The following objects are masked from 'package:reshape2':
##
##     dcast, melt
##
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose

#Converting our variables into factors
World_Telcochurn <- World_Telcochurn %>%
  mutate(COLLEGE = ifelse(COLLEGE == 'zero', 0, 1),
         LEAVE = ifelse(LEAVE == 'STAY', 0, 1),
         REPORTED_SATISFACTION = factor(REPORTED_SATISFACTION),
         REPORTED_USAGE_LEVEL = factor(REPORTED_USAGE_LEVEL),
         CONSIDERING_CHANGE_OF_PLAN = factor(CONSIDERING_CHANGE_OF_PLAN))

#Exploratory data analysis and feature selection
#Correlation between numerical values
numeric.var <- sapply(World_Telcochurn, is.numeric)

corr.matrix <- cor(World_Telcochurn[,numeric.var])
```

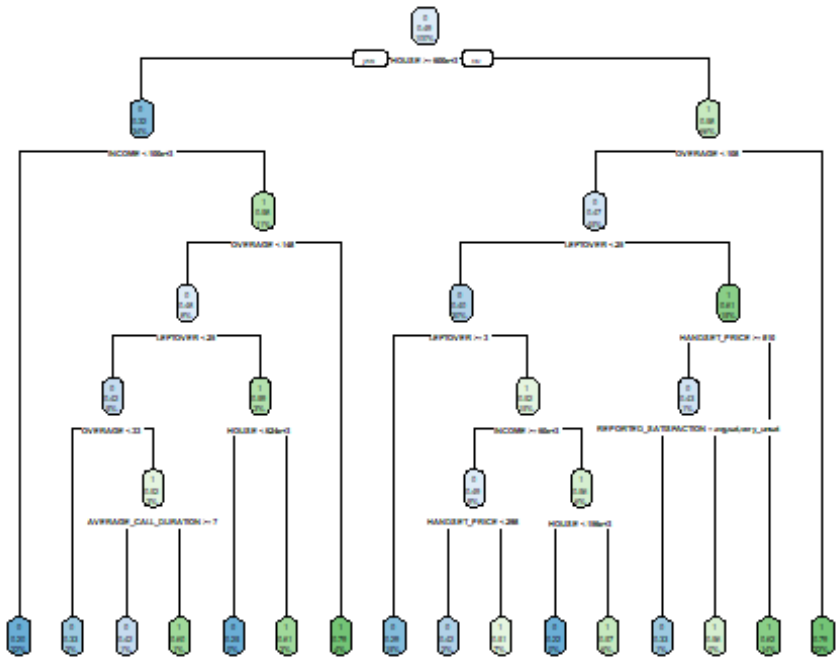
```
corrplot(corr.matrix, main="\n\nCorrelation Plot for Numerical Variables",
method="number")
```




```
#Setting seed and Splitting the data
treedata <- World_Telcochurn[, -1]

str(treedata)
```

```
## 'data.frame':    20000 obs. of  12 variables:
##  $ COLLEGE                   : num  0 1 0 1 0 1 1 0 0 0 ...
##  $ INCOME                    : int  89318 142814 55675 39559 145081
120631 59162 117488 82304 46786 ...
##  $ OVERAGE                   : int  0 187 0 0 0 66 0 53 170 44 ...
##  $ LEFTOVER                  : int  0 17 32 0 0 17 55 12 34 0 ...
##  $ HOUSE                     : int  162233 346690 792662 416439 341108
467811 251345 810740 517128 964756 ...
##  $ HANDSET_PRICE             : int  266 716 257 165 583 884 396 205 369
193 ...
##  $ OVER_15MINS_CALLS_PER_MONTH: int  1 24 1 0 0 4 1 4 26 5 ...
##  $ AVERAGE_CALL_DURATION     : int  12 4 1 15 9 6 1 4 2 8 ...
##  $ REPORTED_SATISFACTION     : Factor w/ 5 levels "avg","sat","unsat",..:
3 3 5 4 1 2 4 5 5 5 ...
##  $ REPORTED_USAGE_LEVEL      : Factor w/ 5 levels
"avg","high","little",..: 5 2 5 2 1 4 4 2 3 3 ...
##  $ CONSIDERING_CHANGE_OF_PLAN : Factor w/ 5 levels
"actively_looking_into_it",..: 2 2 3 2 4 2 2 2 1 2 ...
##  $ LEAVE                     : num  0 1 0 0 1 1 1 0 1 0 ...
```

```r
set.seed(123)
sample_split <- sample.split(treedata$LEAVE, SplitRatio = 0.70)
Train <- subset(treedata, sample_split == TRUE)
Test <- subset(treedata, sample_split == FALSE)

#Building a Decision Tree Model
decisiontree_model <- rpart(LEAVE ~ ., data = Train, method = "class",
minbucket = 5, maxdepth = 6, cp = 0.001)
predictions <- predict(decisiontree_model, Test, type = "class")
conf_matrix <- table(predictions, Test$LEAVE)
rpart.plot(decisiontree_model)
```



```r
# save the model
saveRDS(decisiontree_model, "C:/Users/johnf/Documents/BUSINESS DATA
ANALYTICS/DATA SCIENCE/Assessment/decisiontreeModel.RDS")

#Confusion Matrix, TP, FP, TN, FN
print(conf_matrix)

##
## predictions    0    1
##           0 1989  651
##           1 1055 2305

TP <- conf_matrix[2, 2]
FP <- conf_matrix[2,1]
TN <- conf_matrix[1,1]
FN <- conf_matrix[1,2]
```

```r
accuracy <- (TP + TN)/ sum(conf_matrix)
precision <- TP/(TP + FP)
recall <- TP / (TP + FN)
f1_score <- 2 * (precision * recall) / (precision + recall)

