DM_Models

Group 5

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com (http://rmarkdown.rstudio.com).

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
#load the packages
library(readr)
library(readxl)
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.2.1
──
```

```
## ✔ ggplot2 3.2.1     ✔ purrr   0.3.3
## ✔ tibble  2.1.3     ✔ dplyr   0.8.3
## ✔ tidyr   1.0.0     ✔ stringr 1.4.0
## ✔ ggplot2 3.2.1     ✔ forcats 0.4.0
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts()
──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

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

```
##
## Attaching package: 'caret'
```

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

```
library(e1071)
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

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

```
library(randomForest)
```

```
## randomForest 4.6-14
```

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

```
##
## Attaching package: 'randomForest'
```

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

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

```
library(leaps)
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

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

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:randomForest':
##
##     combine
```

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

```
library(formattable)
```

```
##
## Attaching package: 'formattable'
```

```
## The following object is masked from 'package:MASS':
##
##     area
```

```
library(outliers)
```

```
##
## Attaching package: 'outliers'
```

```
## The following object is masked from 'package:randomForest':
##
##     outlier
```

```
library(rpivotTable)
library(InformationValue)
```

```
##
## Attaching package: 'InformationValue'
```

```
## The following objects are masked from 'package:caret':
##
##     confusionMatrix, precision, sensitivity, specificity
```

```
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

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

```
#upload the target dataset
churn_data <- read_csv("~/Desktop/MBRChurnModel_FirstYear_MSK (1).csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   RENEW = col_character(),
##   M2EXCFLG = col_character(),
##   F2HOMRGN = col_character(),
##   HOMEFCTYCHANGE = col_character(),
##   RECENTMOVING = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
#check the missing values
sapply(churn_data, function(x) sum(is.na(x)))
```

```
##          RENEW       A2ACCIPK       A2ACCTYP       M2EXCFLG       B2BUSTYP
##              0              0              0              0              0
##       F2HOMRGN       F2HOMFCY            AGE         TENURE        ZIPCODE
##              0              0              0              0              0
##       MBRCOUNT       DISTANCE  EARLYFAREWELL HOMEFCTYCHANGE   RECENTMOVING
##              0              0              0              0              0
##        SHOP1YR         SHOP6M         SHOP3M       ECOMSHOP        GASSHOP
##              0              0              0              0              0
##    MEDICALSHOP    GROCERYSHOP
##              0              0
```

```
head(churn_data)
```

```
## # A tibble: 6 x 22
##   RENEW A2ACCIPK A2ACCTYP M2EXCFLG B2BUSTYP F2HOMRGN F2HOMFCY   AGE TENURE
##   <chr>    <dbl>    <dbl> <chr>       <dbl> <chr>       <dbl> <dbl>  <dbl>
## 1 N       280928        1 N               0 NE           1078    42       1
## 2 N       280100        1 N               0 BO            847    61       1
## 3 N       279886        1 N               0 BO            847    52       1
## 4 N       279912        1 N               0 SE            185    32       1
## 5 N       279896        1 E               0 BA            472    46       1
## 6 N       279912        1 E               0 BD            823    36       1
## # … with 13 more variables: ZIPCODE <dbl>, MBRCOUNT <dbl>, DISTANCE <dbl>,
## #   EARLYFAREWELL <dbl>, HOMEFCTYCHANGE <chr>, RECENTMOVING <chr>,
## #   SHOP1YR <dbl>, SHOP6M <dbl>, SHOP3M <dbl>, ECOMSHOP <dbl>,
## #   GASSHOP <dbl>, MEDICALSHOP <dbl>, GROCERYSHOP <dbl>
```

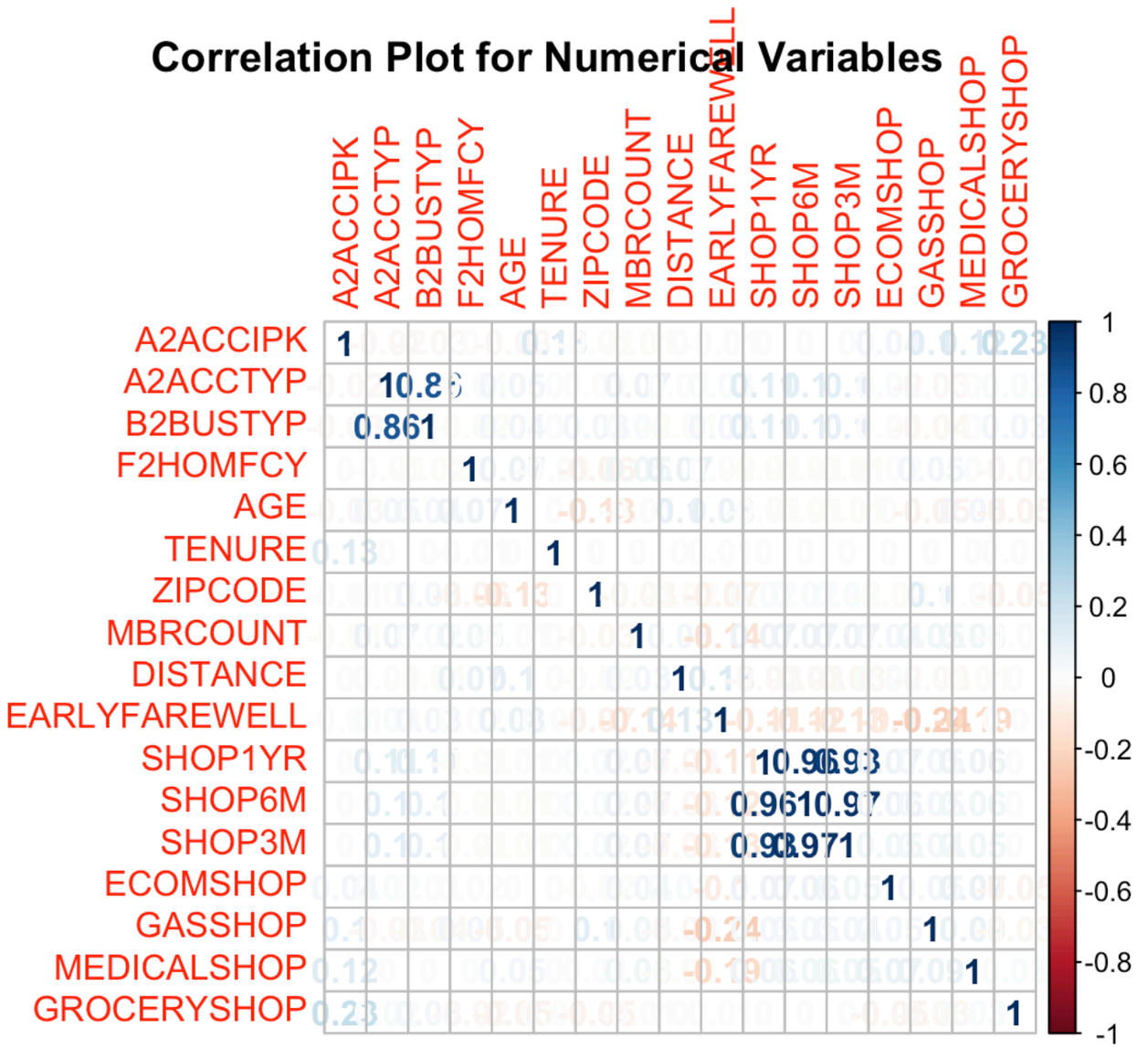
```
str(churn_data)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 120450 obs. of  22 var
iables:
##  $ RENEW         : chr  "N" "N" "N" "N" ...
##  $ A2ACCIPK      : num  280928 280100 279886 279912 279896 ...
##  $ A2ACCTYP      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ M2EXCFLG      : chr  "N" "N" "N" "N" ...
##  $ B2BUSTYP      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ F2HOMRGN      : chr  "NE" "BO" "BO" "SE" ...
##  $ F2HOMFCY      : num  1078 847 847 185 472 ...
##  $ AGE           : num  42 61 52 32 46 36 34 45 52 32 ...
##  $ TENURE        : num  1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ ZIPCODE       : num  20715 77346 91024 32789 93960 ...
##  $ MBRCOUNT      : num  2 2 2 2 2 1 2 2 2 2 ...
##  $ DISTANCE      : num  7.53 6.05 7.89 3.43 26.29 ...
##  $ EARLYFAREWELL : num  75 320 350 137 41 53 38 363 53 64 ...
##  $ HOMEFCTYCHANGE: chr  "Y" "N" "N" "N" ...
##  $ RECENTMOVING  : chr  "N" "N" "N" "N" ...
##  $ SHOP1YR       : num  1385 3500 114 997 12579 ...
##  $ SHOP6M        : num  827.7 0 0 23.2 73 ...
##  $ SHOP3M        : num  253 0 0 0 73 ...
##  $ ECOMSHOP      : num  0 1 0 0 0 0 0 0 0 0 ...
##  $ GASSHOP       : num  0.0293 0 0 0.0251 0 ...
##  $ MEDICALSHOP   : num  0.0173 0 0.30377 0 0.00818 ...
##  $ GROCERYSHOP   : num  0.523 0 0.234 0.405 0.936 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   RENEW = col_character(),
##   ..   A2ACCIPK = col_double(),
##   ..   A2ACCTYP = col_double(),
##   ..   M2EXCFLG = col_character(),
##   ..   B2BUSTYP = col_double(),
##   ..   F2HOMRGN = col_character(),
##   ..   F2HOMFCY = col_double(),
##   ..   AGE = col_double(),
##   ..   TENURE = col_double(),
##   ..   ZIPCODE = col_double(),
##   ..   MBRCOUNT = col_double(),
##   ..   DISTANCE = col_double(),
##   ..   EARLYFAREWELL = col_double(),
##   ..   HOMEFCTYCHANGE = col_character(),
##   ..   RECENTMOVING = col_character(),
##   ..   SHOP1YR = col_double(),
##   ..   SHOP6M = col_double(),
##   ..   SHOP3M = col_double(),
##   ..   ECOMSHOP = col_double(),
##   ..   GASSHOP = col_double(),
##   ..   MEDICALSHOP = col_double(),
##   ..   GROCERYSHOP = col_double()
##   .. )
```

```
#check the duplicated ID column
#churn_data <-churn_data[!duplicated(churn_data$A2ACCIPK), ]
#this dataset is the combination of multiple datasets with unique customer ID
```

```
#Correlation between numeric variables
numeric_var <- sapply(churn_data, is.numeric)
matrix <- cor(churn_data[,numeric_var])

corrplot(matrix,main="\n\nCorrelation Plot for Numerical Variables", method="numbe
r")
```



From the correlation plot, we can see that: B2B and A2A are correlated;(0.86) Shop1yr and shop6m are correlated;(0.96) Shop1yr and shop3m are correlated ;(0.93) shop6m and shop3m are correlated;(0.97)

```
#get the numerical variables
numeric_var <- sapply(churn_data, is.numeric)

#round the numerical variables in two decimals
mynew03 <- churn_data %>% mutate_if(is.numeric, ~round(., 2))

# drop the irrelevant columns ( member No.)
mynew04 <- mynew03[,-2]

# drop the highly related columns
mynew04$A2ACCTYP <- NULL
mynew04$SHOP6M <- NULL
mynew04$SHOP3M <- NULL

mynew04$RENEW <- ifelse(mynew04$RENEW == "Y", 1, 0)
mynew04$RENEW <- factor(mynew04$RENEW, levels = c(0, 1))
str(mynew04)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    120450 obs. of  18 variables:
##  $ RENEW         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ M2EXCFLG      : chr  "N" "N" "N" "N" ...
##  $ B2BUSTYP      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ F2HOMRGN      : chr  "NE" "BO" "BO" "SE" ...
##  $ F2HOMFCY      : num  1078 847 847 185 472 ...
##  $ AGE           : num  42 61 52 32 46 36 34 45 52 32 ...
##  $ TENURE        : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ ZIPCODE       : num  20715 77346 91024 32789 93960 ...
##  $ MBRCOUNT      : num  2 2 2 2 2 1 2 2 2 2 ...
##  $ DISTANCE      : num  7.53 6.05 7.89 3.43 26.29 ...
##  $ EARLYFAREWELL : num  75 320 350 137 41 53 38 363 53 64 ...
##  $ HOMEFCTYCHANGE: chr  "Y" "N" "N" "N" ...
##  $ RECENTMOVING  : chr  "N" "N" "N" "N" ...
##  $ SHOP1YR       : num  1385 3500 114 997 12579 ...
##  $ ECOMSHOP      : num  0 1 0 0 0 0 0 0 0 0 ...
##  $ GASSHOP       : num  0.03 0 0 0.03 0 0 0 0 0.43 0.17 ...
##  $ MEDICALSHOP   : num  0.02 0 0.3 0 0.01 0 0 0 0.02 0 ...
##  $ GROCERYSHOP   : num  0.52 0 0.23 0.41 0.94 0.84 0.51 0.93 0.26 0.17 ...
```

```
#deal with the outliers
#get the numerical variables
numeric_var <- sapply(mynew04, is.numeric)

#get the mean, max, min for numerical varianles columns from the dataframe
colMeans(mynew04[numeric_var])
```

```
##      B2BUSTYP      F2HOMFCY           AGE        TENURE       ZIPCODE
##  3.138016e+02  6.708431e+02  4.321900e+01  9.997592e-01  6.063731e+04
##      MBRCOUNT      DISTANCE EARLYFAREWELL       SHOP1YR      ECOMSHOP
##  1.796920e+00  1.040180e+01  7.518697e+01  2.504813e+03  5.571660e-01
##       GASSHOP   MEDICALSHOP   GROCERYSHOP
##  1.624577e+00  2.038109e+00  3.496127e+00
```

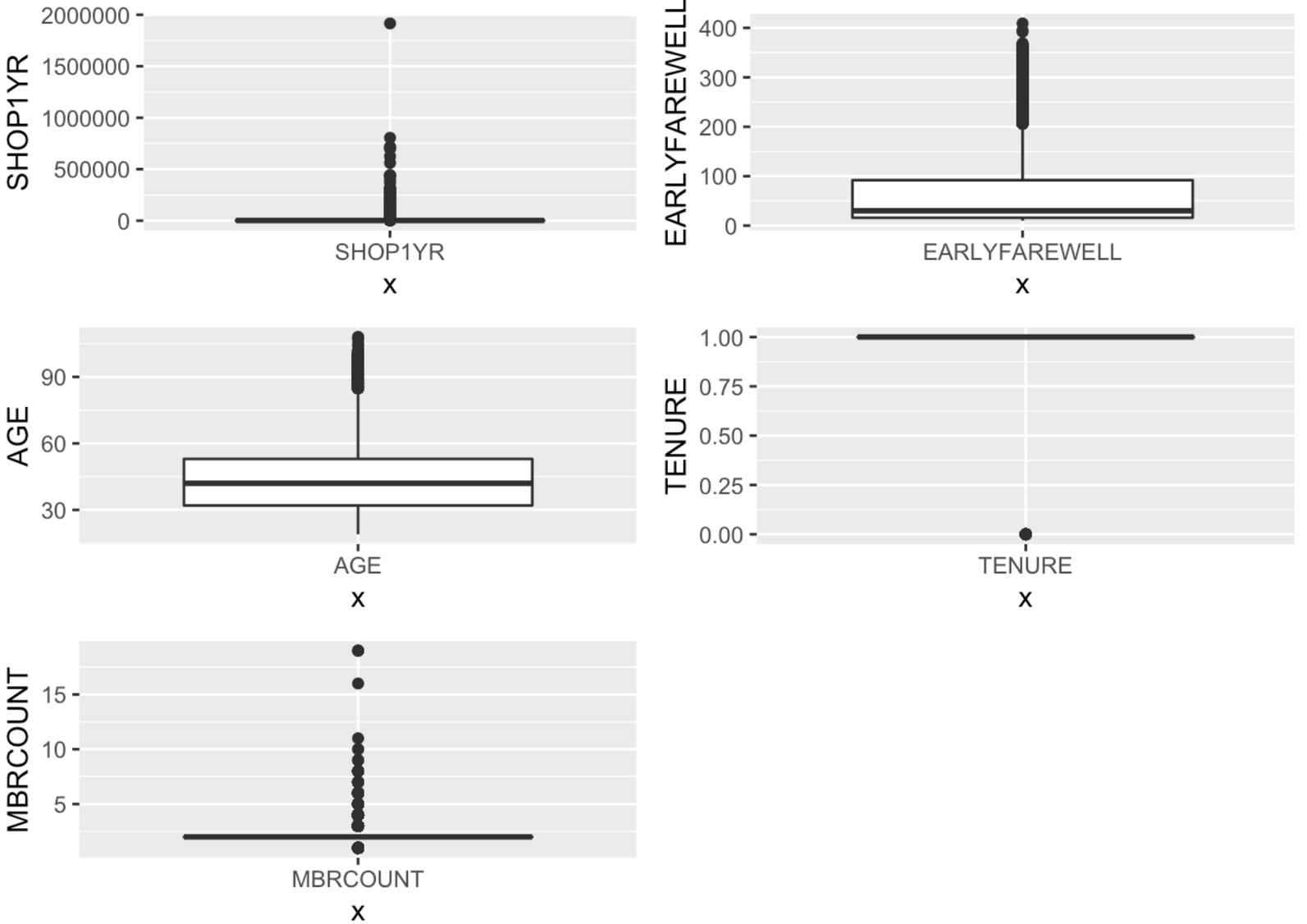
```
sapply(mynew04[numeric_var],max)
```

```
##      B2BUSTYP      F2HOMFCY           AGE        TENURE       ZIPCODE
##       9999.00       1342.00        108.00          1.00      99925.00
##      MBRCOUNT      DISTANCE EARLYFAREWELL       SHOP1YR      ECOMSHOP
##         19.00        463.34        409.00    1915896.00          9.00
##       GASSHOP   MEDICALSHOP   GROCERYSHOP
##          9.00          9.00          9.00
```

```
sapply(mynew04[numeric_var],min)
```

```
##      B2BUSTYP      F2HOMFCY           AGE        TENURE       ZIPCODE
##          0.00          1.00         19.00          0.00        601.00
##      MBRCOUNT      DISTANCE EARLYFAREWELL       SHOP1YR      ECOMSHOP
##          1.00          0.12         10.00          0.00          0.00
##       GASSHOP   MEDICALSHOP   GROCERYSHOP
##          0.00          0.00          0.00
```

```
p1 <- ggplot(mynew04, aes(x = "SHOP1YR", y = SHOP1YR)) +
    geom_boxplot()

p2<-ggplot(mynew04, aes(x = "EARLYFAREWELL", y = EARLYFAREWELL)) +
    geom_boxplot()
p3 <- ggplot(churn_data, aes(x = "AGE", y = AGE)) +
    geom_boxplot()
p4 <- ggplot(churn_data, aes(x = "TENURE", y = TENURE)) +
    geom_boxplot()

p5 <- ggplot(churn_data, aes(x = "MBRCOUNT", y = MBRCOUNT)) +
    geom_boxplot()

grid.arrange(p1,p2,p3,p4,p5)
```



we removes outliers when: we don’t have a lot of time to figure out why you have outliers we have a large amount of data without outliers we have outliers due to measurement or data entry errors

```
#One way to identify outliers is to determine which points have a z-score that's f
ar from 0.
#We can use the scores() function in the outliers package
#identify which roes contain outliers (SHOP1YR)
library(outliers)
# get the z-scores for
outlier_scores_1YR <- scores(mynew04$SHOP1YR)

#use threshold =3
#it is "TRUE" if outlier_scores is greater than 3
# it is false if outlier_scores is less than negative 3
is_outlier1YR <- outlier_scores_1YR > 3 | outlier_scores_1YR < -3

# add a column with info whether the refund_value is an outlier
mynew04$is_outlier <- is_outlier1YR

# create a dataframe with only outliers
churn_outliers_1YR <- mynew04[outlier_scores_1YR > 3| outlier_scores_1YR < -3, ]
str(churn_outliers_1YR)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    254 obs. of  19 variables:
##  $ RENEW          : Factor w/ 2 levels "0","1": 2 2 1 2 2 2 1 2 2 2 ...
##  $ M2EXCFLG       : chr  "E" "E" "E" "E" ...
##  $ B2BUSTYP       : num  0 5993 0 0 0 ...
##  $ F2HOMRGN       : chr  "MW" "BD" "NW" "LA" ...
##  $ F2HOMFCY       : num  1040 767 10 741 473 847 214 230 6 128 ...
##  $ AGE            : num  38 58 27 42 35 48 47 54 39 40 ...
##  $ TENURE         : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ ZIPCODE        : num  60175 98002 99654 93420 91789 ...
##  $ MBRCOUNT       : num  2 2 2 2 2 2 2 5 2 2 ...
##  $ DISTANCE       : num  3.7 4.88 29.52 13.58 4.11 ...
##  $ EARLYFAREWELL  : num  25 15 15 101 12 12 14 11 11 16 ...
##  $ HOMEFCTYCHANGE: chr  "N" "N" "N" "Y" ...
##  $ RECENTMOVING   : chr  "N" "N" "N" "N" ...
##  $ SHOP1YR        : num  31225 392954 41426 58166 61630 ...
##  $ ECOMSHOP       : num  0.04 0 0 0 0.28 0.85 0 0 0 0 ...
##  $ GASSHOP        : num  0 0 0 0 0.02 0 3 2 9 8 ...
##  $ MEDICALSHOP    : num  0.38 0 0.01 0 0.3 0.33 0 6 4 2 ...
##  $ GROCERYSHOP    : num  0.08 0.01 0.44 0.01 0.15 0.08 1 9 5 3 ...
##  $ is_outlier     : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
```

