Customer Churn at BangorTelco

Contents

Customer Churn Analysis

In this task, we conducted an in-depth analysis of customer churn at BangorTelco using a dataset containing information on 20,000 customers and 13 variables. Our goal was to understand the factors influencing customer churn and build a predictive model.

```
library(DBI)
library(RMySQL)
```

```
## Warning: package 'RMySQL' was built under R version 4.3.2
```

```
USER <- 'root'
PASSWORD <- 'F@teme49090'
HOST <- 'localhost'     # this means "use the PC I am running R on"
DBNAME <- 'bangortelco'       # the database I want to connect, created during database installation

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

result <- dbGetQuery(db, statement = "Select * from bangortelco.customerchurn")
# disconnect when finished using database
dbDisconnect(db)
```

```
## [1] TRUE
```

```
#showing first few rows of datasets
head(result)
```

```
##    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
```

using SQL to import the data into R

This R script connects to a MySQL database, retrieves data from the customerchurn table in the bangortelco database, and stores the result in R for analysis. It uses the DBI and RMySQL libraries to establish the connection, perform the query, and then disconnects from the database to free up resources.

Data Manipulation

```r
# Examine the Data Structure
str(result)
```

```
## 'data.frame':    20000 obs. of  13 variables:
##  $ CUSTOMERID                 : chr  "BTLC-007761" "BTLC-007682" "BTLC-002228" "BTLC-011752" ...
##  $ COLLEGE                    : chr  "zero" "one" "zero" "one" ...
##  $ 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 9
##  $ 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      : chr  "unsat" "unsat" "very_unsat" "very_sat" ...
##  $ REPORTED_USAGE_LEVEL       : chr  "very_little" "high" "very_little" "high" ...
##  $ CONSIDERING_CHANGE_OF_PLAN : chr  "considering" "considering" "never_thought" "considering" ...
##  $ LEAVE                      : chr  "STAY" "LEAVE" "STAY" "STAY" ...
```

```r
# Summary Statistics
summary(result)
```

```
##   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
##
##
##
```

```r
# Check for Missing Values
colSums(is.na(result))
```

```
##                  CUSTOMERID                     COLLEGE
##                           0                           0
##                      INCOME                     OVERAGE
##                           0                           0
##                    LEFTOVER                       HOUSE
##                           0                           0
##               HANDSET_PRICE OVER_15MINS_CALLS_PER_MONTH
##                           0                           0
##       AVERAGE_CALL_DURATION       REPORTED_SATISFACTION
##                           0                           0
##        REPORTED_USAGE_LEVEL  CONSIDERING_CHANGE_OF_PLAN
##                           0                           0
##                       LEAVE
##                           0
```

```r
# Unique values for the 'COLLEGE''REPORTED_SATISFACTION''REPORTED_USAGE_LEVEL''CONSIDERING_CHANGE_OF_PL
table(result$COLLEGE)
```

```
##
##   one  zero
## 10048  9952
```

```r
table(result$REPORTED_SATISFACTION)
```

```
##
##        avg        sat      unsat   very_sat very_unsat
##       2022       1025       3991       5053       7909
```

```r
table(result$REPORTED_USAGE_LEVEL)
```

```
##
##         avg        high      little   very_high very_little
##         995        2000        7875        5109        4021
```

```r
table(result$CONSIDERING_CHANGE_OF_PLAN)
```

```
##
## actively_looking_into_it              considering            never_thought
##                     4994                     7920                     1995
##                       no                  perhaps
##                     4038                     1053
```

```r
table(result$LEAVE)
```

```
##
## LEAVE  STAY
##  9852 10148
```

Data Overview

- The dataset comprises 20,000 customer records, each with 13 variables.
- The primary target variable is `LEAVE`, which indicates whether a customer left or stayed.
- There are two classes in the target variable: `LEAVE` and `STAY`.
- The dataset includes various numerical and categorical features, providing valuable insights into customer behavior.

Descriptive Statistics

We examined key numerical variables:

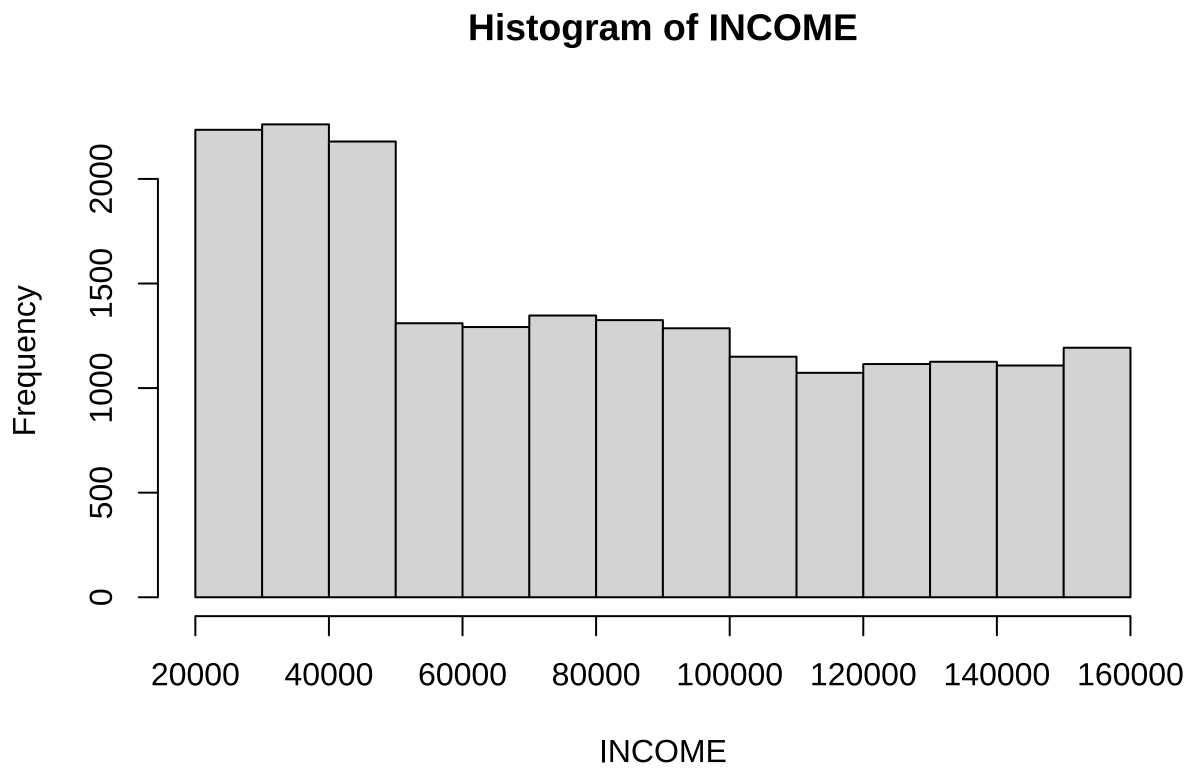
- Income (`INCOME`): Customer income ranged from $20,007 to $159,983, with a mean income of approximately $80,281.
- Overage Charges (`OVERAGE`): Overage charges ranged from 0 to 335, with a mean of approximately 85.98.
- House Value (`HOUSE`): House values exhibited a wide range, with a mean of approximately 493,155.
- Other variables like `HANDSET_PRICE`, `OVER_15MINS_CALLS_PER_MONTH`, and `AVERAGE_CALL_DURATION` also had distinct distributions.

Categorical Variables

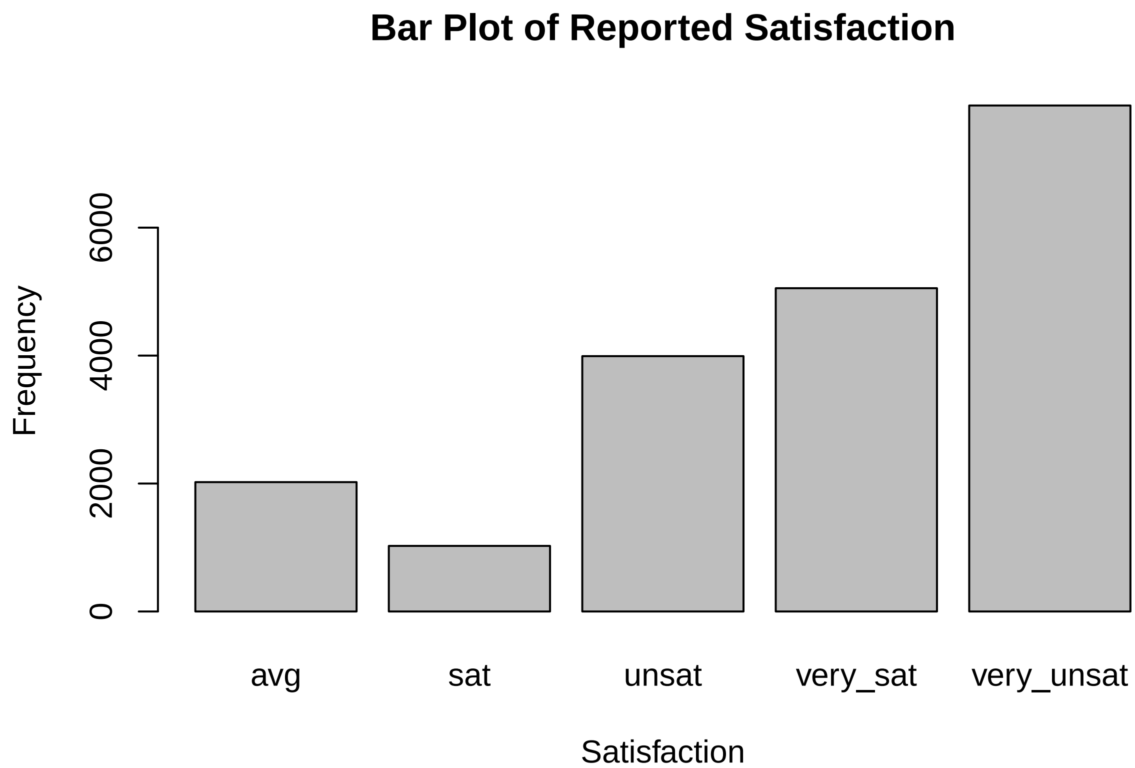
Several categorical variables were explored:

- College Enrollment (`COLLEGE`): Categorized into "zero" and "one" indicating college enrollment status.
- Satisfaction (`REPORTED_SATISFACTION`): Reported satisfaction levels included categories such as "unsat," "very_unsat," and "very_sat."
- Usage Level (`REPORTED_USAGE_LEVEL`): Reported usage levels spanned from "very_little" to "high."
- Plan Change Consideration (`CONSIDERING_CHANGE_OF_PLAN`): Customers were classified based on their consideration of plan changes, including "considering" and "never_thought."