#Print the Metrics
print(paste('Accuracy:', accuracy))
```

```
## [1] "Accuracy: 0.715666666666667"
```

```r
print(paste('Precision:', precision))
```

```
## [1] "Precision: 0.686011904761905"
```

```r
print(paste('Recall:', recall))
```

```
## [1] "Recall: 0.779769959404601"
```

```r
print(paste('F1_score:', f1_score))
```

```
## [1] "F1_score: 0.729892336922103"
```

```r
#BUILDING A LOGISTIC REGRESSION MODEL
summary(Train)
```

```
##     COLLEGE           INCOME          OVERAGE          LEFTOVER
##  Min.   :0.0000   Min.   : 20009   Min.   : -2.00   Min.   : 0.00
##  1st Qu.:0.0000   1st Qu.: 42290   1st Qu.:  0.00   1st Qu.: 0.00
##  Median :1.0000   Median : 75847   Median : 59.00   Median :15.00
##  Mean   :0.5009   Mean   : 80526   Mean   : 86.12   Mean   :23.96
##  3rd Qu.:1.0000   3rd Qu.:116213   3rd Qu.:180.00   3rd Qu.:42.00
##  Max.   :1.0000   Max.   :159983   Max.   :335.00   Max.   :89.00
##      HOUSE        HANDSET_PRICE OVER_15MINS_CALLS_PER_MONTH
##  Min.   :150015   Min.   :130   Min.   : 0.000
##  1st Qu.:264002   1st Qu.:219   1st Qu.: 1.000
##  Median :451815   Median :328   Median : 4.000
##  Mean   :492462   Mean   :391   Mean   : 8.005
##  3rd Qu.:700791   3rd Qu.:536   3rd Qu.:15.000
##  Max.   :999970   Max.   :899   Max.   :29.000
##  AVERAGE_CALL_DURATION REPORTED_SATISFACTION  REPORTED_USAGE_LEVEL
##  Min.   : 1.000        avg       :1403       avg        : 697
##  1st Qu.: 2.000        sat       : 711       high       :1398
##  Median : 5.000        unsat     :2853       little     :5524
##  Mean   : 6.027        very_sat  :3510       very_high  :3593
##  3rd Qu.:10.000        very_unsat:5523       very_little:2788
##  Max.   :15.000
##             CONSIDERING_CHANGE_OF_PLAN     LEAVE
##  actively_looking_into_it:3565         Min.   :0.0000
##  considering              :5510         1st Qu.:0.0000
##  never_thought            :1406         Median :0.0000
##  no                       :2810         Mean   :0.4926
```

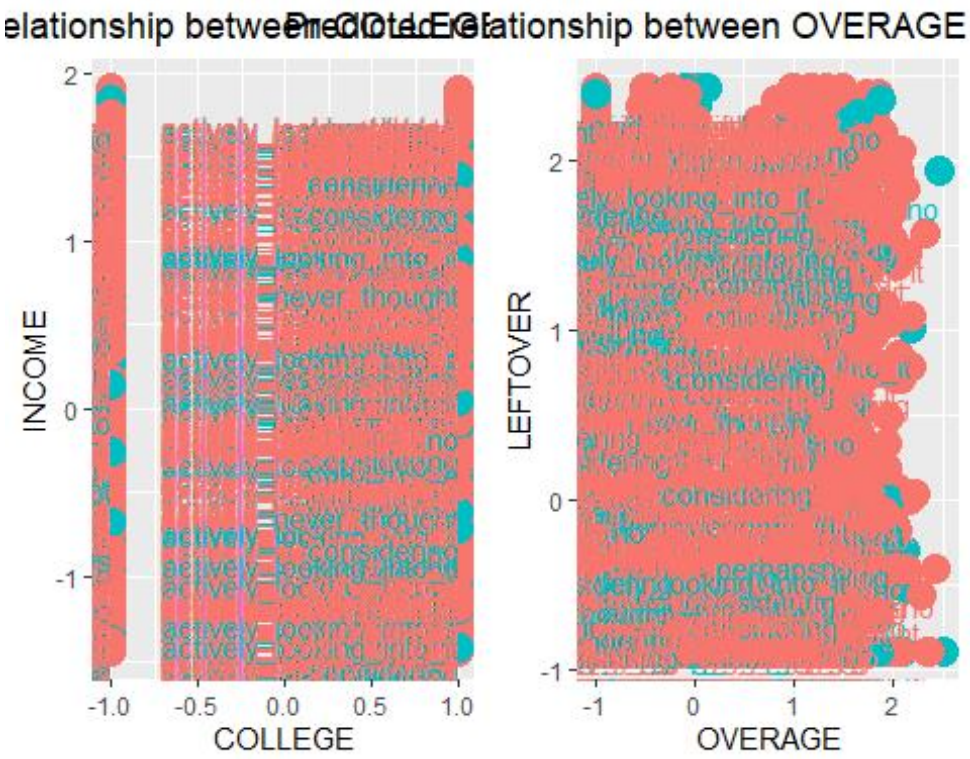
```
##  perhaps                 : 709         3rd Qu.:1.0000
##                                        Max.   :1.0000

Train$CUSTOMERID  <- NULL

Telco_model <- glm(LEAVE ~ COLLEGE + INCOME + OVERAGE + LEFTOVER + HOUSE +
HANDSET_PRICE + OVER_15MINS_CALLS_PER_MONTH + AVERAGE_CALL_DURATION +
REPORTED_SATISFACTION + REPORTED_USAGE_LEVEL + CONSIDERING_CHANGE_OF_PLAN,
                    data= Train,
                    family="binomial")

predicted_probabilities <- predict(Telco_model,
                                   newdata=Train,
                                   type="response")

summary(Telco_model)