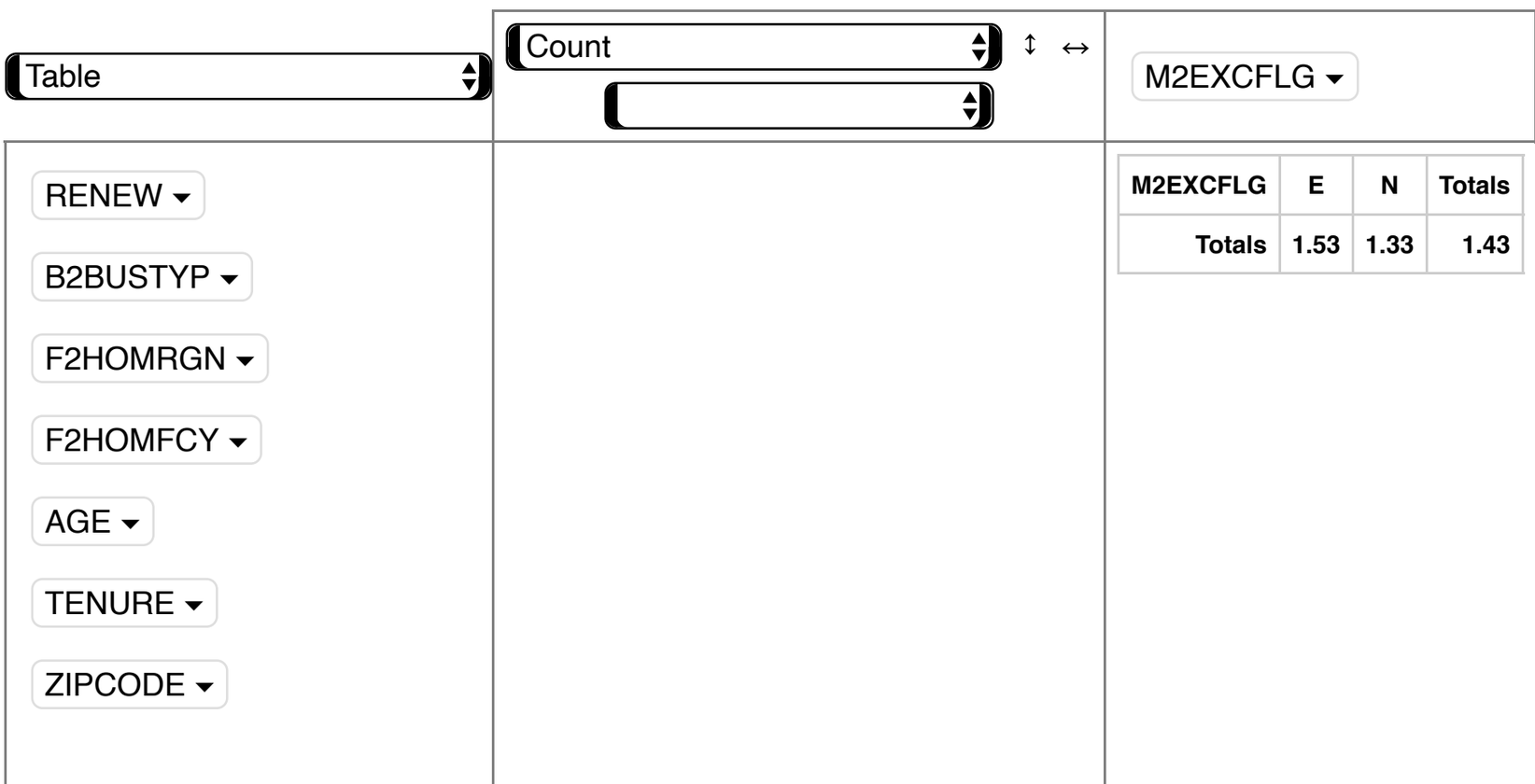
```
#Remove rows with outliers from churn dataset
churn_clean1<- mynew04[mynew04$is_outlier== F, ]
churn_clean1$is_outlier <- NULL
str(churn_clean1)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    120196 obs. of  18 variables:
##  $ RENEW          : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ M2EXCFLG       : chr  "N" "N" "N" "N" ...
##  $ B2BUSTYP       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ F2HOMRGN       : chr  "NE" "BO" "BO" "SE" ...
##  $ F2HOMFCY       : num  1078 847 847 185 472 ...
##  $ AGE            : num  42 61 52 32 46 36 34 45 52 32 ...
##  $ TENURE         : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ ZIPCODE        : num  20715 77346 91024 32789 93960 ...
##  $ MBRCOUNT       : num  2 2 2 2 2 1 2 2 2 2 ...
##  $ DISTANCE       : num  7.53 6.05 7.89 3.43 26.29 ...
##  $ EARLYFAREWELL  : num  75 320 350 137 41 53 38 363 53 64 ...
##  $ HOMEFCTYCHANGE: chr  "Y" "N" "N" "N" ...
##  $ RECENTMOVING   : chr  "N" "N" "N" "N" ...
##  $ SHOP1YR        : num  1385 3500 114 997 12579 ...
##  $ ECOMSHOP       : num  0 1 0 0 0 0 0 0 0 0 ...
##  $ GASSHOP        : num  0.03 0 0 0.03 0 0 0 0 0.43 0.17 ...
##  $ MEDICALSHOP    : num  0.02 0 0.3 0 0.01 0 0 0 0.02 0 ...
##  $ GROCERYSHOP    : num  0.52 0 0.23 0.41 0.94 0.84 0.51 0.93 0.26 0.17 ...
```

```
#encode the response variable into a factor variable of 1 and 0
churn_clean1$RENEW <- as.numeric(churn_clean1$RENEW)
str(churn_clean1)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    120196 obs. of  18 variables:
##  $ RENEW          : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ M2EXCFLG       : chr  "N" "N" "N" "N" ...
##  $ B2BUSTYP       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ F2HOMRGN       : chr  "NE" "BO" "BO" "SE" ...
##  $ F2HOMFCY       : num  1078 847 847 185 472 ...
##  $ AGE            : num  42 61 52 32 46 36 34 45 52 32 ...
##  $ TENURE         : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ ZIPCODE        : num  20715 77346 91024 32789 93960 ...
##  $ MBRCOUNT       : num  2 2 2 2 2 1 2 2 2 2 ...
##  $ DISTANCE       : num  7.53 6.05 7.89 3.43 26.29 ...
##  $ EARLYFAREWELL  : num  75 320 350 137 41 53 38 363 53 64 ...
##  $ HOMEFCTYCHANGE: chr  "Y" "N" "N" "N" ...
##  $ RECENTMOVING   : chr  "N" "N" "N" "N" ...
##  $ SHOP1YR        : num  1385 3500 114 997 12579 ...
##  $ ECOMSHOP       : num  0 1 0 0 0 0 0 0 0 0 ...
##  $ GASSHOP        : num  0.03 0 0 0.03 0 0 0 0 0.43 0.17 ...
##  $ MEDICALSHOP    : num  0.02 0 0.3 0 0.01 0 0 0 0.02 0 ...
##  $ GROCERYSHOP    : num  0.52 0 0.23 0.41 0.94 0.84 0.51 0.93 0.26 0.17 ...
```

```
library(rpivotTable)
#Categorical variables (M2EXCFLG,F2HOMRGN,HOMEFCTYCHANGE,RECENTMOVING)
rpivotTable(churn_clean1, cols=c("M2EXCFLG"),vals = "RENEW", aggregatorName = "Ave
rage", width="100%", height="400px")
```



M2EXCFLG	E	N	Totals
Totals	1.53	1.33	1.43

MBRCOUNT

DISTANCE

EARLYFAREWELL

HOMEFCTYCHANGE

RECENTMOVING

SHOP1YR

ECOMSHOP

GASSHOP

MEDICALSHOP

GROCERYSHOP

```
#E=1.53 & N= 1.33

#F2HOMRGN
#BO=1.17
#TE=1.39
#(1.4-1.45):BD, NE,NW,SD,SE
#(1.46-1.5): BA; MW;LA

#HOMEFCTYCHANGE
#N=1.41 & Y=1.51

#RECENTMOVING)
#N=1.43 & Y=1.44

#deal with the region column
churn_clean1$F2HOMRGN_BO<- churn_clean1$F2HOMRGN %in% c("BO")
churn_clean1$F2HOMRGN_TE<- churn_clean1$F2HOMRGN %in% c("TE")
churn_clean1$F2HOMRGN_middle<- churn_clean1$F2HOMRGN %in% c("BD","NE","NW","SD","S
E")
churn_clean1$F2HOMRGN_high<- churn_clean1$F2HOMRGN %in% c("BA","LA","MW")
```

```
      BA    BD       BO       LA    MW       NE         NW  SD  SE  TE  Totals
    1.48    1.42    1.17    1.46    1.48    1.43    1.45    1.44    1.41    1.39
1.44
```

```
#Convert characters to binary factors
library(caret)
dmy <- dummyVars(" ~ .", data = churn_clean1[c(2,12,13,19:22)])
trsf <- data.frame(predict(dmy, newdata = churn_clean1))
churn_clean1 <- data.frame(c(churn_clean1,trsf))

#get the dataset for model processing
mynew05<- churn_clean1[,c(1,3,5:11,14:18,23:36)]
mynew05$RENEW <- as.factor(mynew05$RENEW)

mynew05$RENEW <- ifelse(mynew05$RENEW == "2", 1, 0)
mynew05$RENEW <- as.factor(mynew05$RENEW)
str(mynew05)
```

```
## 'data.frame':    120196 obs. of  28 variables:
##  $ RENEW               : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ B2BUSTYP            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ F2HOMFCY            : num  1078 847 847 185 472 ...
##  $ AGE                 : num  42 61 52 32 46 36 34 45 52 32 ...
##  $ TENURE              : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ ZIPCODE             : num  20715 77346 91024 32789 93960 ...
##  $ MBRCOUNT            : num  2 2 2 2 2 1 2 2 2 2 ...
##  $ DISTANCE            : num  7.53 6.05 7.89 3.43 26.29 ...
##  $ EARLYFAREWELL       : num  75 320 350 137 41 53 38 363 53 64 ...
##  $ SHOP1YR             : num  1385 3500 114 997 12579 ...
##  $ ECOMSHOP            : num  0 1 0 0 0 0 0 0 0 0 ...
##  $ GASSHOP             : num  0.03 0 0 0.03 0 0 0 0 0.43 0.17 ...
##  $ MEDICALSHOP         : num  0.02 0 0.3 0 0.01 0 0 0 0.02 0 ...
##  $ GROCERYSHOP         : num  0.52 0 0.23 0.41 0.94 0.84 0.51 0.93 0.26 0.17 ..
.
##  $ M2EXCFLGE           : num  0 0 0 0 1 1 0 0 0 1 ...
##  $ M2EXCFLGN           : num  1 1 1 1 0 0 1 1 1 0 ...
##  $ HOMEFCTYCHANGEN     : num  0 1 1 1 1 1 1 1 1 0 ...
##  $ HOMEFCTYCHANGEY     : num  1 0 0 0 0 0 0 0 0 1 ...
##  $ RECENTMOVINGN       : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ RECENTMOVINGY       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ F2HOMRGN_BOFALSE    : num  1 0 0 1 1 1 1 1 1 1 ...
##  $ F2HOMRGN_BOTRUE     : num  0 1 1 0 0 0 0 0 0 0 ...
##  $ F2HOMRGN_TEFALSE    : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ F2HOMRGN_TETRUE     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ F2HOMRGN_middleFALSE: num  0 1 1 0 1 0 0 1 0 1 ...
##  $ F2HOMRGN_middleTRUE : num  1 0 0 1 0 1 1 0 1 0 ...
##  $ F2HOMRGN_highFALSE  : num  1 1 1 1 0 1 1 0 1 0 ...
##  $ F2HOMRGN_highTRUE   : num  0 0 0 0 1 0 0 1 0 1 ...
```

```
# we need to make sure the training data has approximately equal proportion of cla
ss
table(mynew05$RENEW)
```

```
##
##     0     1
## 67943 52253
```

```
#set up the traing and testing dataset
set.seed(500)
index <- createDataPartition(mynew05$RENEW, p = 0.7, list = FALSE)
mytrain_data <- mynew05[index, ]
mytest_data  <- mynew05[-index, ]

table(mytrain_data$RENEW)
```

```
##
##     0     1
## 47561 36578
```

```
table(mytest_data$RENEW)
```

```
##
##     0     1
## 20382 15675
```

```
#logistic regression model
#train the model
set.seed(111)
logitmodel <- glm(RENEW ~.,family=binomial(link="logit"), data=mytrain_data)
summary(logitmodel)
```

```
##
## Call:
## glm(formula = RENEW ~ ., family = binomial(link = "logit"), data = mytrain_data
)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.6057  -0.9601  -0.3696   1.0089   2.9452
##
## Coefficients: (8 not defined because of singularities)
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -1.283e+00  5.465e-01  -2.347  0.01893 *
## B2BUSTYP            -4.626e-06  5.612e-06  -0.824  0.40978
## F2HOMFCY             5.154e-05  2.090e-05   2.466  0.01367 *
## AGE                  2.085e-02  5.617e-04  37.117  < 2e-16 ***
## TENURE              -7.313e-01  5.384e-01  -1.358  0.17439
## ZIPCODE              1.941e-07  2.796e-07   0.694  0.48754
## MBRCOUNT             2.453e-02  1.885e-02   1.301  0.19325
## DISTANCE            -9.502e-04  4.817e-04  -1.973  0.04854 *
## EARLYFAREWELL       -8.499e-03  1.412e-04 -60.205  < 2e-16 ***
## SHOP1YR              2.420e-04  4.878e-06  49.609  < 2e-16 ***
## ECOMSHOP             2.865e-02  5.041e-03   5.683 1.32e-08 ***
## GASSHOP              7.659e-03  3.367e-03   2.275  0.02292 *
## MEDICALSHOP          9.831e-03  3.107e-03   3.165  0.00155 **
## GROCERYSHOP         -6.070e-02  3.672e-03 -16.529  < 2e-16 ***
## M2EXCFLGE            4.517e-01  1.640e-02  27.547  < 2e-16 ***
## M2EXCFLGN                   NA         NA      NA       NA
## HOMEFCTYCHANGEN     -1.228e-01  1.827e-02  -6.724 1.77e-11 ***
## HOMEFCTYCHANGEY             NA         NA      NA       NA
## RECENTMOVINGN        6.495e-02  2.537e-02   2.560  0.01046 *
```

```
## RECENTMOVINGY                 NA         NA      NA       NA
## F2HOMRGN_BOFALSE      3.573e-01  6.414e-02   5.570 2.55e-08 ***
## F2HOMRGN_BOTRUE               NA         NA      NA       NA
## F2HOMRGN_TEFALSE      3.039e-01  3.143e-02   9.669  < 2e-16 ***
## F2HOMRGN_TETRUE               NA         NA      NA       NA
## F2HOMRGN_middleFALSE  1.550e-01  1.814e-02   8.545  < 2e-16 ***
## F2HOMRGN_middleTRUE           NA         NA      NA       NA
## F2HOMRGN_highFALSE            NA         NA      NA       NA
## F2HOMRGN_highTRUE             NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 115204  on 84138  degrees of freedom
## Residual deviance:  94986  on 84119  degrees of freedom
## AIC: 95026
##
## Number of Fisher Scoring iterations: 5
```

```
#predict the churn posibility
logpred_prob <- predict(logitmodel, newdata = mytest_data,type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
## == : prediction from a rank-deficient fit may be misleading
```

```
#show the confusion matrix
#evaluate the accuracy
caret::confusionMatrix(as.factor(ifelse(logpred_prob> 0.5, 1, 0)), mytest_data$REN
EW)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 15674  5799
##          1  4708  9876
##
##                Accuracy : 0.7086
##                  95% CI : (0.7039, 0.7133)
##     No Information Rate : 0.5653
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4023
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7690
##             Specificity : 0.6300
##          Pos Pred Value : 0.7299
##          Neg Pred Value : 0.6772
##              Prevalence : 0.5653
##          Detection Rate : 0.4347
##    Detection Prevalence : 0.5955
##       Balanced Accuracy : 0.6995
##
##        'Positive' Class : 0
##
```

```
#the model accuracy is 0.7086
```