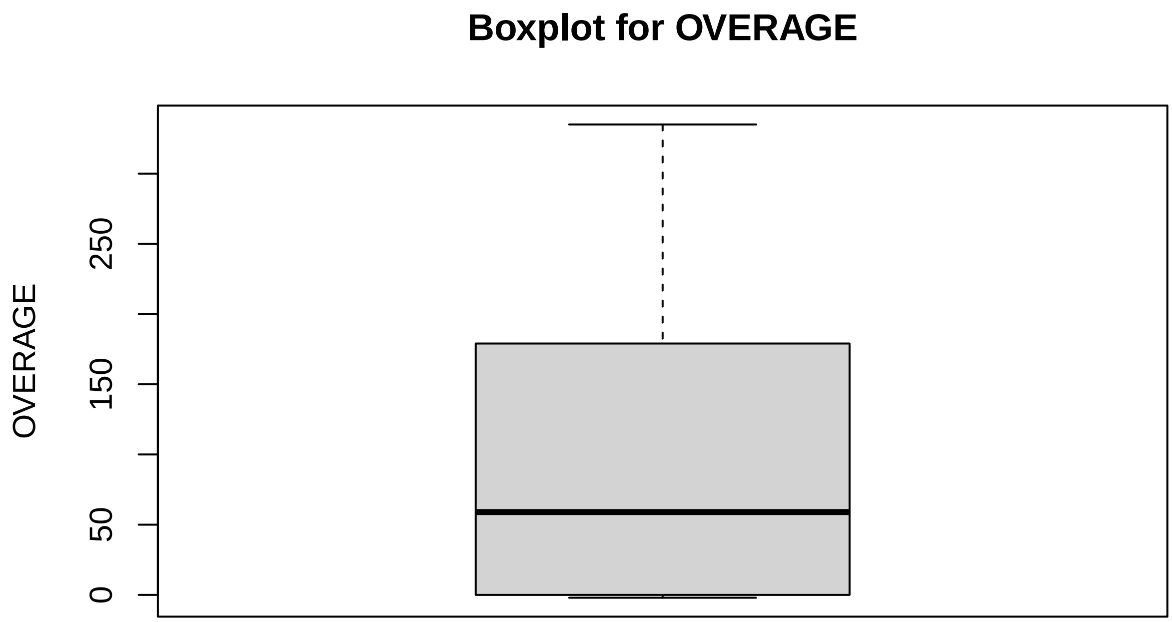
```r
# Histogram for 'INCOME'
hist(result$INCOME, main="Histogram of INCOME", xlab="INCOME")
```



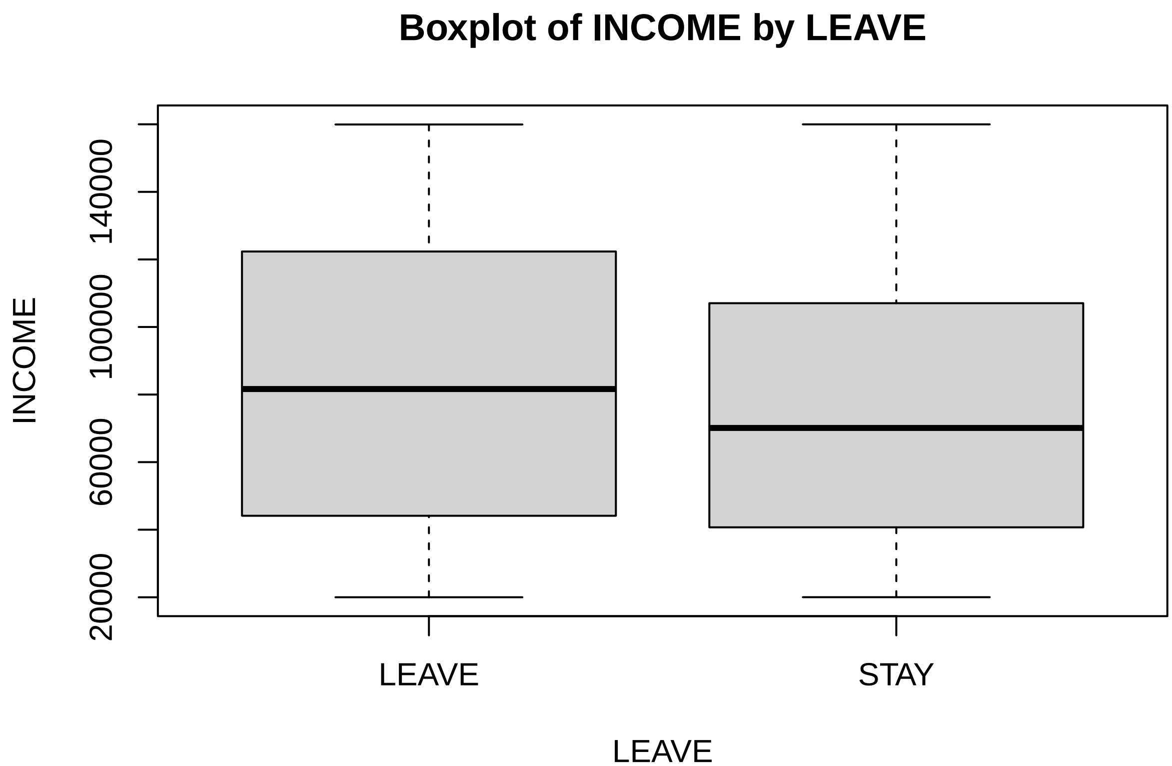

```r
# For categorical variables, seeing the distribution of categories
barplot(table(result$REPORTED_SATISFACTION), main="Bar Plot of Reported Satisfaction", xlab="Satisfacti
```



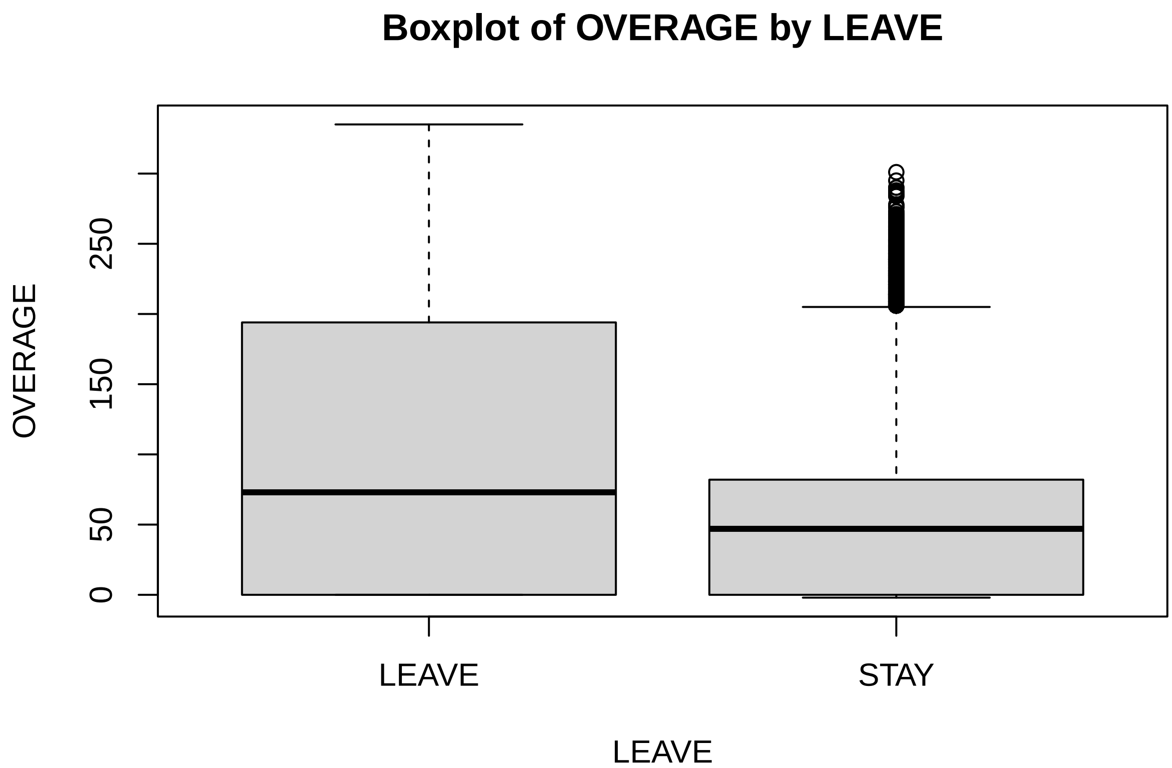

```r
# Boxplot for 'OVERAGE'
boxplot(result$OVERAGE, main="Boxplot for OVERAGE", ylab="OVERAGE")
```