##
## Call:
## glm(formula = LEAVE ~ COLLEGE + INCOME + OVERAGE + LEFTOVER +
##     HOUSE + HANDSET_PRICE + OVER_15MINS_CALLS_PER_MONTH +
AVERAGE_CALL_DURATION +
##     REPORTED_SATISFACTION + REPORTED_USAGE_LEVEL +
CONSIDERING_CHANGE_OF_PLAN,
##     family = "binomial", data = Train)
##
## Coefficients:
##                                        Estimate Std. Error z value
Pr(>|z|)
## (Intercept)                          -6.158e-01  1.260e-01  -4.887
1.02e-06
## COLLEGE                               6.794e-02  3.586e-02   1.894
0.058169
## INCOME                                3.404e-06  6.265e-07   5.434
5.50e-08
## OVERAGE                               5.075e-03  3.322e-04  15.275  <
2e-16
## LEFTOVER                              8.491e-03  8.894e-04   9.546  <
2e-16
## HOUSE                                -1.790e-06  7.309e-08 -24.489  <
2e-16
## HANDSET_PRICE                         3.858e-04  1.218e-04   3.168
0.001535
## OVER_15MINS_CALLS_PER_MONTH           1.066e-02  3.185e-03   3.348
0.000813
## AVERAGE_CALL_DURATION                 2.802e-02  5.399e-03   5.191
2.10e-07
## REPORTED_SATISFACTIONsat             -1.294e-01  9.819e-02  -1.318
0.187549
## REPORTED_SATISFACTIONunsat            8.399e-02  6.923e-02   1.213
```

```
0.225025
## REPORTED_SATISFACTIONvery_sat            5.824e-02  6.709e-02   0.868
0.385343
## REPORTED_SATISFACTIONvery_unsat          6.423e-02  6.356e-02   1.011
0.312236
## REPORTED_USAGE_LEVELhigh                 6.158e-02  9.838e-02   0.626
0.531318
## REPORTED_USAGE_LEVELlittle               4.852e-02  8.529e-02   0.569
0.569480
## REPORTED_USAGE_LEVELvery_high            8.457e-02  8.786e-02   0.962
0.335811
## REPORTED_USAGE_LEVELvery_little          8.690e-02  8.989e-02   0.967
0.333681
## CONSIDERING_CHANGE_OF_PLANconsidering   -1.297e-02  4.557e-02  -0.285
0.775885
## CONSIDERING_CHANGE_OF_PLANnever_thought  2.338e-02  6.680e-02   0.350
0.726367
## CONSIDERING_CHANGE_OF_PLANno             5.384e-04  5.350e-02   0.010
0.991971
## CONSIDERING_CHANGE_OF_PLANperhaps        2.952e-02  8.724e-02   0.338
0.735119
##
## (Intercept)                             ***
## COLLEGE                                 .
## INCOME                                  ***
## OVERAGE                                 ***
## LEFTOVER                                ***
## HOUSE                                   ***
## HANDSET_PRICE                           **
## OVER_15MINS_CALLS_PER_MONTH             ***
## AVERAGE_CALL_DURATION                   ***
## REPORTED_SATISFACTIONsat
## REPORTED_SATISFACTIONunsat
## REPORTED_SATISFACTIONvery_sat
## REPORTED_SATISFACTIONvery_unsat
## REPORTED_USAGE_LEVELhigh
## REPORTED_USAGE_LEVELlittle
## REPORTED_USAGE_LEVELvery_high
## REPORTED_USAGE_LEVELvery_little
## CONSIDERING_CHANGE_OF_PLANconsidering
## CONSIDERING_CHANGE_OF_PLANnever_thought
## CONSIDERING_CHANGE_OF_PLANno
## CONSIDERING_CHANGE_OF_PLANperhaps
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 19405  on 13999  degrees of freedom
## Residual deviance: 17786  on 13979  degrees of freedom
```

```
## AIC: 17828
##
## Number of Fisher Scoring iterations: 4

# save the model
saveRDS(Telco_model, "C:/Users/johnf/Documents/BUSINESS DATA ANALYTICS/DATA
SCIENCE/Assessment/Telco_model.RDS")

#Convert to 0, 1 predictions
class_prediction <- ifelse(predicted_probabilities >= 0.5, 1, 0)

# Make a table of predictions vs. actual
result_table <- table(class_prediction,
                      Train$LEAVE)

result_table

##
## class_prediction    0    1
##                0 4699 2703
##                1 2405 4193

#Confusion Matrix For Logistic Regression
print(result_table)

##
## class_prediction    0    1
##                0 4699 2703
##                1 2405 4193

LGR_TP <- result_table[2, 2]
LGR_FP <- result_table[2,1]
LGR_TN <- result_table[1,1]
LGR_FN <- result_table[1,2]

LGR_accuracy <- (LGR_TP + LGR_TN)/ sum(result_table)
LGR_precision <- LGR_TP/(LGR_TP + LGR_FP)
LGR_recall <- LGR_TP / (LGR_TP + LGR_FN)
LGR_f1_score <- 2 * (LGR_precision * LGR_recall) / (LGR_precision +
LGR_recall)

#Print the Metrics
print(paste('Accuracy:', LGR_accuracy))

## [1] "Accuracy: 0.635142857142857"

print(paste('Precision:', LGR_precision))

## [1] "Precision: 0.635495604728706"

print(paste('Recall:', LGR_recall))
```

```
## [1] "Recall: 0.608033642691415"

print(paste('F1_score:', LGR_f1_score))