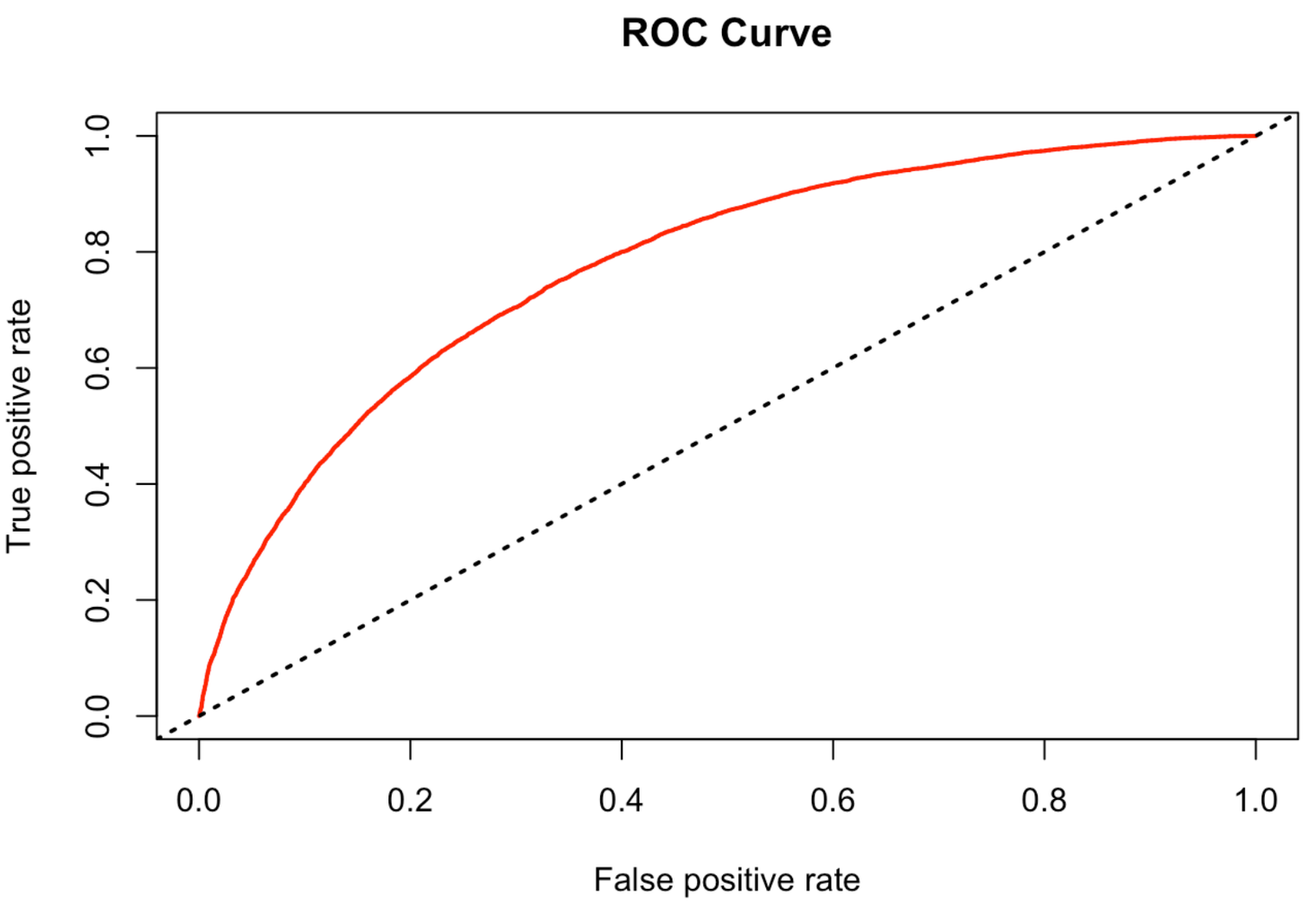
```
#The InformationValue::optimalCutoff function provides ways to find the optimal cu
toff to improve the prediction
library(InformationValue)
predicted <- predict(logitmodel, mytest_data,  type="response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
## == : prediction from a rank-deficient fit may be misleading
```

```
optimalCutoff(mytest_data$RENEW, predicted)[1]
```

```
## [1] 0.4993099
```

```
#look at the ROC curve and the AUC value
library(ROCR)
pr <- prediction(logpred_prob, mytest_data$RENEW)
# plotting ROC curve
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf,main = "ROC Curve",col = 2,lwd = 2)
abline(a = 0,b = 1,lwd = 2,lty = 3,col = "black")
```




```
# AUC value
#AUC stands for "Area under the ROC Curve
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.7761531
```

```
# the auc value is 0.776, which represents the quality of the model's predictions
irrespective of what classification threshold is chosen.
#the larger the area under the ROC curve, the better is your model
```

```
#we need to do feature selections to tune the model to get the higer model accurac
y
#A good feature is when we can distinguish between churn and non-churn customers
set.seed(111)
library(MASS)
fit_1 <- glm(RENEW ~., family=binomial(link="logit"), data=mytrain_data)
step <-stepAIC(fit_1, trace=FALSE,direction = "both")
step$anova
```

```
## Stepwise Model Path
## Analysis of Deviance Table
##
## Initial Model:
## RENEW ~ B2BUSTYP + F2HOMFCY + AGE + TENURE + ZIPCODE + MBRCOUNT +
##     DISTANCE + EARLYFAREWELL + SHOP1YR + ECOMSHOP + GASSHOP +
##     MEDICALSHOP + GROCERYSHOP + M2EXCFLGE + M2EXCFLGN + HOMEFCTYCHANGEN +
##     HOMEFCTYCHANGEY + RECENTMOVINGN + RECENTMOVINGY + F2HOMRGN_BOFALSE +
##     F2HOMRGN_BOTRUE + F2HOMRGN_TEFALSE + F2HOMRGN_TETRUE + F2HOMRGN_middleFALSE
+
##     F2HOMRGN_middleTRUE + F2HOMRGN_highFALSE + F2HOMRGN_highTRUE
##
## Final Model:
## RENEW ~ F2HOMFCY + AGE + TENURE + DISTANCE + EARLYFAREWELL +
##     SHOP1YR + ECOMSHOP + GASSHOP + MEDICALSHOP + GROCERYSHOP +
##     M2EXCFLGE + HOMEFCTYCHANGEN + RECENTMOVINGN + F2HOMRGN_BOFALSE +
##     F2HOMRGN_TEFALSE + F2HOMRGN_middleFALSE
##
##
##                     Step Df  Deviance Resid. Df Resid. Dev      AIC
## 1                                         84119   94986.02 95026.02
## 2    - F2HOMRGN_highTRUE  0 0.0000000     84119   94986.02 95026.02
## 3   - F2HOMRGN_highFALSE  0 0.0000000     84119   94986.02 95026.02
## 4  - F2HOMRGN_middleTRUE  0 0.0000000     84119   94986.02 95026.02
## 5      - F2HOMRGN_TETRUE  0 0.0000000     84119   94986.02 95026.02
## 6      - F2HOMRGN_BOTRUE  0 0.0000000     84119   94986.02 95026.02
## 7        - RECENTMOVINGY  0 0.0000000     84119   94986.02 95026.02
## 8      - HOMEFCTYCHANGEY  0 0.0000000     84119   94986.02 95026.02
## 9            - M2EXCFLGN  0 0.0000000     84119   94986.02 95026.02
## 10             - ZIPCODE  1 0.4819438     84120   94986.51 95024.51
## 11            - B2BUSTYP  1 0.6553322     84121   94987.16 95023.16
## 12            - MBRCOUNT  1 1.5842402     84122   94988.75 95022.75
```

```
summary(step)
```

```
##
## Call:
## glm(formula = RENEW ~ F2HOMFCY + AGE + TENURE + DISTANCE + EARLYFAREWELL +
##     SHOP1YR + ECOMSHOP + GASSHOP + MEDICALSHOP + GROCERYSHOP +
##     M2EXCFLGE + HOMEFCTYCHANGEN + RECENTMOVINGN + F2HOMRGN_BOFALSE +
##     F2HOMRGN_TEFALSE + F2HOMRGN_middleFALSE, family = binomial(link = "logit"),
##     data = mytrain_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.6140  -0.9600  -0.3695   1.0090   2.9495
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -1.230e+00  5.450e-01  -2.257  0.02403 *
## F2HOMFCY              5.094e-05  2.066e-05   2.465  0.01368 *
## AGE                   2.077e-02  5.570e-04  37.294  < 2e-16 ***
## TENURE               -7.308e-01  5.383e-01  -1.358  0.17457
## DISTANCE             -9.080e-04  4.809e-04  -1.888  0.05903 .
## EARLYFAREWELL        -8.519e-03  1.406e-04 -60.576  < 2e-16 ***
## SHOP1YR               2.424e-04  4.843e-06  50.040  < 2e-16 ***
## ECOMSHOP              2.850e-02  5.035e-03   5.659 1.52e-08 ***
## GASSHOP               7.911e-03  3.354e-03   2.359  0.01834 *
## MEDICALSHOP           9.898e-03  3.106e-03   3.187  0.00144 **
## GROCERYSHOP          -6.093e-02  3.666e-03 -16.620  < 2e-16 ***
## M2EXCFLGE             4.527e-01  1.631e-02  27.752  < 2e-16 ***
## HOMEFCTYCHANGEN      -1.227e-01  1.816e-02  -6.759 1.39e-11 ***
## RECENTMOVINGN         6.467e-02  2.536e-02   2.550  0.01078 *
## F2HOMRGN_BOFALSE      3.616e-01  6.396e-02   5.653 1.57e-08 ***
## F2HOMRGN_TEFALSE      3.027e-01  3.138e-02   9.647  < 2e-16 ***
## F2HOMRGN_middleFALSE  1.595e-01  1.683e-02   9.477  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 115204  on 84138  degrees of freedom
## Residual deviance:  94989  on 84122  degrees of freedom
## AIC: 95023
##
## Number of Fisher Scoring iterations: 5
```

```
#train the model after the features selection
set.seed(111)
logitmodel01 <- glm(RENEW ~ F2HOMFCY + AGE + TENURE + DISTANCE + EARLYFAREWELL +
    SHOP1YR + ECOMSHOP + GASSHOP + MEDICALSHOP + GROCERYSHOP +
    M2EXCFLGE + HOMEFCTYCHANGEN + RECENTMOVINGN + F2HOMRGN_BOFALSE +
    F2HOMRGN_TEFALSE + F2HOMRGN_middleFALSE,family=binomial(link="logit"), data=my
train_data)

summary(logitmodel01)
```

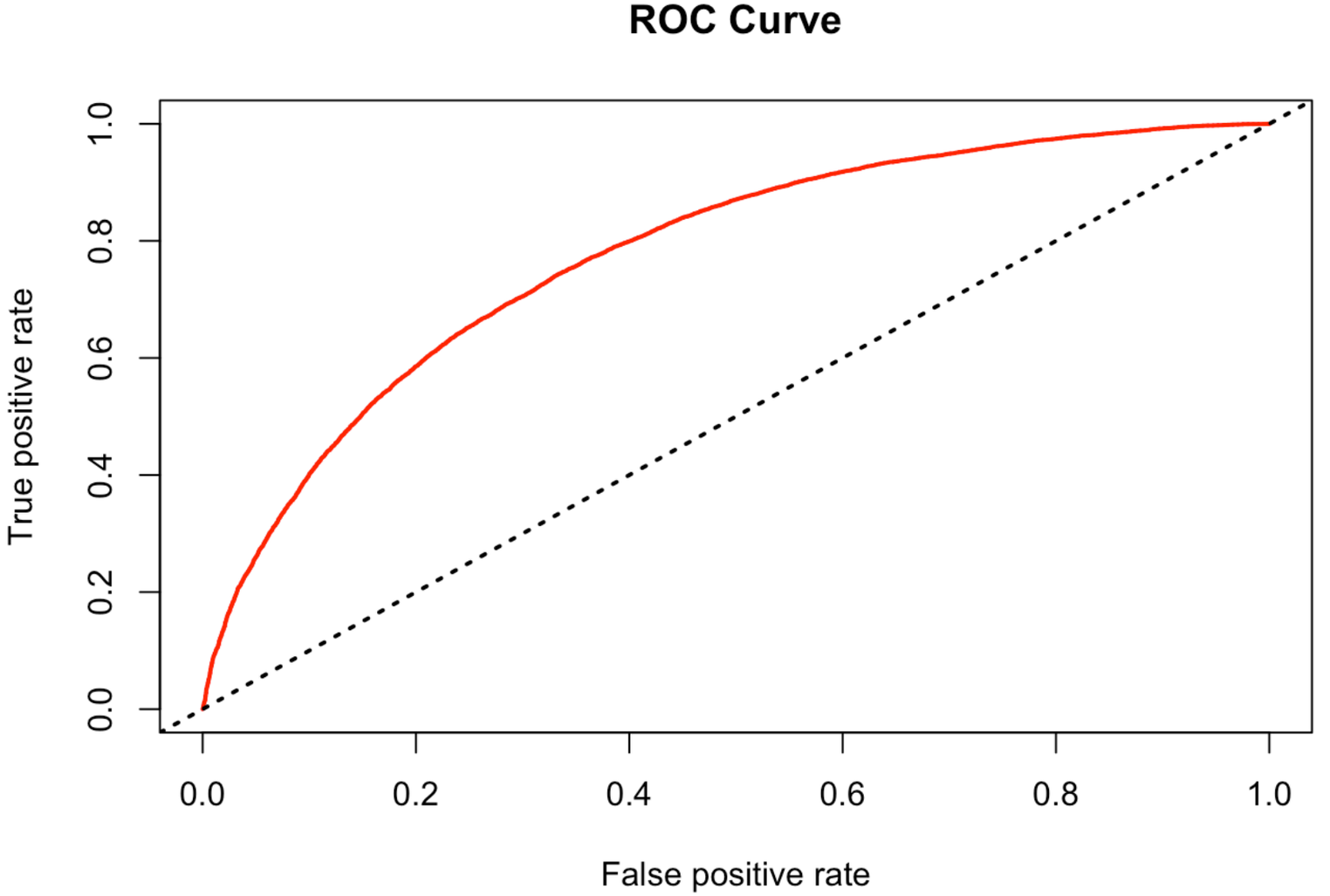
```
##
## Call:
## glm(formula = RENEW ~ F2HOMFCY + AGE + TENURE + DISTANCE + EARLYFAREWELL +
##     SHOP1YR + ECOMSHOP + GASSHOP + MEDICALSHOP + GROCERYSHOP +
##     M2EXCFLGE + HOMEFCTYCHANGEN + RECENTMOVINGN + F2HOMRGN_BOFALSE +
##     F2HOMRGN_TEFALSE + F2HOMRGN_middleFALSE, family = binomial(link = "logit"),
##     data = mytrain_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.6140  -0.9600  -0.3695   1.0090   2.9495
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -1.230e+00  5.450e-01  -2.257  0.02403 *
## F2HOMFCY              5.094e-05  2.066e-05   2.465  0.01368 *
## AGE                   2.077e-02  5.570e-04  37.294  < 2e-16 ***
## TENURE               -7.308e-01  5.383e-01  -1.358  0.17457
## DISTANCE             -9.080e-04  4.809e-04  -1.888  0.05903 .
## EARLYFAREWELL        -8.519e-03  1.406e-04 -60.576  < 2e-16 ***
## SHOP1YR               2.424e-04  4.843e-06  50.040  < 2e-16 ***
## ECOMSHOP              2.850e-02  5.035e-03   5.659 1.52e-08 ***
## GASSHOP               7.911e-03  3.354e-03   2.359  0.01834 *
## MEDICALSHOP           9.898e-03  3.106e-03   3.187  0.00144 **
## GROCERYSHOP          -6.093e-02  3.666e-03 -16.620  < 2e-16 ***
## M2EXCFLGE             4.527e-01  1.631e-02  27.752  < 2e-16 ***
## HOMEFCTYCHANGEN      -1.227e-01  1.816e-02  -6.759 1.39e-11 ***
## RECENTMOVINGN         6.467e-02  2.536e-02   2.550  0.01078 *
## F2HOMRGN_BOFALSE      3.616e-01  6.396e-02   5.653 1.57e-08 ***
## F2HOMRGN_TEFALSE      3.027e-01  3.138e-02   9.647  < 2e-16 ***
## F2HOMRGN_middleFALSE  1.595e-01  1.683e-02   9.477  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 115204  on 84138  degrees of freedom
## Residual deviance:  94989  on 84122  degrees of freedom
## AIC: 95023
##
## Number of Fisher Scoring iterations: 5
```

```
library(caret)
#predict the churn posibility
logpred_prob01 <- predict(logitmodel01, newdata = mytest_data, type = "response")

#evaluate the accuracy
caret::confusionMatrix(factor(ifelse(logpred_prob01> 0.5, 1, 0)), mytest_data$RENE
W)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 15677  5804
##          1  4705  9871
##
##                Accuracy : 0.7085
##                  95% CI : (0.7038, 0.7132)
##     No Information Rate : 0.5653
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4021
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7692
##             Specificity : 0.6297
##          Pos Pred Value : 0.7298
##          Neg Pred Value : 0.6772
##              Prevalence : 0.5653
##          Detection Rate : 0.4348
##    Detection Prevalence : 0.5958
##       Balanced Accuracy : 0.6994
##
##        'Positive' Class : 0
##
```

```
#the model accuracy is 0.7085
```

```
#look at the ROC curve and the AUC value
library(ROCR)
pr <- prediction(logpred_prob01, mytest_data$RENEW)
# plotting ROC curve
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf,main = "ROC Curve",col = 2,lwd = 2)
abline(a = 0,b = 1,lwd = 2,lty = 3,col = "black")
```




```
# AUC value
#AUC stands for "Area under the ROC Curve
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

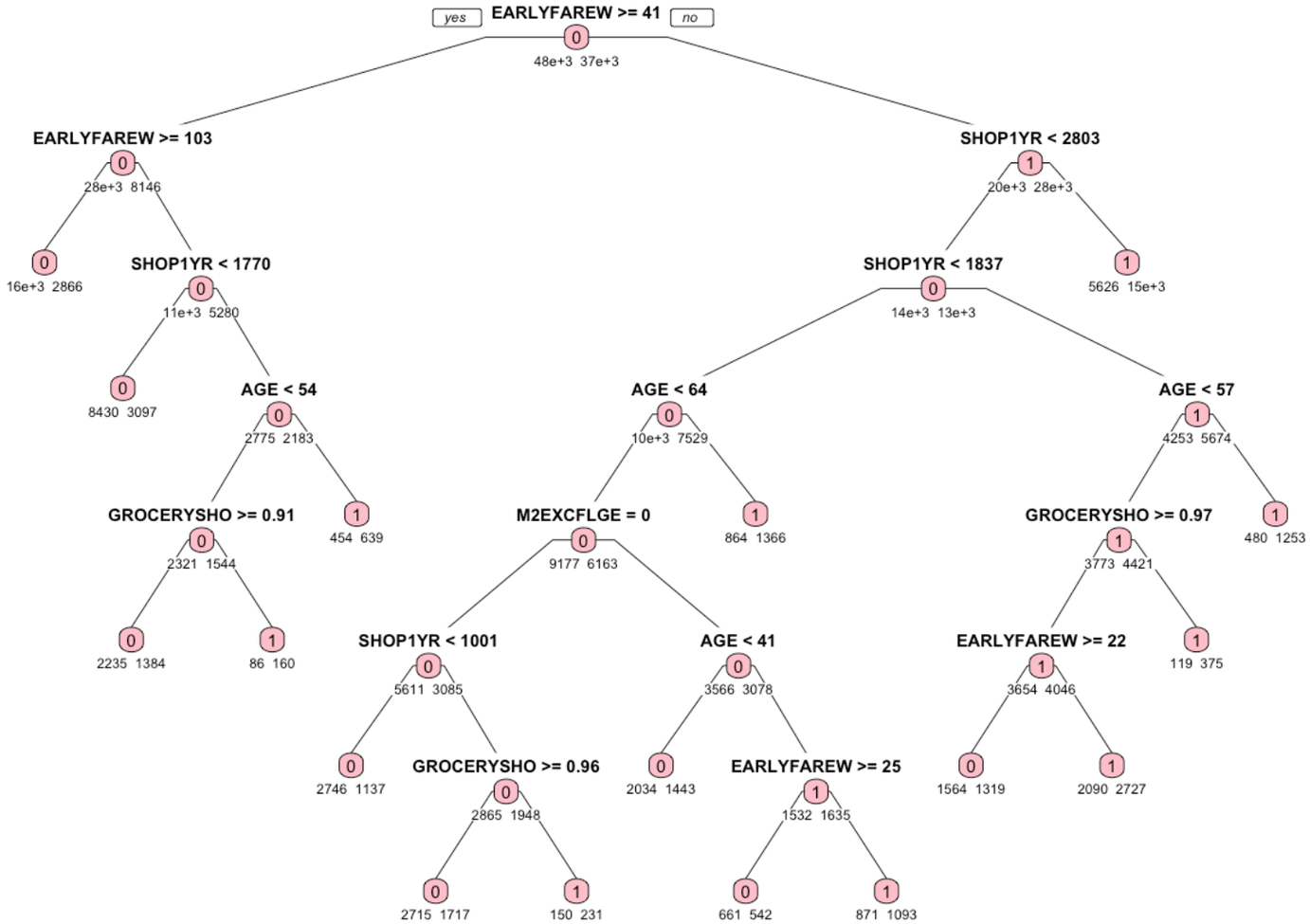
```
## [1] 0.7761827
```

```
#auc=0.776
```

```r
# classification tree
#unpruned tree
library(rpart)
library(rpart.plot)

#The minbucket provides the smallest number of observations that are allowed in a terminal node
#The minsplit parameter is the smallest number of observations in the parent node that could be split further
#the maxdepth parameter prevents the tree from growing past a certain depth / height
#cp: the minimum improvement in the model needed at each node
set.seed(1050)
class.tree <- rpart(RENEW ~., data=mytrain_data,
                    control = rpart.control(minbucket =7,minsplit=20,cp=0.001), method = "class")
printcp(class.tree)
```