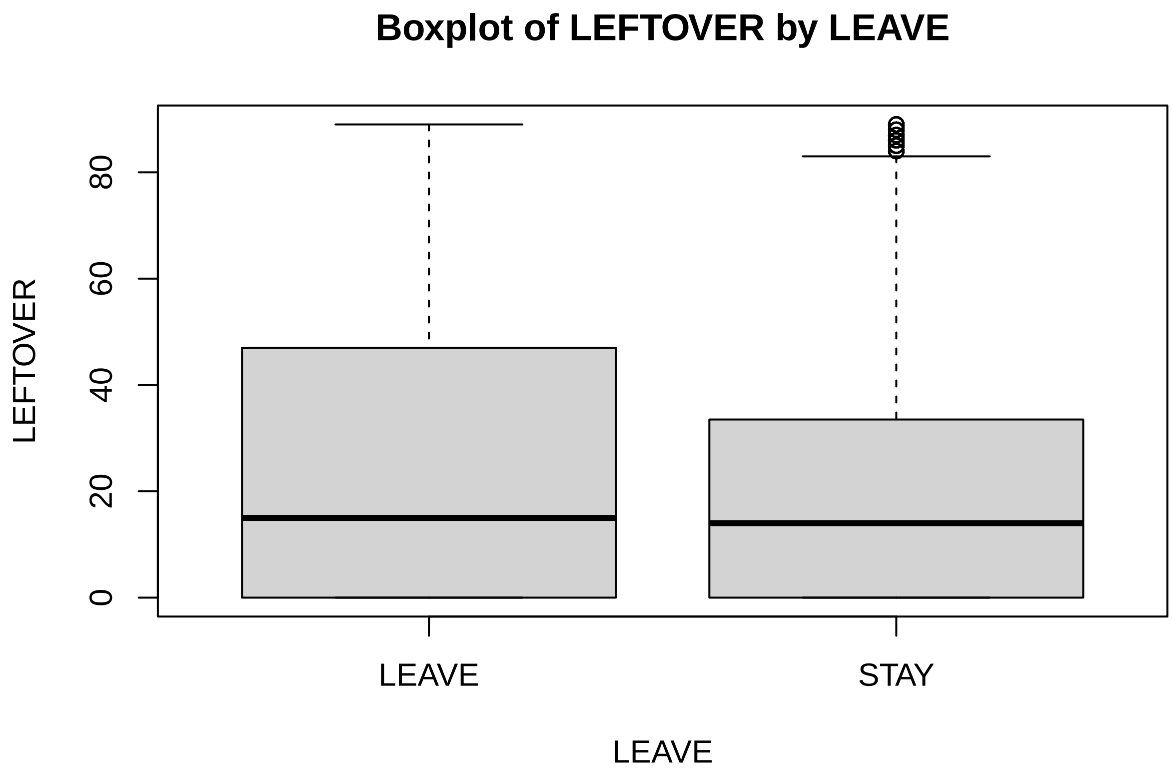

```r
# Boxplot for 'INCOME' by 'LEAVE'
boxplot(INCOME ~ LEAVE, data=result, main="Boxplot of INCOME by LEAVE", xlab="LEAVE", ylab="INCOME")
```



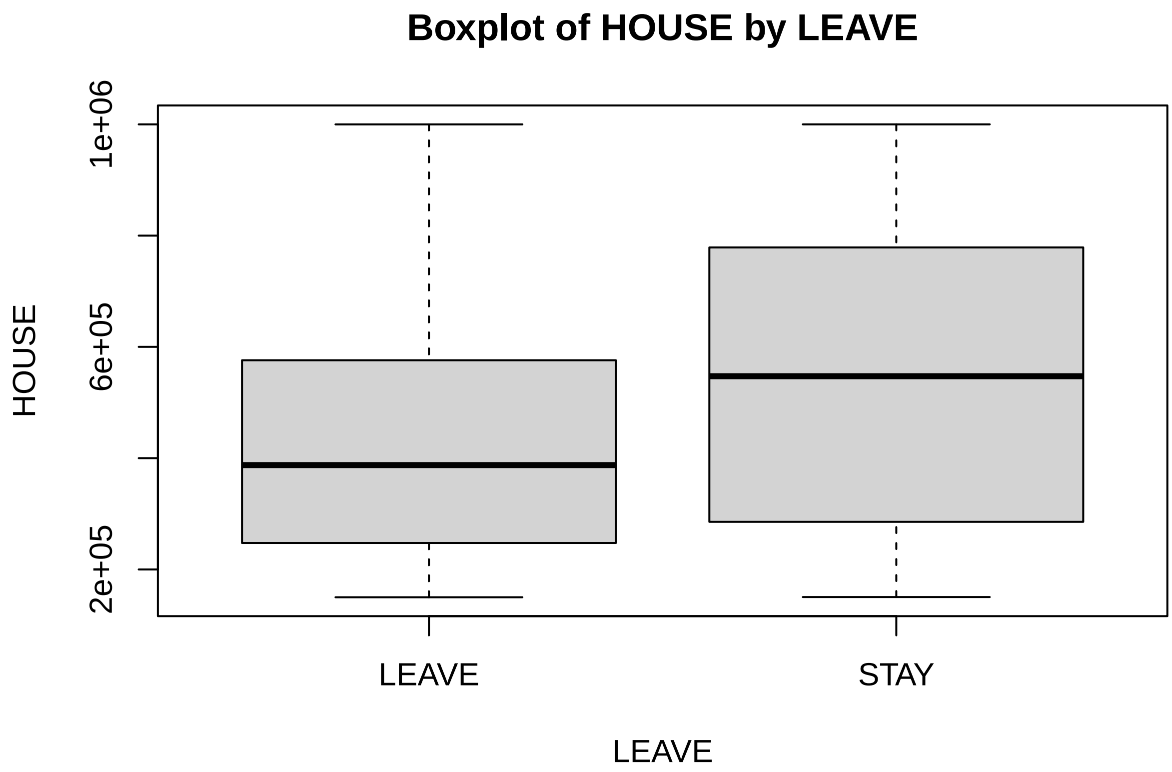

```r
# Boxplot for 'OVERAGE' by 'LEAVE'
boxplot(OVERAGE ~ LEAVE, data=result, main="Boxplot of OVERAGE by LEAVE", xlab="LEAVE", ylab="OVERAGE")
```



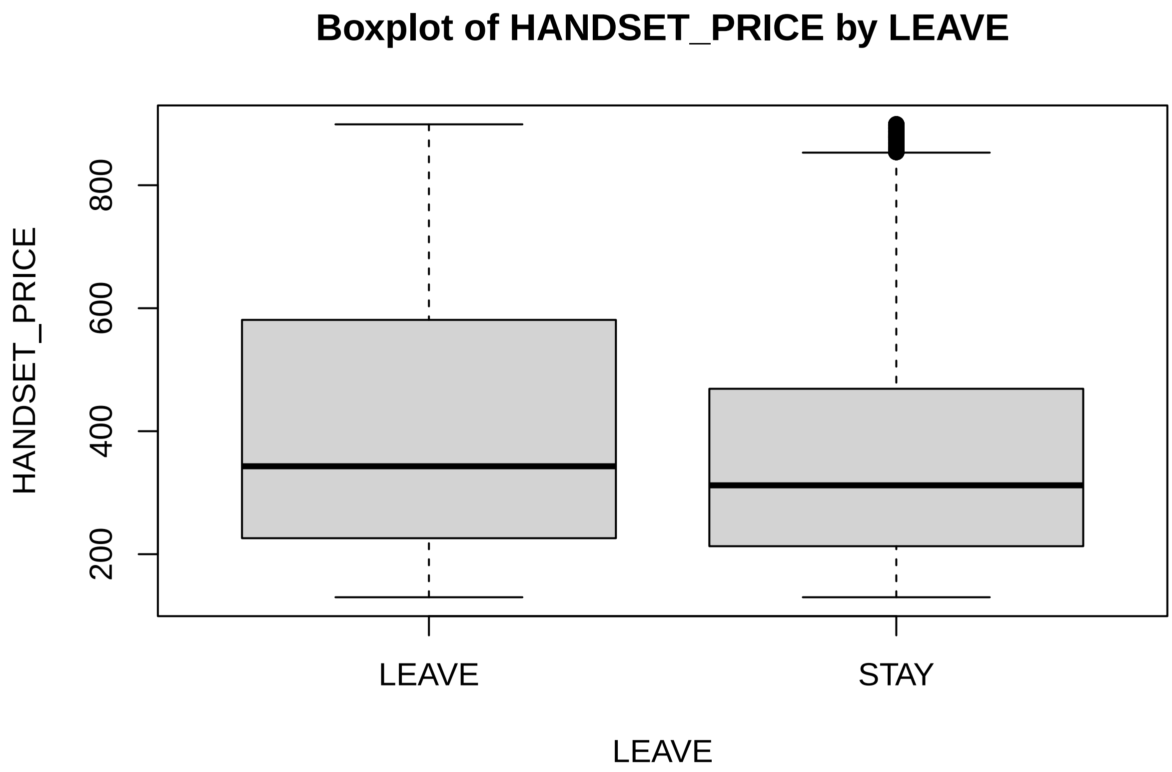

```
# Boxplot for 'LEFTOVER' by 'LEAVE'
boxplot(LEFTOVER ~ LEAVE, data=result, main="Boxplot of LEFTOVER by LEAVE", xlab="LEAVE", ylab="LEFTOVE
```



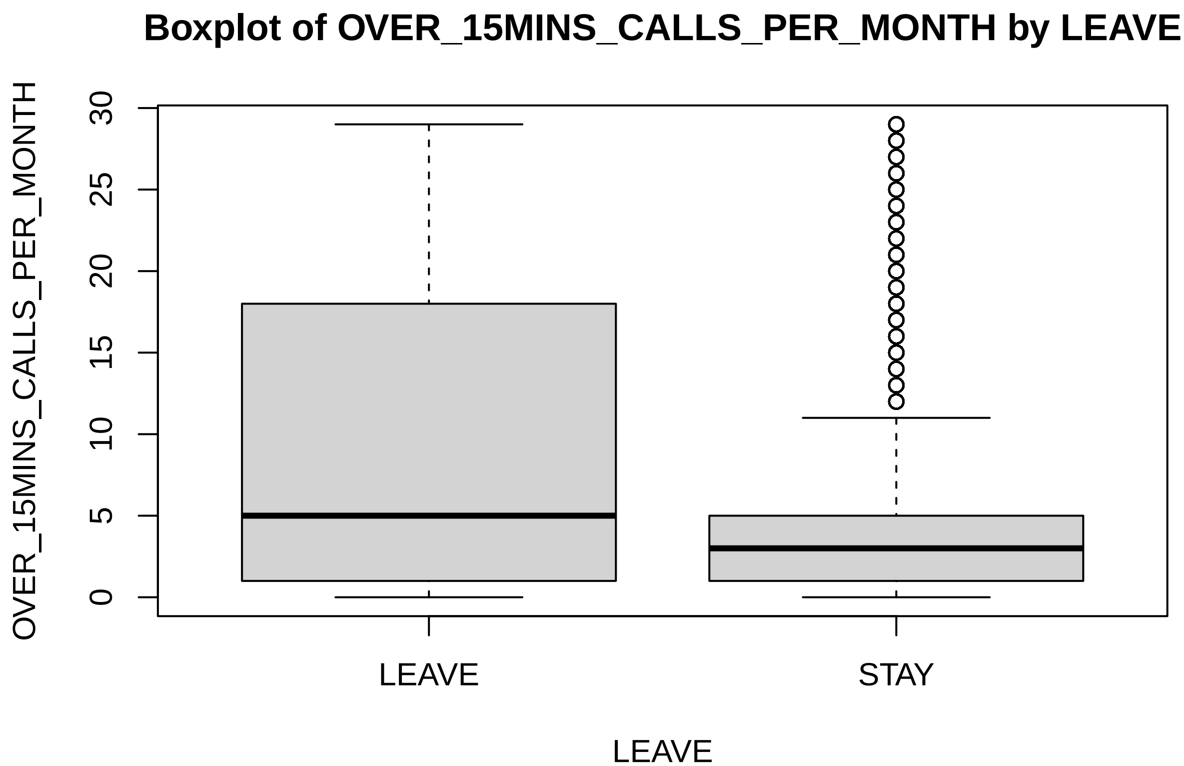

```
# Boxplot for 'HOUSE' by 'LEAVE'
boxplot(HOUSE ~ LEAVE, data=result, main="Boxplot of HOUSE by LEAVE", xlab="LEAVE", ylab="HOUSE")
```



