Practicum 2

YKNM (Fall 2022)

Load the required packages for both problems

```
library(klaR)          #for Naive Bayes
```

```
## Loading required package: MASS
```

```
library(dplyr)         #for data wrangling
```

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

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

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

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

```
library(gmodels)       #for proportion matrix
library(fastDummies)   #for dummy column
library(rpart)         #for fitting decision trees
library(rpart.plot)    #for plotting decision trees
library(ipred)         #for fitting bagged decision trees
library(gbm)           #for fitting boosting decision trees
```

```
## Loaded gbm 2.1.8.1
```

```
library(Metrics)       #for dummy column
library(psych)         #for plotting pairs panel
library(corrplot)      #for correlation matrix
```

```
## corrplot 0.92 loaded
```

Problem 1

1. Download the heart failure prediciton dataset.

```r
# heart failure data
raw.heartf <- read.csv("heart.csv")
head(raw.heartf, 5)
```

```
##   Age Sex ChestPainType RestingBP Cholesterol FastingBS RestingECG MaxHR
## 1  40   M           ATA       140         289         0     Normal   172
## 2  49   F           NAP       160         180         0     Normal   156
## 3  37   M           ATA       130         283         0         ST    98
## 4  48   F           ASY       138         214         0     Normal   108
## 5  54   M           NAP       150         195         0     Normal   122
##   ExerciseAngina Oldpeak ST_Slope HeartDisease
## 1              N     0.0       Up            0
## 2              N     1.0     Flat            1
## 3              N     0.0       Up            0
## 4              Y     1.5     Flat            1
## 5              N     0.0       Up            0
```

```r
# rm(list = ls()) # clear env
```

2. Explore the data set as you see fit to get a sense of the data and to get comfortable with it.This could include understanding the structure and statistical details of the dataset.

```r
# check structure
str(raw.heartf)
```

```
## 'data.frame':    918 obs. of  12 variables:
##  $ Age           : int  40 49 37 48 54 39 45 54 37 48 ...
##  $ Sex           : chr  "M" "F" "M" "F" ...
##  $ ChestPainType : chr  "ATA" "NAP" "ATA" "ASY" ...
##  $ RestingBP     : int  140 160 130 138 150 120 130 110 140 120 ...
##  $ Cholesterol   : int  289 180 283 214 195 339 237 208 207 284 ...
##  $ FastingBS     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ RestingECG    : chr  "Normal" "Normal" "ST" "Normal" ...
##  $ MaxHR         : int  172 156 98 108 122 170 170 142 130 120 ...
##  $ ExerciseAngina: chr  "N" "N" "N" "Y" ...
##  $ Oldpeak       : num  0 1 0 1.5 0 0 0 0 1.5 0 ...
##  $ ST_Slope      : chr  "Up" "Flat" "Up" "Flat" ...
##  $ HeartDisease  : int  0 1 0 1 0 0 0 0 1 0 ...
```

```r
# factorize `FastingBS`: as it being a categorical data
raw.heartf$FastingBS <- as.character(raw.heartf$FastingBS)
# check the distribution
table(raw.heartf$FastingBS)
```

```
## 
##   0   1 
## 704 214
```

```r
# summary stats
summary(raw.heartf) # no missing data
```

```
##       Age            Sex            ChestPainType        RestingBP
##  Min.   :28.00   Length:918         Length:918         Min.   :  0.0
##  1st Qu.:47.00   Class :character   Class :character   1st Qu.:120.0
##  Median :54.00   Mode  :character   Mode  :character   Median :130.0
##  Mean   :53.51                                         Mean   :132.4
##  3rd Qu.:60.00                                         3rd Qu.:140.0
##  Max.   :77.00                                         Max.   :200.0
##   Cholesterol     FastingBS          RestingECG            MaxHR
##  Min.   :  0.0   Length:918         Length:918         Min.   : 60.0
##  1st Qu.:173.2   Class :character   Class :character   1st Qu.:120.0
##  Median :223.0   Mode  :character   Mode  :character   Median :138.0
##  Mean   :198.8                                         Mean   :136.8
##  3rd Qu.:267.0                                         3rd Qu.:156.0
##  Max.   :603.0                                         Max.   :202.0
##  ExerciseAngina        Oldpeak          ST_Slope          HeartDisease
##  Length:918         Min.   :-2.6000   Length:918         Min.   :0.0000
##  Class :character   1st Qu.: 0.0000   Class :character   1st Qu.:0.0000
##  Mode  :character   Median : 0.6000   Mode  :character   Median :1.0000
##                     Mean   : 0.8874                      Mean   :0.5534
##                     3rd Qu.: 1.5000                      3rd Qu.:1.0000
##                     Max.   : 6.2000                      Max.   :1.0000
```

```r
# Transform data for Analysis
# get target variable column
heartf.target <- raw.heartf$HeartDisease
```

3. Split the data set 70/30% so you retain 30% for validation and tuning using random sampling without replacement. Use a fixed seed so you produce the same results each time you run the code. Going forward you will use the 70% data set for training and the 30% data set for validation and determining accuracy. Compare the target variable distribution between the train and test data.

```r
# get target variable column
heartf.target <- raw.heartf$HeartDisease

## create a dataset of selective attribute (predictors)
# assign the predictors/attribute
attr <- c('ChestPainType', 'Oldpeak', 'ExerciseAngina','MaxHR', 'Age',
          'FastingBS', 'ST_Slope', 'Cholesterol')
# filter
heartf.data <- raw.heartf %>% select(all_of(attr))
colnames(heartf.data) # check the filtered columns
```

```
## [1] "ChestPainType"  "Oldpeak"        "ExerciseAngina" "MaxHR"
## [5] "Age"            "FastingBS"      "ST_Slope"       "Cholesterol"
```

```r
# Pre-processing step 1
## Binning: transforming continuous variables into categorical variables
heartf <- heartf.data
contvar <- colnames(heartf[, sapply(heartf, class) %in% c('integer', 'numeric')])
contvar # list of continuous variables from the data
```

```
## [1] "Oldpeak"     "MaxHR"       "Age"         "Cholesterol"
```

```r
# create 4 equal bins
heartf[,contvar] <- lapply(heartf[,contvar],
                           function(x) (cut(x, breaks = 4)))

# check the new categorical distribution
table(heartf$Oldpeak)
```

```
##
## (-2.61,-0.4]   (-0.4,1.8]      (1.8,4]     (4,6.21]
##           11          724          177            6
```

```r
table(heartf$Age)
```

```
##
##   (28,40.2] (40.2,52.5] (52.5,64.8]   (64.8,77]
##          93         294         428         103
```

```r
table(heartf$MaxHR)
```

```
##
## (59.9,95.5]  (95.5,131]   (131,166]   (166,202]
##          45         356         391         126
```

```r
table(heartf$Cholesterol)
```

```
##
## (-0.603,151]    (151,302]    (302,452]    (452,604]
##          192          623           95            8
```

```r
# add target variable back to the data set
heartf$HeartDisease <- heartf.target

# check the pre-processed data
str(heartf)
```

```
## 'data.frame':    918 obs. of  9 variables:
##  $ ChestPainType : chr  "ATA" "NAP" "ATA" "ASY" ...
##  $ Oldpeak       : Factor w/ 4 levels "(-2.61,-0.4]",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ ExerciseAngina: chr  "N" "N" "N" "Y" ...
##  $ MaxHR         : Factor w/ 4 levels "(59.9,95.5]",..: 4 3 2 2 2 4 4 3 2 2 ...
##  $ Age           : Factor w/ 4 levels "(28,40.2]","(40.2,52.5]",..: 1 2 1 2 3 1 2 3 1 2 ...
##  $ FastingBS     : chr  "0" "0" "0" "0" ...
##  $ ST_Slope      : chr  "Up" "Flat" "Up" "Flat" ...
##  $ Cholesterol   : Factor w/ 4 levels "(-0.603,151]",..: 2 2 2 2 2 3 2 2 2 2 ...
##  $ HeartDisease  : int  0 1 0 1 0 0 0 0 1 0 ...
```

```r
head(heartf, 3)
```

```
##   ChestPainType    Oldpeak ExerciseAngina      MaxHR         Age FastingBS
## 1           ATA (-0.4,1.8]              N  (166,202]   (28,40.2]         0
## 2           NAP (-0.4,1.8]              N  (131,166] (40.2,52.5]         0
## 3           ATA (-0.4,1.8]              N (95.5,131]   (28,40.2]         0
##   ST_Slope Cholesterol HeartDisease
## 1       Up   (151,302]            0
## 2     Flat   (151,302]            1
## 3       Up   (151,302]            0
```

```r
# split the data set for generating training and valid sets
set.seed(101)  # set seed
# getting 30% sampled index of the data without replacement
sample_idx <- sample(rownames(heartf), nrow(heartf) * 0.3, replace = FALSE)

# valid/test set (30%)
valid_heartf <- heartf[sample_idx, ]
nrow(valid_heartf)
```

```
## [1] 275
```

```r
head(valid_heartf, 3)
```

```
##     ChestPainType    Oldpeak ExerciseAngina      MaxHR         Age FastingBS
## 841           ATA (-0.4,1.8]              N  (131,166] (40.2,52.5]         0
## 825           NAP    (1.8,4]              N  (166,202]   (28,40.2]         0
## 430           NAP (-0.4,1.8]              Y (95.5,131] (52.5,64.8]         0
##     ST_Slope  Cholesterol HeartDisease
## 841     Flat    (151,302]            0
## 825     Down    (151,302]            0
## 430     Flat (-0.603,151]            1
```

```r
# train set(70%)
train_heartf <- heartf[!rownames(heartf) %in% sample_idx,]
nrow(train_heartf)
```

```
## [1] 643
```

```r
head(train_heartf, 3)
```

```
##   ChestPainType    Oldpeak ExerciseAngina      MaxHR         Age FastingBS
## 1           ATA (-0.4,1.8]              N  (166,202]   (28,40.2]         0
## 4           ASY (-0.4,1.8]              Y (95.5,131] (40.2,52.5]         0
## 6           NAP (-0.4,1.8]              N  (166,202]   (28,40.2]         0
##   ST_Slope Cholesterol HeartDisease
## 1       Up   (151,302]            0
## 4     Flat   (151,302]            1
## 6       Up   (302,452]            0
```

```r
# target variable distribution of the main dataset
table(heartf$HeartDisease) /nrow(heartf) * 100
```

```
##
##        0        1
## 44.66231 55.33769
```

```r
# # compare the target variable distribution between the train and calid/test data.
table(valid_heartf$HeartDisease)/ nrow(valid_heartf) * 100
```

```
##
##        0        1
## 44.36364 55.63636
```

```r
table(train_heartf$HeartDisease)/ nrow(train_heartf) * 100
```

```
##
##        0        1
## 44.79005 55.20995
```

```r
# assign target variable of train and valid/test data
target.train <- train_heartf$HeartDisease
target.test <- valid_heartf$HeartDisease
```

4. Using the Naive Bayes Classification algorithm from the KlaRpackage, build a binary classifier that predicts whether the person with certain conditions has heart disease or not. Only use the following input attributes: ChestPainType, Oldpeak, ExerciseAngina, MaxHR, Age, FastingBS, ST_Slope, Cholesterol and Ignore any other features in your model. You need to transform continuous variables into categorical variables by binning (use equal size bins from min to max).

Naive Bayes Classification

```r
# train and test data for Naive Bayes
nb_train <- select(train_heartf, -"HeartDisease")
nb_test <- select(valid_heartf, -"HeartDisease")

# Create Naive Bayes function
NB <- function(train, target) {
  ## Naive Bayes modeling
  NB_model <- NaiveBayes(as.factor(target) ~., train)

  ## Predictions
  NB_pred <- predict(NB_model)

  ## Class distribution from the classifier
  NB_table <- table(NB_pred$class, target)

  ## Model accuracy
  NB_accuracy <- round(sum(diag(NB_table)) / sum(NB_table) * 100, 2)
```

```r
  ## Model outputs
  NB_out <- list("model"= NB_model, "acc" = NB_accuracy)
  return(NB_out)
}

# call the model function
naiveB <- NB(train = nb_train, target = target.train)
# model
nb_model <- naiveB$model
nb_model$tables
```

```
## $ChestPainType
##          var
## grouping        ASY        ATA        NAP         TA
##        0 0.26388889 0.35416667 0.31944444 0.06250000
##        1 0.76619718 0.04225352 0.14366197 0.04788732
##
## $Oldpeak
##          var
## grouping (-2.61,-0.4]  (-0.4,1.8]     (1.8,4]    (4,6.21]
##        0  0.003472222 0.958333333 0.038194444 0.000000000
##        1  0.016901408 0.676056338 0.301408451 0.005633803
##
## $ExerciseAngina
##          var
## grouping         N         Y
##        0 0.8715278 0.1284722
##        1 0.3690141 0.6309859
##
## $MaxHR
##          var
## grouping (59.9,95.5] (95.5,131]  (131,166]  (166,202]
##        0  0.01388889 0.21875000 0.50694444 0.26041667
##        1  0.07887324 0.52112676 0.35211268 0.04788732
##
## $Age
##          var
## grouping  (28,40.2] (40.2,52.5] (52.5,64.8]  (64.8,77]
##        0 0.14930556  0.39930556  0.37847222 0.07291667
##        1 0.05070423  0.26478873  0.55211268 0.13239437
##
## $FastingBS
##          var
## grouping          0          1
##        0 0.90277778 0.09722222
##        1 0.67042254 0.32957746
##
## $ST_Slope
##          var
## grouping       Down       Flat         Up
##        0 0.02083333 0.18402778 0.79513889
##        1 0.08169014 0.75774648 0.16056338
##
```

```
## $Cholesterol
##         var
## grouping (-0.603,151]  (151,302]  (302,452]  (452,604]
##        0   0.06944444 0.80902778 0.11111111 0.01041667
##        1   0.31830986 0.56901408 0.10140845 0.01126761
```

5. Build a confusion matrix for the classifier from (4) and comment on it, e.g., explain what it means.

```r
# Evaluate the model performance
## predictions
nb_pred <- predict(nb_model, nb_test, type="class")

## correlation
cor(as.numeric(nb_pred$class), target.test)
```

```
## [1] 0.660308
```

```r
## confusion matrix
nb_confmat <- CrossTable(nb_pred$class, target.test,
                         prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
                         dnn = c('Predicted', 'Actual'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  275
##
##
##              | Actual
##    Predicted |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |        97 |        21 |       118 |
##              |     0.353 |     0.076 |           |
## -------------|-----------|-----------|-----------|
##            1 |        25 |       132 |       157 |
##              |     0.091 |     0.480 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       122 |       153 |       275 |
## -------------|-----------|-----------|-----------|
##
##
```

```r
## Accuracy of the model
c("NaiveB.Accuaracy"= round((97 + 132) / 275 * 100, 2))
```

```
## NaiveB.Accuaracy
##            83.27
```

From the evaluation, we draw the following conclusion (1) The prediction class and target are moderately correlated. (2) From the confusion matrix, we get 229 true positives and 46 true negatives. (3) The model generates an accuracy of 83.27%.