```
##
## Classification tree:
## rpart(formula = RENEW ~ ., data = mytrain_data, method = "class",
##     control = rpart.control(minbucket = 7, minsplit = 20, cp = 0.001))
##
## Variables actually used in tree construction:
## [1] AGE           EARLYFAREWELL GROCERYSHOP   M2EXCFLGE     SHOP1YR
##
## Root node error: 36578/84139 = 0.43473
##
## n= 84139
##
##          CP nsplit rel error  xerror      xstd
## 1 0.2327082      0   1.00000 1.00000 0.0039311
## 2 0.0343376      1   0.76729 0.76975 0.0037419
## 3 0.0137241      3   0.69862 0.70231 0.0036521
## 4 0.0022327      4   0.68489 0.68979 0.0036336
## 5 0.0020231      7   0.67819 0.68350 0.0036240
## 6 0.0016859     10   0.67213 0.68309 0.0036234
## 7 0.0011072     14   0.66504 0.67809 0.0036157
## 8 0.0010000     16   0.66283 0.67368 0.0036088
```

```r
# plot tree
prp(class.tree,type = 1,extra = 1, under = TRUE, split.font = 2,varlen = -10, box.palette="pink")
```




```
#get the rule
rpart.rules(class.tree)
```

```
##  RENEW
##    0.15 when EARLYFAREWELL >=       103
##    0.27 when EARLYFAREWELL is 41 to 103 & SHOP1YR <  1770
##    0.29 when EARLYFAREWELL <  41        & SHOP1YR <  1001         & AGE <  64
& M2EXCFLGE is 0
##    0.38 when EARLYFAREWELL is 41 to 103 & SHOP1YR >=         1770 & AGE <  54
& GROCERYSHOP >= 0.91
##    0.39 when EARLYFAREWELL <  41        & SHOP1YR is 1001 to 1837 & AGE <  64
& GROCERYSHOP >= 0.96 & M2EXCFLGE is 0
##    0.42 when EARLYFAREWELL <  41        & SHOP1YR <  1837         & AGE <  41
& M2EXCFLGE is 1
##    0.45 when EARLYFAREWELL is 25 to  41 & SHOP1YR <  1837         & AGE is 41 to
64                       & M2EXCFLGE is 1
##    0.46 when EARLYFAREWELL is 22 to  41 & SHOP1YR is 1837 to 2803 & AGE <  57
& GROCERYSHOP >= 0.97
##    0.56 when EARLYFAREWELL <  25        & SHOP1YR <  1837         & AGE is 41 to
64                       & M2EXCFLGE is 1
##    0.57 when EARLYFAREWELL <  22        & SHOP1YR is 1837 to 2803 & AGE <  57
& GROCERYSHOP >= 0.97
##    0.58 when EARLYFAREWELL is 41 to 103 & SHOP1YR >=         1770 & AGE >=
54
##    0.61 when EARLYFAREWELL <  41        & SHOP1YR is 1001 to 1837 & AGE <  64
& GROCERYSHOP <  0.96 & M2EXCFLGE is 0
##    0.61 when EARLYFAREWELL <  41        & SHOP1YR <  1837         & AGE >=
64
##    0.65 when EARLYFAREWELL is 41 to 103 & SHOP1YR >=         1770 & AGE <  54
& GROCERYSHOP <  0.91
##    0.72 when EARLYFAREWELL <  41        & SHOP1YR is 1837 to 2803 & AGE >=
57
##    0.73 when EARLYFAREWELL <  41        & SHOP1YR >=         2803
##    0.76 when EARLYFAREWELL <  41        & SHOP1YR is 1837 to 2803 & AGE <  57
& GROCERYSHOP <  0.97
```

```
#get the importance
t(t(class.tree$variable.importance))
```

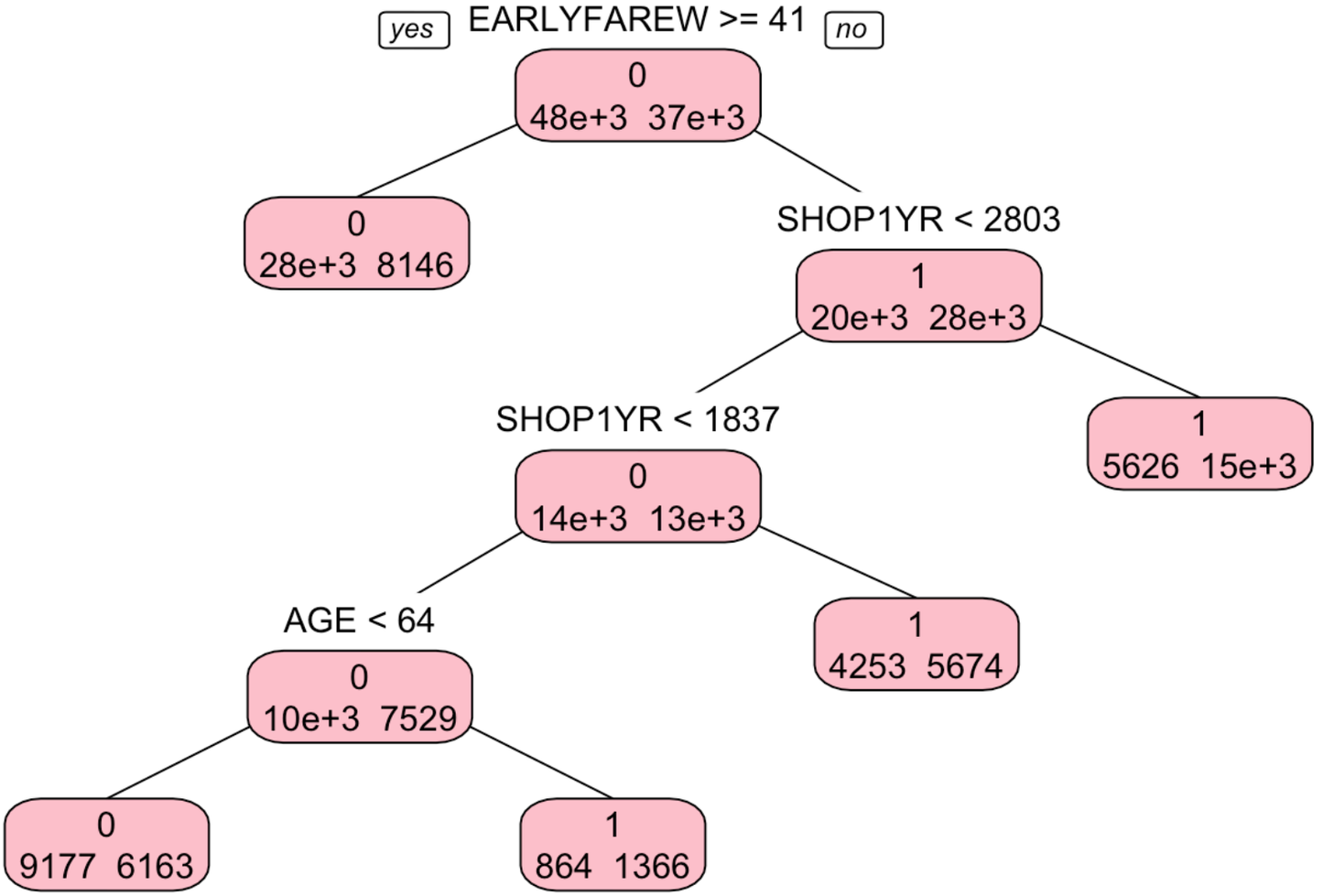
```
##                          [,1]
## EARLYFAREWELL    6.074814e+03
## SHOP1YR          4.329767e+03
## GASSHOP          1.165429e+03
## MEDICALSHOP      1.020142e+03
## GROCERYSHOP      4.055647e+02
## AGE              3.618981e+02
## DISTANCE         3.167036e+02
## M2EXCFLGE        2.524327e+02
## M2EXCFLGN        2.524327e+02
## ECOMSHOP         2.147290e+02
## HOMEFCTYCHANGEN  4.989633e+01
## HOMEFCTYCHANGEY  4.989633e+01
## ZIPCODE          4.560886e+00
## F2HOMFCY         2.445467e+00
## B2BUSTYP         2.247312e+00
## MBRCOUNT         1.131504e+00
## F2HOMRGN_BOFALSE 1.081570e+00
## F2HOMRGN_BOTRUE  1.081570e+00
## TENURE           1.475908e-02
```

```
#prediction
pred_tree<- predict(class.tree, newdata = mytest_data,type="class")
pred_prob <- predict(class.tree, newdata = mytest_data,type="prob")

#evaluation
caret::confusionMatrix(pred_tree,mytest_data$RENEW)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 15601  5721
##          1  4781  9954
##
##                Accuracy : 0.7087
##                  95% CI : (0.704, 0.7134)
##     No Information Rate : 0.5653
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4032
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7654
##             Specificity : 0.6350
##          Pos Pred Value : 0.7317
##          Neg Pred Value : 0.6755
##              Prevalence : 0.5653
##          Detection Rate : 0.4327
##    Detection Prevalence : 0.5913
##       Balanced Accuracy : 0.7002
##
##        'Positive' Class : 0
##
```

```
#accuracy=0.7087
```

```
# classification tree
#pruned tree
set.seed(1050)
ct <- rpart(RENEW ~., data=mytrain_data,, control = rpart.control(minbucket =7,min
split=20), method = "class")

# prune by lower cp
#returns the optimal cp value associated with the minimum error.
pruned.ct <- prune(ct,
                   cp = ct$cptable[which.min(ct$cptable[,"xerror"]),"CP"])
# plot tree
prp(pruned.ct, type = 1, extra = 1, split.font = 1, varlen = -10, box.palette="pin
k")
```




```
#get the rule
rpart.rules(pruned.ct)
```

```
##  RENEW
##    0.23 when EARLYFAREWELL >= 41
##    0.40 when EARLYFAREWELL <  41 & SHOP1YR <  1837         & AGE <  64
##    0.57 when EARLYFAREWELL <  41 & SHOP1YR is 1837 to 2803
##    0.61 when EARLYFAREWELL <  41 & SHOP1YR <  1837         & AGE >= 64
##    0.73 when EARLYFAREWELL <  41 & SHOP1YR >=         2803
```

```
#get the importance
t(t(ct$variable.importance))
```

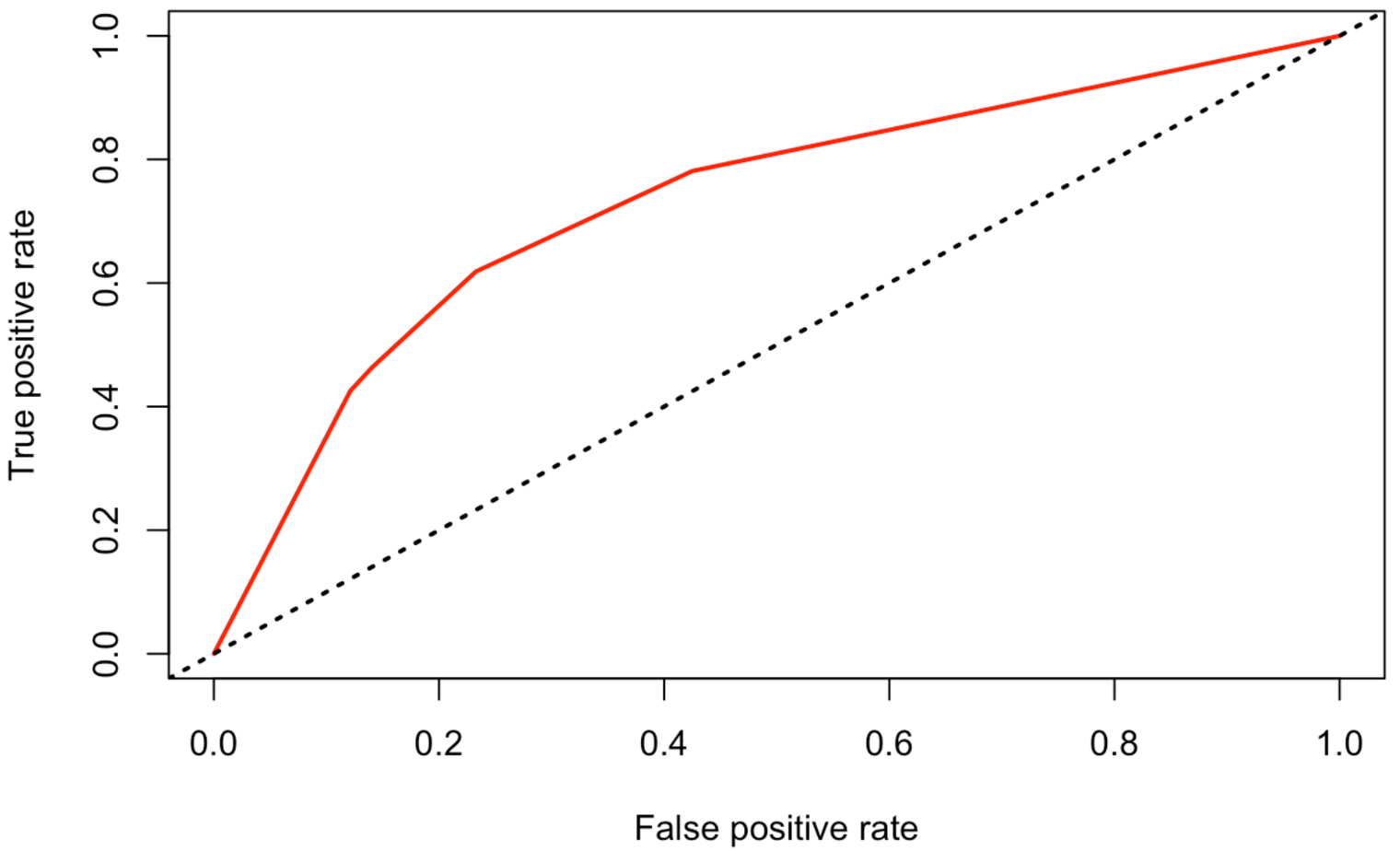
```
##                       [,1]
## EARLYFAREWELL 5.489385e+03
## SHOP1YR       3.904721e+03
## GASSHOP       1.080655e+03
## MEDICALSHOP   9.288826e+02
## DISTANCE      3.150643e+02
## GROCERYSHOP   2.873686e+02
## ECOMSHOP      2.113100e+02
## AGE           1.730266e+02
## M2EXCFLGE     1.637318e+02
## M2EXCFLGN     1.637318e+02
## ZIPCODE       1.307707e-01
## MBRCOUNT      5.230829e-02
## F2HOMFCY      2.615415e-02
```

```
#prediction
pred_tree01<- predict(ct, newdata = mytest_data,type="class")
pred_prob <- predict(ct, newdata = mytest_data,type="prob")

#evaluation
caret::confusionMatrix(pred_tree01 ,mytest_data$RENEW)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 15638  5979
##          1  4744  9696
##
##                Accuracy : 0.7026
##                  95% CI : (0.6979, 0.7073)
##     No Information Rate : 0.5653
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3894
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7672
##             Specificity : 0.6186
##          Pos Pred Value : 0.7234
##          Neg Pred Value : 0.6715
##              Prevalence : 0.5653
##          Detection Rate : 0.4337
##    Detection Prevalence : 0.5995
##       Balanced Accuracy : 0.6929
##
##        'Positive' Class : 0
##
```

```
#accuracy=0.7026
```

```
#look at the ROC curve and the AUC value
library(ROCR)
pr <- prediction(pred_prob[,2], mytest_data$RENEW)
# plotting ROC curve
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf,main = "ROC Curve",col = 2,lwd = 2)
abline(a = 0,b = 1,lwd = 2,lty = 3,col = "black")
```

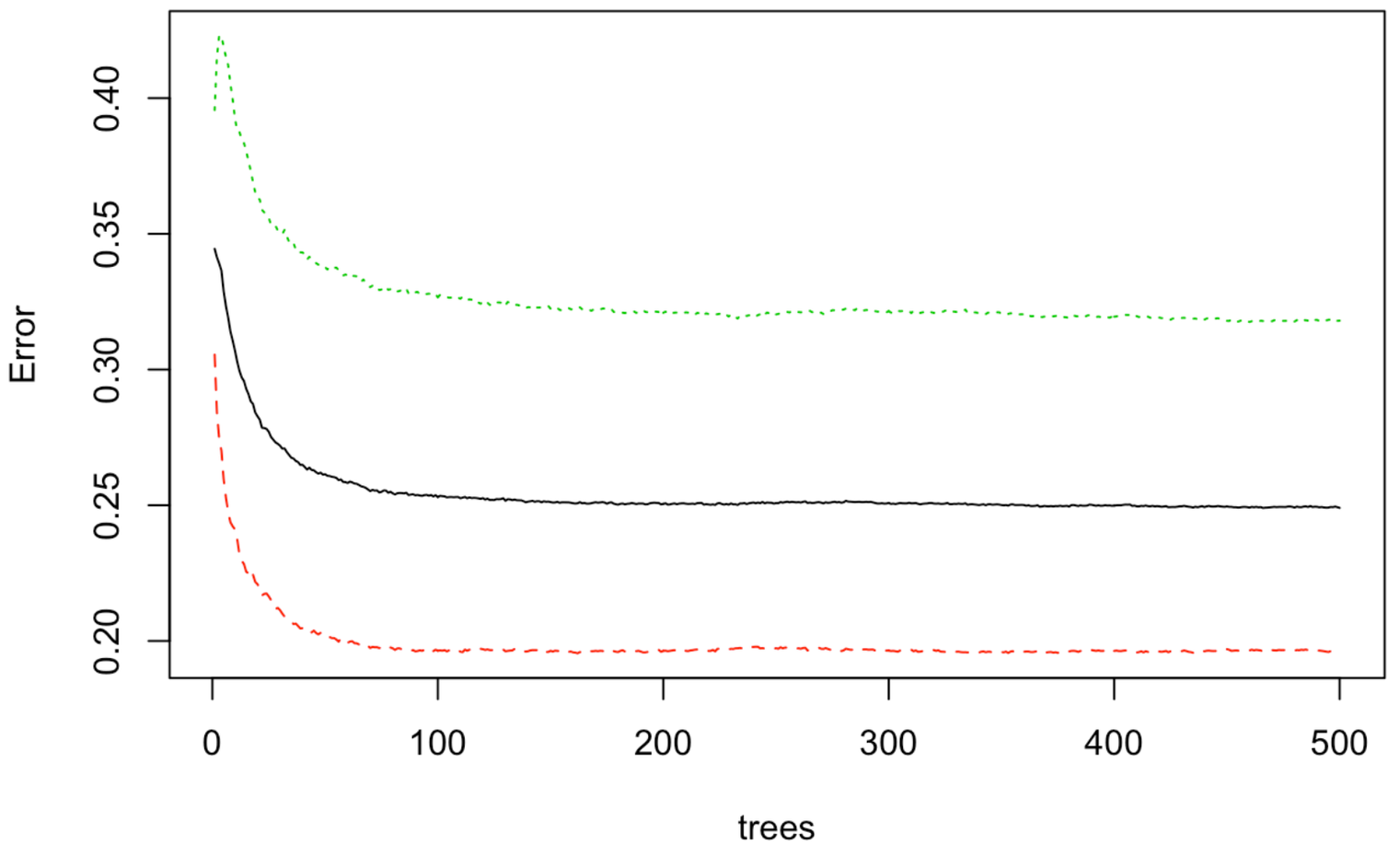



```
# AUC value
#AUC stands for "Area under the ROC Curve
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.7308922
```

```
#auc=0.7309
```

```
#Random Forest
#build the model
rfModel <- randomForest(RENEW ~., data=mytrain_data)