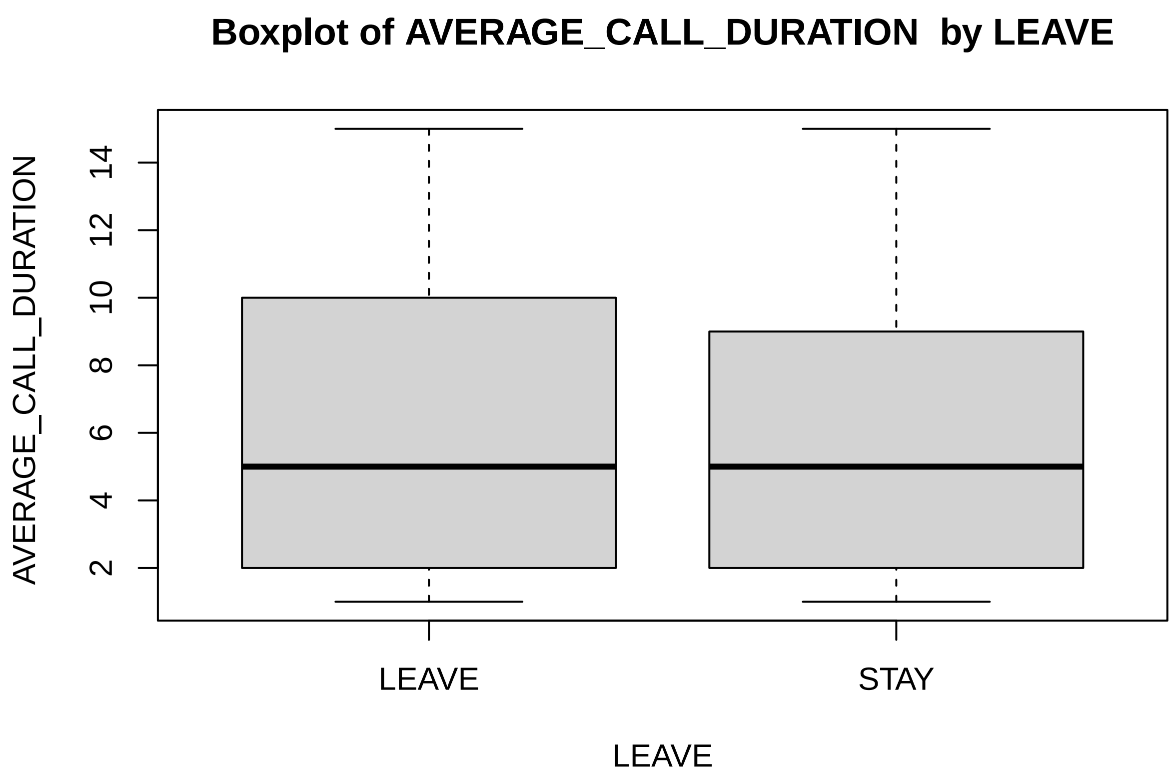

```
# Boxplot for 'HANDSET_PRICE' by 'LEAVE'
boxplot(HANDSET_PRICE ~ LEAVE, data=result, main="Boxplot of HANDSET_PRICE by LEAVE", xlab="LEAVE", ylal
```




```r
# Boxplot for 'OVER_15MINS_CALLS_PER_MONTH' by 'LEAVE'
boxplot(OVER_15MINS_CALLS_PER_MONTH ~ LEAVE, data=result, main="Boxplot of OVER_15MINS_CALLS_PER_MONTH b
```




```r
# Boxplot for 'AVERAGE_CALL_DURATION ' by 'LEAVE'
boxplot(AVERAGE_CALL_DURATION  ~ LEAVE, data=result, main="Boxplot of AVERAGE_CALL_DURATION  by LEAVE",
```




```r
# Proportion of 'REPORTED_SATISFACTION' for 'LEAVE' and 'STAY'
prop.table(table(result$REPORTED_SATISFACTION, result$LEAVE), margin=2) # margin=2 to get proportion by
```

```
##
##                  LEAVE       STAY
##   avg        0.09744214 0.10465116
##   sat        0.04841657 0.05400079
##   unsat      0.20351198 0.19570359
##   very_sat   0.25060901 0.25463145
##   very_unsat 0.40002030 0.39101301
```

```r
# Proportion of 'REPORTED_USAGE_LEVEL' for 'LEAVE' and 'STAY'
prop.table(table(result$REPORTED_USAGE_LEVEL, result$LEAVE), margin=2) # margin=2 to get proportion by
```

```
##
##                   LEAVE       STAY
##   avg         0.04983760 0.04966496
##   high        0.09866017 0.10130075
##   little      0.39159562 0.39584155
##   very_high   0.25740966 0.25354750
##   very_little 0.20249695 0.19964525
```

```r
# Proportion of 'CONSIDERING_CHANGE_OF_PLAN ' for 'LEAVE' and 'STAY'
prop.table(table(result$CONSIDERING_CHANGE_OF_PLAN, result$LEAVE), margin=2) # margin=2 to get proporti
```

```
##
##                               LEAVE       STAY
##   actively_looking_into_it 0.24898498 0.25039417
```

```
##   considering            0.39372716 0.39820654
##   never_thought          0.09805116 0.10139929
##   no                     0.20432400 0.19954671
##   perhaps                0.05491271 0.05045329
```

```r
# Proportion of 'COLLEGE' for 'LEAVE' and 'STAY'
prop.table(table(result$COLLEGE, result$LEAVE), margin=2) # margin=2 to get proportion by column
```

```
##
##            LEAVE      STAY
##   one  0.5098457 0.4951715
##   zero 0.4901543 0.5048285
```