## [1] "F1_score: 0.621461390247517"

## Assigning the Leave variable into a matrix
LEAVE_labels = World_Telcochurn[,12]

# Encoding the target feature as factor
World_Telcochurn$LEAVE <- as.numeric(World_Telcochurn$LEAVE)

# Identify Categorical Variables
categorical_vars <- c("REPORTED_SATISFACTION", "REPORTED_USAGE_LEVEL",
"CONSIDERING_CHANGE_OF_PLAN")

# Convert to Factors
World_Telcochurn[, categorical_vars] <- lapply(World_Telcochurn[,
categorical_vars], as.factor)

# One-Hot Encoding
encoded_data <- model.matrix(~ . - 1, data = World_Telcochurn[,
categorical_vars])

# Combine Data
World_Telcochurn <- cbind(World_Telcochurn, encoded_data)

# Remove the original categorical variables
World_Telcochurn <- World_Telcochurn[, !(names(World_Telcochurn) %in%
categorical_vars)]

# Select only numeric columns for scaling
numeric_cols <- sapply(World_Telcochurn, is.numeric)
scaled_data <- scale(World_Telcochurn[, numeric_cols])

# Convert the scaled data back to a dataframe
scaled_World_Telcochurn <- as.data.frame(scaled_data)

# Split into test and train 80/20
set.seed(123)

size <- floor(0.8 *  nrow(scaled_World_Telcochurn))

train_ind <- sample(seq_len(nrow(scaled_World_Telcochurn)), size = size)

train_labels <- scaled_World_Telcochurn[train_ind, 12]

knn_train <- scaled_World_Telcochurn[train_ind,1:22]
knn_test <- scaled_World_Telcochurn[-train_ind,1:22]
```

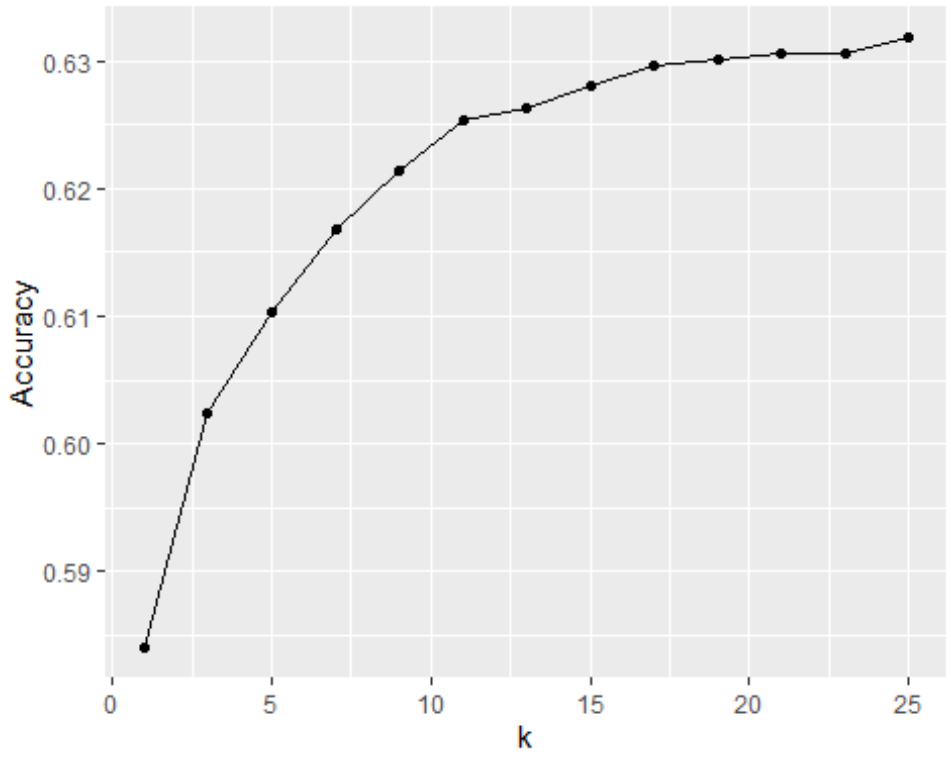
```r
test_labels <- LEAVE_labels[-train_ind]

# Fit KNN Model
predictions <- knn(train = knn_train,
                   test = knn_test,
                   cl = train_labels,
                   k= round(sqrt(nrow(knn_train))))


# Create a dataframe for plotting predictions
plot_predictions <- cbind(knn_test, predicted = predictions)

view(plot_predictions)




require(gridExtra)

p1 <- ggplot(plot_predictions, aes(COLLEGE, INCOME, color = predicted, fill =
predicted)) +
  geom_point(size = 5) +
  geom_text(aes(label=test_labels),hjust=1, vjust=2) +
  ggtitle("Predicted relationship between COLLEGE AND INCOME") +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme(legend.position = "none")

p2 <- ggplot(plot_predictions, aes(OVERAGE, LEFTOVER, color = predicted, fill
= predicted)) +
  geom_point(size = 5) +
  geom_text(aes(label=test_labels),hjust=1, vjust=2) +
  ggtitle("Predicted relationship between OVERAGE AND LEFTOVER")+
  theme(plot.title = element_text(hjust = 0.5)) +
  theme(legend.position = "none")