6. Create a full Logistic Regression model of the same features as in (4) (i.e., do not eliminate any features regardless of p-value). Be sure to either use dummy coding for categorical features or convert them to factor variables and ensure that the glm function does the dummy coding.

Logistic Regression Classification

```
# Data for model
# Pre-processing step 2
## Dummy coding of the categorical variables

# function for dummy coding of character columns
dummycols <- function(x) {
  # select character columns
  charcols <- colnames(x[, sapply(x, class) %in% c('character')])
  # add dummy columns to the data set
  dc <- dummy_cols(x, select_columns = all_of(charcols))
  # remove the character columns from the dummy data set
  data <- dc %>% select(-charcols)
  return(data)
}

# train and test data for Logistic regression
lgr1 <- select(train_heartf, -"HeartDisease")
lgr2 <- select(valid_heartf, -"HeartDisease")

# call the dummy column function
# train set
lgr_train <- dummycols(lgr1)
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use ‘all_of(charcols)‘ instead of ‘charcols‘ to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
# test set
lgr_test <- dummycols(lgr2)
# check
head(lgr_train,3)
```

```
##       Oldpeak      MaxHR         Age Cholesterol ChestPainType_ASY
## 1 (-0.4,1.8]  (166,202]   (28,40.2]   (151,302]                 0
## 2 (-0.4,1.8] (95.5,131] (40.2,52.5]   (151,302]                 1
## 3 (-0.4,1.8]  (166,202]   (28,40.2]   (302,452]                 0
##   ChestPainType_ATA ChestPainType_NAP ChestPainType_TA ExerciseAngina_N
## 1                 1                 0                0                1
## 2                 0                 0                0                0
## 3                 0                 1                0                1
##   ExerciseAngina_Y FastingBS_0 FastingBS_1 ST_Slope_Down ST_Slope_Flat
```

```
## 1                0           1           0             0             0
## 2                1           1           0             0             1
## 3                0           1           0             0             0
##   ST_Slope_Up
## 1           1
## 2           0
## 3           1
```

```
head(lgr_test,3)
```

```
##      Oldpeak      MaxHR         Age  Cholesterol ChestPainType_ASY
## 1 (-0.4,1.8]  (131,166] (40.2,52.5]    (151,302]                 0
## 2    (1.8,4]  (166,202]   (28,40.2]    (151,302]                 0
## 3 (-0.4,1.8] (95.5,131] (52.5,64.8] (-0.603,151]                 0
##   ChestPainType_ATA ChestPainType_NAP ChestPainType_TA ExerciseAngina_N
## 1                 1                 0                0                1
## 2                 0                 1                0                1
## 3                 0                 1                0                0
##   ExerciseAngina_Y FastingBS_0 FastingBS_1 ST_Slope_Down ST_Slope_Flat
## 1                0           1           0             0             1
## 2                0           1           0             1             0
## 3                1           1           0             0             1
##   ST_Slope_Up
## 1           0
## 2           0
## 3           0
```

```
LGR <- function(train, target){
  ## Logistic Regression modeling
  LGR_model <- glm(target ~., data = train,
                   family = binomial(link="logit"))
  # summary(logreg_model)
  ## Predictions
  LGR_prob <- predict(LGR_model)
  LGR_pred <- ifelse(LGR_prob > 0.5, "1", "0")

  ## Class distribution from the classifier
  LGR_table <- table(LGR_pred, target)

  ## Model accuracy
  LGR_accuracy <- round(sum(diag(LGR_table)) / sum(LGR_table) * 100, 2)

  ## Model outputs
  LGR_out <- list('model' = LGR_model, 'acc' = LGR_accuracy)
  return(LGR_out)
}

# Call the model function
logReg <- LGR(train = lgr_train, target = target.train)
# model
lgr_model <- logReg$model
lgr_model
```

```
##
## Call:  glm(formula = target ~ ., family = binomial(link = "logit"),
##     data = train)
##
## Coefficients:
##          (Intercept)     Oldpeak(-0.4,1.8]        Oldpeak(1.8,4]
##              2.96690              -0.85672               0.47300
##      Oldpeak(4,6.21]       MaxHR(95.5,131]        MaxHR(131,166]
##             12.97159              -0.59148              -0.70537
##       MaxHR(166,202]        Age(40.2,52.5]        Age(52.5,64.8]
##             -0.86927              -0.24767              -0.01421
##         Age(64.8,77]  Cholesterol(151,302]  Cholesterol(302,452]
##              0.14707              -1.39706              -1.44692
## Cholesterol(452,604]     ChestPainType_ASY     ChestPainType_ATA
##             -1.47768               1.01185              -0.99342
##    ChestPainType_NAP      ChestPainType_TA      ExerciseAngina_N
##             -0.53535                    NA              -1.25322
##     ExerciseAngina_Y           FastingBS_0           FastingBS_1
##                   NA              -1.14257                    NA
##        ST_Slope_Down         ST_Slope_Flat           ST_Slope_Up
##              1.43736               2.36095                    NA
##
## Degrees of Freedom: 642 Total (i.e. Null);  623 Residual
## Null Deviance:       884.4
## Residual Deviance: 414.2     AIC: 454.2
```

7. Build a confusion matrix for the classifier from (6) and comment on it, e.g., explain what it means.

```r
# Evaluate the model performance
## predictions
lgr_prob <- predict(lgr_model, lgr_test, type = "response")
```

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

```r
lgr_pred <- ifelse(lgr_prob > 0.5, 1, 0)

## correlation
cor(as.numeric(lgr_pred), target.test)
```

```
## [1] 0.6897456
```

```r
## confusion matrix
lgr_confmat <- CrossTable(lgr_pred, target.test,
                          prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
                          dnn = c('Predicted', 'Actual'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
```

```
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  275
##
##
##              | Actual
##    Predicted |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |        98 |        18 |       116 |
##              |     0.356 |     0.065 |           |
## -------------|-----------|-----------|-----------|
##            1 |        24 |       135 |       159 |
##              |     0.087 |     0.491 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       122 |       153 |       275 |
## -------------|-----------|-----------|-----------|
##
##
```

```r
## model accuracy
c("LogReg.Accuracy" = round((98 + 135) / 275 * 100, 2))
```

```
## LogReg.Accuracy
##           84.73
```

From the evaluation, we draw the following conclusion (1) The prediction class and target are moderately correlated. (2) From the confusion matrix, we get 235 true positives and 46 true negatives. (3) The model generates an accuracy of 84.73%.

8. Create a Decision Tree model from rpart package to predict the target variable. Use the same features as (4).

Decision Tree Classification

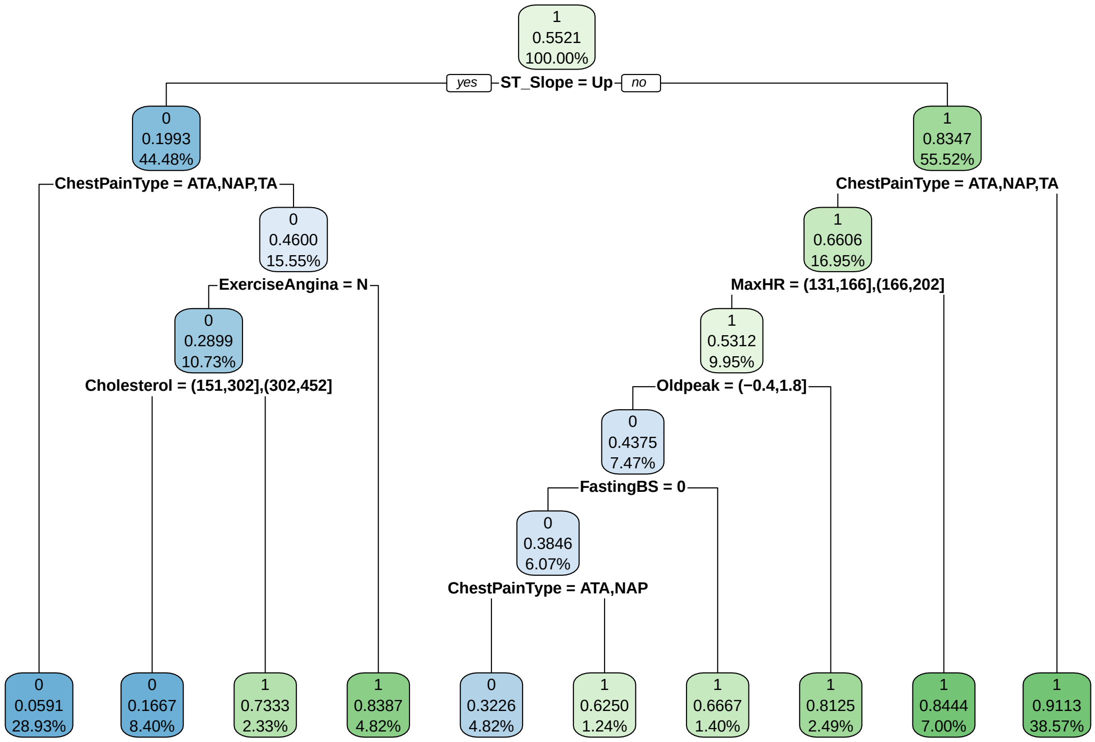
```r
# data for model
## train and test data for Logistic regression
rp_train <- select(train_heartf, -"HeartDisease")
rp_test <- select(valid_heartf, -"HeartDisease")

# Create Decision Tree function
DT <- function(train, target) {
  ## Decision Tree modeling using rpart
  RP_model <- rpart(target ~., data = train, method = "class",
                    control = rpart.control(cp = 0))

  ## Predictions
  RP_pred <- predict(RP_model, type = "class")

  ## Class distribution for the classifier
  RP_table <- table(RP_pred, target)
```

```r
  ## Accuracy
  RP_accuracy <- round(sum(diag(RP_table)) / sum(RP_table) * 100, 2)

  ## Model outputs
  RP_out <- list('model'= RP_model, 'acc'= RP_accuracy)
  return(RP_out)
}

# call the model function
decisionTree <- DT(train = rp_train, target = target.train)
# model
rp_model <- decisionTree$model
rp_model
```

```
## n= 643
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 643 288 1 (0.44790047 0.55209953)
##    2) ST_Slope=Up 286  57 0 (0.80069930 0.19930070)
##      4) ChestPainType=ATA,NAP,TA 186  11 0 (0.94086022 0.05913978) *
##      5) ChestPainType=ASY 100  46 0 (0.54000000 0.46000000)
##       10) ExerciseAngina=N 69  20 0 (0.71014493 0.28985507)
##         20) Cholesterol=(151,302],(302,452] 54   9 0 (0.83333333 0.16666667) *
##         21) Cholesterol=(-0.603,151] 15   4 1 (0.26666667 0.73333333) *
##       11) ExerciseAngina=Y 31   5 1 (0.16129032 0.83870968) *
##    3) ST_Slope=Down,Flat 357  59 1 (0.16526611 0.83473389)
##      6) ChestPainType=ATA,NAP,TA 109  37 1 (0.33944954 0.66055046)
##       12) MaxHR=(131,166],(166,202] 64  30 1 (0.46875000 0.53125000)
##         24) Oldpeak=(-0.4,1.8] 48  21 0 (0.56250000 0.43750000)
##           48) FastingBS=0 39  15 0 (0.61538462 0.38461538)
##             96) ChestPainType=ATA,NAP 31  10 0 (0.67741935 0.32258065) *
##             97) ChestPainType=TA 8   3 1 (0.37500000 0.62500000) *
##           49) FastingBS=1 9   3 1 (0.33333333 0.66666667) *
##         25) Oldpeak=(-2.61,-0.4],(1.8,4] 16   3 1 (0.18750000 0.81250000) *
##       13) MaxHR=(59.9,95.5],(95.5,131] 45   7 1 (0.15555556 0.84444444) *
##      7) ChestPainType=ASY 248  22 1 (0.08870968 0.91129032) *
```