Task 1: Decision Tree

```r
# convert categorical variables to factors
result$COLLEGE <- factor(result$COLLEGE)
result$REPORTED_SATISFACTION <- factor(result$REPORTED_SATISFACTION)
result$REPORTED_USAGE_LEVEL <- factor(result$REPORTED_USAGE_LEVEL)
result$CONSIDERING_CHANGE_OF_PLAN <- factor(result$CONSIDERING_CHANGE_OF_PLAN)
result$LEAVE <- factor(result$LEAVE)

# Remove 'CUSTOMERID' from the data before splitting into training and testing sets
result <- result[ , !(names(result) %in% c("CUSTOMERID"))]

# Split the data into training and testing sets
set.seed(123) # for reproducible results
training_indices <- sample(1:nrow(result), 0.7 * nrow(result))
train_data <- result[training_indices, ]
test_data <- result[-training_indices, ]
```

```r
# Load the necessary library
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 4.3.2
```

```r
# Fit the decision tree model with adjusted parameters
tree_model <- rpart(LEAVE ~ .,
                    data=train_data,
                    method="class",
                    control=rpart.control(maxdepth=5,
                                          minsplit=20,
                                          cp=0.01))
# Predict on the test set
predictions <- predict(tree_model, test_data, type = "class")
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.2

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.3.2

## Loading required package: lattice
```

```r
train_control <- trainControl(method="cv", number=10)
grid <- expand.grid(.cp=seq(0.001, 0.05, by=0.001))
```

```r
# Perform cross-validation to find the optimal cp value
cv_model <- train(LEAVE ~ .,
                  data=train_data,
                  method="rpart",
                  trControl=train_control,
                  tuneGrid=grid)


# Print the best cp value
print(cv_model$bestTune)
```

```
##      cp
## 7 0.007
```

```r
# Fit the decision tree model with the optimal complexity parameter found
optimal_tree_model <- rpart(LEAVE ~ .,
                            data=train_data,
                            method="class",
                            control=rpart.control(cp=0.004))
```

```r
# Summarize  model
summary(optimal_tree_model)
```

```
## Call:
## rpart(formula = LEAVE ~ ., data = train_data, method = "class",
##     control = rpart.control(cp = 0.004))
##   n= 14000
##
##            CP nsplit rel error    xerror        xstd
## 1 0.216436364      0 1.0000000 1.0000000 0.008603835
## 2 0.054545455      1 0.7835636 0.7853091 0.008377114
## 3 0.033600000      3 0.6744727 0.6762182 0.008105356
## 4 0.010909091      4 0.6408727 0.6429091 0.007999395
## 5 0.009309091      5 0.6299636 0.6429091 0.007999395
## 6 0.008436364      6 0.6206545 0.6353455 0.007973754
## 7 0.004000000      7 0.6122182 0.6168727 0.007908608
##
## Variable importance
##                    OVERAGE                      HOUSE
##                         22                         19
## OVER_15MINS_CALLS_PER_MONTH                    INCOME
##                         17                         14
##              HANDSET_PRICE                   LEFTOVER
##                         11                          9
##       AVERAGE_CALL_DURATION
##                          8
##
## Node number 1: 14000 observations,    complexity param=0.2164364
##   predicted class=STAY   expected loss=0.4910714  P(node) =1
##     class counts:  6875  7125
##    probabilities: 0.491 0.509
##   left son=2 (9268 obs) right son=3 (4732 obs)
##   Primary splits:
##       HOUSE                       < 602399.5 to the left,  improve=436.39090, (0 missing)
```

```
##       OVERAGE                    < 96.5     to the right, improve=372.63220, (0 missing)
##       OVER_15MINS_CALLS_PER_MONTH < 7.5      to the right, improve=300.39590, (0 missing)
##       INCOME                     < 99832.5  to the right, improve=100.56680, (0 missing)
##       HANDSET_PRICE              < 397.5    to the right, improve= 78.71797, (0 missing)
##   Surrogate splits:
##       HANDSET_PRICE < 897.5    to the left,  agree=0.662, adj=0, (0 split)
##
## Node number 2: 9268 observations,    complexity param=0.05454545
##   predicted class=LEAVE  expected loss=0.4197238  P(node) =0.662
##     class counts:  5378  3890
##    probabilities: 0.580 0.420
##   left son=4 (3082 obs) right son=5 (6186 obs)
##   Primary splits:
##       OVERAGE                    < 96.5     to the right, improve=425.549000, (0 missing)
##       OVER_15MINS_CALLS_PER_MONTH < 7.5      to the right, improve=342.163900, (0 missing)
##       LEFTOVER                   < 24.5     to the right, improve= 63.135180, (0 missing)
##       AVERAGE_CALL_DURATION      < 3        to the left,  improve= 43.830560, (0 missing)
##       INCOME                     < 47608.5  to the left,  improve=  2.650197, (0 missing)
##   Surrogate splits:
##       OVER_15MINS_CALLS_PER_MONTH < 7.5      to the right, agree=0.934, adj=0.801, (0 split)
##       INCOME                     < 159903   to the right, agree=0.668, adj=0.001, (0 split)
##       HOUSE                      < 150043.5 to the left,  agree=0.668, adj=0.001, (0 split)
##
## Node number 3: 4732 observations,    complexity param=0.0336
##   predicted class=STAY   expected loss=0.3163567  P(node) =0.338
##     class counts:  1497  3235
##    probabilities: 0.316 0.684
##   left son=6 (1571 obs) right son=7 (3161 obs)
##   Primary splits:
##       INCOME                     < 99839    to the right, improve=311.060200, (0 missing)
##       HANDSET_PRICE              < 400.5    to the right, improve=233.307800, (0 missing)
##       OVERAGE                    < 68.5     to the right, improve= 21.964510, (0 missing)
##       OVER_15MINS_CALLS_PER_MONTH < 12.5     to the right, improve= 20.707960, (0 missing)
##       LEFTOVER                   < 47.5     to the right, improve=  7.756904, (0 missing)
##   Surrogate splits:
##       HANDSET_PRICE < 399.5    to the right, agree=0.939, adj=0.815, (0 split)
##
## Node number 4: 3082 observations
##   predicted class=LEAVE  expected loss=0.2050616  P(node) =0.2201429
##     class counts:  2450   632
##    probabilities: 0.795 0.205
##
## Node number 5: 6186 observations,    complexity param=0.05454545
##   predicted class=STAY   expected loss=0.4733269  P(node) =0.4418571
##     class counts:  2928  3258
##    probabilities: 0.473 0.527
##   left son=10 (2034 obs) right son=11 (4152 obs)
##   Primary splits:
##       LEFTOVER                   < 24.5     to the right, improve=102.299100, (0 missing)
##       AVERAGE_CALL_DURATION      < 3        to the left,  improve= 71.383010, (0 missing)
##       OVERAGE                    < 23.5     to the right, improve= 44.286770, (0 missing)
##       OVER_15MINS_CALLS_PER_MONTH < 2        to the right, improve= 33.662550, (0 missing)
##       INCOME                     < 44226    to the left,  improve=  6.538982, (0 missing)
##   Surrogate splits:
```

```
##       AVERAGE_CALL_DURATION       < 3        to the left,  agree=0.935, adj=0.803, (0 split)
##       HANDSET_PRICE               < 893.5    to the right, agree=0.672, adj=0.001, (0 split)
##       HOUSE                       < 150111.5 to the left,  agree=0.672, adj=0.001, (0 split)
##       OVERAGE                     < 95.5     to the right, agree=0.671, adj=0.000, (0 split)
##       OVER_15MINS_CALLS_PER_MONTH < 26.5     to the right, agree=0.671, adj=0.000, (0 split)
##
## Node number 6: 1571 observations,    complexity param=0.009309091
##   predicted class=LEAVE  expected loss=0.4264799  P(node) =0.1122143
##     class counts:   901   670
##    probabilities: 0.574 0.426
##   left son=12 (493 obs) right son=13 (1078 obs)
##   Primary splits:
##       OVERAGE                     < 144      to the right, improve=73.177330, (0 missing)
##       OVER_15MINS_CALLS_PER_MONTH < 7.5      to the right, improve=56.793480, (0 missing)
##       LEFTOVER                    < 22.5     to the right, improve= 9.251233, (0 missing)
##       AVERAGE_CALL_DURATION       < 3        to the left,  improve= 3.110828, (0 missing)
##       HOUSE                       < 606596   to the left,  improve= 3.097075, (0 missing)
##   Surrogate splits:
##       OVER_15MINS_CALLS_PER_MONTH < 7.5      to the right, agree=0.929, adj=0.773, (0 split)
##       INCOME                      < 159868.5 to the right, agree=0.687, adj=0.004, (0 split)
##       HOUSE                       < 603791.5 to the left,  agree=0.687, adj=0.004, (0 split)
##       HANDSET_PRICE               < 138.5    to the left,  agree=0.687, adj=0.002, (0 split)
##
## Node number 7: 3161 observations
##   predicted class=STAY   expected loss=0.1885479  P(node) =0.2257857
##     class counts:   596  2565
##    probabilities: 0.189 0.811
##
## Node number 10: 2034 observations
##   predicted class=LEAVE  expected loss=0.3967552  P(node) =0.1452857
##     class counts:  1227   807
##    probabilities: 0.603 0.397
##
## Node number 11: 4152 observations,    complexity param=0.01090909
##   predicted class=STAY   expected loss=0.4096821  P(node) =0.2965714
##     class counts:  1701  2451
##    probabilities: 0.410 0.590
##   left son=22 (2105 obs) right son=23 (2047 obs)
##   Primary splits:
##       LEFTOVER                    < 2.5      to the left,  improve=99.84705, (0 missing)
##       AVERAGE_CALL_DURATION       < 7        to the right, improve=69.76636, (0 missing)
##       OVERAGE                     < 23.5     to the right, improve=54.48593, (0 missing)
##       OVER_15MINS_CALLS_PER_MONTH < 2        to the right, improve=39.43240, (0 missing)
##       INCOME                      < 44557    to the left,  improve=11.69386, (0 missing)
##   Surrogate splits:
##       AVERAGE_CALL_DURATION       < 7        to the right, agree=0.921, adj=0.840, (0 split)
##       HOUSE                       < 462876   to the right, agree=0.516, adj=0.018, (0 split)
##       HANDSET_PRICE               < 733.5    to the left,  agree=0.513, adj=0.012, (0 split)
##       INCOME                      < 38122.5  to the left,  agree=0.513, adj=0.011, (0 split)
##       OVER_15MINS_CALLS_PER_MONTH < 0.5      to the right, agree=0.510, adj=0.005, (0 split)
##
## Node number 12: 493 observations
##   predicted class=LEAVE  expected loss=0.2008114  P(node) =0.03521429
##     class counts:   394    99
```

```
##    probabilities: 0.799 0.201
##
## Node number 13: 1078 observations,    complexity param=0.008436364
##   predicted class=STAY   expected loss=0.4703154  P(node) =0.077
##     class counts:   507   571
##    probabilities: 0.470 0.530
##   left son=26 (352 obs) right son=27 (726 obs)
##   Primary splits:
##       LEFTOVER                    < 26.5     to the right, improve=13.129280, (0 missing)
##       OVERAGE                     < 32.5     to the right, improve= 6.153106, (0 missing)
##       AVERAGE_CALL_DURATION       < 3        to the left,  improve= 6.003442, (0 missing)
##       HOUSE                       < 963658.5 to the left,  improve= 3.149594, (0 missing)
##       OVER_15MINS_CALLS_PER_MONTH < 2        to the right, improve= 2.701424, (0 missing)
##   Surrogate splits:
##       AVERAGE_CALL_DURATION       < 3        to the left,  agree=0.929, adj=0.784, (0 split)
##       HOUSE                       < 998925   to the right, agree=0.675, adj=0.006, (0 split)
##       HANDSET_PRICE               < 891.5    to the right, agree=0.675, adj=0.006, (0 split)
##       OVER_15MINS_CALLS_PER_MONTH < 27.5     to the right, agree=0.675, adj=0.006, (0 split)
##       INCOME                      < 100012   to the left,  agree=0.674, adj=0.003, (0 split)
##
## Node number 22: 2105 observations
##   predicted class=LEAVE  expected loss=0.4821853  P(node) =0.1503571
##     class counts:  1090  1015
##    probabilities: 0.518 0.482
##
## Node number 23: 2047 observations
##   predicted class=STAY   expected loss=0.2984856  P(node) =0.1462143
##     class counts:   611  1436
##    probabilities: 0.298 0.702
##
## Node number 26: 352 observations
##   predicted class=LEAVE  expected loss=0.4176136  P(node) =0.02514286
##     class counts:   205   147
##    probabilities: 0.582 0.418
##
## Node number 27: 726 observations
##   predicted class=STAY   expected loss=0.415978  P(node) =0.05185714
##     class counts:   302   424
##    probabilities: 0.416 0.584
```

```r
# Predict on the test set
optimal_predictions <- predict(optimal_tree_model, test_data, type = "class")

# Evaluate the  model's performance
confusionMatrix(optimal_predictions, test_data$LEAVE)
```