grid.arrange(p1, p2, ncol=2)
```



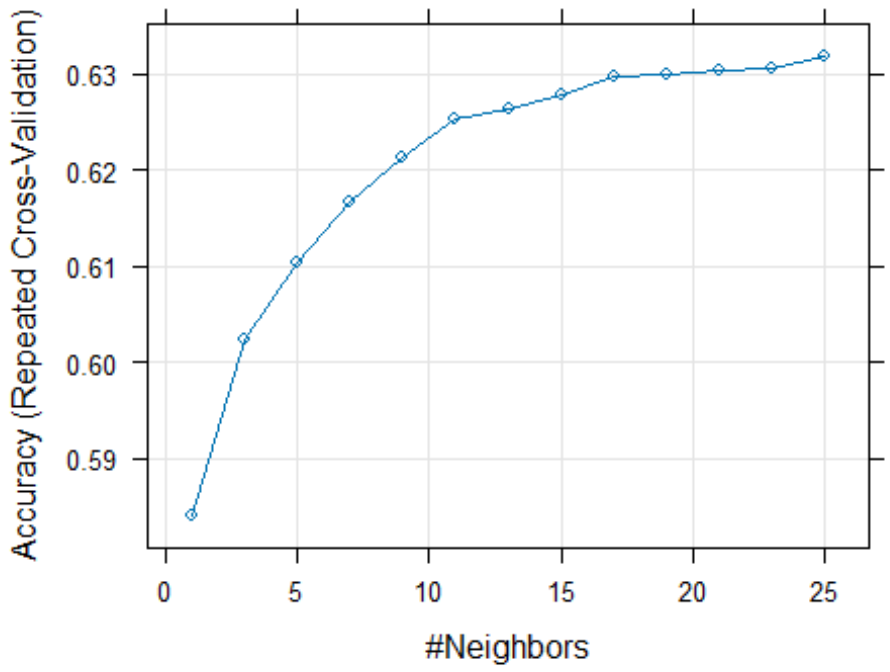

```
## Remove Customer ID
World_Telcochurn$CUSTOMERID  <- NULL

#TASK 4: Build a KNN Model;
KNNdata<-World_Telcochurn
KNNdata<- KNNdata %>% mutate( LEAVE = factor(LEAVE))

## Scaling The Data
KNNdata[,2:8]<-scale(KNNdata[,2:8])

## Splitting the data
set.seed(123)
intrain<-createDataPartition(KNNdata$LEAVE, p=0.70, list = FALSE)
KNNTrainData<-KNNdata[intrain,]
KNNTestData<-KNNdata[-intrain,]

?knn

## starting httpd help server ... done

Grid_values<- expand.grid(k=seq(1, 25, by =2))

KnnModel<- train(LEAVE~.,data = KNNTrainData, method = 'knn',
    preProcess= c('center', 'scale'),
    trControl= trainControl(method = 'repeatedcv',number =10, repeats = 5),
tuneGrid = Grid_values)
KnnModel
```

```
## k-Nearest Neighbors
##
## 14001 samples
##    21 predictor
##     2 classes: '0', '1'
##
## Pre-processing: centered (21), scaled (21)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 12601, 12601, 12602, 12600, 12601, 12601, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.5840728  0.1678795
##    3  0.6024408  0.2044767
##    5  0.6102838  0.2200513
##    7  0.6167410  0.2327255
##    9  0.6214257  0.2418743
##   11  0.6253684  0.2496148
##   13  0.6263395  0.2513725
##   15  0.6279688  0.2545138
##   17  0.6296969  0.2578875
##   19  0.6300396  0.2584904
##   21  0.6304968  0.2593515
##   23  0.6306105  0.2595151
##   25  0.6318536  0.2619353
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 25.
```

```
#Plot the Model
KnnResult <- KnnModel$results
KnnResult |> ggplot(aes(x = k, y = Accuracy)) + geom_point() + geom_line()
```




```
plot(KnnModel)
```




```
confusionMatrix(KnnModel)
```

```
## Cross-Validated (10 fold, repeated 5 times) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    0    1
##          0 36.0 22.1
##          1 14.7 27.2
##
##  Accuracy (average) : 0.6319
```

```
# save the model
saveRDS(KnnModel, "C:/Users/johnf/Documents/BUSINESS DATA ANALYTICS/DATA
SCIENCE/Assessment/KnnModel.RDS")

# Make predictions on test data
KnnPredictions <- predict(KnnModel, newdata = KNNTestData)

# Generate confusion matrix
Knnconfusionmat <- confusionMatrix(data = KnnPredictions, reference =
KNNTestData$LEAVE)

# Extracting metrics
KNNaccuracy <- Knnconfusionmat$overall['Accuracy']
KNNprecision <- Knnconfusionmat$byClass['Precision']
KNNrecall <- Knnconfusionmat$byClass['Recall']
KNNF1_score <- Knnconfusionmat$byClass['F1']