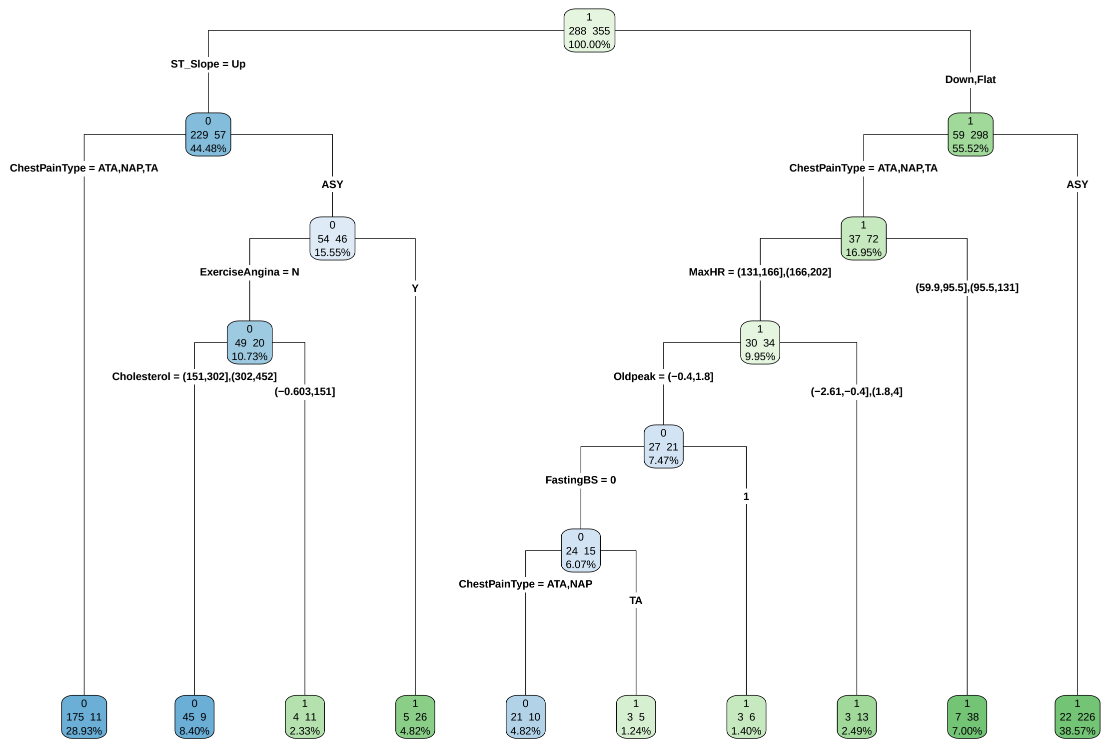
```r
# visualize the decision tree model
rpart.plot(rp_model, digits = 4)
```

```
## Warning: Cannot retrieve the data used to build the model (so cannot determine roundint and is.binary
## To silence this warning:
##     Call rpart.plot with roundint=FALSE,
##     or rebuild the rpart model with model=TRUE.
```




```
# forces leaf nodes to align at the bottom of the tree
rpart.plot(rp_model, digits = 4, fallen.leaves = TRUE,
           type = 4, extra = 101)
```

```
## Warning: Cannot retrieve the data used to build the model (so cannot determine roundint and is.binary
## To silence this warning:
##     Call rpart.plot with roundint=FALSE,
##     or rebuild the rpart model with model=TRUE.
```



9. How do you compare the decision tree of rpart package with C5.0 package which is used in the book? which one do you prefer and why?

Talking about **C5.0** that generates decision tree for classification and is implemented as C50. - By reducing the estimated entropy value, this approach employs an information entropy computation to find the best rule that divides the data at that node into purer classes - this means that each subset of data split by the rule will initially contain less diversity of classes and will finally contain only one class [full purity] as each node splits the data depending on the rule at that node. Both numerical and category data can be used; this example uses both. Additionally, it can accept missing data values. New unclassified data items can be given `predicted` a class using the output model.

Speaking of **Rpart**, another decision tree generating algorithm used for classification as well as regression (will be applied further). Its principle is based on Recursive partitioning. Similar to C50, rpart employs a computational measure to choose the optimum rule for dividing the data at a given node into more pure classes.The Gini coefficient serves as the computational metric in the rpart method. Rpart divides the data into purer class subsets at each node by minimizing the Gini coefficient, placing the class leaf nodes at the base of the decision tree. The tree also used for predicting classes to unknown data items.

10. Build a confusion matrix for the classifier from (8) and comment on it, e.g., explain what it means.

```r
# pruning
# rp_model_pruned <- prune(decisionTree$model, cp = 0.00015)

# Evaluate the model performance
## predictions
```

```r
rp_pred <- predict(rp_model, rp_test, type = "class")

## correlation
cor(as.numeric(rp_pred), target.test)
```

```
## [1] 0.6530715
```

```r
## confusion matrix for the classifier
rp_confmat <- CrossTable(rp_pred, target.test,
                         prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
                         dnn = c('Predicted', 'Actual'))
```

```
## 
## 
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
## 
## 
## Total Observations in Table:  275 
## 
## 
##              | Actual 
##    Predicted |         0 |         1 | Row Total | 
## -------------|-----------|-----------|-----------|
##            0 |        92 |        17 |       109 | 
##              |     0.335 |     0.062 |           | 
## -------------|-----------|-----------|-----------|
##            1 |        30 |       136 |       166 | 
##              |     0.109 |     0.495 |           | 
## -------------|-----------|-----------|-----------|
## Column Total |       122 |       153 |       275 | 
## -------------|-----------|-----------|-----------|
## 
## 
```

```r
# model accuracy
c("decisionTree.Accuracy" = round((92 + 136) / 275 * 100, 2))
```

```
## decisionTree.Accuracy 
##                 82.91
```

From the evaluation, we draw the following conclusion (1) The prediction class and target are moderately correlated. (2) From the confusion matrix, we get 228 true positives and 46 true negatives. (3) The model generates an accuracy of 82.91%.

11. Build a function called predictHeartDisease() that predicts the heart disease of an individual and that combines the three predictive models from (4) and (6) and (8) into a simple ensemble. If the models disagree on a prediction, then the prediction that gets the majority vote is the winner, you can also

weight each vote by its corresponding classifier's accuracy over the train set – make sure you do not hard code that as the training data may change over time and the same model may not be the more accurate forever.

Ensemble function: predictHeartDisease

```
# define the function
PredictHeartDisease <-
  function(model1, model2, model3, data, train, target, test, dcols) {
  # model1 = Naive Bayes
  # model2 = Logistic regression
  # model3 = Decision tree
    ## All these individual model func returns a list of their model structure
    ## and respective accuracy only on train set
  # train = training set of heart failure data
  # target = response variable (target) i.e. HeartDisease column from the train set
  # test = new test cases (transformed according to our train set for prediction)
  # _dcols = helper function i.e. dummy column

  ##Model1: Naive Bayes##
  nbtrain <- train
  # modeling
  naiveB <- model1(nbtrain, target)

  ##Model2: Logistic regression##
  # pre-process
  lgrtrain <- dcols(train)
  # modeling
  logReg <- model2(lgrtrain, target)

  ##Model3: Decision Tree##
  rptrain <- train
  # modeling
  decisionTree <- model3(rptrain, target)

  ## Weighing each vote
  # comparing accuracy of each model
  Model <- c('Naive Bayes', 'Logistic Reg', 'Decision Tree')
  ModelAcc <- c(naiveB$acc, logReg$acc, decisionTree$acc)
  # wrapping into a data frame
  modelTable <- data.frame(Model, ModelAcc)
  # Choose the best model having maximum accuracy
  bestModel <- modelTable$Model[which.max(modelTable$ModelAcc)]

  # Every model returns predicted probability on test data and the final
  # prediction is the one that receives majority of the votes.
  # Generating class for the new test case
  # In the factorized target column, 1 is assigned to individual have/may have
  # a "heart disease" whereas 0 means they do not, with the respective to their
  # predictors/characteristics

  while(class(bestModel) == "character") {
    if (bestModel == "Naive Bayes") {
      pred1 <- predict(naiveB$model, test, type= "class")
```

```
      m1class <- ifelse(pred1$class == 1, "Yes", "No")
      return(list("bestmodel"= bestModel, "predClass"= m1class))
    }
    if (bestModel == "Logistic Reg") {
      prob2 <- predict(logReg$model, test, type = "response")
      pred2 <- ifelse(prob2 > 0.5, 1, 0)
      m2class <- ifelse(pred2 == 1, "Yes", "No")
      return(list("bestmodel"= bestModel, "predClass"= m2class))
    }
    if (bestModel == "Decision Tree") {
      pred3 <- predict(decisionTree$model, test, type = "class")
      m3class <- ifelse(pred3 == 1, "Yes", "No")
      return(list("bestmodel"= bestModel, "predClass"= m3class))
    }
    break
  }
}
```

12. Using the ensemble model from (11), predict the heart disease indicator of the following individual: ChestPainType : ASY | Old Peak : 1.0 | ExerciseAngina : Y | MaxHR : 138 | Age : 56 | FastingBS : 1 | ST_Slope :Flat | Cholesterol : 280.

```
# assign the new test cases
test_ensemble <- data.frame(ChestPainType= "ASY", Oldpeak= 1.0,
                            ExerciseAngina= "Y", MaxHR= 138, Age= 56,
                            FastingBS = 1, ST_Slope= "Flat", Cholesterol= 280)

# impute missing data using median
# test_ensemble$HeartDisease <- median(sort(heartf.target)) # not necessary

# Transform the test case
## combine test with the main data
## heartf.data <- raw.heartf %>% select(all_of(attr))
data.ens <- rbind(heartf.data, test_ensemble)
tail(data.ens,3)
```

```
##     ChestPainType Oldpeak ExerciseAngina MaxHR Age FastingBS ST_Slope
## 917           ATA       0              N   174  57         0     Flat
## 918           NAP       0              N   173  38         0       Up
## 919           ASY       1              Y   138  56         1     Flat
##     Cholesterol
## 917         236
## 918         175
## 919         280
```

```
## continous to categorical
data.ens[,contvar] <- lapply(data.ens[,contvar],
                             function(x) (cut(x, breaks = 4)))

## get the transformed test cases
test.ens <- data.ens[nrow(data.ens),]
test.ens
```

```
##     ChestPainType    Oldpeak ExerciseAngina     MaxHR         Age FastingBS
## 919           ASY (-0.4,1.8]              Y (131,166] (52.5,64.8]         1
##     ST_Slope Cholesterol
## 919     Flat   (151,302]
```

```
## remove the transformed test case from the data
data.ens <- data.ens[-nrow(data.ens),]

## get the train data for the ensemble function
data.ens$HeartDisease <- heartf.target
training <- data.ens[!rownames(data.ens) %in% sample_idx,]

## get the train and target
train.target <- training$HeartDisease
train.ens <- select(train_heartf, -"HeartDisease")

# call the ensemble model function
predHDclass <- PredictHeartDisease(model1 = NB, model2 = LGR, model3 = DT,
                                   train = train.ens, target = train.target,
                                   test = test.ens, dcols = dummycols)
# get predicted class
HDclass <- predHDclass$predClass

# print out the class
if (HDclass == "Yes") {
  print(paste("The individual may have a heart disease!"))
}else {
  print(paste("The individual may not have a heart disease!"))
}
```

```
## [1] "The individual may have a heart disease!"
```

Bagging and Boosting

```
# using the ensemble learning frameworks of Bagging and Boosting to improve the performance
# of the best Model on train set
bestModel <- predHDclass$bestmodel
bestModel
```

```
## [1] "Decision Tree"
```

So, the best model chosen after majority voting is **Decision Tree**.

Bagging

```
# train =  rp_train
# target = target.train

# bagging involves bootstraping, so will set seed
set.seed(25)
```

```r
# create the model with `bagging`
bagged_rp.model <- bagging(
  formula = target.train ~ .,
  data = rp_train,
  coob = TRUE,
  control = rpart.control(cp = 0)
)

bagged_rp.model
```

```
##
## Bagging regression trees with 25 bootstrap replications
##
## Call: bagging.data.frame(formula = target.train ~ ., data = rp_train,
##     coob = TRUE, control = rpart.control(cp = 0))
##
## Out-of-bag estimate of root mean squared error:  0.3336
```

Note that the model use 25 bootstrapped samples to build the bagged model and we specified coob to be TRUE to obtain the estimated out-of-bag error. control parameter are chosen so that those two arguments allow the individual trees to grow extremely deep, which leads to trees with high variance but low bias

From the output of the model we can see that the out-of-bag estimated RMSE is 0.36. This is the average difference between the predicted value for `HeartDisease` and the actual observed value.

```r
# evalutating the model
## predicitons
bagged_rp_pred <- round(predict(bagged_rp.model))

## proportion table
bagged_rp_table <- table(bagged_rp_pred, target.train)

## bagged accuracy
bag_rp_accuracy <- round(sum(diag(bagged_rp_table)) / sum(bagged_rp_table) * 100, 2)

## comparing the accuracy from the previous model
c("decisionTree"= decisionTree$acc, "bagged.DT"= bag_rp_accuracy)
```

```
## decisionTree    bagged.DT
##        88.02        86.31
```

Bagging did not improved the model to any extent.