#We use this plot to help us determine the number of trees
plot(rfModel)
```



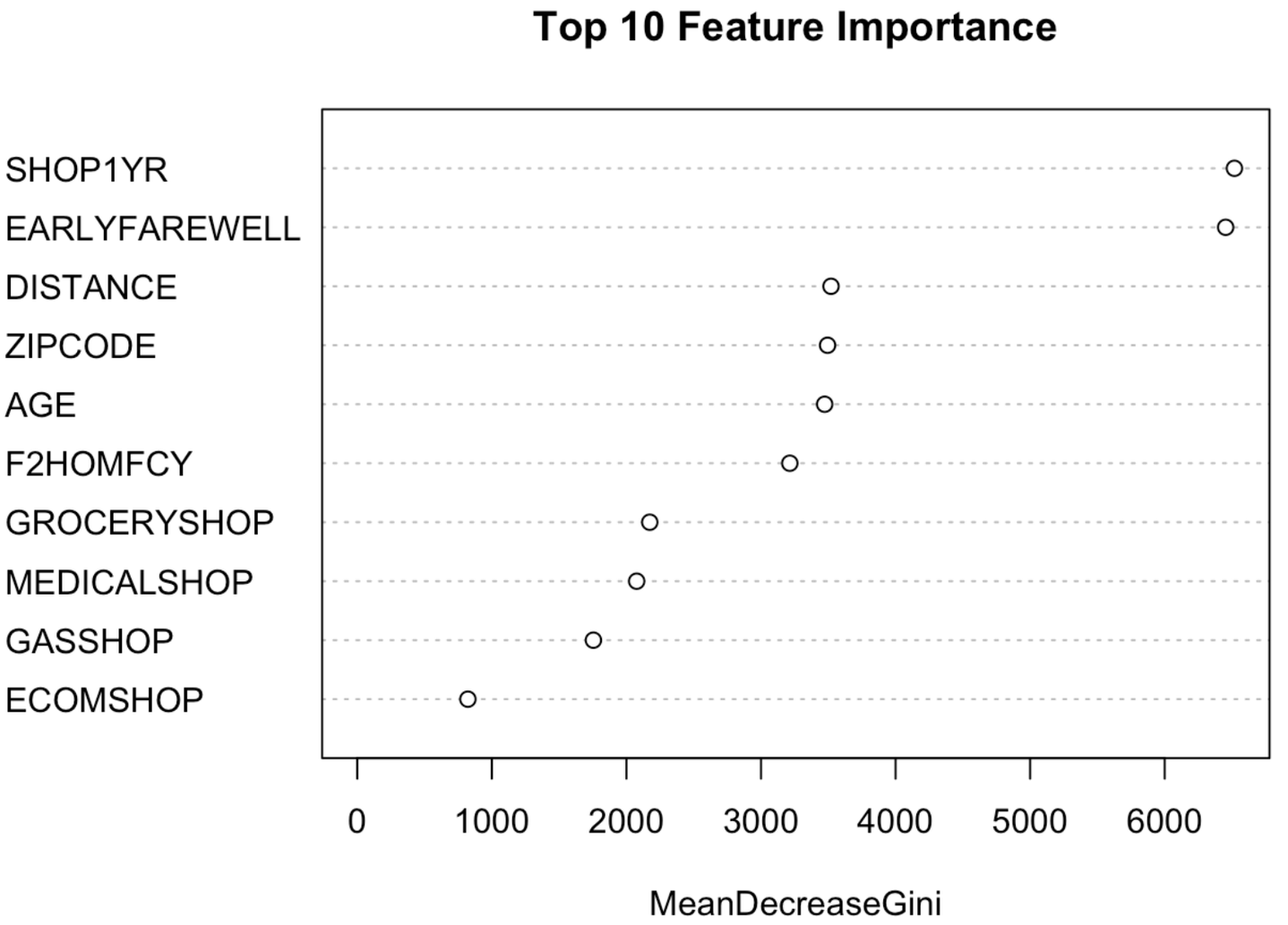

```
summary(rfModel)
```

```
##                 Length Class  Mode
## call                 3 -none- call
## type                 1 -none- character
## predicted        84139 factor numeric
## err.rate          1500 -none- numeric
## confusion            6 -none- numeric
## votes           168278 matrix numeric
## oob.times        84139 -none- numeric
## classes              2 -none- character
## importance          27 -none- numeric
## importanceSD         0 -none- NULL
## localImportance      0 -none- NULL
## proximity            0 -none- NULL
## ntree                1 -none- numeric
## mtry                 1 -none- numeric
## forest              14 -none- list
## y                84139 factor numeric
## test                 0 -none- NULL
## inbag                0 -none- NULL
## terms                3 terms  call
```

```
print(rfModel)
```

```
##
## Call:
##  randomForest(formula = RENEW ~ ., data = mytrain_data)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 24.9%
## Confusion matrix:
##       0     1 class.error
## 0 38240  9321   0.1959799
## 1 11632 24946   0.3180054
```

```
## to look at variable importance
varImpPlot(rfModel,sort=T, n.var = 10, main = 'Top 10 Feature Importance')
```

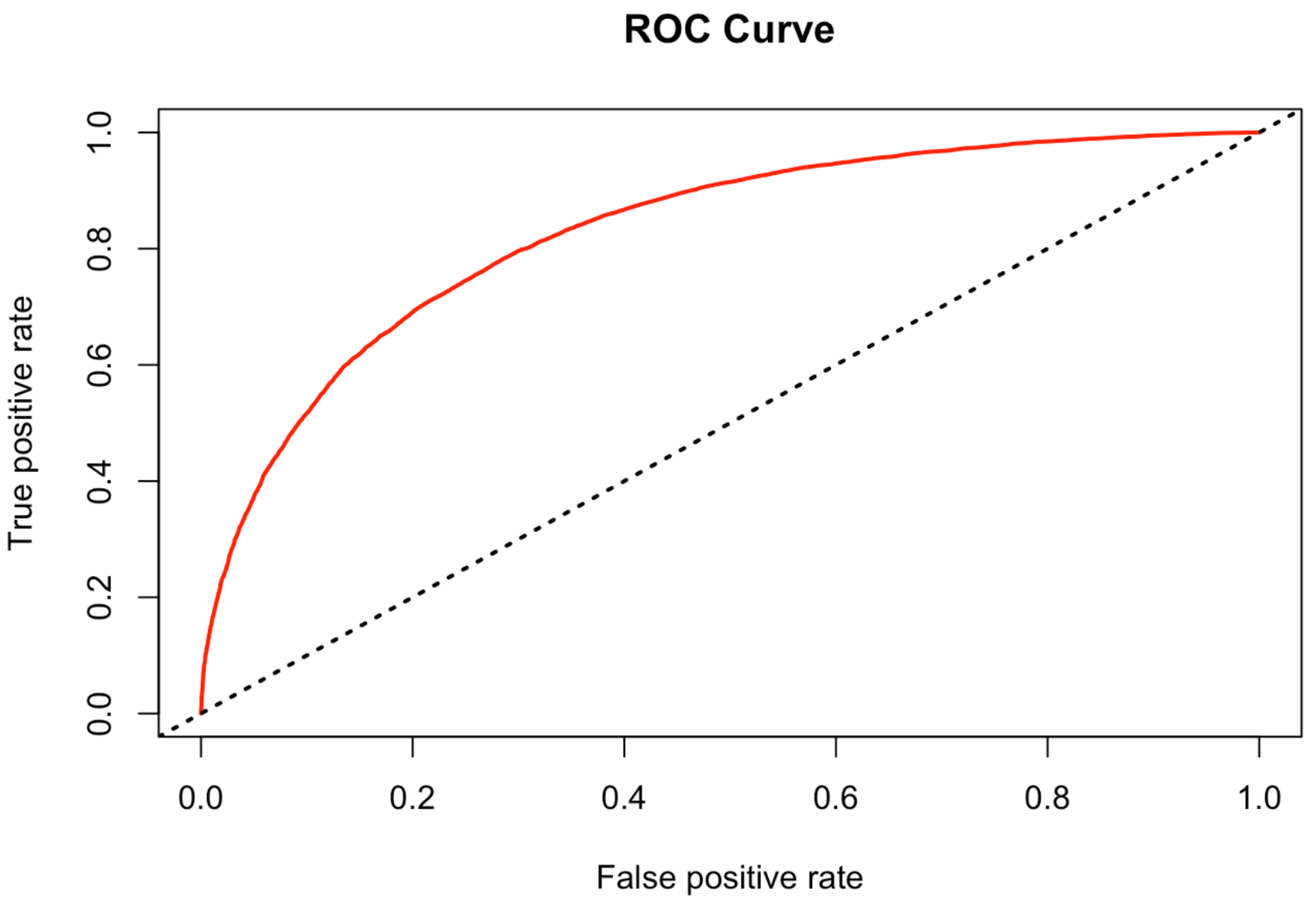



```
#prediction
pred_rf <- predict(rfModel, newdata = mytest_data)
pred_prob <- predict(rfModel, newdata = mytest_data,type="prob")

#confusion matrix for prediction
caret::confusionMatrix(pred_rf,mytest_data$RENEW)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 16334  4874
##          1  4048 10801
##
##                Accuracy : 0.7526
##                  95% CI : (0.7481, 0.757)
##     No Information Rate : 0.5653
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4935
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8014
##             Specificity : 0.6891
##          Pos Pred Value : 0.7702
##          Neg Pred Value : 0.7274
##              Prevalence : 0.5653
##          Detection Rate : 0.4530
##    Detection Prevalence : 0.5882
##       Balanced Accuracy : 0.7452
##
##        'Positive' Class : 0
##
```

```
#Accuracy = (True Negatives + True Positives)/ Total records
```

```
#look at the ROC curve and the AUC value
library(ROCR)
pr <- prediction(pred_prob[,2], mytest_data$RENEW)
# plotting ROC curve
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf,main = "ROC Curve",col = 2,lwd = 2)
abline(a = 0,b = 1,lwd = 2,lty = 3,col = "black")
```



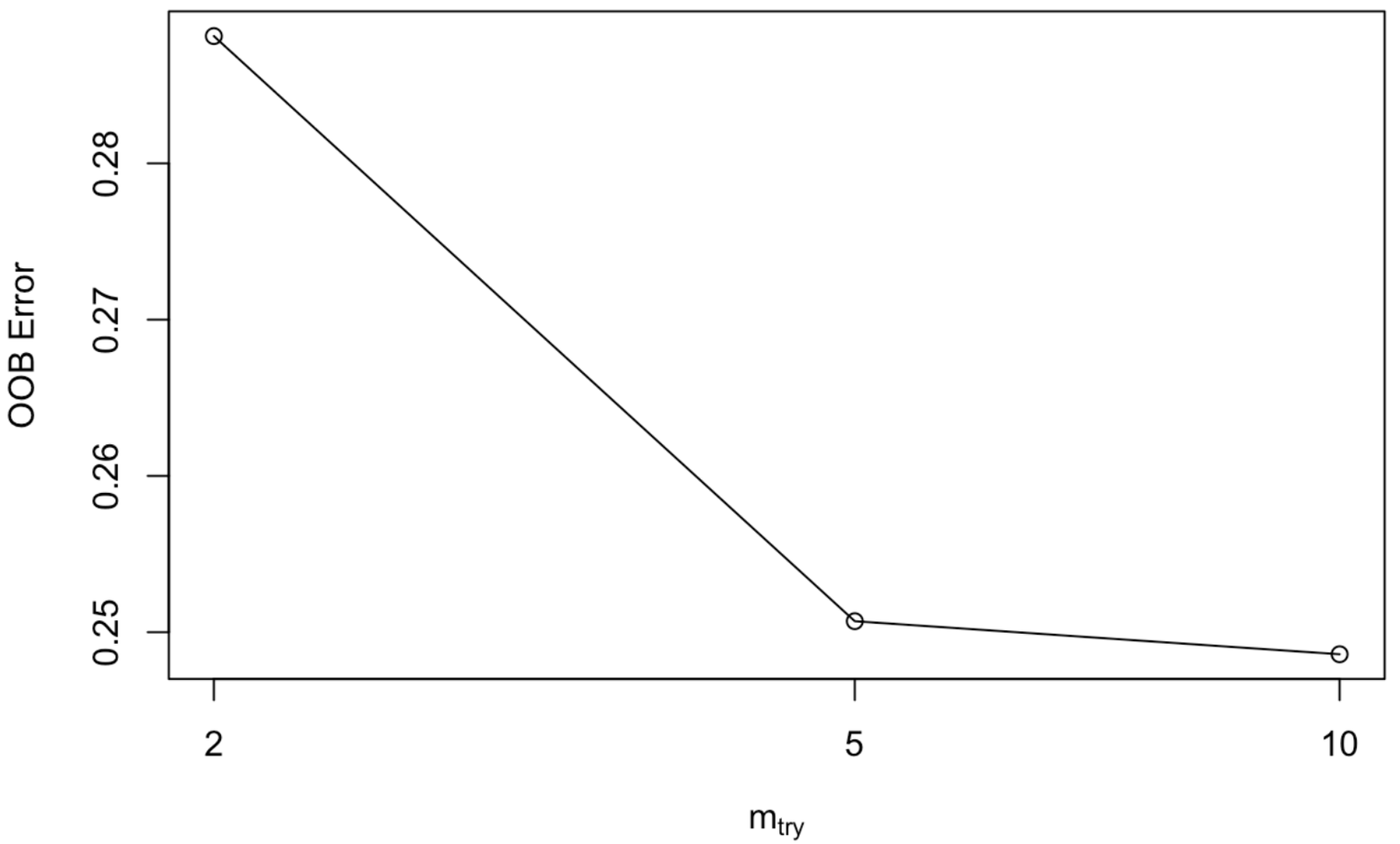

```
# AUC value
#AUC stands for "Area under the ROC Curve
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.8288224
```

```
#auc=0.8286
```

```
#tune the random forest model
#we tune the model by selecting the number of trees, conducting feature selection;
minimize the OOB error
a <- mytrain_data[, -1]
b <- mytrain_data$RENEW

t <- tuneRF(a, b, stepFactor = 0.5, plot = TRUE,
            ntreeTry =180, trace = TRUE, improve = 0.05)
```

```
## mtry = 5  OOB error = 25.07%
## Searching left ...
## mtry = 10    OOB error = 24.86%
## 0.008438419 0.05
## Searching right ...
## mtry = 2     OOB error = 28.81%
## -0.1493316 0.05
```

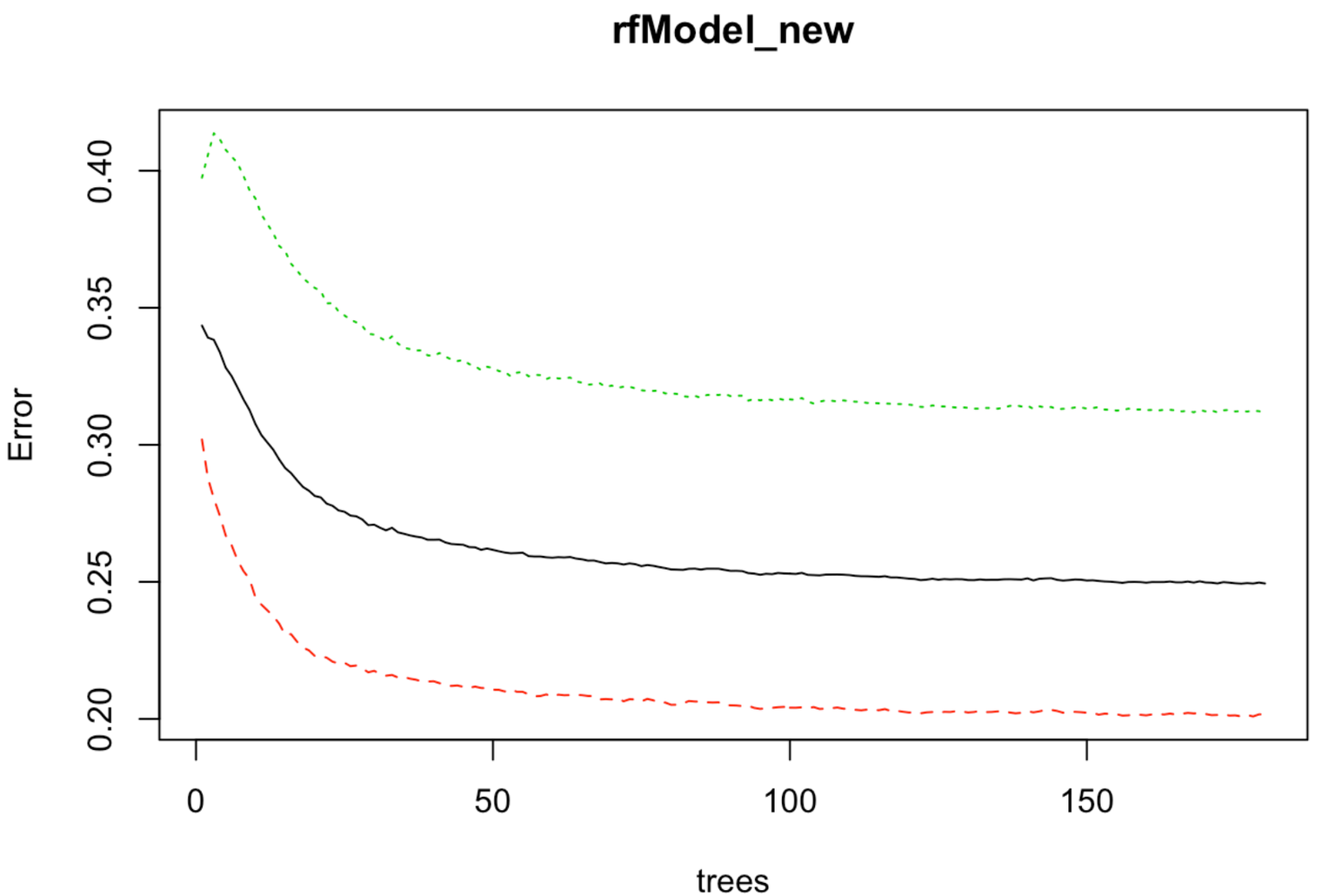



```
##m(try)=10 when tree=100 #accuracy=0.7511 OOB error = 25.3%
##m(try)=10 when tree=150 #accuracy=0.7521 OOB error = 25.1%
##m(try)=10 when tree=180 #accuracy=0.7525 OOB error = 24.9%
##m(try)=10 when tree=185 #accuracy=0.7523 OOB error = 24.9%
##m(try)=10 when tree=200 #accuracy=0.7519 OOB error = 24.9%
```

```
#run the Random Forest model after tuning
set.seed(100)
rfModel_new <- randomForest(RENEW ~., data=mytrain_data, ntree = 180,
                            mtry = 10, importance = TRUE)
print(rfModel_new)
```

```
##
## Call:
##  randomForest(formula = RENEW ~ ., data = mytrain_data, ntree = 180,      mtry
= 10, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 180
## No. of variables tried at each split: 10
##
##         OOB estimate of  error rate: 24.94%
## Confusion matrix:
##       0     1 class.error
## 0 37970  9591   0.2016568
## 1 11395 25183   0.3115261
```