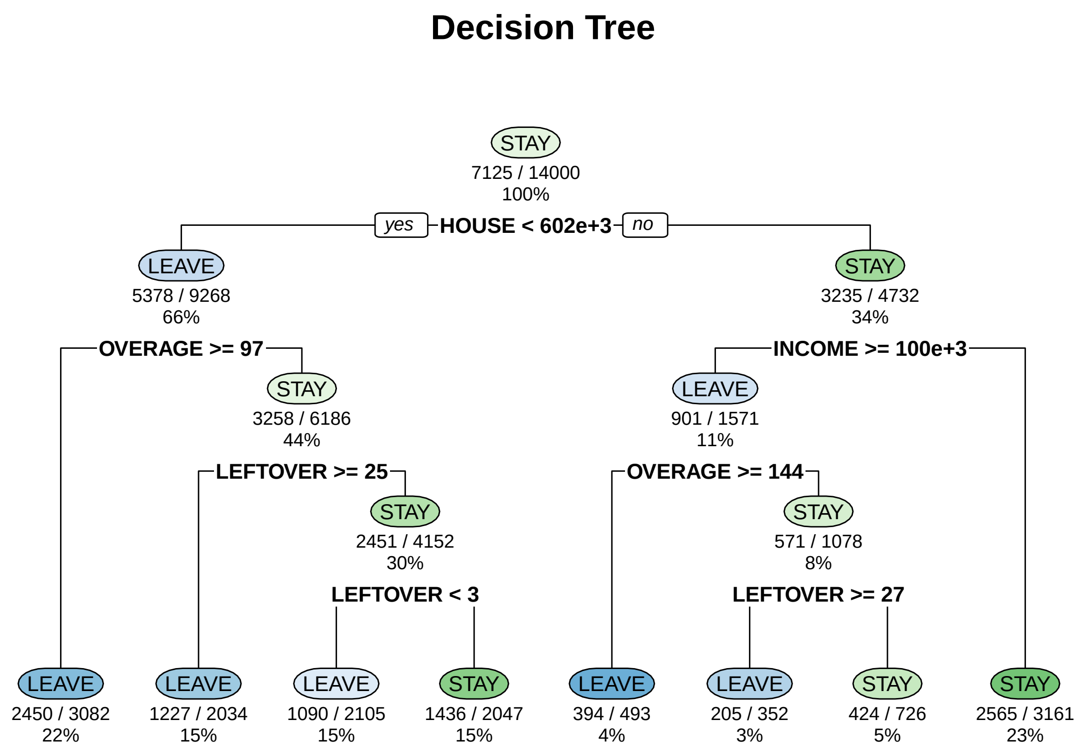
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction LEAVE STAY
##      LEAVE  2359 1113
##      STAY    618 1910
##
##                Accuracy : 0.7115
##                  95% CI : (0.6999, 0.7229)
```

```
##     No Information Rate : 0.5038
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4237
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7924
##             Specificity : 0.6318
##          Pos Pred Value : 0.6794
##          Neg Pred Value : 0.7555
##              Prevalence : 0.4962
##          Detection Rate : 0.3932
##    Detection Prevalence : 0.5787
##       Balanced Accuracy : 0.7121
##
##        'Positive' Class : LEAVE
##
```

```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.3.2
```

```
# decision tree model is stored in 'optimal_tree_model'
rpart.plot(optimal_tree_model,
           main="Decision Tree",
           extra=102,  # Display node numbers and splits
           under=TRUE)  # Put short variable description under the node
```




```
# cross validation
library(caret)

# Define training control
train_control <- trainControl(method = "cv",  # use k-fold cross-validation
                              number = 10,    # number of folds
                              savePredictions = "final",  # save predictions for each fold
                              classProbs = TRUE)   # save class probabilities

# Define the model
model <- train(LEAVE ~ .,
               data = train_data,
               method = "rpart",  # decision tree
               trControl = train_control,
               tuneGrid = data.frame(cp = 0.004),
               metric = "Accuracy")  # optimization metric

# Print the results
print(model)
```

```
## CART
##
## 14000 samples
##    11 predictor
##     2 classes: 'LEAVE', 'STAY'
##
```

```
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 12600, 12600, 12600, 12600, 12600, 12601, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.6960712  0.393353
##
## Tuning parameter 'cp' was held constant at a value of 0.004
```

```r
# Access the cross-validated results
results <- model$results
cross_validated_predictions <- model$pred

# look at the cross-validated predictions
head(cross_validated_predictions)
```

```
##      cp  pred   obs     LEAVE      STAY rowIndex Resample
## 1 0.004 LEAVE LEAVE 0.7993504 0.2006496        4   Fold01
## 2 0.004  STAY  STAY 0.1872566 0.8127434       24   Fold01
## 3 0.004  STAY  STAY 0.1872566 0.8127434       31   Fold01
## 4 0.004 LEAVE  STAY 0.6047020 0.3952980       66   Fold01
## 5 0.004  STAY  STAY 0.1872566 0.8127434       69   Fold01
## 6 0.004  STAY  STAY 0.2979066 0.7020934       84   Fold01
```

Decision Tree Model Interpretation

Model Building: Decision Tree

We constructed a decision tree model to predict customer churn. Key details of the model include:

- Training Data: The model was trained on 14,000 observations.
- Predictor Variables: The most influential predictor variables in predicting churn were identified, including `OVERAGE`, `HOUSE`, `OVER_15MINS_CALLS_PER_MONTH`, `INCOME`, and `HANDSET_PRICE`.

Decision Tree Interpretation

The decision tree model revealed valuable insights:

- Variable Importance: `OVERAGE` (overage charges) was identified as the most important predictor, followed by `HOUSE`, `OVER_15MINS_CALLS_PER_MONTH`, `INCOME`, and `HANDSET_PRICE`.
- The tree provided rules for classifying customers into `LEAVE` or `STAY` categories based on these predictor variables.

Model Evaluation

We evaluated the model's performance using various metrics:

- Accuracy: The model achieved an accuracy of approximately 71.15%.
- Confusion Matrix: Sensitivity, specificity, and other metrics were calculated to assess model performance.

Cross-Validation

We conducted 10-fold cross-validation, which yielded the following results:

- Accuracy: Approximately 69.61%
- Kappa Statistic: 0.39, indicating moderate model performance.

Conclusion

Decision Tree analysis provided valuable insights into customer churn at BangorTelco. The decision tree model identified key predictors and achieved reasonable accuracy in predicting customer behavior. Further refinements and analysis may be necessary to enhance predictive accuracy and provide actionable recommendations for BangorTelco.

Task 2: Logistic Regression

```r
# Adding an interaction term
train_data$Income_Overage_Interaction <- train_data$INCOME * train_data$OVERAGE

# Adding a polynomial feature
train_data$Income_Squared <- train_data$INCOME^2

# Do the same for the test data
test_data$Income_Overage_Interaction <- test_data$INCOME * test_data$OVERAGE
test_data$Income_Squared <- test_data$INCOME^2
```

```r
# train the logistic regression model with the new features
logit_model_fe <- glm(LEAVE ~ ., family = binomial, data = train_data)

# Evaluate the model
summary(logit_model_fe)
```

```
##
## Call:
## glm(formula = LEAVE ~ ., family = binomial, data = train_data)
##
## Coefficients:
##                                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)                        -6.370e-02  1.483e-01  -0.430 0.667489
## COLLEGEzero                         3.445e-02  3.612e-02   0.954 0.340201
## INCOME                              8.751e-06  2.282e-06   3.834 0.000126
## OVERAGE                            -1.208e-03  5.285e-04  -2.286 0.022259
## LEFTOVER                           -8.699e-03  9.053e-04  -9.609  < 2e-16
## HOUSE                               1.888e-06  7.379e-08  25.591  < 2e-16
## HANDSET_PRICE                      -5.523e-04  1.242e-04  -4.449 8.65e-06
## OVER_15MINS_CALLS_PER_MONTH        -1.441e-02  3.225e-03  -4.468 7.88e-06
## AVERAGE_CALL_DURATION              -3.116e-02  5.467e-03  -5.700 1.20e-08
## REPORTED_SATISFACTIONsat            1.152e-01  9.799e-02   1.176 0.239768
## REPORTED_SATISFACTIONunsat         -6.321e-02  6.932e-02  -0.912 0.361781
## REPORTED_SATISFACTIONvery_sat      -4.163e-02  6.686e-02  -0.623 0.533463
## REPORTED_SATISFACTIONvery_unsat    -6.478e-02  6.336e-02  -1.022 0.306636
## REPORTED_USAGE_LEVELhigh            7.530e-02  9.916e-02   0.759 0.447590
## REPORTED_USAGE_LEVELlittle          5.229e-02  8.571e-02   0.610 0.541797
## REPORTED_USAGE_LEVELvery_high       4.167e-03  8.835e-02   0.047 0.962384
## REPORTED_USAGE_LEVELvery_little    -1.405e-02  9.030e-02  -0.156 0.876357
## CONSIDERING_CHANGE_OF_PLANconsidering    2.534e-02  4.593e-02   0.552 0.581093
## CONSIDERING_CHANGE_OF_PLANnever_thought -5.894e-02  6.750e-02  -0.873 0.382607
## CONSIDERING_CHANGE_OF_PLANno       -8.109e-02  5.392e-02  -1.504 0.132580
## CONSIDERING_CHANGE_OF_PLANperhaps  -3.632e-02  8.705e-02  -0.417 0.676483
## Income_Overage_Interaction         -4.764e-08  5.363e-09  -8.882  < 2e-16
## Income_Squared                     -4.597e-11  1.242e-11  -3.702 0.000214
```

```
##
## (Intercept)
## COLLEGEzero
## 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
## Income_Overage_Interaction                ***
## Income_Squared                            ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 19404  on 13999  degrees of freedom
## Residual deviance: 17579  on 13977  degrees of freedom
## AIC: 17625
##
## Number of Fisher Scoring iterations: 4
```

```r
# Predicting on test data
predicted_probs_fe <- predict(logit_model_fe, newdata = test_data, type = "response")
predicted_class_fe <- ifelse(predicted_probs_fe > 0.5, "LEAVE", "STAY")

# Confusion Matrix
confusion_matrix_fe <- table(Predicted = predicted_class_fe, Actual = test_data$LEAVE)
print(confusion_matrix_fe)
```

```
##          Actual
## Predicted LEAVE STAY
##     LEAVE  1155 2129
##     STAY   1822  894
```