Boosting

```r
# Boosting
# factorize character variables
rp_train$ChestPainType <- as.factor(rp_train$ChestPainType)
rp_train$ExerciseAngina <- as.factor(rp_train$ExerciseAngina)
rp_train$ST_Slope <- as.factor(rp_train$ST_Slope)
rp_train$FastingBS <- as.factor(rp_train$FastingBS)

# gradient boosting with `gbm`
boost_rp.model <- gbm(target.train ~., data = rp_train)
```

```
## Distribution not specified, assuming bernoulli ...
```

```
boost_rp.model
```

```
## gbm(formula = target.train ~ ., data = rp_train)
## A gradient boosted model with bernoulli loss function.
## 100 iterations were performed.
## There were 8 predictors of which 8 had non-zero influence.
```

```
## predicitons
boost_rp_pred <- round(predict.gbm(object = boost_rp.model,
                            newdata = rp_train,
                            type = "response"))
```

```
## Using 100 trees...
```

```
## proportion table
boost_rp_table <- table(boost_rp_pred, target.train)

## boosting accuracy
boost_rp_accuracy <- round(sum(diag(boost_rp_table)) / sum(boost_rp_table) * 100, 2)

## comparing the accuracy from the previous model
c("decisionTree"= decisionTree$acc, "boost.DT"= boost_rp_accuracy)
```

```
## decisionTree     boost.DT
##        88.02        87.25
```

Bagging did not improved the model to any extent too.

Problem 2

1. Download the Demand Forecasting dataset from UCI repositoryLinks to an external site. Load the dataset into an R dataframe and call it orders.df.

```
# load the dataset
orders.df <- read.table("Daily_Demand_Forecasting_Orders.csv",
                         sep = ";", header = TRUE)
# investigate the data
## assign new column names
colnames(orders.df) <- c("Week.of.the.month", "Day.of.the.week", "Non.urgent",
                         "Urgent", "TypeA", "TypeB", "TypeC", "Fiscal.sector",
                         "Traffic.control.sector","Banking.orders1",
                         "Banking.orders2","Banking.orders3", "Total.orders")
str(orders.df) # structure
```

```
## 'data.frame':    60 obs. of  13 variables:
##  $ Week.of.the.month     : int  1 1 1 2 2 2 2 2 3 3 ...
##  $ Day.of.the.week       : int  4 5 6 2 3 4 5 6 2 3 ...
##  $ Non.urgent            : num  316.3 128.6 43.7 171.3 90.5 ...
```

```
##  $ Urgent               : num  223.3 96 84.4 127.7 113.5 ...
##  $ TypeA                : num  61.5 38.1 21.8 41.5 37.7 ...
##  $ TypeB                : num  175.6 56 25.1 113.3 56.6 ...
##  $ TypeC                : num  302.4 130.6 82.5 162.3 116.2 ...
##  $ Fiscal.sector        : num  0 0 1.39 18.16 6.46 ...
##  $ Traffic.control.sector: int  65556 40419 11992 49971 48534 52042 46573 35033 66612 58224 ...
##  $ Banking.orders1      : int  44914 21399 3452 33703 19646 8773 33597 26278 19461 7742 ...
##  $ Banking.orders2      : int  188411 89461 21305 69054 16411 47522 48269 56665 103376 82395 ...
##  $ Banking.orders3      : int  14793 7679 14947 18423 20257 24966 20973 18502 10458 11948 ...
##  $ Total.orders         : num  540 225 129 317 211 ...
```

```
head(orders.df, 5) # subset
```

```
##   Week.of.the.month Day.of.the.week Non.urgent  Urgent  TypeA   TypeB   TypeC
## 1                 1               4    316.307 223.270 61.543 175.586 302.448
## 2                 1               5    128.633  96.042 38.058  56.037 130.580
## 3                 1               6     43.651  84.375 21.826  25.125  82.461
## 4                 2               2    171.297 127.667 41.542 113.294 162.284
## 5                 2               3     90.532 113.526 37.679  56.618 116.220
##   Fiscal.sector Traffic.control.sector Banking.orders1 Banking.orders2
## 1         0.000                  65556           44914          188411
## 2         0.000                  40419           21399           89461
## 3         1.386                  11992            3452           21305
## 4        18.156                  49971           33703           69054
## 5         6.459                  48534           19646           16411
##   Banking.orders3 Total.orders
## 1           14793      539.577
## 2            7679      224.675
## 3           14947      129.412
## 4           18423      317.120
## 5           20257      210.517
```

```
# check for missing values
summary(orders.df)
```

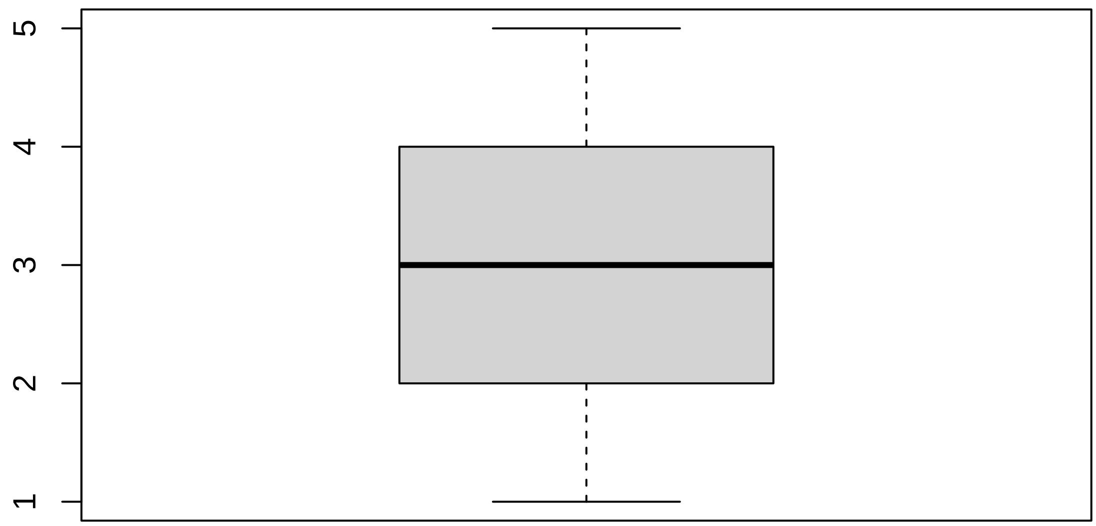
```
##  Week.of.the.month Day.of.the.week   Non.urgent         Urgent
##  Min.   :1.000     Min.   :2.000   Min.   : 43.65   Min.   : 77.37
##  1st Qu.:2.000     1st Qu.:3.000   1st Qu.:125.35   1st Qu.:100.89
##  Median :3.000     Median :4.000   Median :151.06   Median :113.11
##  Mean   :3.017     Mean   :4.033   Mean   :172.55   Mean   :118.92
##  3rd Qu.:4.000     3rd Qu.:5.000   3rd Qu.:194.61   3rd Qu.:132.11
##  Max.   :5.000     Max.   :6.000   Max.   :435.30   Max.   :223.27
##      TypeA             TypeB             TypeC         Fiscal.sector
##  Min.   : 21.83   Min.   : 25.12   Min.   : 74.37   Min.   :  0.000
##  1st Qu.: 39.46   1st Qu.: 74.92   1st Qu.:113.63   1st Qu.:  1.243
##  Median : 47.17   Median : 99.48   Median :127.99   Median :  7.832
##  Mean   : 52.11   Mean   :109.23   Mean   :139.53   Mean   : 77.396
##  3rd Qu.: 58.46   3rd Qu.:132.17   3rd Qu.:160.11   3rd Qu.: 20.361
##  Max.   :118.18   Max.   :267.34   Max.   :302.45   Max.   :865.000
##  Traffic.control.sector Banking.orders1  Banking.orders2  Banking.orders3
##  Min.   :11992          Min.   :  3452   Min.   : 16411   Min.   : 7679
##  1st Qu.:34994          1st Qu.: 20130   1st Qu.: 50680   1st Qu.:12610
##  Median :44312          Median : 32528   Median : 67181   Median :18012
```

```
##  Mean   :44504          Mean   : 46641   Mean   : 79401   Mean   :23115
##  3rd Qu.:52112          3rd Qu.: 45119   3rd Qu.: 94788   3rd Qu.:31048
##  Max.   :71772          Max.   :210508   Max.   :188411   Max.   :73839
##   Total.orders
##  Min.   :129.4
##  1st Qu.:238.2
##  Median :288.0
##  Mean   :300.9
##  3rd Qu.:334.2
##  Max.   :616.5
```

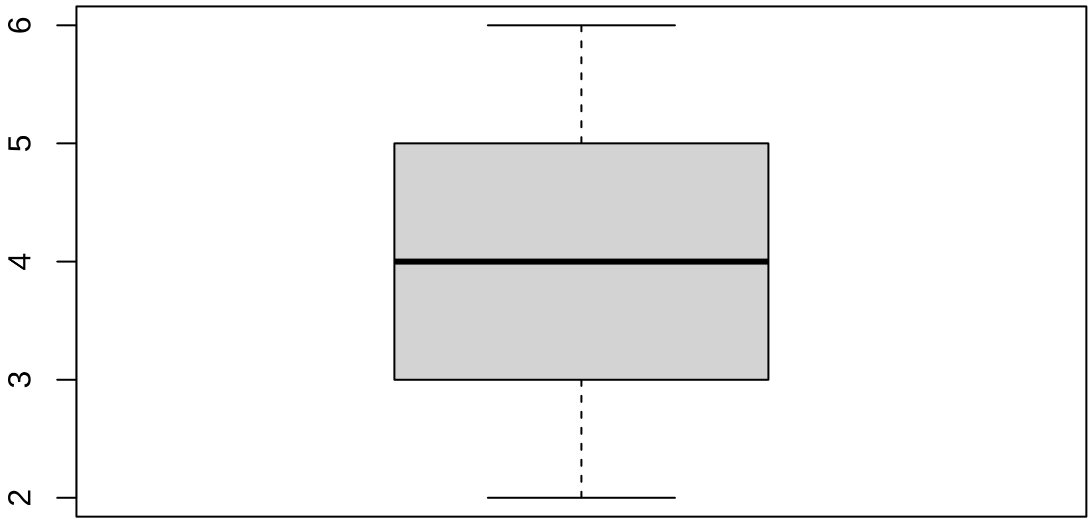
From the summary statistics, the data appears to have no missing values.

2. Are there outliers in any one of the features in the data set? How do you identify outliers? Remove them but create a second data set with outliers removed called orders.no.df. Keep the original data set orders.df. (Note: Use filter only on columns: Order.Type.C <250, Urgent.Order<175 and Banking.Orders..3 <60000 to get the outlier free dataframe orders.no.df. Use boxplots to make sure the thresholds make sense to filter outliers)

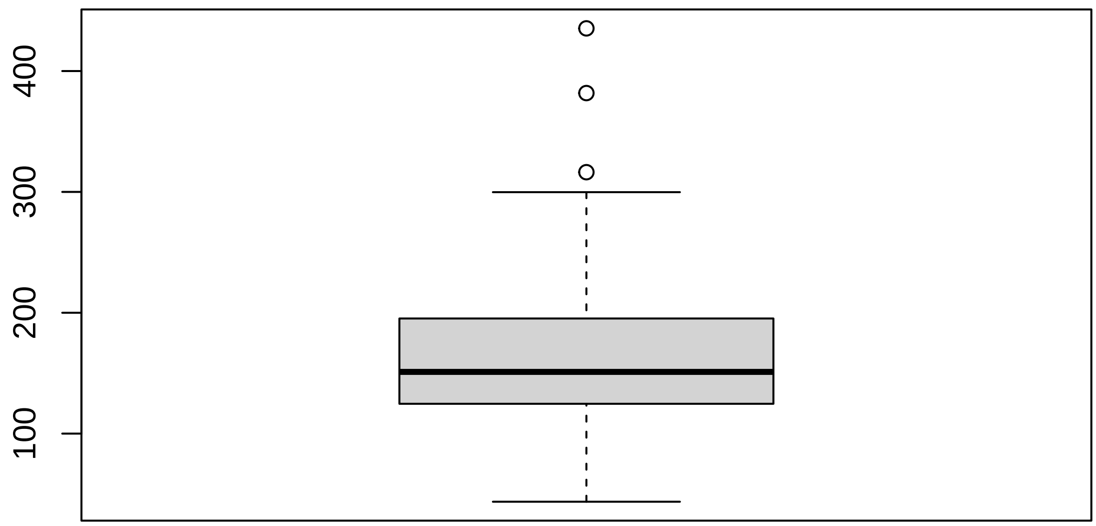
```r
# one of the quickest way to check outliers is through `Boxplots`
# create an outlier function to check all the outliers in the data
outliers <- function(data){
  columns <- colnames(data)
  for (col in columns){
    outlier = boxplot(data[,col])$out
    cat('Outliers for', col, 'are:')
    print(outlier)
  }
}
# call the function
outliers(data = orders.df)
```