```
plot(rfModel_new)
```



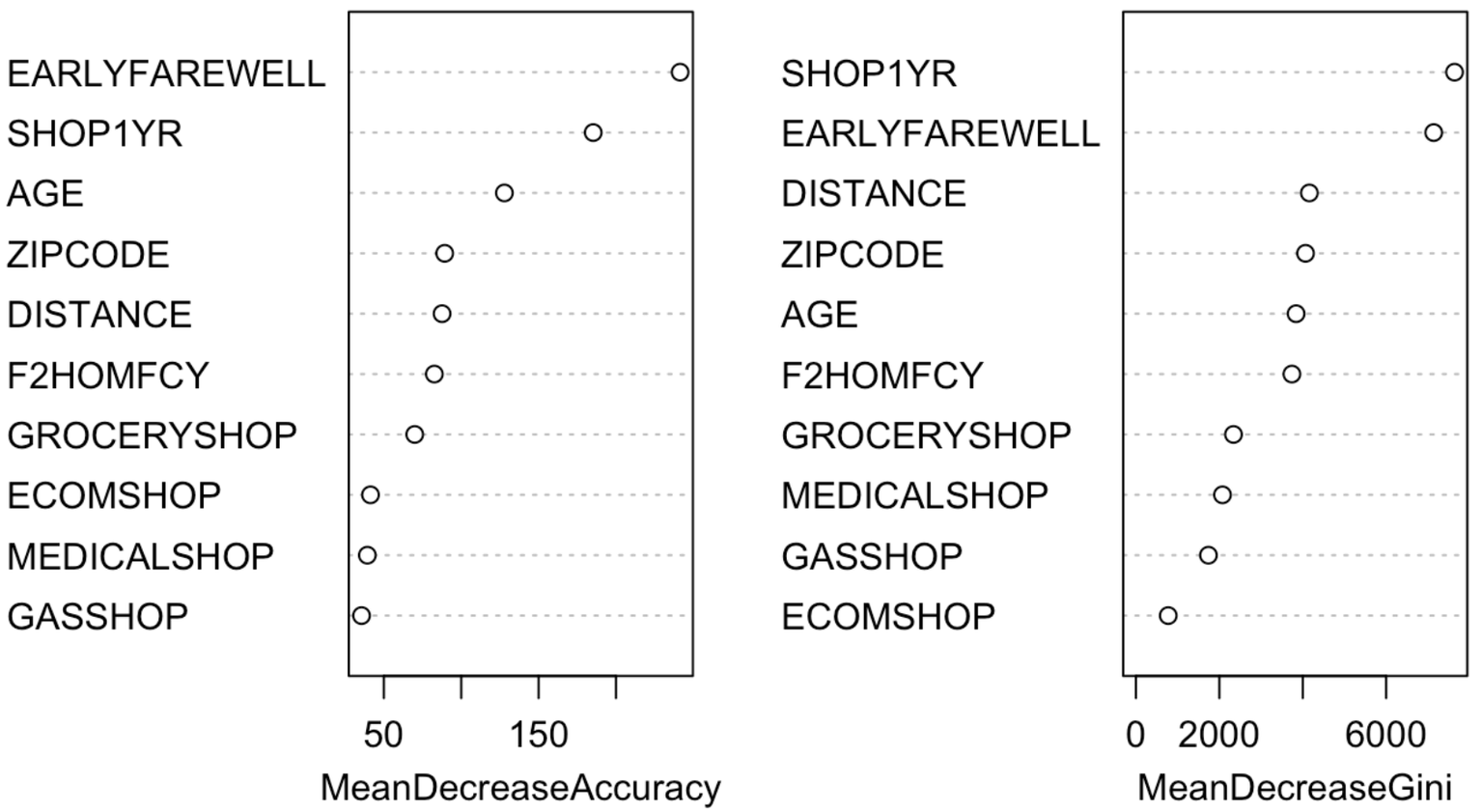

```
summary(rfModel_new)
```

```
##                 Length Class  Mode
## call                 6 -none- call
## type                 1 -none- character
## predicted        84139 factor numeric
## err.rate           540 -none- numeric
## confusion            6 -none- numeric
## votes           168278 matrix numeric
## oob.times        84139 -none- numeric
## classes              2 -none- character
## importance         108 -none- numeric
## importanceSD        81 -none- numeric
## localImportance      0 -none- NULL
## proximity            0 -none- NULL
## ntree                1 -none- numeric
## mtry                 1 -none- numeric
## forest              14 -none- list
## y                84139 factor numeric
## test                 0 -none- NULL
## inbag                0 -none- NULL
## terms                3 terms  call
```

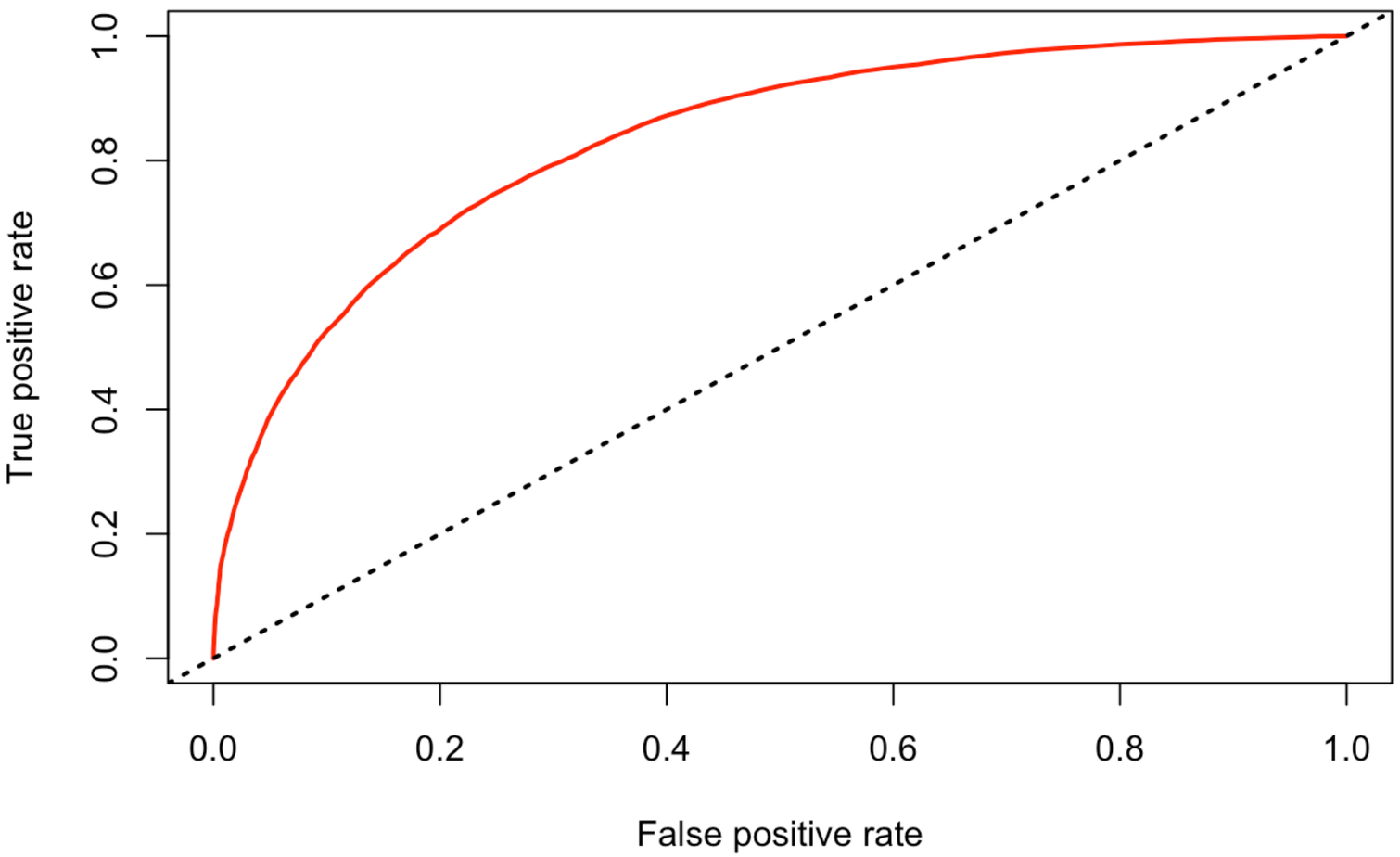
```
#prediction
pred_rf <- predict(rfModel_new, newdata = mytest_data)
pred_prob <- predict(rfModel_new, newdata = mytest_data,type="prob")

#confusion matrix for prediction
caret::confusionMatrix(pred_rf,mytest_data$RENEW)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 16218  4761
##          1  4164 10914
##
##                Accuracy : 0.7525
##                  95% CI : (0.748, 0.7569)
##     No Information Rate : 0.5653
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4941
##
##  Mcnemar's Test P-Value : 2.813e-10
##
##             Sensitivity : 0.7957
##             Specificity : 0.6963
##          Pos Pred Value : 0.7731
##          Neg Pred Value : 0.7238
##              Prevalence : 0.5653
##          Detection Rate : 0.4498
##    Detection Prevalence : 0.5818
##       Balanced Accuracy : 0.7460
##
##        'Positive' Class : 0
##
```

```
varImpPlot(rfModel_new, sort=T, n.var = 10, main = 'Top 10 Feature Importance')
```




```
#look at the ROC curve and the AUC value
library(ROCR)
pr <- prediction(pred_prob[,2], mytest_data$RENEW)
# plotting ROC curve
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf,main = "ROC Curve",col = 2,lwd = 2)
abline(a = 0,b = 1,lwd = 2,lty = 3,col = "black")
```




```
# AUC value
#AUC stands for "Area under the ROC Curve
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.8321789
```

```
#auc=0.8321
```

```
#KNN
#Based on the DT importance, we use that sub-dataset of 10 features for KNN
newsubset <- mynew05[,c("RENEW","EARLYFAREWELL","SHOP1YR","DISTANCE","GASSHOP","ME
DICALSHOP","AGE","GROCERYSHOP",
"ECOMSHOP","M2EXCFLGE","M2EXCFLGN")]
str(newsubset)
```

```
## 'data.frame':    120196 obs. of  11 variables:
##  $ RENEW        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ EARLYFAREWELL: num  75 320 350 137 41 53 38 363 53 64 ...
##  $ SHOP1YR      : num  1385 3500 114 997 12579 ...
##  $ DISTANCE     : num  7.53 6.05 7.89 3.43 26.29 ...
##  $ GASSHOP      : num  0.03 0 0 0.03 0 0 0 0 0.43 0.17 ...
##  $ MEDICALSHOP  : num  0.02 0 0.3 0 0.01 0 0 0 0.02 0 ...
##  $ AGE          : num  42 61 52 32 46 36 34 45 52 32 ...
##  $ GROCERYSHOP  : num  0.52 0 0.23 0.41 0.94 0.84 0.51 0.93 0.26 0.17 ...
##  $ ECOMSHOP     : num  0 1 0 0 0 0 0 0 0 0 ...
##  $ M2EXCFLGE    : num  0 0 0 0 1 1 0 0 0 1 ...
##  $ M2EXCFLGN    : num  1 1 1 1 0 0 1 1 1 0 ...
```

```
#load and partition the dataset: training (70%) and validation (30%) sets

set.seed(105)
indexknn<- sample(1:nrow(newsubset),size=nrow(newsubset)*0.7,replace = FALSE)
train_knn<- newsubset[indexknn,] # 70% training data
test_knn<- newsubset[-indexknn,]

#create the separate dataframe
train_knn_pl<- newsubset[indexknn,1]

# initialize normalized training, validation data, complete data frames to origina
ls
train.norm.df <- train_knn
valid.norm.df <- test_knn

# use preProcess() from the caret package to normalize features
norm.values <- preProcess(train_knn[, -1], method=c("center", "scale"))

train.norm.df[, -1] <- predict(norm.values, train_knn[, -1])
valid.norm.df[, -1] <- predict(norm.values, test_knn[, -1])
```

```
#KNN
#compute knn for different k on validation to find the best k
# initialize a data frame with two columns: k, and accuracy
library(class)
```

```
##
## Attaching package: 'class'
```

```
## The following objects are masked from 'package:FNN':
##
##     knn, knn.cv
```

```
set.seed(105)
cl <- train_knn_pl

i=1
k.optm=1
for (i in 30:60){
    knn.mod <-  knn(train=train.norm.df[,-1], test=valid.norm.df[, -1], cl, k=i)
    k.optm[i] <- 100 * sum(knn.mod == test_knn$RENEW)/NROW(test_knn$RENEW)
    k=i
    cat(k,'=',k.optm[i],'\n')
}
```

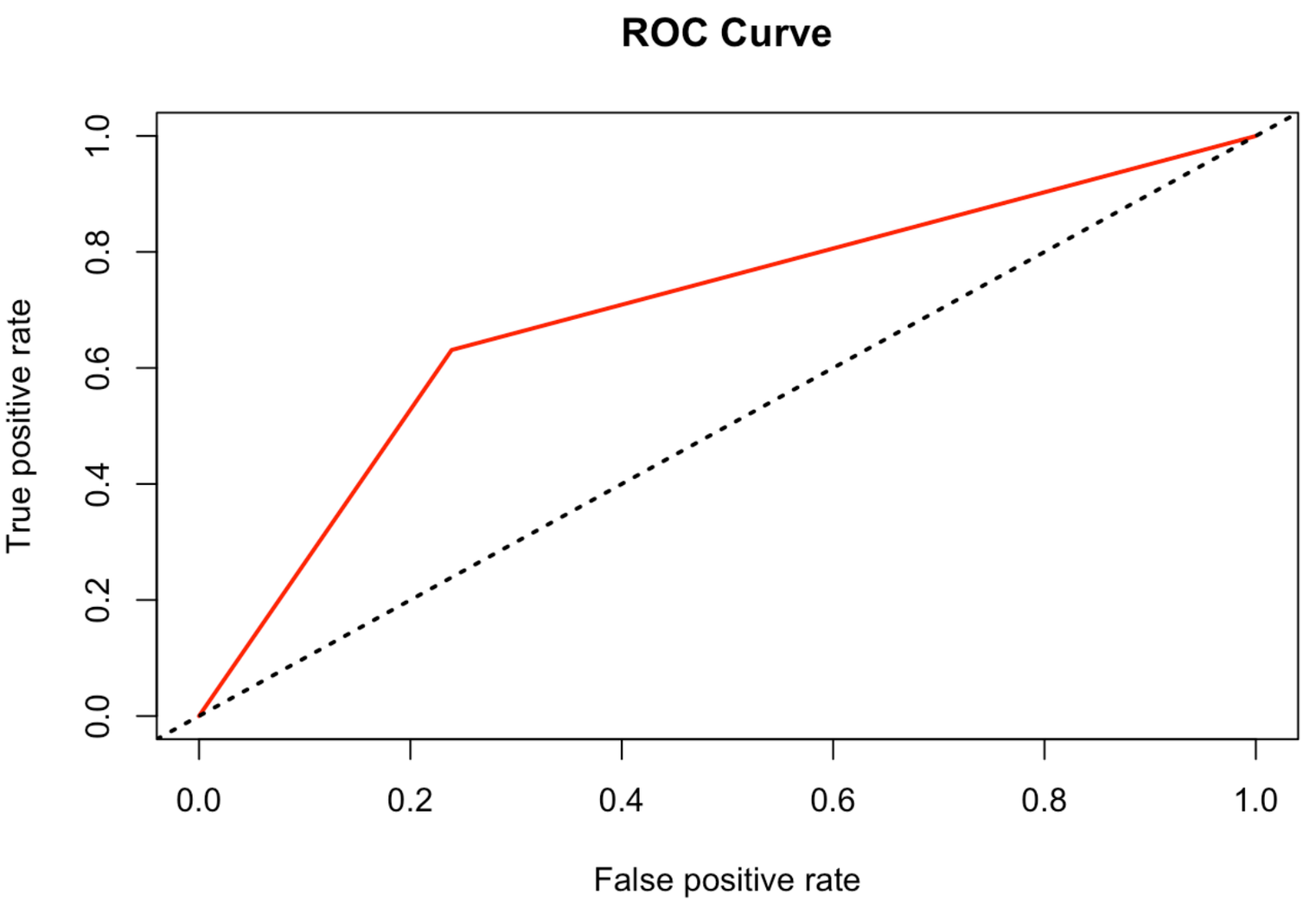
```
## 30 = 69.8716
## 31 = 69.94925
## 32 = 70.01581
## 33 = 70.00471
## 34 = 69.96312
## 35 = 70.1101
## 36 = 70.05463
## 37 = 70.14615
## 38 = 70.07959
## 39 = 70.19884
## 40 = 70.09068
## 41 = 70.18775
## 42 = 70.07682
## 43 = 70.15724
## 44 = 70.23212
## 45 = 70.18497
## 46 = 70.13506
## 47 = 70.24321
## 48 = 70.24044
## 49 = 70.30145
## 50 = 70.35969
## 51 = 70.25431
## 52 = 70.2654
## 53 = 70.38465
## 54 = 70.35692
## 55 = 70.37633
## 56 = 70.27372
## 57 = 70.33473
## 58 = 70.36246
## 59 = 70.2654
## 60 = 70.27094
```

```
library(FNN)
set.seed(105)
cl <- train_knn_pl
#the best k=53, with the highest accuracy
knn.53 <- knn(train=train.norm.df[,-1], test=valid.norm.df[, -1], cl, k=53,prob=TR
UE)

#show the confusion matrix for the validation data
library(caret)
caret::confusionMatrix(knn.53,valid.norm.df$RENEW)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 15375  5849
##          1  4831 10004
##
##                Accuracy : 0.7038
##                  95% CI : (0.6991, 0.7085)
##     No Information Rate : 0.5604
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3947
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7609
##             Specificity : 0.6310
##          Pos Pred Value : 0.7244
##          Neg Pred Value : 0.6744
##              Prevalence : 0.5604
##          Detection Rate : 0.4264
##    Detection Prevalence : 0.5886
##       Balanced Accuracy : 0.6960
##
##        'Positive' Class : 0
##
```

```
#accuracy=0.7039
```

```
#look at the ROC curve and the AUC value
library(ROCR)
pr <- prediction(as.numeric(knn.53), valid.norm.df$RENEW)
# plotting ROC curve
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf,main = "ROC Curve",col = 2,lwd = 2)
abline(a = 0,b = 1,lwd = 2,lty = 3,col = "black")
```




```
# AUC value
#AUC stands for "Area under the ROC Curve
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.6959802
```

```
#auc=0.6960
```

```
#KNN
#Based on the stepwise selection, we use that sub-dataset for KNN
newsubset01 <- mynew05[,c("RENEW","F2HOMFCY", "AGE", "MBRCOUNT","DISTANCE","EARLYF
AREWELL",
    "SHOP1YR","ECOMSHOP","GASSHOP", "MEDICALSHOP","GROCERYSHOP",
    "M2EXCFLGE", "HOMEFCTYCHANGEN","RECENTMOVINGN","F2HOMRGN_BOFALSE",
    "F2HOMRGN_TEFALSE","F2HOMRGN_middleFALSE")]
str(newsubset01)
```

```
## 'data.frame':    120196 obs. of  17 variables:
##  $ RENEW               : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ F2HOMFCY            : num  1078 847 847 185 472 ...
##  $ AGE                 : num  42 61 52 32 46 36 34 45 52 32 ...
##  $ MBRCOUNT            : num  2 2 2 2 2 1 2 2 2 2 ...
##  $ DISTANCE            : num  7.53 6.05 7.89 3.43 26.29 ...
##  $ EARLYFAREWELL       : num  75 320 350 137 41 53 38 363 53 64 ...
##  $ SHOP1YR             : num  1385 3500 114 997 12579 ...
##  $ ECOMSHOP            : num  0 1 0 0 0 0 0 0 0 0 ...
##  $ GASSHOP             : num  0.03 0 0 0.03 0 0 0 0 0.43 0.17 ...
##  $ MEDICALSHOP         : num  0.02 0 0.3 0 0.01 0 0 0 0.02 0 ...
##  $ GROCERYSHOP         : num  0.52 0 0.23 0.41 0.94 0.84 0.51 0.93 0.26 0.17 ..
.
##  $ M2EXCFLGE           : num  0 0 0 0 1 1 0 0 0 1 ...
##  $ HOMEFCTYCHANGEN     : num  0 1 1 1 1 1 1 1 1 0 ...
##  $ RECENTMOVINGN       : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ F2HOMRGN_BOFALSE    : num  1 0 0 1 1 1 1 1 1 1 ...
##  $ F2HOMRGN_TEFALSE    : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ F2HOMRGN_middleFALSE: num  0 1 1 0 1 0 0 1 0 1 ...
```

```
#load and partition the dataset: training (70%) and validation (30%) sets

set.seed(105)
indexknn<- sample(1:nrow(newsubset01),size=nrow(newsubset01)*0.7,replace = FALSE)
train_knn<- newsubset01[indexknn,] # 70% training data
test_knn<- newsubset01[-indexknn,]

#create the separate dataframe
train_knn_pl<- newsubset01[indexknn,1]

# initialize normalized training, validation data, complete data frames to origina
ls
train.norm.df <- train_knn
valid.norm.df <- test_knn

# use preProcess() from the caret package to normalize features
norm.values <- preProcess(train_knn[, -1], method=c("center", "scale"))

train.norm.df[, -1] <- predict(norm.values, train_knn[, -1])
valid.norm.df[, -1] <- predict(norm.values, test_knn[, -1])
```