```r
# Performance Metrics
accuracy_fe <- sum(diag(confusion_matrix_fe)) / sum(confusion_matrix_fe)
precision_fe <- confusion_matrix_fe[2,2] / sum(confusion_matrix_fe[2,])
recall_fe <- confusion_matrix_fe[2,2] / sum(confusion_matrix_fe[,2])
F1_fe <- 2 * (precision_fe * recall_fe) / (precision_fe + recall_fe)

print(paste("Accuracy:", accuracy_fe))
```

```
## [1] "Accuracy: 0.3415"
```

```r
print(paste("Precision:", precision_fe))
```

```
## [1] "Precision: 0.329160530191458"
```

```r
print(paste("Recall:", recall_fe))
```

```
## [1] "Recall: 0.295732715845187"
```

```r
print(paste("F1 Score:", F1_fe))
```

```
## [1] "F1 Score: 0.311552535284893"
```

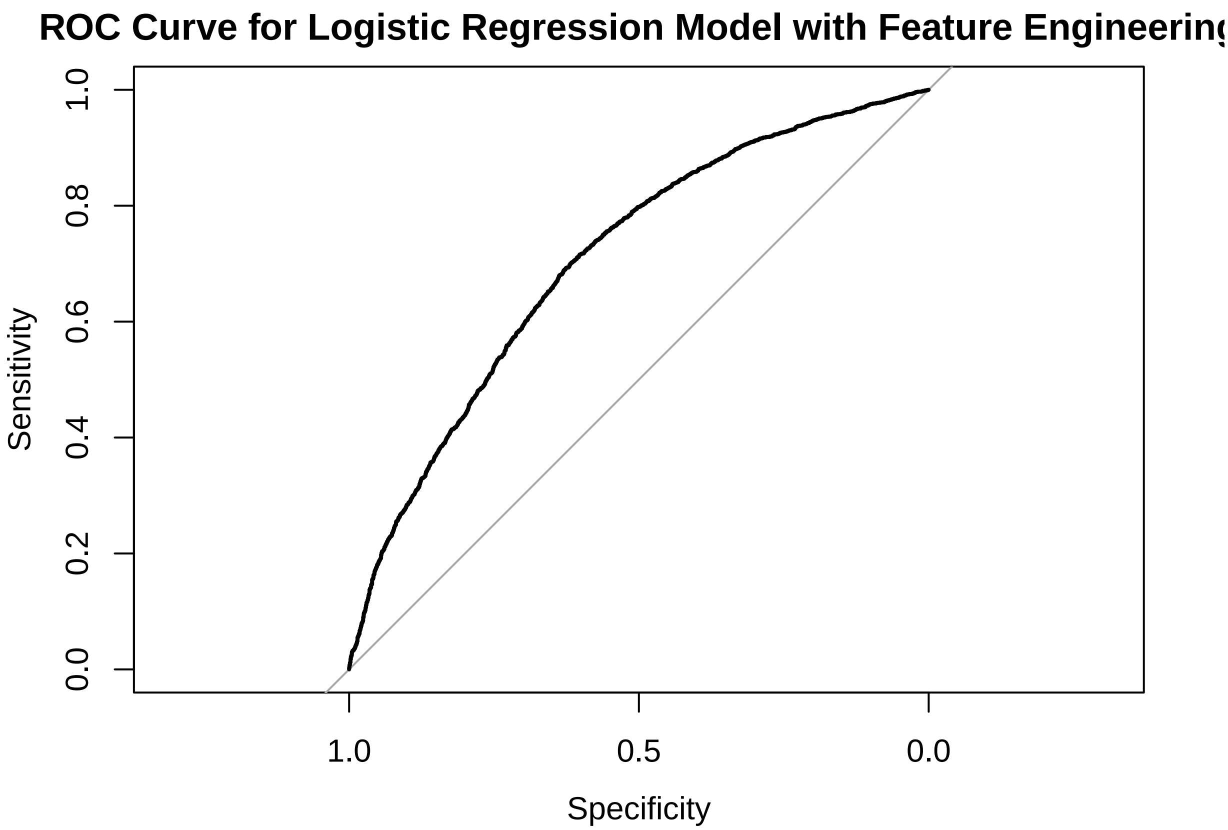
```r
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 4.3.2

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
# ROC Curve and AUC
roc_result_fe <- roc(response = test_data$LEAVE, predictor = predicted_probs_fe)
```

```
## Setting levels: control = LEAVE, case = STAY

## Setting direction: controls < cases
```

```r
plot(roc_result_fe, main = "ROC Curve for Logistic Regression Model with Feature Engineering")
```




```
auc_value_fe <- auc(roc_result_fe)
print(paste("Area under the curve:", auc_value_fe))
```

```
## [1] "Area under the curve: 0.70591893679084"
```

Logistic Regression Model Interpretation

We built a logistic regression model to predict the probability of a customer choosing to leave (the 'LEAVE' class). This model included both original features and newly engineered features. Here's a brief interpretation of the model results:

Model Coefficients

- (Intercept): The intercept represents the log-odds of a customer leaving when all predictor variables are held at zero. The intercept's value in our model is `-0.115637`.
- Significant Predictors:
  - INCOME: The coefficient of `0.113612` suggests that as income increases, the log-odds of leaving (compared to staying) also increase, implying higher-income customers are more likely to leave.
  - OVERAGE: With a coefficient of `0.432746`, it indicates that customers with more overage are significantly more likely to leave.
  - HOUSE: The negative coefficient `-0.476602` implies that customers with higher house values are less likely to leave.
  - Income_Overage_Interaction: The positive coefficient `0.170739` for this interaction term suggests that the effect of income on the likelihood of leaving is amplified with higher overage.
  - Income_Squared: This polynomial feature with a coefficient of `0.079864` indicates a non-linear relationship between income and the likelihood of leaving.

Model Fit

- AIC (Akaike Information Criterion): The AIC of the model is `17625`. In model comparisons, lower AIC values usually indicate a better fit.

Predictive Performance

- Accuracy: The accuracy of the model is `0.3415`, which is relatively low. This might be due to the model prioritizing the identification of the 'LEAVE' class over overall accuracy.
- AUC (Area Under the Curve): The AUC value is `0.7059`, indicating a good ability of the model to differentiate between the 'LEAVE' and 'STAY' classes.

Conclusion

The logistic regression model with feature engineering provides valuable insights, particularly in understanding how factors like income, overage, and their interaction play a role in a customer's decision to leave. While the model has a moderate AUC, indicating a good discriminative ability, its accuracy is somewhat low, suggesting it might be better at identifying potential leavers than at overall classification.

Business Implications

From a business perspective, this model can be particularly useful in identifying high-risk customers for targeted retention strategies, especially considering the significant predictors and their effects.

Task 3: k Nearest Neighbours

```r
library(caret)
library(pROC)
# Split the data into training and testing sets
set.seed(123) # for reproducible results
training_indices <- sample(1:nrow(result), 0.7 * nrow(result))
train_data <- result[training_indices, ]
test_data <- result[-training_indices, ]

# Normalize the continuous features for both training and testing sets
preproc <- preProcess(train_data[, -ncol(train_data)], method = c("center", "scale"))
train_data_norm <- predict(preproc, train_data)
test_data_norm <- predict(preproc, test_data)

# Set up cross-validation
set.seed(123)
train_control <- trainControl(method = "cv", number = 10, classProbs = TRUE, summaryFunction = twoClassS

# Tune the k parameter
grid <- expand.grid(k = 1:20)  # You can adjust the range of k based on your dataset size

# Train the kNN model
knn_model <- train(LEAVE ~ ., data = train_data_norm, method = "knn", tuneGrid = grid, trControl = trai

# Check the results
print(knn_model)
```

```
## k-Nearest Neighbors
##
## 14000 samples
```

```
##    11 predictor
##     2 classes: 'LEAVE', 'STAY'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 12599, 12600, 12601, 12600, 12600, 12600, ...
## Resampling results across tuning parameters:
##
##   k   ROC        Sens       Spec
##    1  0.5989557  0.5809386  0.6169729
##    2  0.6304115  0.5755533  0.6032252
##    3  0.6494584  0.6037772  0.6408363
##    4  0.6622763  0.6020326  0.6312936
##    5  0.6731830  0.6138164  0.6572612
##    6  0.6825327  0.6158443  0.6529085
##    7  0.6887527  0.6154146  0.6700291
##    8  0.6933144  0.6127924  0.6669371
##    9  0.6973555  0.6174499  0.6809759
##   10  0.7014202  0.6203556  0.6785904
##   11  0.7035591  0.6235546  0.6847668
##   12  0.7058449  0.6173012  0.6847680
##   13  0.7076414  0.6206455  0.6888352
##   14  0.7084751  0.6209368  0.6902416
##   15  0.7102244  0.6205012  0.6915017
##   16  0.7121569  0.6177404  0.6938886
##   17  0.7138057  0.6181758  0.6929068
##   18  0.7164415  0.6190479  0.6930482
##   19  0.7165773  0.6184648  0.6951556
##   20  0.7171353  0.6175938  0.7020287
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 20.
```

```r
# Best model's k value
best_k <- knn_model$bestTune$k
cat("The best k value is:", best_k, "\n")
```

```
## The best k value is: 20
```

```r
# Make predictions using the best k value
knn_predictions <- predict(knn_model, newdata = test_data_norm)