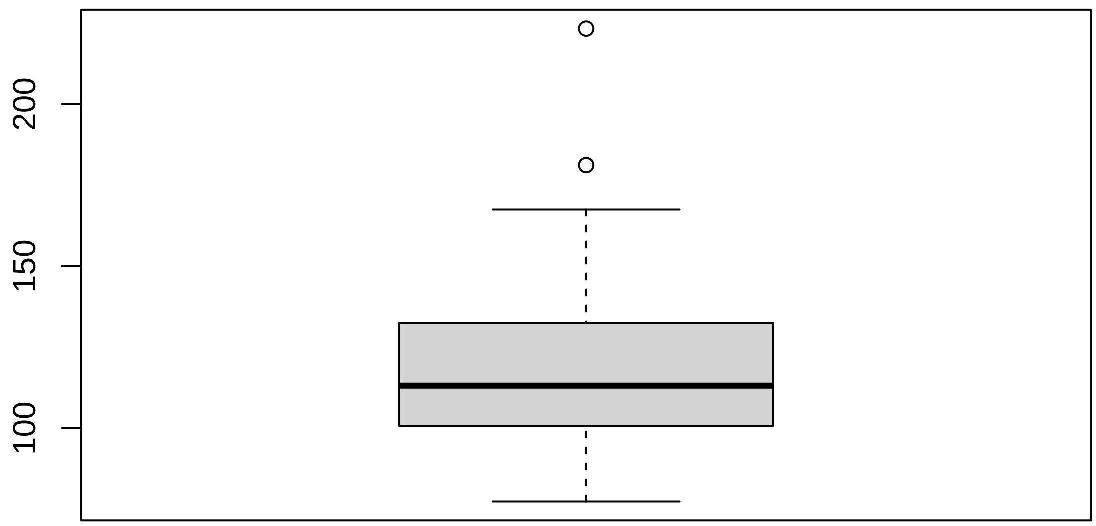

```
## Outliers for Week.of.the.month are:numeric(0)
```



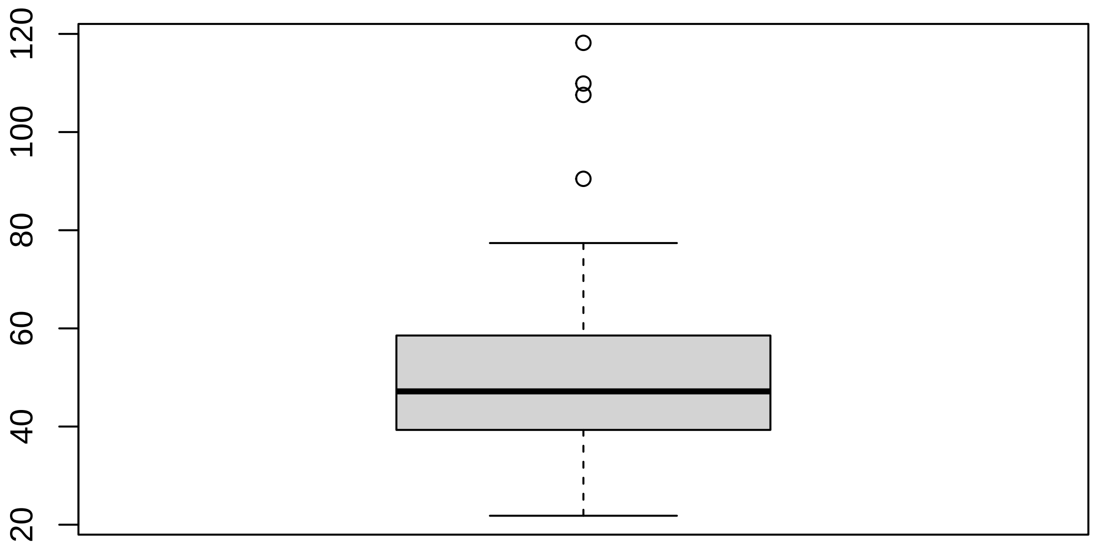

```
## Outliers for Day.of.the.week are:numeric(0)
```



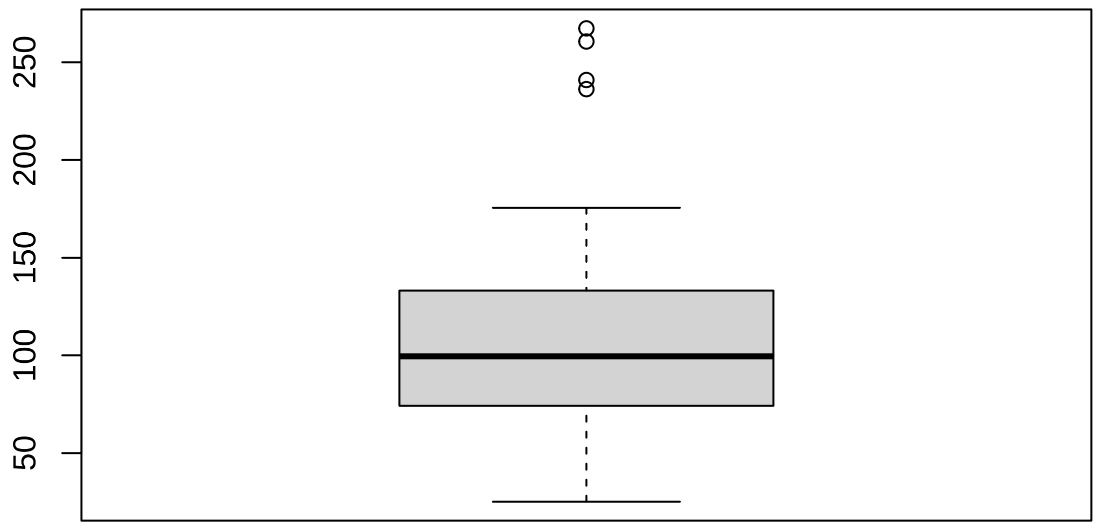

```
## Outliers for Non.urgent are:[1] 316.307 435.304 381.768
```



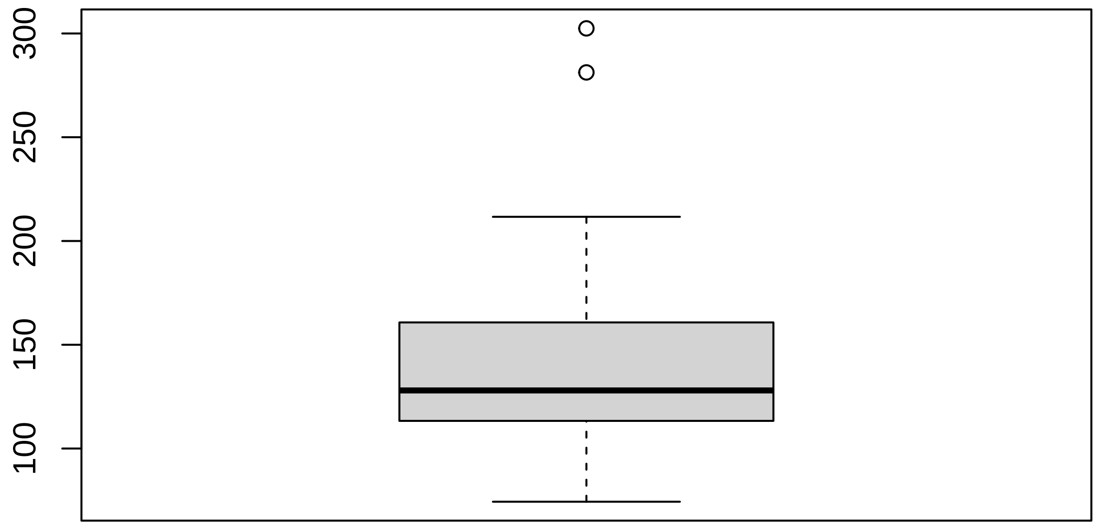

```
## Outliers for Urgent are:[1] 223.270 181.149
```



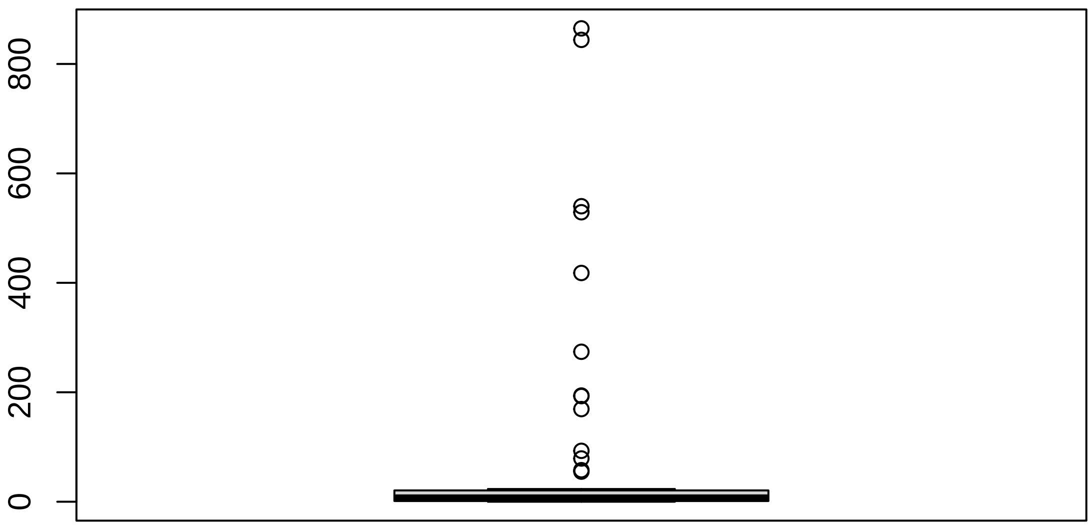

```
## Outliers for TypeA are:[1]  90.476 118.178 109.888 107.568
```



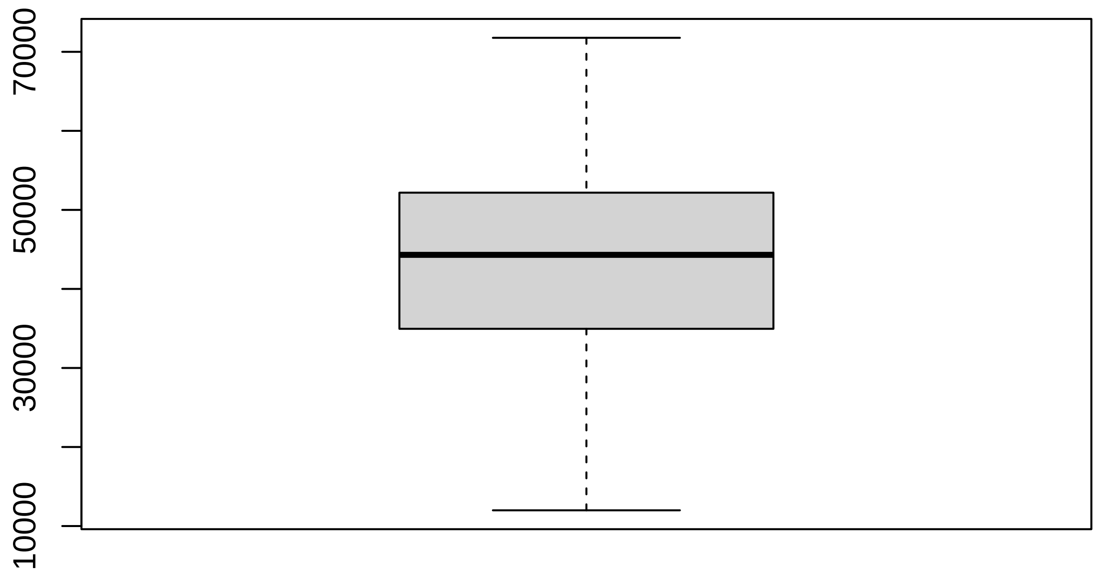

```
## Outliers for TypeB are:[1] 236.248 267.342 260.632 240.922
```



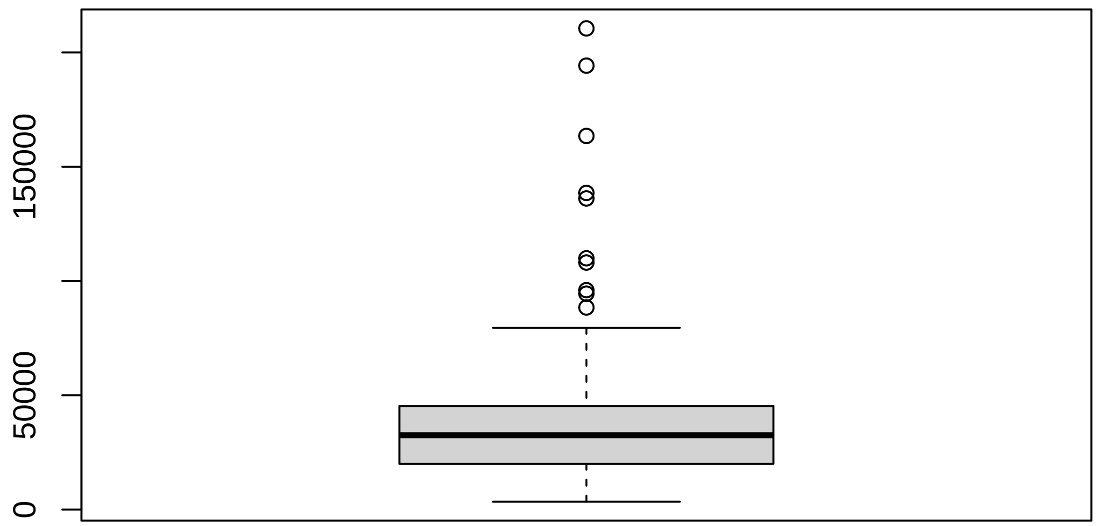

```
## Outliers for TypeC are:[1] 302.448 281.227
```