# Displaying metrics
KNNaccuracy
```

```
##  Accuracy
## 0.6204367
```

```
KNNprecision
```

```
## Precision
## 0.6106144
```

```
KNNrecall
```

```
##    Recall
## 0.6954665
```

```
KNNF1_score
```

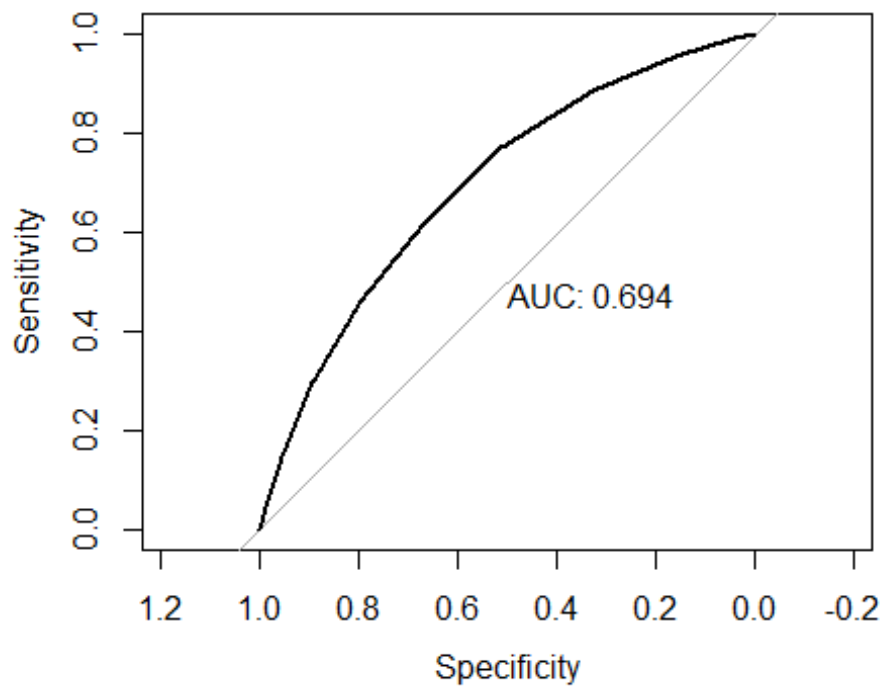
```
##        F1
## 0.6502841
```

```
Knnconfusionmat
```

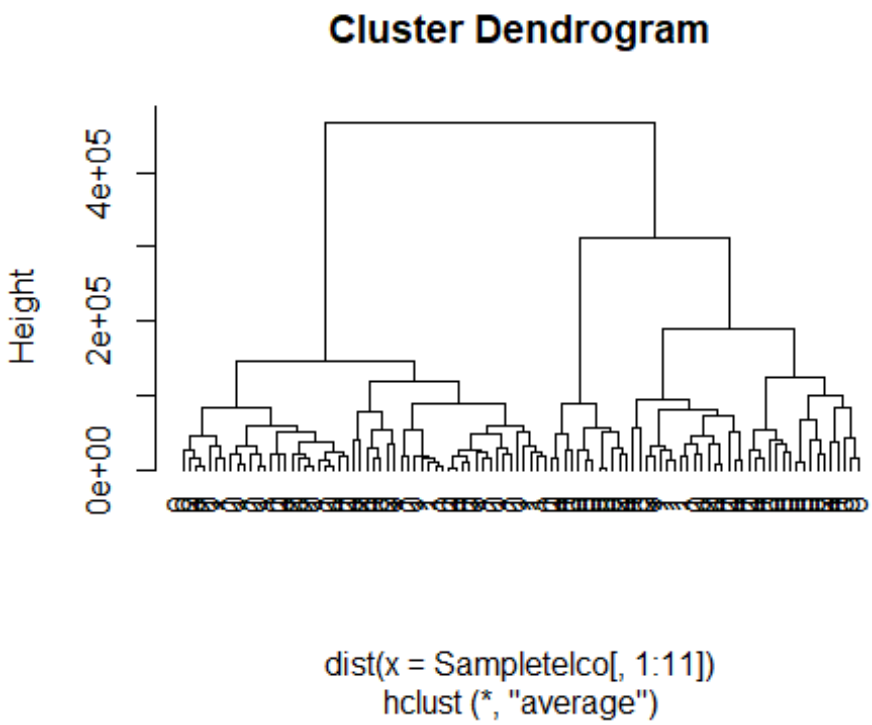
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2117 1350
##          1  927 1605
##
##                Accuracy : 0.6204
##                  95% CI : (0.608, 0.6327)
##     No Information Rate : 0.5074
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2391
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6955
##             Specificity : 0.5431
##          Pos Pred Value : 0.6106
##          Neg Pred Value : 0.6339
##              Prevalence : 0.5074
##          Detection Rate : 0.3529
##    Detection Prevalence : 0.5779
##       Balanced Accuracy : 0.6193
##
##        'Positive' Class : 0
##
```

```
#split train and test equally for ROC.
set.seed(123)
SplitIndex <- sample(x = c("Train", "Test"), size = nrow(KNNdata), replace =
T, prob = c(0.5,0.5))
KNNTrainData <- filter(KNNdata, SplitIndex == "Train")
KNNTestData <- filter(KNNdata, SplitIndex == "Test")

#Build the model on training data
set.seed(123)
KnnModel2 <- train(form = LEAVE ~ .,
                   data = KNNTrainData,
                   method = 'knn')

#Predicted probabilities
KNNprobability <- predict(object = KnnModel2, newdata = KNNTestData, type =
"prob")
# head(KnnProbs)
KNNprobability <- KNNprobability[,2]
#Generate the ROC
KnnROC <- roc(response = KNNTestData$LEAVE, predictor = KNNprobability)

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