```
#KNN
#compute knn for different k on validation to find the best k
# initialize a data frame with two columns: k, and accuracy
library(class)
set.seed(105)
cl <- train_knn_pl

i=1
k.optm=1
for (i in 30:60){
    knn.mod <-  knn(train=train.norm.df[,-1], test=valid.norm.df[, -1], cl, k=i)
    k.optm[i] <- 100 * sum(knn.mod == test_knn$RENEW)/NROW(test_knn$RENEW)
    k=i
    cat(k,'=',k.optm[i],'\n')
}
```

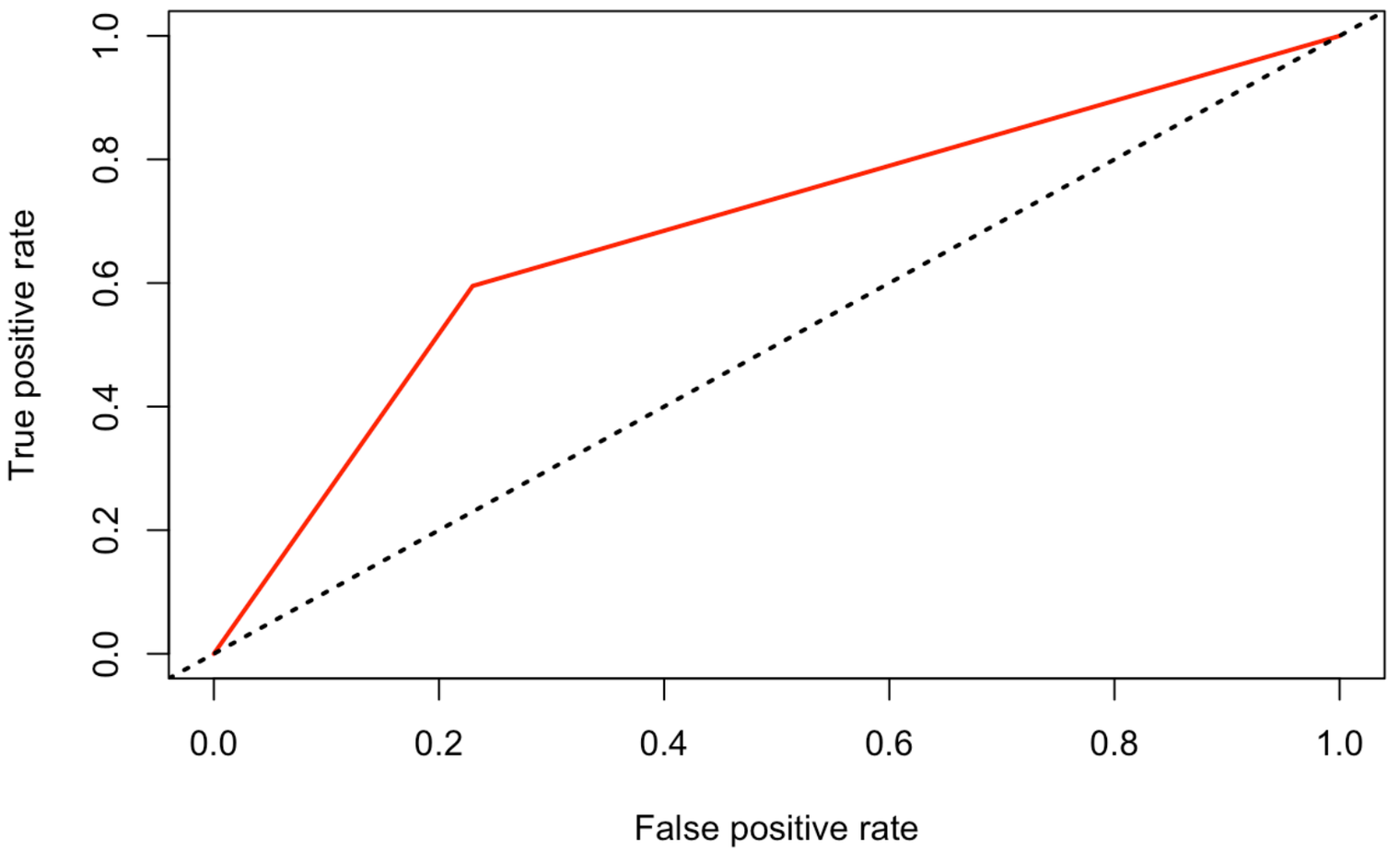
```
## 30 = 68.63196
## 31 = 68.86214
## 32 = 68.93147
## 33 = 68.83996
## 34 = 68.96753
## 35 = 68.99803
## 36 = 68.91206
## 37 = 68.98139
## 38 = 68.97585
## 39 = 68.98971
## 40 = 69.07291
## 41 = 69.0535
## 42 = 69.02299
## 43 = 69.09232
## 44 = 69.14779
## 45 = 69.06182
## 46 = 69.21157
## 47 = 69.084
## 48 = 69.1256
## 49 = 69.15056
## 50 = 69.19493
## 51 = 69.22544
## 52 = 69.06459
## 53 = 69.22821
## 54 = 69.22821
## 55 = 69.31695
## 56 = 69.21434
## 57 = 69.16997
## 58 = 69.11173
## 59 = 69.15888
## 60 = 69.08677
```

```
library(FNN)
set.seed(105)
cl <- train_knn_pl
#the best k=53, with the highest accuracy
knn.55 <- knn(train=train.norm.df[,-1], test=valid.norm.df[, -1], cl, k=55)
#show the confusion matrix for the validation data
library(caret)
caret::confusionMatrix(knn.55,valid.norm.df$RENEW)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 15561  6417
##          1  4645  9436
##
##                Accuracy : 0.6932
##                  95% CI : (0.6884, 0.698)
##     No Information Rate : 0.5604
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3698
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7701
##             Specificity : 0.5952
##          Pos Pred Value : 0.7080
##          Neg Pred Value : 0.6701
##              Prevalence : 0.5604
##          Detection Rate : 0.4315
##    Detection Prevalence : 0.6095
##       Balanced Accuracy : 0.6827
##
##        'Positive' Class : 0
##
```

```
#accuracy=0.6932
```

```
#look at the ROC curve and the AUC value
library(ROCR)
pr <- prediction(as.numeric(knn.55), valid.norm.df$RENEW)
# plotting ROC curve
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf,main = "ROC Curve",col = 2,lwd = 2)
abline(a = 0,b = 1,lwd = 2,lty = 3,col = "black")
```




```
# AUC value
#AUC stands for "Area under the ROC Curve
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.6826682
```

```
#auc=0.6826
```

```
#SVM
##build model: radial kernel, default params
##Non-linear boundary
library(e1071)
set.seed(105)
svm_model <- svm(RENEW ~., data=mytrain_data, method="C-classification", kernel="r
adial")
```

```
#print params
svm_model$cost
```

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

```
svm_model$gamma
```

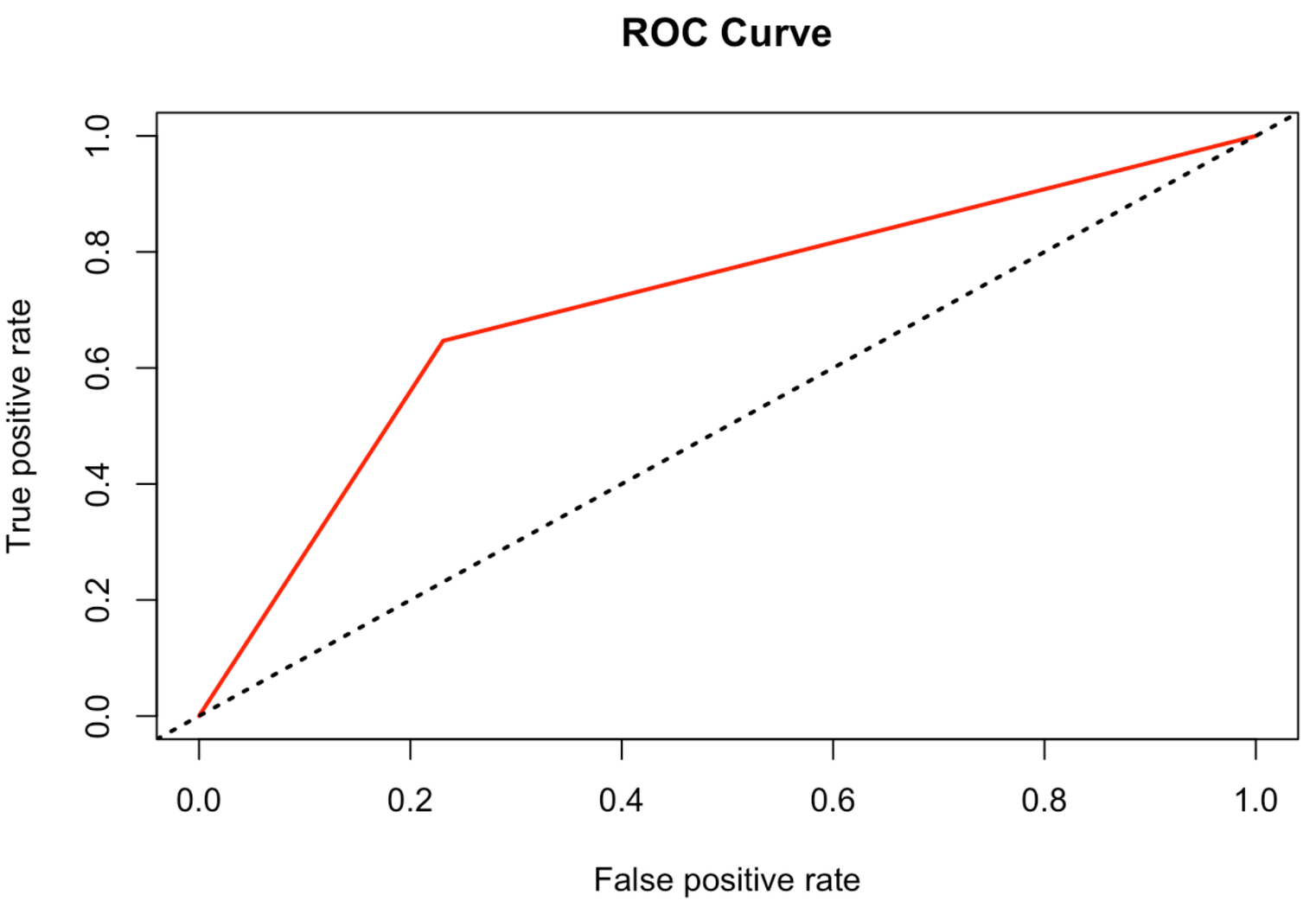
```
## [1] 0.03703704
```

```
# prediction
pred_test <-predict(svm_model,newdata = mytest_data)
caret::confusionMatrix(pred_test, mytest_data$RENEW)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 15674  5537
##          1  4708 10138
##
##                Accuracy : 0.7159
##                  95% CI : (0.7112, 0.7205)
##     No Information Rate : 0.5653
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4183
##
##  Mcnemar's Test P-Value : 2.829e-16
##
##             Sensitivity : 0.7690
##             Specificity : 0.6468
##          Pos Pred Value : 0.7390
##          Neg Pred Value : 0.6829
##              Prevalence : 0.5653
##          Detection Rate : 0.4347
##    Detection Prevalence : 0.5883
##       Balanced Accuracy : 0.7079
##
##        'Positive' Class : 0
##
```

```
#accuracy= 0.7159
```

```
#look at the ROC curve and the AUC value
library(ROCR)
pr <- prediction(as.numeric(pred_test), mytest_data$RENEW)
# plotting ROC curve
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf,main = "ROC Curve",col = 2,lwd = 2)
abline(a = 0,b = 1,lwd = 2,lty = 3,col = "black")
```




```
# AUC value
#AUC stands for "Area under the ROC Curve
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.7078871
```

```
#auc=0.7078
```

```
#Based on the DT importance, we use that sub-dataset of 10 features for RF
 #accuracy=0.7511
```

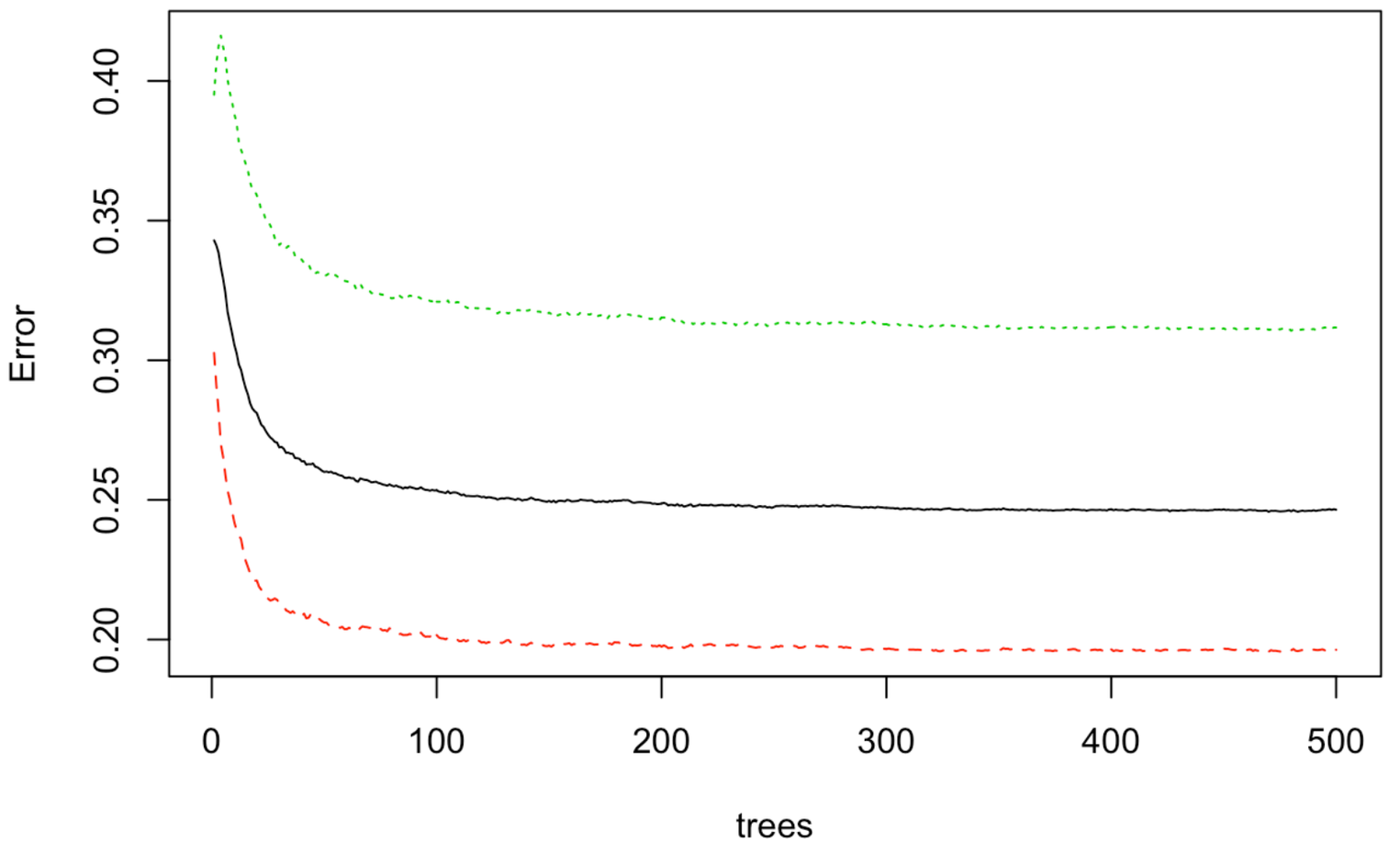
```
#Based on the stepwise selection result, we use that sub-dataset of 10 features fo
r RF
#set up the traing and testing dataset
set.seed(500)
index <- createDataPartition(newsubset01$RENEW, p = 0.7, list = FALSE)
mytrain_data <- newsubset01[index, ]
mytest_data  <- newsubset01[-index, ]

table(mytrain_data$RENEW)
```

```
##
##     0     1
## 47561 36578
```

```
table(mytest_data$RENEW)
```

```
##
##     0     1
## 20382 15675
```

```
#build the model
rfModel <- randomForest(RENEW ~., data=mytrain_data)

#We use this plot to help us determine the number of trees
plot(rfModel)
```



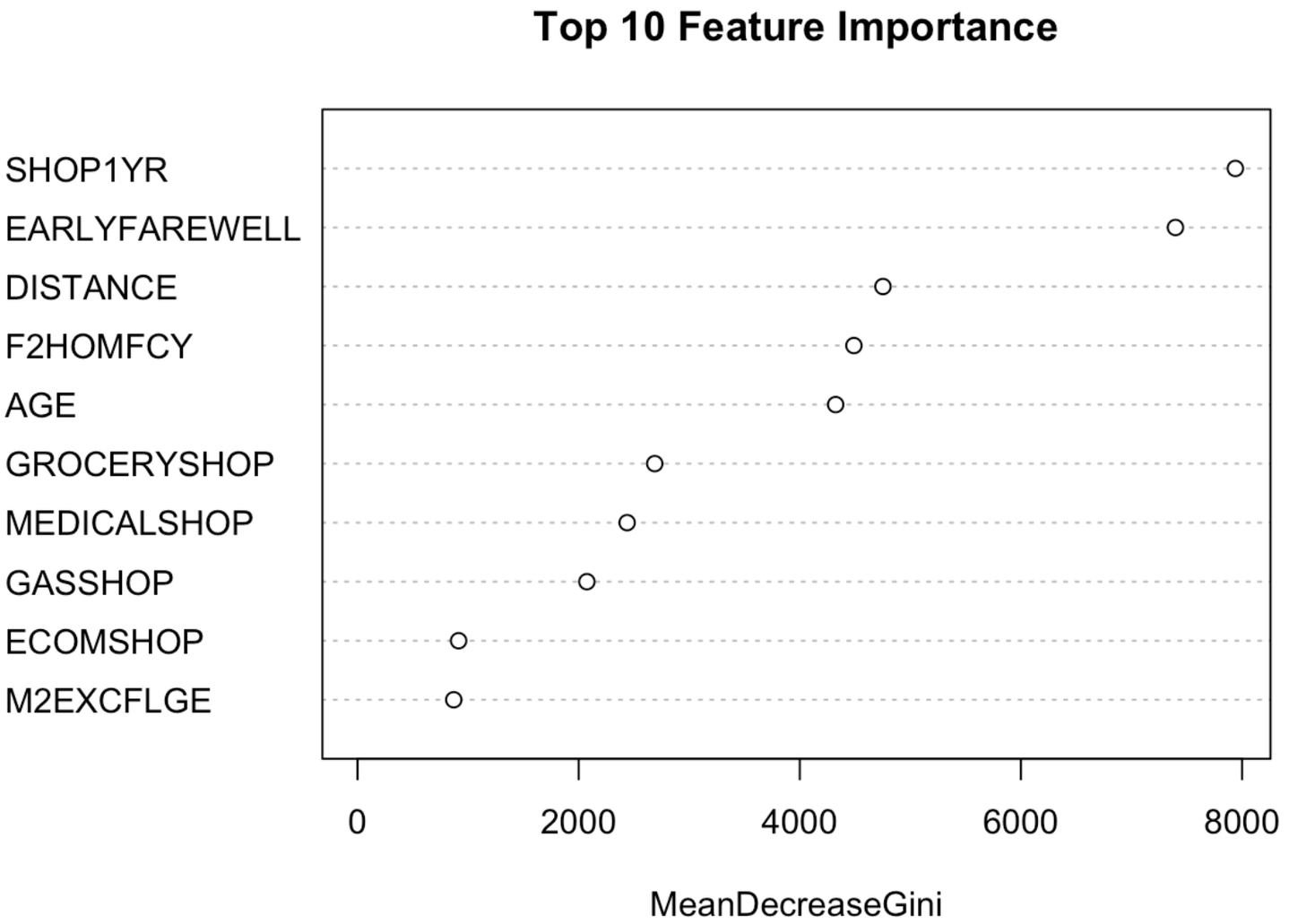

```
summary(rfModel)
```

```
##                 Length Class  Mode
## call                 3 -none- call
## type                 1 -none- character
## predicted        84139 factor numeric
## err.rate          1500 -none- numeric
## confusion            6 -none- numeric
## votes           168278 matrix numeric
## oob.times        84139 -none- numeric
## classes              2 -none- character
## importance          16 -none- numeric
## importanceSD         0 -none- NULL
## localImportance      0 -none- NULL
## proximity            0 -none- NULL
## ntree                1 -none- numeric
## mtry                 1 -none- numeric
## forest              14 -none- list
## y                84139 factor numeric
## test                 0 -none- NULL
## inbag                0 -none- NULL
## terms                3 terms  call
```