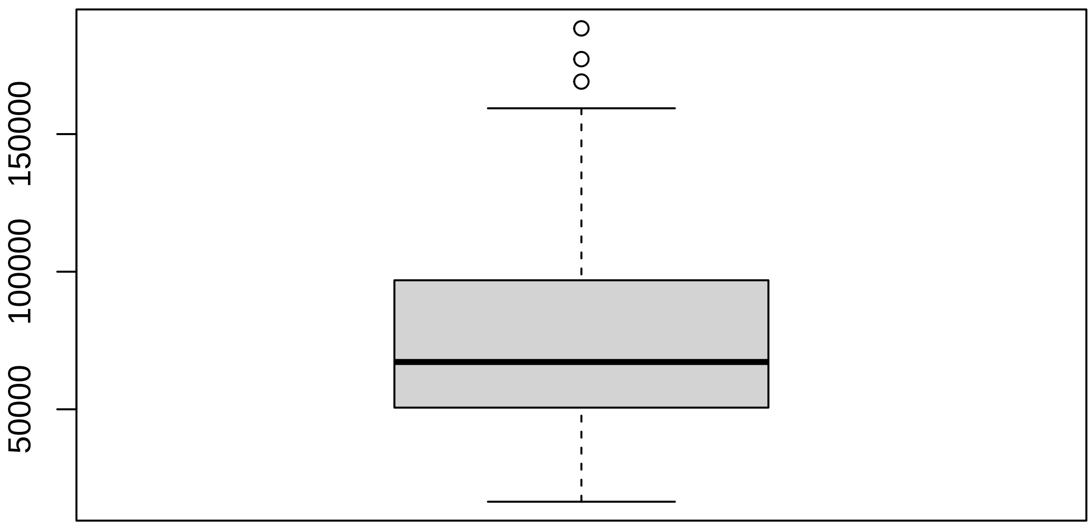
```
## Outliers for Fiscal.sector are: [1]  79.000 865.000 194.000 844.000 193.000  57.645  93.000  55.000 5
## [10] 540.000 418.000 169.275 274.000
```



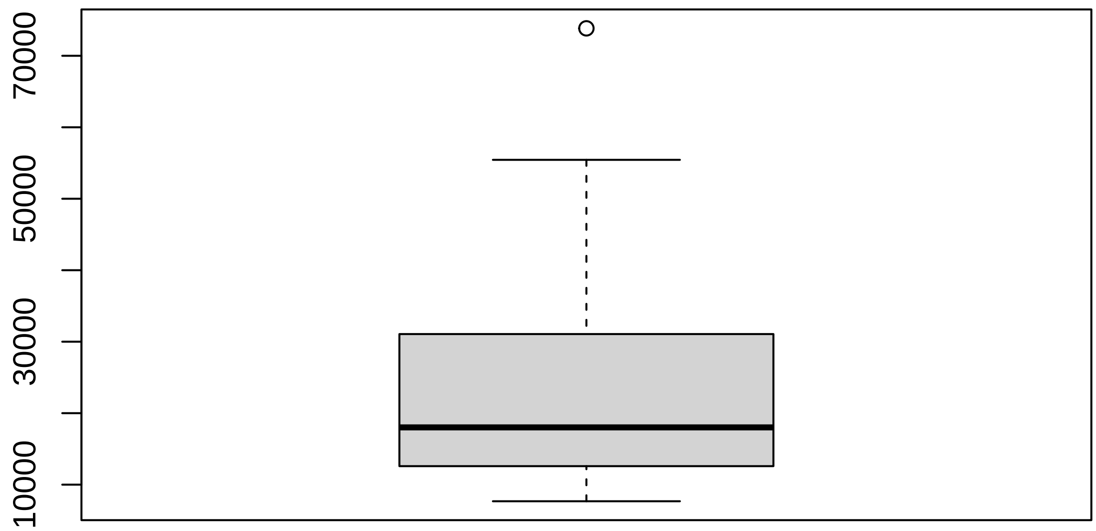

```
## Outliers for Traffic.control.sector are:numeric(0)
```



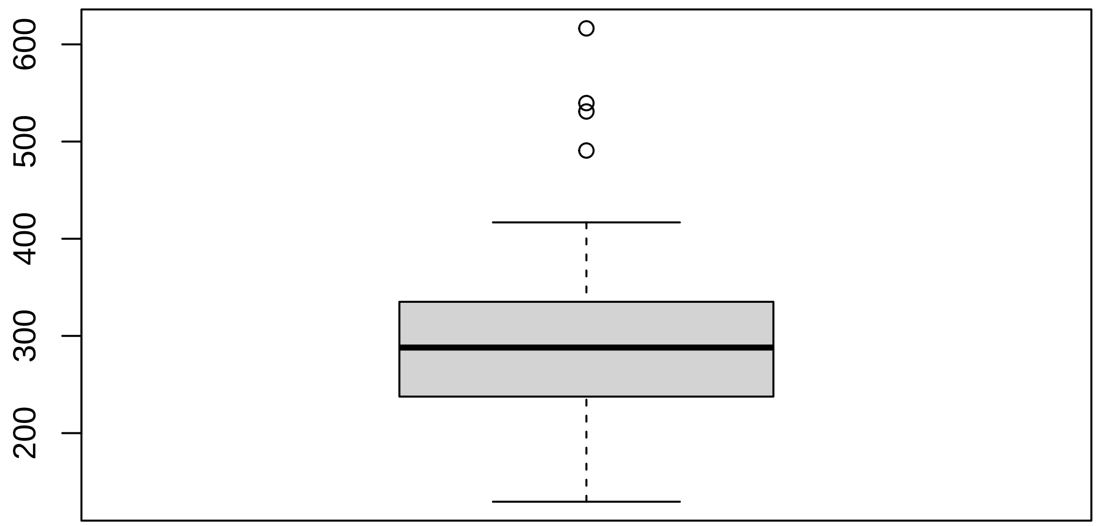

```
## Outliers for Banking.orders1 are: [1] 210508 163452  95989 194216 136119  94460 138536  88404 109931
```




```
## Outliers for Banking.orders2 are:[1] 188411 177229 169088
```




```
## Outliers for Banking.orders3 are:[1] 73839
```




```
## Outliers for Total.orders are:[1] 539.577 490.790 616.453 530.944
```

```r
## Ideally, we can also use z-score standardization for outlier removal but as
## we are just apply the filter using 3 variables. The approach does not suite here.

# filter columns specified:
## Order.Type.C <250, Urgent.Order<175 and Banking.Orders..3 <60000

# Order.Type.C <250
filter1 <- which(orders.df$TypeC > 250)
filter1
```

```
## [1]  1 33
```

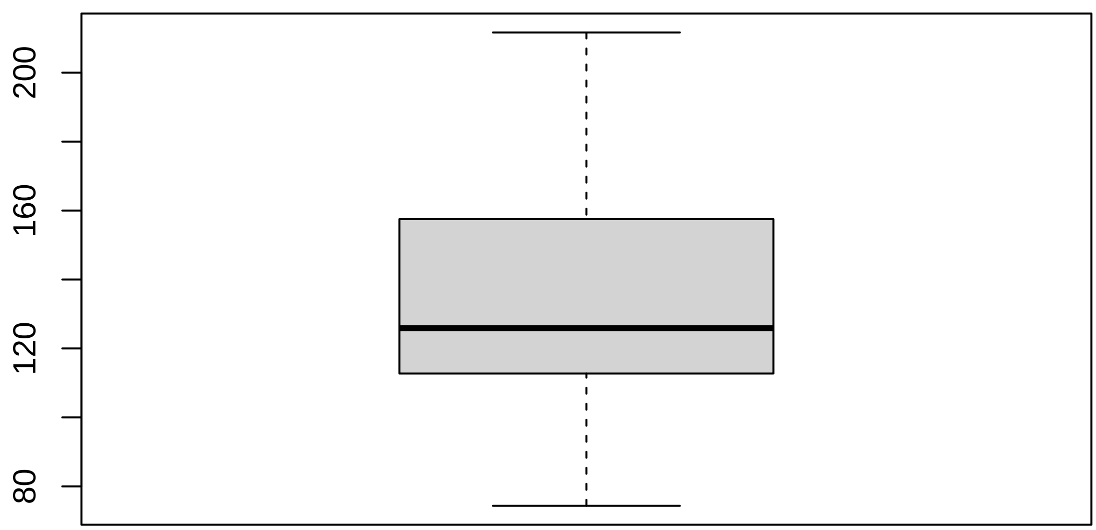
```r
#  Urgent.Order<175
filter2 <- which(orders.df$Urgent > 175)
filter2
```

```
## [1]  1 33
```

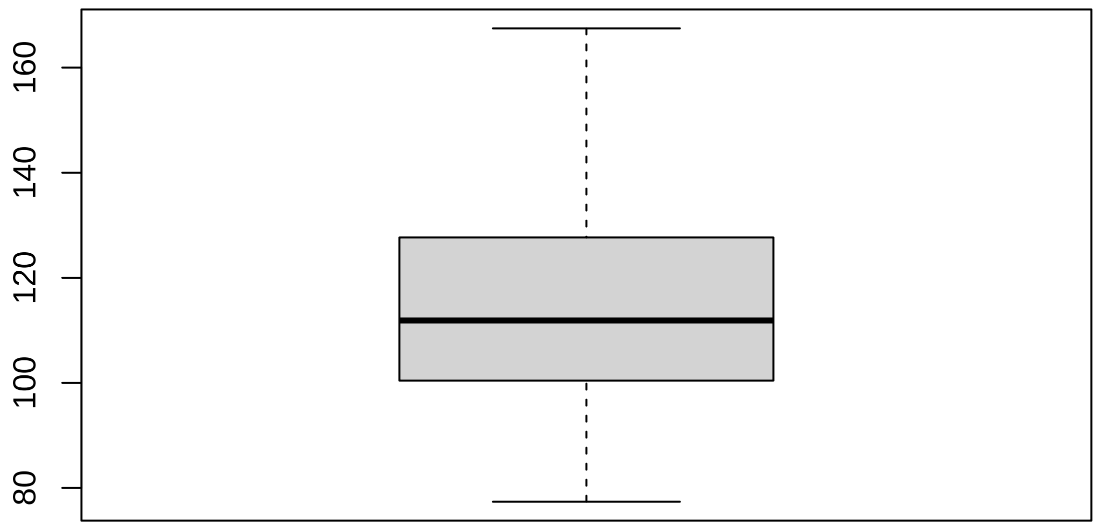
```r
# apply either of the first two filter for the data
orders.no.df <- orders.df[-filter1,]

# Banking.Orders..3 <60000
filter3 <- which(orders.no.df$Banking.orders3 > 60000)
filter3
```

```
## [1] 19
```

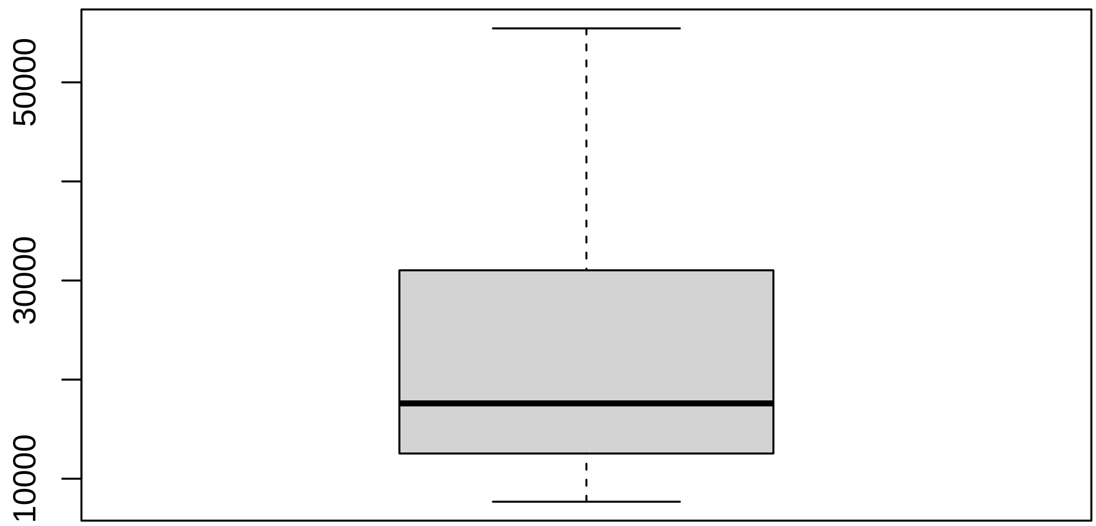
```r
# apply the second filter for the data
orders.no.df <- orders.no.df[-filter3,]

# draw boxplots for variables to check whether our filtering steps were made
# at appropriate threshold
boxplot(orders.no.df$TypeC)
```




```r
boxplot(orders.no.df$Urgent)
```




```
boxplot(orders.no.df$Banking.orders3)
```