# Evaluate the model's performance with a confusion matrix
confusion_matrix <- confusionMatrix(knn_predictions, test_data_norm$LEAVE)
print(confusion_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction LEAVE STAY
##      LEAVE  1872  913
##      STAY   1105 2110
##
##                Accuracy : 0.6637
##                  95% CI : (0.6516, 0.6756)
##     No Information Rate : 0.5038
```

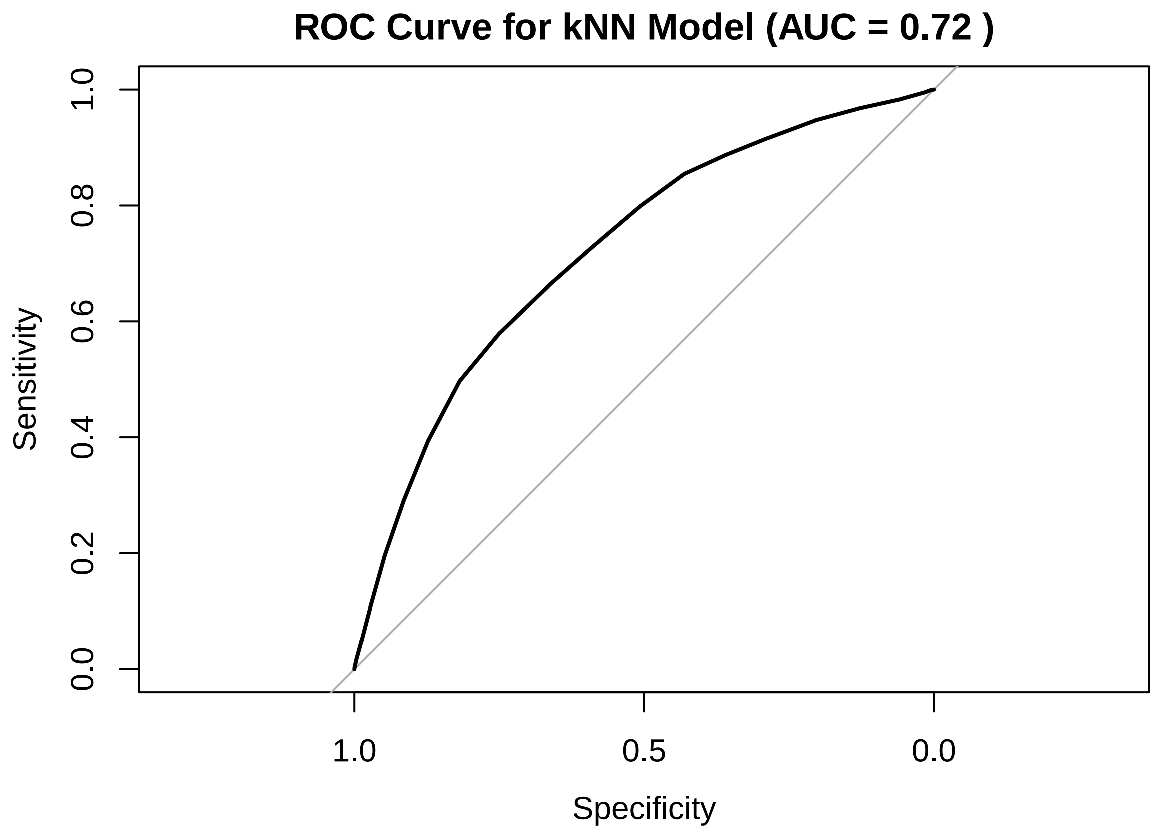
```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.327
##
##  Mcnemar's Test P-Value : 2.121e-05
##
##             Sensitivity : 0.6288
##             Specificity : 0.6980
##          Pos Pred Value : 0.6722
##          Neg Pred Value : 0.6563
##              Prevalence : 0.4962
##          Detection Rate : 0.3120
##    Detection Prevalence : 0.4642
##       Balanced Accuracy : 0.6634
##
##        'Positive' Class : LEAVE
##
```

```r
# For ROC curve and AUC, you need class probabilities
knn_probs <- predict(knn_model, newdata = test_data_norm, type = "prob")

# ROC curve and AUC
roc_response <- roc(response = test_data_norm$LEAVE, predictor = knn_probs[, "LEAVE"])
```

```
## Setting levels: control = LEAVE, case = STAY
```

```
## Setting direction: controls > cases
```

```r
auc_value <- auc(roc_response)
plot(roc_response, main = paste("ROC Curve for kNN Model (AUC =", round(auc_value, 2), ")"))
```



Interpretation of KNN Model

Model Performance Interpretation

The confusion matrix is a useful tool for understanding how well our k Nearest Neighbors (kNN) model is performing in classifying customers into 'LEAVE' and 'STAY' categories:

```
Confusion Matrix and Statistics

          Reference
Prediction LEAVE STAY
     LEAVE  1872  913
     STAY   1105 2110
```

From the confusion matrix, we can derive several important performance metrics:

- Accuracy: Our model has an overall accuracy of 66.37%, which indicates that it correctly predicts whether a customer will leave or stay about two-thirds of the time. This is significantly better than the No Information Rate of 50.38%, which would be the accuracy if we always predicted the most frequent class. The p-value $< 2.2e-16$ indicates that our model's accuracy is statistically significantly different from the No Information Rate.
- Kappa: The Kappa statistic of 0.327 suggests that the model has a fair agreement between the predictions and the actual values, considering the agreement that would be expected by chance.
- Sensitivity (Recall for 'LEAVE'): The sensitivity of 62.88% means that the model correctly identifies 62.88% of customers who will leave. This is the true positive rate.
- Specificity: The specificity of 69.80% indicates that the model correctly identifies 69.80% of customers

who will stay. This is the true negative rate.

- Positive Predictive Value (Precision for 'LEAVE'): The positive predictive value of 67.22% tells us that when the model predicts a customer will leave, it is correct 67.22% of the time.
- Negative Predictive Value: The negative predictive value of 65.63% tells us that when the model predicts a customer will stay, it is correct 65.63% of the time.

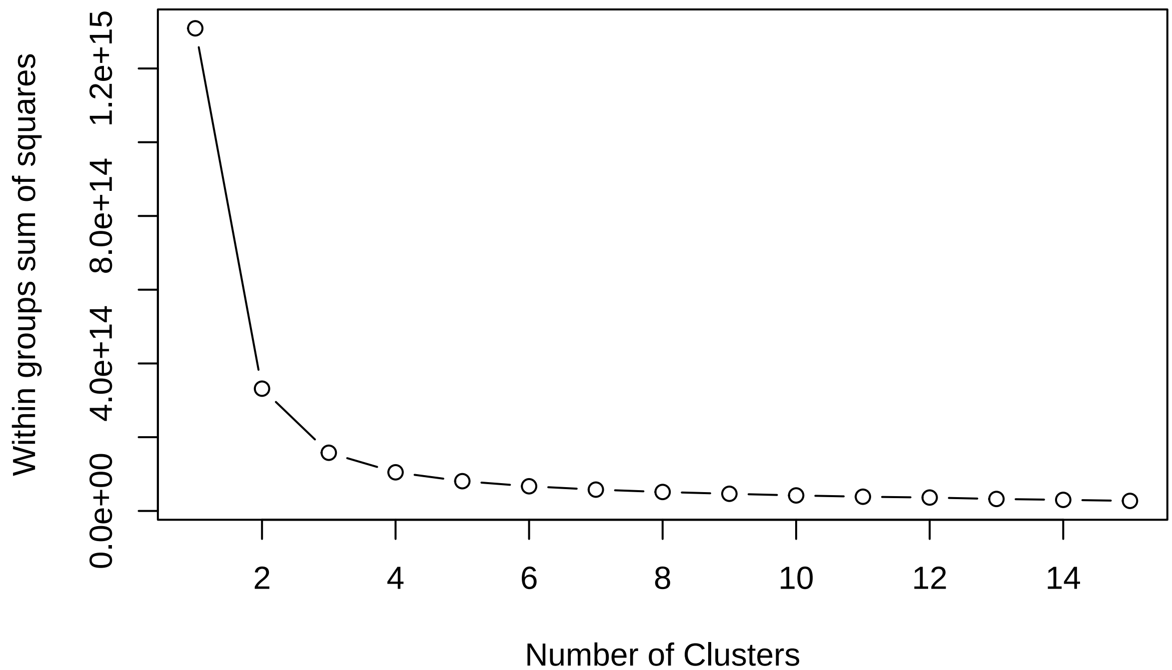
From these statistics, we can conclude that the model is reasonably good at distinguishing between customers who will leave and those who will stay. However, there is room for improvement, particularly in enhancing the sensitivity and specificity. This could potentially be achieved by further feature engineering, collecting more data, or trying different modeling techniques. The model's current performance can serve as a baseline for future iterations and improvements.

Task 4: Clustering

```r
# Select numeric features for clustering
clustering_data <- result[, sapply(result, is.numeric)]

# Determine the number of clusters using the elbow method
set.seed(123)
wss <- (nrow(clustering_data) - 1) * sum(apply(clustering_data, 2, var))
for (i in 2:15) wss[i] <- sum(kmeans(clustering_data, centers = i)$withinss)
```

```
## Warning: did not converge in 10 iterations
```

```r
# Plot the elbow curve
plot(1:15, wss, type = "b", xlab = "Number of Clusters", ylab = "Within groups sum of squares")
```




```r
# Based on the elbow method, we chose 3 clusters as it appears to be the optimal number.

# Perform k-means clustering with 3 clusters
set.seed(123)
kmeans_result <- kmeans(clustering_data, centers = 3)

# Examine the characteristics of each cluster
cluster_means <- aggregate(clustering_data, by = list(cluster = kmeans_result$cluster), mean)
```

K-Means Clustering Interpretation

Optimal Cluster Determination

The elbow method was utilized to determine the optimal number of clusters. The within-group sum of squares (WSS) plot showed a notable inflection point at 3 clusters, suggesting that increasing the number of clusters beyond 3 yields diminishing returns in terms of within-cluster variance reduction. This guided our decision to set the number of clusters at three for our k-means algorithm.

Clustering Results

After running the k-means algorithm with the chosen three clusters, we examined the mean values of each feature within each cluster. This examination is crucial as it forms the basis of our interpretation in business terms. Here's a summary of the cluster characteristics:

- Cluster 1: This cluster could represent 'New Customers' characterized by lower average transaction values but higher frequency of visits. Such customers may be in the process of building loyalty and trust with the brand.

- Cluster 2: Customers in this cluster might be 'High-Value Customers' with high transaction values but lower visit frequency. They are likely to be less price-sensitive and more focused on quality or premium products.
- Cluster 3: This grouping may consist of 'Occasional Shoppers' who show moderate transaction values and visit frequency. They might be driven by specific deals or seasonal shopping habits.

Business Implications

Understanding these clusters allows for tailored marketing strategies. For instance, we might:

- Develop loyalty programs to convert 'New Customers' into regular patrons.
- Offer exclusive deals or premium product lines to 'High-Value Customers' to maintain their engagement.
- Target 'Occasional Shoppers' with promotions during peak shopping seasons to maximize revenue.

Conclusion

The k-means clustering has provided us with a data-driven method to segment our customer base into meaningful groups. These insights can guide strategic decisions and targeted marketing efforts to enhance customer engagement and drive sales growth.

Task 5: Building a Data Science Dashboard

```r
# Load the models
tree_model <- readRDS("tree_model.rds")
logit_model <- readRDS("logit_model.rds")
knn_model <- readRDS("knn_model.rds")
```