```
print(rfModel)
```

```
##
## Call:
##  randomForest(formula = RENEW ~ ., data = mytrain_data)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 24.65%
## Confusion matrix:
##       0     1 class.error
## 0 38224  9337   0.1963163
## 1 11402 25176   0.3117174
```

```
## to look at variable importance
varImpPlot(rfModel,sort=T, n.var = 10, main = 'Top 10 Feature Importance')
```

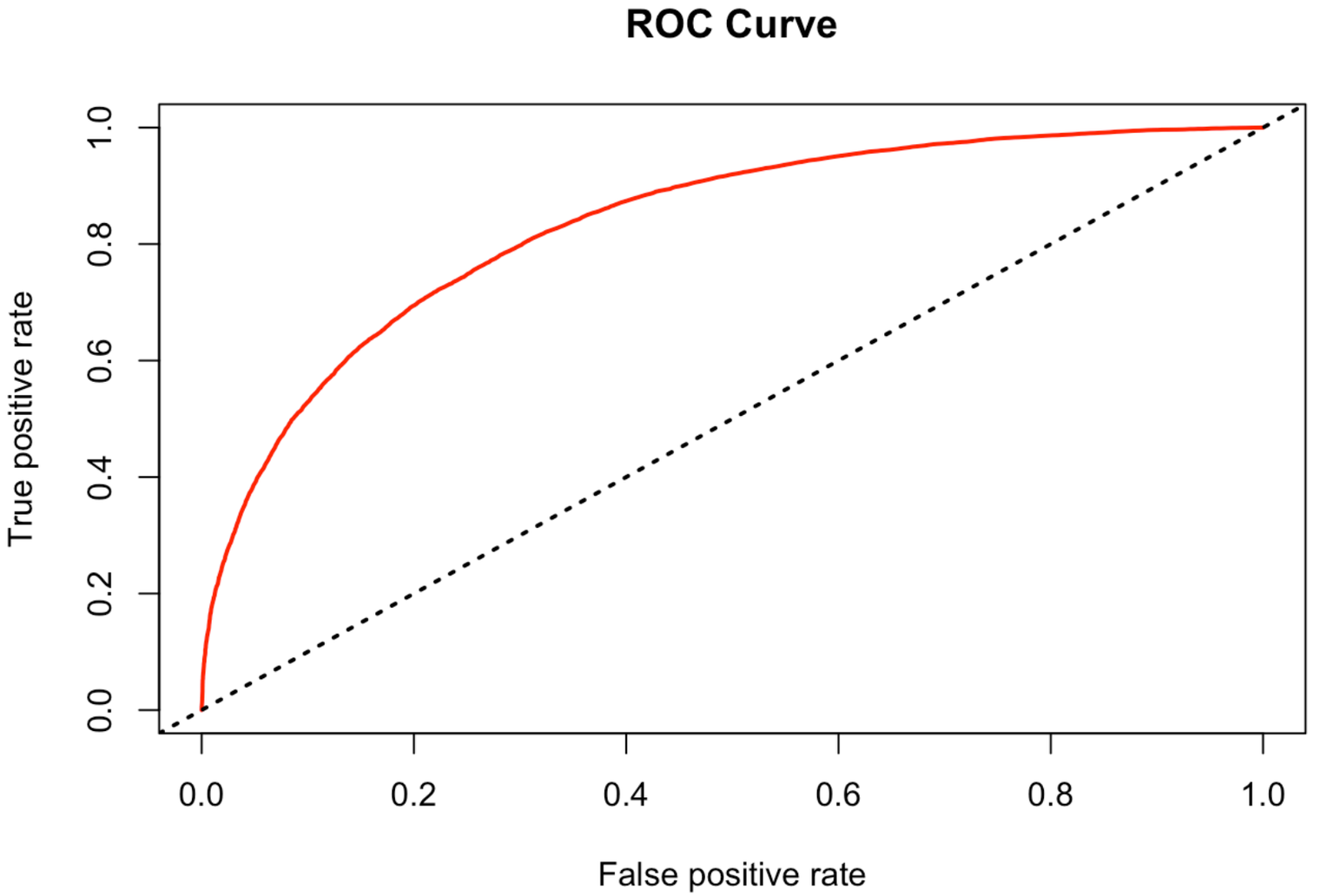



```
#prediction
pred_rf <- predict(rfModel, newdata = mytest_data)
pred_prob <- predict(rfModel, newdata = mytest_data,type="prob")

#confusion matrix for prediction
caret::confusionMatrix(pred_rf,mytest_data$RENEW)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 16323  4798
##          1  4059 10877
##
##                Accuracy : 0.7544
##                  95% CI : (0.7499, 0.7588)
##     No Information Rate : 0.5653
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4975
##
##  Mcnemar's Test P-Value : 4.443e-15
##
##             Sensitivity : 0.8009
##             Specificity : 0.6939
##          Pos Pred Value : 0.7728
##          Neg Pred Value : 0.7282
##              Prevalence : 0.5653
##          Detection Rate : 0.4527
##    Detection Prevalence : 0.5858
##       Balanced Accuracy : 0.7474
##
##        'Positive' Class : 0
##
```

```
#accuracy=0.7544
```

```
#look at the ROC curve and the AUC value
library(ROCR)
pr <- prediction(pred_prob[,2], mytest_data$RENEW)
# plotting ROC curve
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf,main = "ROC Curve",col = 2,lwd = 2)
abline(a = 0,b = 1,lwd = 2,lty = 3,col = "black")
```



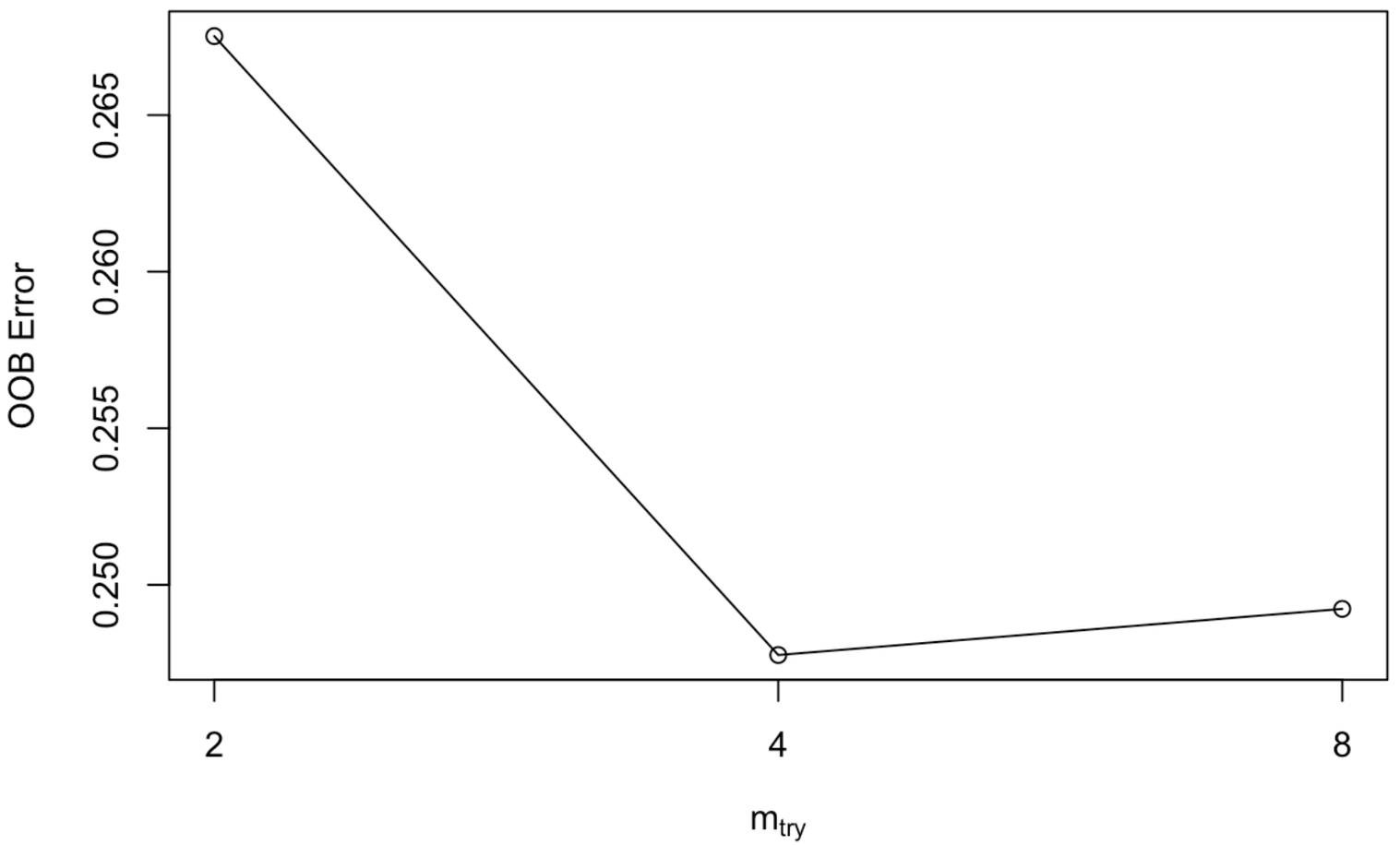

```
# AUC value
#AUC stands for "Area under the ROC Curve
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.8336141
```

```
#auc=0.8336
```

```
#tune the model
a <- mytrain_data[, -1]
b <- mytrain_data$RENEW

t <- tuneRF(a, b, stepFactor = 0.5, plot = TRUE,
            ntreeTry =300, trace = TRUE, improve = 0.05)
```

```
## mtry = 4  OOB error = 24.78%
## Searching left ...
## mtry = 8     OOB error = 24.92%
## -0.005948383 0.05
## Searching right ...
## mtry = 2     OOB error = 26.75%
## -0.0797755 0.05
```

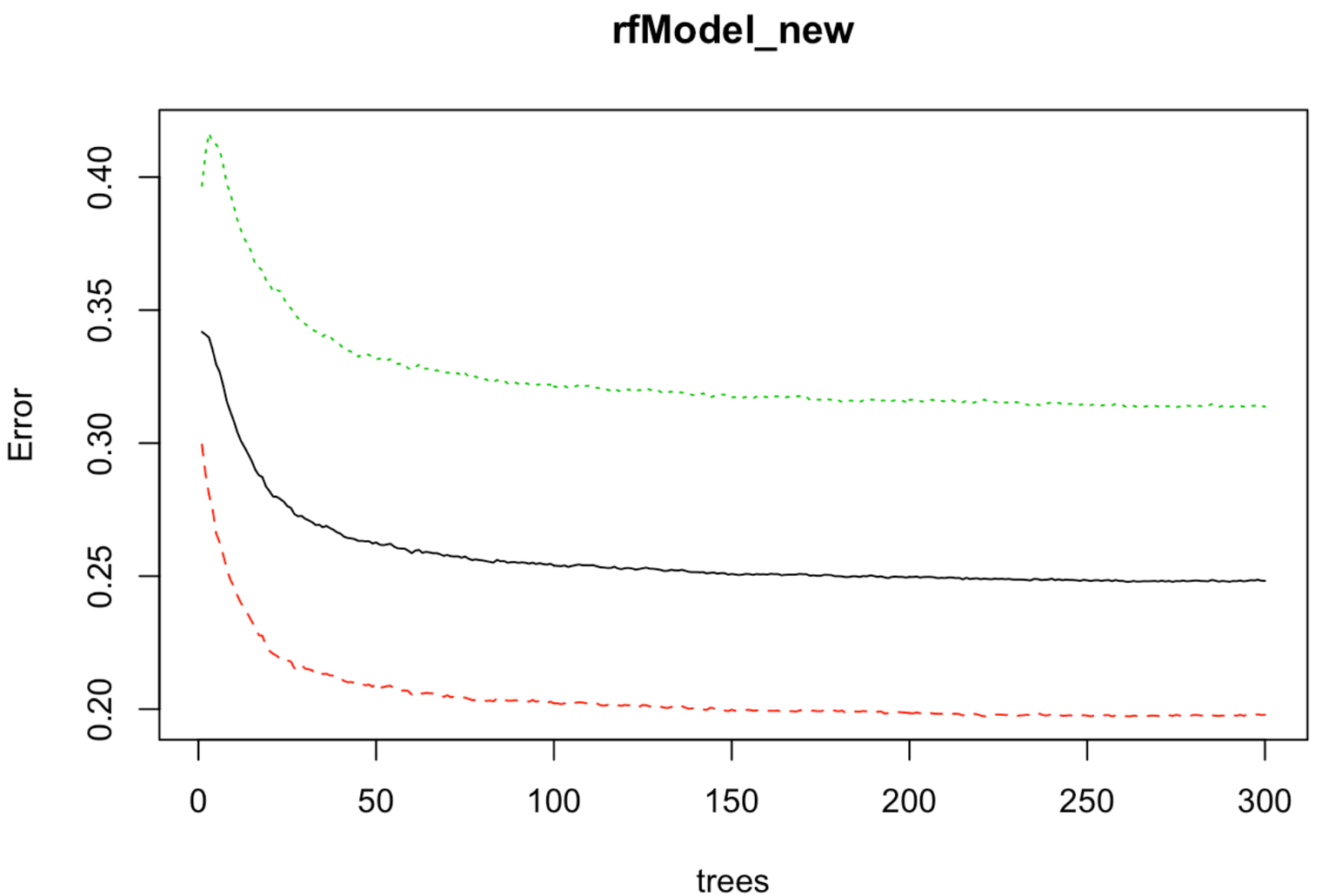



```
##m(try)=4 when tree=180 #accuracy=0.7525 OOB error = 25%
##m(try)=4 when tree=190 #accuracy=0.7525 OOB error = 25%
##m(try)=4 when tree=200 #accuracy=0.7523 OOB error = 24.88%
##m(try)=4 when tree=300 #accuracy=0.7531 OOB error = 24.82%
##m(try)=4 when tree=350 #accuracy=0.753 OOB error = 24.8%
```

```
#run the Random Forest model after tuning
set.seed(100)
rfModel_new <- randomForest(RENEW ~., data=mytrain_data, ntree = 300,
                            mtry = 4, importance = TRUE)
print(rfModel_new)
```

```
##
## Call:
##  randomForest(formula = RENEW ~ ., data = mytrain_data, ntree = 300,      mtry
= 4, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 300
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 24.82%
## Confusion matrix:
##       0     1 class.error
## 0 38151  9410   0.1978512
## 1 11475 25103   0.3137132
```

```
plot(rfModel_new)
```




```
summary(rfModel_new)
```

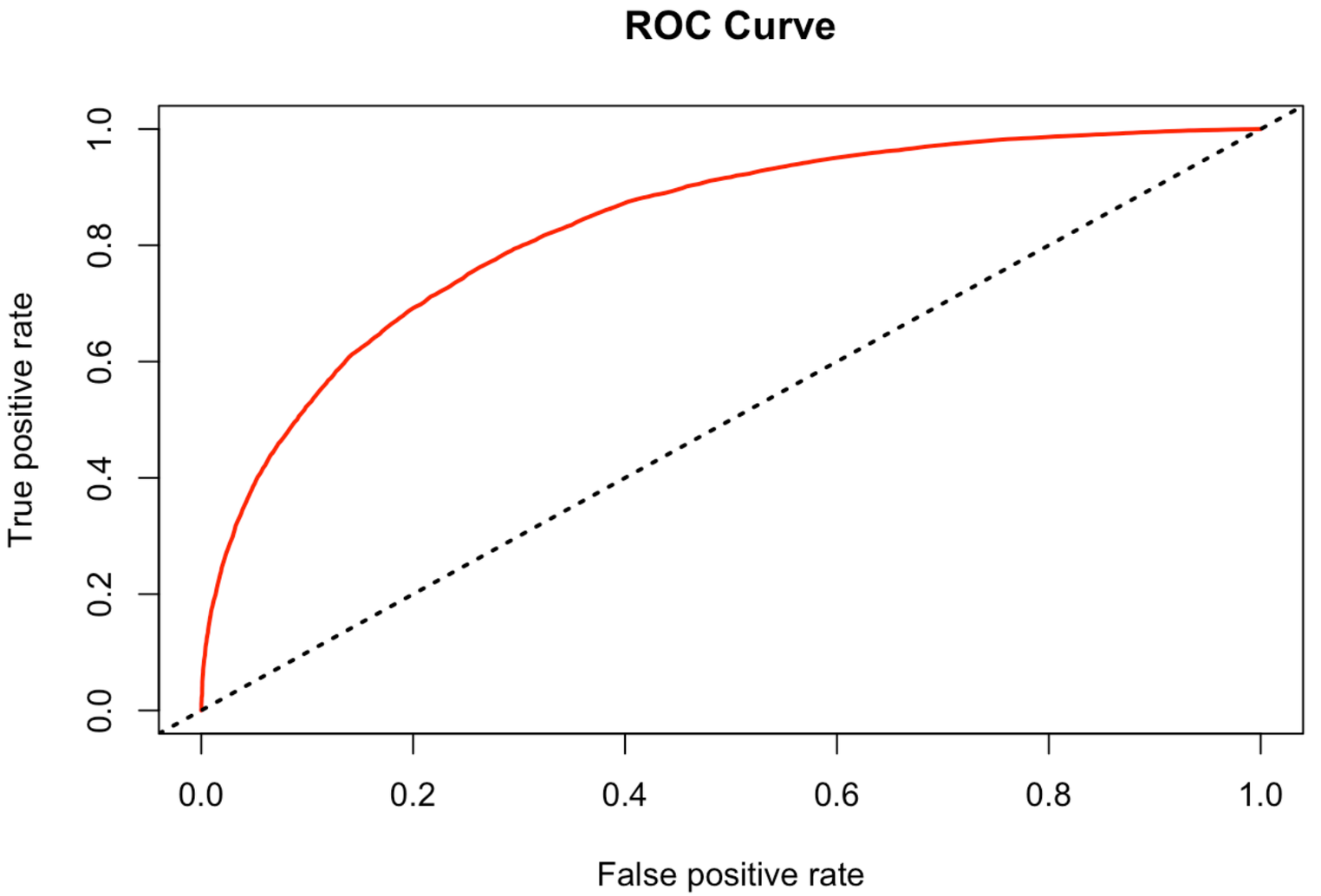
```
##                 Length Class  Mode
## call                 6 -none- call
## type                 1 -none- character
## predicted        84139 factor numeric
## err.rate           900 -none- numeric
## confusion            6 -none- numeric
## votes           168278 matrix numeric
## oob.times        84139 -none- numeric
## classes              2 -none- character
## importance          64 -none- numeric
## importanceSD        48 -none- numeric
## localImportance      0 -none- NULL
## proximity            0 -none- NULL
## ntree                1 -none- numeric
## mtry                 1 -none- numeric
## forest              14 -none- list
## y                84139 factor numeric
## test                 0 -none- NULL
## inbag                0 -none- NULL
## terms                3 terms  call
```

```
#prediction
pred_rf <- predict(rfModel_new, newdata = mytest_data)
pred_prob <- predict(rfModel_new, newdata = mytest_data,type="prob")

#confusion matrix for prediction
caret::confusionMatrix(pred_rf,mytest_data$RENEW)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 16287  4809
##          1  4095 10866
##
##                Accuracy : 0.7531
##                  95% CI : (0.7486, 0.7575)
##     No Information Rate : 0.5653
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4949
##
##  Mcnemar's Test P-Value : 4.154e-14
##
##             Sensitivity : 0.7991
##             Specificity : 0.6932
##          Pos Pred Value : 0.7720
##          Neg Pred Value : 0.7263
##              Prevalence : 0.5653
##          Detection Rate : 0.4517
##    Detection Prevalence : 0.5851
##       Balanced Accuracy : 0.7461
##
##        'Positive' Class : 0
##
```

```
varImpPlot(rfModel_new, sort=T, n.var = 10, main = 'Top 10 Feature Importance')
```


```
#look at the ROC curve and the AUC value
library(ROCR)
pr <- prediction(pred_prob[,2], mytest_data$RENEW)
# plotting ROC curve
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf,main = "ROC Curve",col = 2,lwd = 2)
abline(a = 0,b = 1,lwd = 2,lty = 3,col = "black")
```




```
# AUC value
#AUC stands for "Area under the ROC Curve
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.832326
```

```
#auc=0.832
```