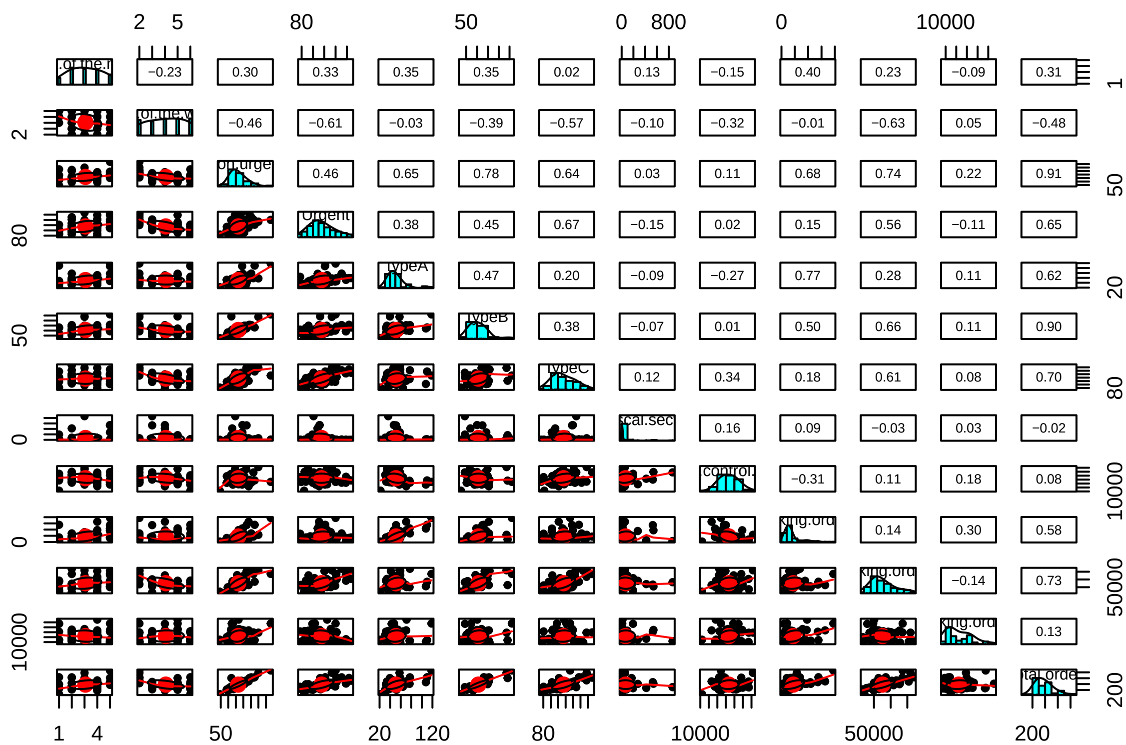

```r
# Check the summary for the limits
summary(orders.no.df)
```

```
##  Week.of.the.month Day.of.the.week   Non.urgent         Urgent
##  Min.   :1.00      Min.   :2.00    Min.   : 43.65   Min.   : 77.37
##  1st Qu.:2.00      1st Qu.:3.00    1st Qu.:123.30   1st Qu.:100.42
##  Median :3.00      Median :4.00    Median :150.78   Median :111.86
##  Mean   :3.07      Mean   :4.07    Mean   :166.15   Mean   :115.16
##  3rd Qu.:4.00      3rd Qu.:5.00    3rd Qu.:193.77   3rd Qu.:127.67
##  Max.   :5.00      Max.   :6.00    Max.   :381.77   Max.   :167.46
##      TypeA             TypeB             TypeC         Fiscal.sector
##  Min.   : 21.83   Min.   : 25.12   Min.   : 74.37   Min.   :  0.000
##  1st Qu.: 39.02   1st Qu.: 72.83   1st Qu.:112.72   1st Qu.:  1.386
##  Median : 46.30   Median : 99.07   Median :125.87   Median :  9.135
##  Mean   : 51.00   Mean   :105.80   Mean   :134.40   Mean   : 66.663
##  3rd Qu.: 57.47   3rd Qu.:130.10   3rd Qu.:157.50   3rd Qu.: 20.057
##  Max.   :118.18   Max.   :260.63   Max.   :211.65   Max.   :865.000
##  Traffic.control.sector Banking.orders1  Banking.orders2  Banking.orders3
##  Min.   :11992          Min.   :  3452   Min.   : 16411   Min.   : 7679
##  1st Qu.:34878          1st Qu.: 20246   1st Qu.: 50433   1st Qu.:12543
##  Median :42737          Median : 32150   Median : 65199   Median :17600
##  Mean   :43496          Mean   : 44279   Mean   : 75791   Mean   :22241
##  3rd Qu.:51235          3rd Qu.: 43333   3rd Qu.: 91784   3rd Qu.:31031
##  Max.   :71772          Max.   :194216   Max.   :169088   Max.   :55445
##   Total.orders
##  Min.   :129.4
```

```
##  1st Qu.:236.3
##  Median :281.4
##  Mean   :291.2
##  3rd Qu.:331.9
##  Max.   :530.9
```

3. Using pairs.panel, what are the distributions of each of the features in the data set with outliers removed (orders.no.df)? Are they reasonably normal so you can apply a statistical learner such as regression? Can you normalize features through a log, inverse, or square-root transform? State which features should be transformed and then transform as needed and build a new data set, orders.tx.

```r
# pairs panel from `psych` package
pairs.panels(orders.no.df)
```



From the pairs.panel plot, we get a comprehensive look on (1) the distribution of the each feature (essentially normality), and (2) the relationship between individual feature with each other and more importantly with our target variable i.e. Total.orders(last column).

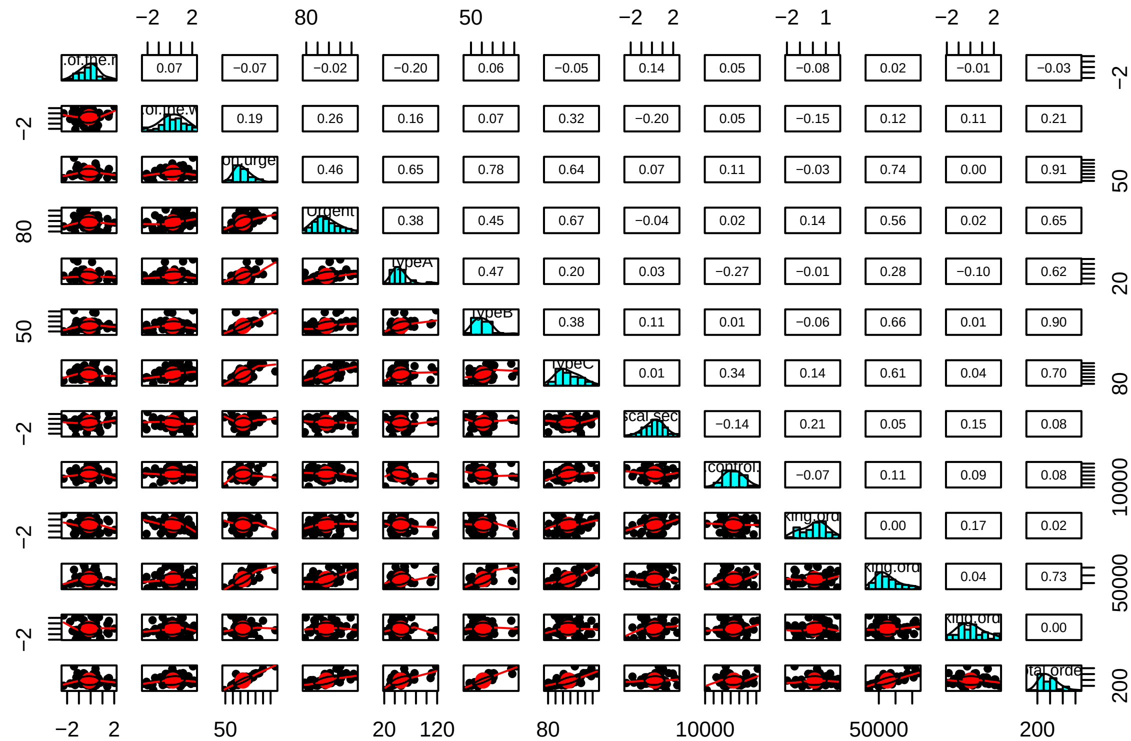
Reflecting from the plot, we are going to drop few variables due to the mulitcollinearity with the target leading to a potential look-ahead bias effect which will mislead our predictive analysis. Features to be eliminated are: `Non.urgent`,`Urgent`, `TypeA`, `TypeB`, `TypeC`

From the plot, it appears that none of the features are reasonably normal except `Banking.orders2`. Hence, it would not be suggestive of applying a statistical learner to the data immediately.

```r
# extract unnormalized predictors
non.normCols <- c("Week.of.the.month", "Day.of.the.week", "Fiscal.sector",
                  "Banking.orders1", "Banking.orders3")

# Create inverse function for normalizing features
inverseTransform <- function(n = 1) {
  # generate 'n' standard uniform samples
  u <- runif(n)
  # pass these samples through our inverse CDF
  x <- qnorm(u)
  # return the new, normally-distributed values
  return(x)
}

# create a function that transform/process the un-normalized column
normCols <- function(df, func, colvar) {
  # apply inverse transform function to columns
  for (idx in colvar) {
    df[,idx] <- func(df[,idx])
  }
  return(df)
}

# get the indices for the unnormalizaed columns (selected from pairs.panel)
colNorm <- which(names(orders.no.df)%in% non.normCols)
colNorm # check
```

```
## [1]  1  2  8 10 12
```

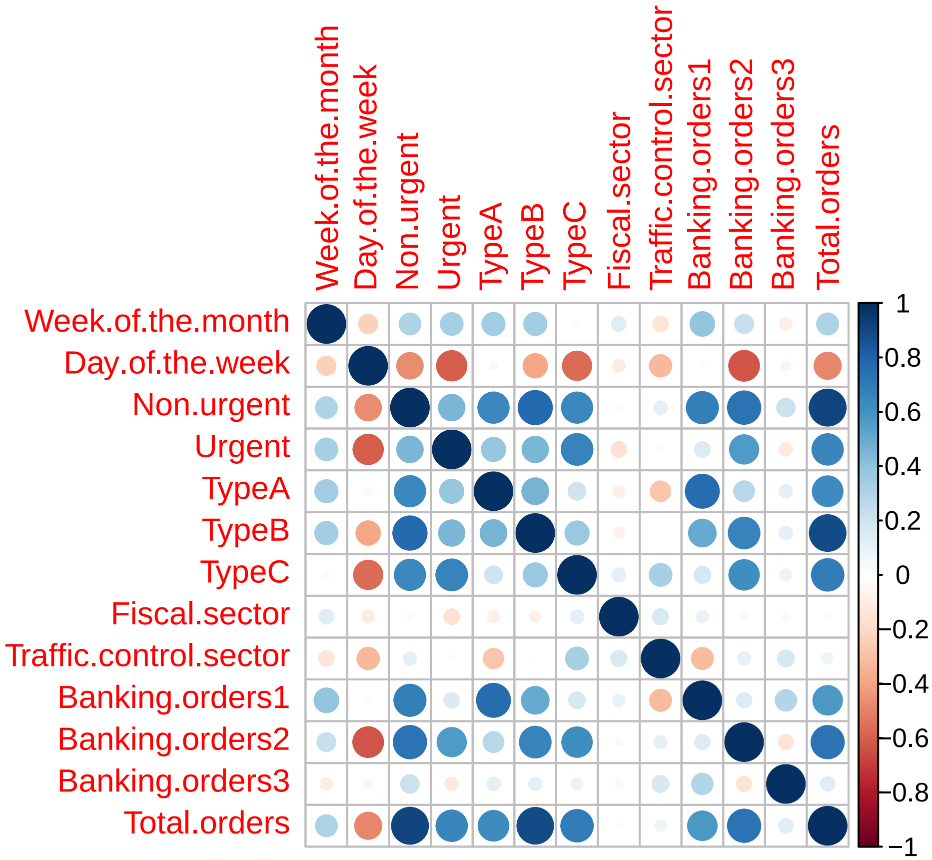
```r
# call the inverse function
orders.tx <- normCols(df = orders.no.df, func = inverseTransform, colvar = colNorm)

# test the inverse standardization of the selected columns using pairs.panels
pairs.panels(orders.tx)
```



From our inverse standardization, it appears that most of our variables have projected a good normality distribution. Therefore, now we can easily apply our statistical learners.

4. What are the correlations to the response variable (Total Orders) for orders.no.df? Are there collinearities? Build a full correlation matrix.

```
# correlation matrix using corrplot()
corrplot::corrplot(cor(orders.no.df))
```



Aforementioned analysis on `pairs.panels` evidently showcased that there are a handful of independent variables that are strongly collinear with the response variable `Total.orders`. Hence, they need to be dropped before regression analysis.