```
## Setting direction: controls < cases

plot(KnnROC, print.auc = T)
```

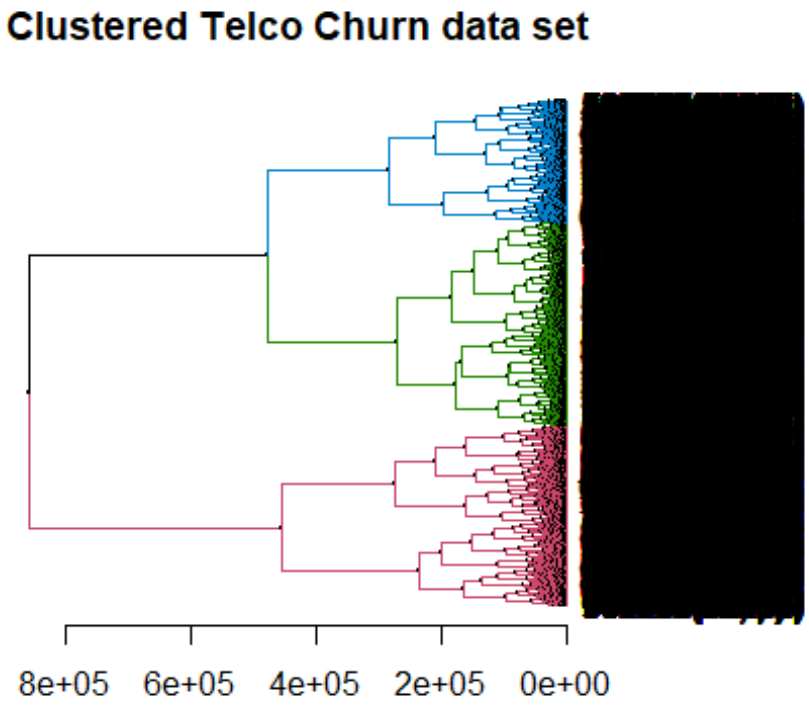



```
## TASK 4
## K MEANS CLUSTERING
# For legibility of the next graphic
set.seed(123)
Sampletelco <- sample_n(tbl = World_Telcochurn, size = 100)

#Hierarchical clutering - calculate and plot
TelcoHclust <- hclust(d = dist(x=Sampletelco[,1:11]), method = "average")

plot(x = TelcoHclust, hang = -1, labels=Sampletelco$LEAVE)
```

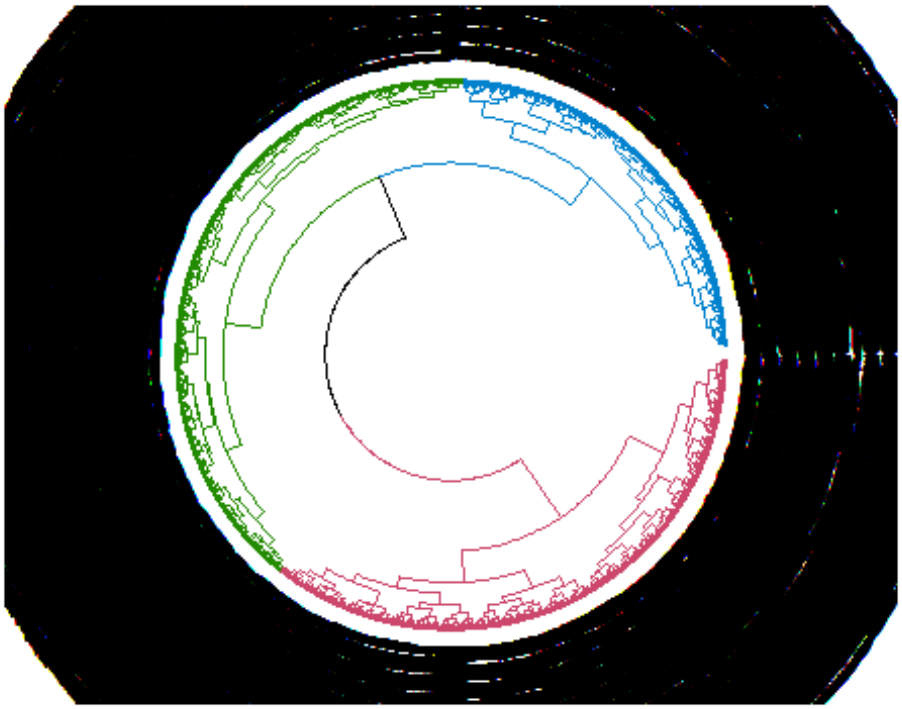



```r
#Compute distances in the WorldTelco data (excluding LEAVE), generate
hierarchical clusters
DistTelco <- dist(x = World_Telcochurn[,1:11], method = "euclidean")

HcTelco <- hclust(d = DistTelco, method = "complete")

# dendrogram object
TelcoDend <- as.dendrogram(HcTelco)


# Save the levels of the Leave column
LeaveLevs <- rev(levels(World_Telcochurn[,2]))

# Color the branches based on the clusters:
TelcoDend <- color_branches(dend = TelcoDend, k=3)

# Manually match the labels, as much as possible, to the real classification
# assign one of three colours to each label
labels_colors(TelcoDend) <-
   rainbow_hcl(3)[sort_levels_values(
     as.numeric(World_Telcochurn[,2])[order.dendrogram(TelcoDend)]
     )]

labels(TelcoDend) <-
paste(as.character(World_Telcochurn[,5])[order.dendrogram(TelcoDend)],
                            "(",labels(TelcoDend),")",
```

```
                              sep = "")
# We hang the dendrogram a bit (distance between end of dendrogram and the
label):
TelcoDend <- hang.dendrogram(TelcoDend,hang_height=0.1)



# plotting the visuals
par(mar = c(3,3,3,7))
plot(TelcoDend,
     main = "Clustered Telco Churn data set",
     horiz =  TRUE,
     nodePar = list(cex = .007))
```



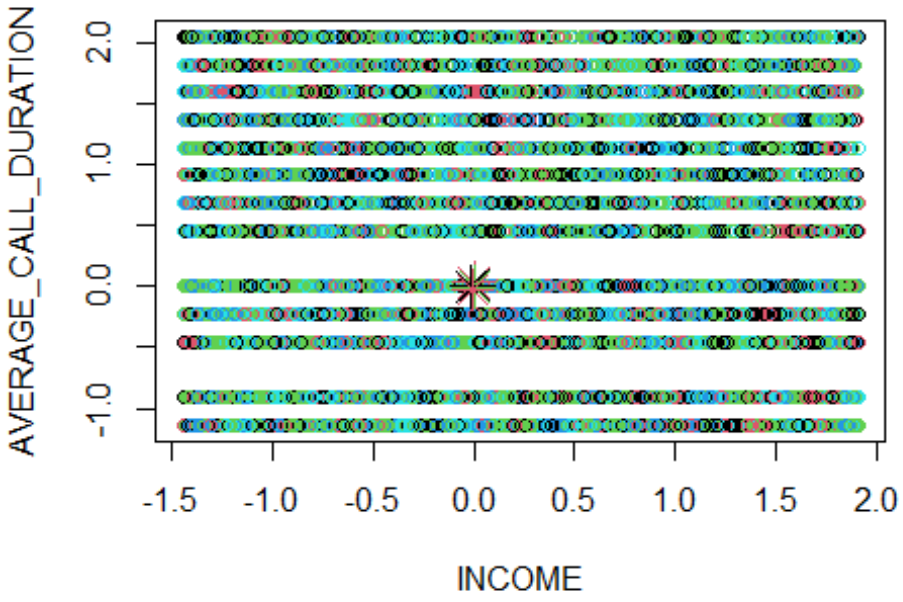

```
# Check if LeaveLevs is not empty before calling legend
if (length(LeaveLevs) > 0) {
  legend("topleft", legend = LeaveLevs, fill =
rainbow_hcl(length(LeaveLevs)))
}

par(mar = rep(1,4))
circlize_dendrogram(TelcoDend)
```



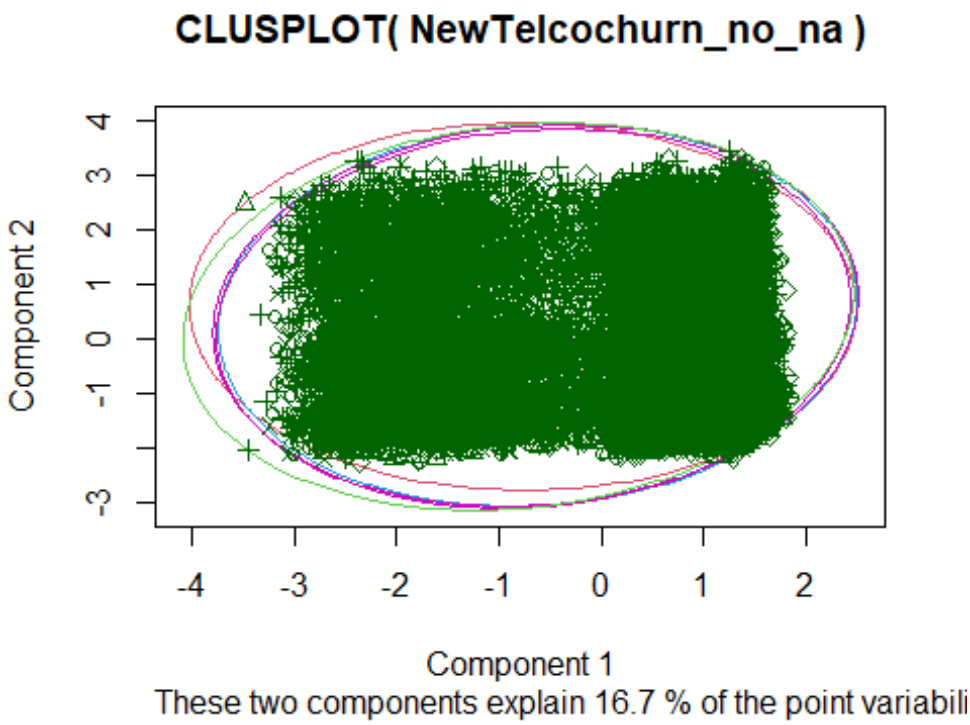
```
# Identify numerical columns
numerical_columns <- sapply(World_Telcochurn, is.numeric)

# Scale numerical columns
World_Telcochurn_scaled <- World_Telcochurn
World_Telcochurn_scaled[, numerical_columns] <- scale(World_Telcochurn[,
numerical_columns])

#Set up a new data set, and remove the LEAVE column
NewTelcochurn <- World_Telcochurn_scaled
NewTelcochurn$LEAVE <- NULL

# Remove rows with missing values
NewTelcochurn_no_na <- na.omit(NewTelcochurn)
# Convert non-numeric columns to numeric if needed
NewTelcochurn_no_na <- as.data.frame(sapply(NewTelcochurn_no_na, as.numeric))

#Use the kmeans algorithm on the new data, specifying we want k=5 clusters
KmeanTelcochurn <- kmeans(x = NewTelcochurn_no_na, center = 5)

#Note how the clusters for the data without species do an okay job of
capturing species
table(LEAVE = World_Telcochurn$LEAVE, Cluster = KmeanTelcochurn$cluster)

##      Cluster
## LEAVE    1    2    3    4    5
```

```
##     0 2079 1028 3646 1781 1614
##     1 2010  972 3489 1793 1588
```

```r
# Plots the INCOME and AVERAGE CALL DURATION of the Dataset,

plot(NewTelcochurn_no_na[c("INCOME", "AVERAGE_CALL_DURATION")],
col=KmeanTelcochurn$cluster)
points(KmeanTelcochurn$centers[,c("INCOME","AVERAGE_CALL_DURATION")],
col=1:3, pch=8, cex=2)
```




```r
#What if we didn't want to use only two of the four dimensions for plotting?
clusplot(NewTelcochurn_no_na, KmeanTelcochurn$cluster, color = TRUE)
```




```
clusplot(NewTelcochurn_no_na, KmeanTelcochurn$cluster, color=TRUE,
shade=TRUE,
         labels=2, lines=0)
```