5. Split each of the three data sets, orders.no.df, orders.df, and orders.tx 70%/30% so you retain 30% for testing using random sampling without replacement. Call the data sets, orders.training and orders.testing, orders.no.training and orders.no.testing, and orders.tx.training and orders.tx.testing.

```
set.seed(100)

# Split the data set 70%/30%

# random sampling (pt1) for orders.df
sample1 <- sample(rownames(orders.df), nrow(orders.df) * 0.3, replace = FALSE)
# test (30%)
orders.testing <- orders.df[sample1,]
nrow(orders.testing)
```

```
## [1] 18
```

```
# train (70%)
orders.training <- orders.df[!rownames(orders.df) %in% sample1,]
nrow(orders.training)
```

```
## [1] 42
```

```r
# random sampling (pt2) for orders.no.df
sample2 <- sample(rownames(orders.no.df), nrow(orders.no.df) * 0.3, replace = FALSE)
# test set (30%)
orders.no.testing <- orders.no.df[sample2,]
nrow(orders.no.testing)
```

```
## [1] 17
```

```r
# train set (70%)
orders.no.training <- orders.no.df[!rownames(orders.no.df) %in% sample2,]
nrow(orders.no.training)
```

```
## [1] 40
```

```r
# random sampling (pt3) for orders.tx
sample3 <- sample(rownames(orders.tx), nrow(orders.tx) * 0.3, replace = FALSE)
# test set (30%)
orders.tx.testing <- orders.tx[sample3,]
nrow(orders.tx.testing)
```

```
## [1] 17
```

```r
# train set (70%)
orders.tx.training <- orders.tx[!rownames(orders.tx) %in% sample3,]
nrow(orders.tx.training)
```

```
## [1] 40
```

6. Build three Multiple Regression models for orders.training, orders.no.training, and orders.tx.training using backward elimination based on p-value for predicting Total Orders.

```r
# Multiple regression model with "lm"
## On orders.df
model1 <- lm(Total.orders ~ Week.of.the.month + Day.of.the.week + Fiscal.sector +
               Traffic.control.sector + Banking.orders1 + Banking.orders2 +
               Banking.orders3 , data = orders.training)

summary(model1) # summary stats
```

```
##
## Call:
## lm(formula = Total.orders ~ Week.of.the.month + Day.of.the.week +
##     Fiscal.sector + Traffic.control.sector + Banking.orders1 +
##     Banking.orders2 + Banking.orders3, data = orders.training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -71.301 -20.436  -0.513  15.784 158.372
##
## Coefficients:
```

```
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            75.0685364 56.4644357   1.329   0.1925
## Week.of.the.month      -2.3166775  5.4173181  -0.428   0.6716
## Day.of.the.week         2.8510714  6.3711599   0.447   0.6574
## Fiscal.sector          -0.0081759  0.0466029  -0.175   0.8618
## Traffic.control.sector  0.0012391  0.0005645   2.195   0.0351 *
## Banking.orders1         0.0009910  0.0001688   5.872 1.26e-06 ***
## Banking.orders2         0.0014699  0.0002036   7.220 2.35e-08 ***
## Banking.orders3         0.0002238  0.0005714   0.392   0.6978
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.97 on 34 degrees of freedom
## Multiple R-squared:  0.8669, Adjusted R-squared:  0.8395
## F-statistic: 31.65 on 7 and 34 DF,  p-value: 4.201e-13
```

```
# Using p-value elimination (P < 0.05), we drop the first three variables i.e. week of the
# month, Fiscal sector, and Banking.orders3.

# backward elimination based on p-values, feature selecion from model1
BE_model1 <- lm(Total.orders ~ Day.of.the.week + Traffic.control.sector +
                  Banking.orders1 + Banking.orders2, data = orders.training)

BE1 <- summary(BE_model1)
BE1
```

```
##
## Call:
## lm(formula = Total.orders ~ Day.of.the.week + Traffic.control.sector +
##     Banking.orders1 + Banking.orders2, data = orders.training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -69.276 -22.335   4.741  18.477 155.547
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)            6.502e+01  4.427e+01   1.469   0.1504
## Day.of.the.week        3.613e+00  5.767e+00   0.626   0.5348
## Traffic.control.sector 1.355e-03  5.105e-04   2.655   0.0116 *
## Banking.orders1        9.876e-04  1.404e-04   7.036 2.53e-08 ***
## Banking.orders2        1.463e-03  1.842e-04   7.943 1.63e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 38.56 on 37 degrees of freedom
## Multiple R-squared:  0.8652, Adjusted R-squared:  0.8507
## F-statistic: 59.39 on 4 and 37 DF,  p-value: 1.34e-15
```

```
# The resulting model equation is given below:
# Total.orders = 6502e+01 + 3613e+00(Day.of.the.week) + 1.355e-03(Traffic.control.sector) +
# 9.876e-04(Banking.orders1) + 1.463e-03(Banking.orders2)
```

```r
## orders.no.df
model2 <- lm(Total.orders ~ Week.of.the.month + Day.of.the.week + Fiscal.sector +
               Traffic.control.sector + Banking.orders1 + Banking.orders2 +
               Banking.orders3 , data = orders.no.training)

summary(model2)
```

```
##
## Call:
## lm(formula = Total.orders ~ Week.of.the.month + Day.of.the.week +
##     Fiscal.sector + Traffic.control.sector + Banking.orders1 +
##     Banking.orders2 + Banking.orders3, data = orders.no.training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -53.537 -19.652  -5.443  16.763 158.286
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             2.265e+02  6.924e+01   3.271  0.00257 **
## Week.of.the.month      -3.307e+00  5.606e+00  -0.590  0.55941
## Day.of.the.week        -1.015e+01  6.549e+00  -1.550  0.13109
## Fiscal.sector          -1.583e-02  3.838e-02  -0.412  0.68283
## Traffic.control.sector  1.052e-04  8.710e-04   0.121  0.90461
## Banking.orders1         1.038e-03  1.977e-04   5.249 9.62e-06 ***
## Banking.orders2         8.867e-04  2.800e-04   3.167  0.00338 **
## Banking.orders3        -2.266e-05  6.832e-04  -0.033  0.97375
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 38.29 on 32 degrees of freedom
## Multiple R-squared:  0.7579, Adjusted R-squared:  0.7049
## F-statistic: 14.31 on 7 and 32 DF,  p-value: 2.919e-08
```

```r
# Using p-value elimination (P < 0.05), we drop the first three variables i.e. week of the
# month, Day.of.the.week, Fiscal sector, Traffic.control.sector, and Banking.orders3.

# backward elimination based on p-values, feature selection from model2
BE_model2 <- lm(Total.orders ~ Banking.orders1 + Banking.orders2 ,
                data = orders.no.training)

BE2 <- summary(BE_model2)
BE2
```

```
##
## Call:
## lm(formula = Total.orders ~ Banking.orders1 + Banking.orders2,
##     data = orders.no.training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -61.162 -18.737  -8.634  17.828 148.102
```

```
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.603e+02  1.568e+01  10.218 2.54e-12 ***
## Banking.orders1 9.459e-04  1.405e-04   6.734 6.44e-08 ***
## Banking.orders2 1.177e-03  1.887e-04   6.240 2.97e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 37.34 on 37 degrees of freedom
## Multiple R-squared:  0.7336, Adjusted R-squared:  0.7192
## F-statistic: 50.96 on 2 and 37 DF,  p-value: 2.349e-11
```

```r
# The resulting model equation is given below:
# Total.orders = 1.603e+02 + 9.459e-04(Banking.orders1) + 1.177e-03(Banking.orders2)

## orders.tx
model3 <- lm(Total.orders ~ Week.of.the.month + Day.of.the.week + Fiscal.sector +
               Traffic.control.sector + Banking.orders1 + Banking.orders2 +
               Banking.orders3 , data = orders.tx.training)
summary(model3)
```

```
##
## Call:
## lm(formula = Total.orders ~ Week.of.the.month + Day.of.the.week +
##     Fiscal.sector + Traffic.control.sector + Banking.orders1 +
##     Banking.orders2 + Banking.orders3, data = orders.tx.training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -89.456 -20.517  -8.353  21.517 124.018
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)            1.502e+02  3.514e+01   4.274 0.000161 ***
## Week.of.the.month      2.254e+00  9.388e+00   0.240 0.811776
## Day.of.the.week        1.532e+01  7.753e+00   1.976 0.056853 .
## Fiscal.sector          1.366e+01  9.523e+00   1.434 0.161303
## Traffic.control.sector  4.689e-04  7.137e-04   0.657 0.515849
## Banking.orders1       -4.127e+00  1.022e+01  -0.404 0.689073
## Banking.orders2        1.561e-03  2.237e-04   6.976 6.65e-08 ***
## Banking.orders3       -2.327e+00  8.009e+00  -0.291 0.773234
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 52.42 on 32 degrees of freedom
## Multiple R-squared:  0.6706, Adjusted R-squared:  0.5986
## F-statistic: 9.308 on 7 and 32 DF,  p-value: 3.03e-06
```

```r
# Using p-value elimination (P < 0.05), we drop the first three variables i.e. week of the
# month, Day.of.the.week, Fiscal sector, Traffic.control.sector, Banking.orders1
# and Banking.orders3.
```

```r
# backward elimination
BE_model3 <- lm(Total.orders ~ Banking.orders2,
                data = orders.tx.training)

BE3 <- summary(BE_model3)
BE3
```

```
##
## Call:
## lm(formula = Total.orders ~ Banking.orders2, data = orders.tx.training)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -93.65 -26.65  -8.03  15.41 135.13
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.695e+02  1.845e+01   9.187 3.41e-11 ***
## Banking.orders2 1.664e-03  2.137e-04   7.785 2.19e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 52.04 on 38 degrees of freedom
## Multiple R-squared:  0.6146, Adjusted R-squared:  0.6045
## F-statistic: 60.61 on 1 and 38 DF,  p-value: 2.187e-09
```

```r
#  The resulting model equation is given below:
# Total.orders = 1.695e+02 + 1.664e-03(Banking.orders2)
```

7. Build three Regression Tree models using rpart package for predicting Total Orders: one with orders.training, one with orders.no.training, and one with orders.tx.training.

```r
# Regression tree models with "rpart"
## orders.df
rp_tree1 <- rpart(Total.orders ~ Week.of.the.month + Day.of.the.week + Fiscal.sector +
                     Traffic.control.sector + Banking.orders1 + Banking.orders2 +
                     Banking.orders3 , data = orders.training, method = "anova",
                   control = rpart.control(cp = 0))
sum.rp1 <- summary(rp_tree1)
```

```
## Call:
## rpart(formula = Total.orders ~ Week.of.the.month + Day.of.the.week +
##     Fiscal.sector + Traffic.control.sector + Banking.orders1 +
##     Banking.orders2 + Banking.orders3, data = orders.training,
##     method = "anova", control = rpart.control(cp = 0))
##   n= 42
##
##           CP nsplit rel error    xerror       xstd
## 1 0.62958576      0 1.0000000 1.0541467 0.29243779
## 2 0.11283240      1 0.3704142 0.4584026 0.10713404
## 3 0.03645882      2 0.2575818 0.4399653 0.10102734
## 4 0.00000000      3 0.2211230 0.4178399 0.09108242
```

```
##
## Variable importance
##        Banking.orders2        Day.of.the.week        Banking.orders1
##                     55                     20                     15
## Traffic.control.sector          Fiscal.sector        Banking.orders3
##                      7                      3                      1
##
## Node number 1: 42 observations,    complexity param=0.6295858
##   mean=311.7614, MSE=9719.131
##   left son=2 (35 obs) right son=3 (7 obs)
##   Primary splits:
##       Banking.orders2        < 131791.5 to the left,  improve=0.62958580, (0 missing)
##       Banking.orders1        < 43222.5  to the left,  improve=0.31675950, (0 missing)
##       Day.of.the.week        < 2.5      to the right, improve=0.22349710, (0 missing)
##       Traffic.control.sector < 57969.5  to the left,  improve=0.16111890, (0 missing)
##       Banking.orders3        < 29103    to the left,  improve=0.05848031, (0 missing)
##   Surrogate splits:
##       Day.of.the.week        < 2.5      to the right, agree=0.905, adj=0.429, (0 split)
##       Banking.orders1        < 166376   to the left,  agree=0.881, adj=0.286, (0 split)
##       Traffic.control.sector < 62705    to the left,  agree=0.857, adj=0.143, (0 split)
##
## Node number 2: 35 observations,    complexity param=0.1128324
##   mean=276.7784, MSE=3192.357
##   left son=4 (8 obs) right son=5 (27 obs)
##   Primary splits:
##       Banking.orders2   < 49878    to the left,  improve=0.41222190, (0 missing)
##       Fiscal.sector     < 17.6535  to the left,  improve=0.24677580, (0 missing)
##       Banking.orders1   < 73027    to the left,  improve=0.20401160, (0 missing)
##       Week.of.the.month < 2.5      to the left,  improve=0.13787330, (0 missing)
##       Banking.orders3   < 14483.5  to the right, improve=0.07311415, (0 missing)
##   Surrogate splits:
##       Banking.orders1 < 9604.5   to the left,  agree=0.8, adj=0.125, (0 split)
##
## Node number 3: 7 observations
##   mean=486.676, MSE=5638.843
##
## Node number 4: 8 observations
##   mean=210.1349, MSE=1274.986
##
## Node number 5: 27 observations,    complexity param=0.03645882
##   mean=296.5247, MSE=2054.593
##   left son=10 (17 obs) right son=11 (10 obs)
##   Primary splits:
##       Fiscal.sector          < 17.1935  to the left,  improve=0.26828090, (0 missing)
##       Traffic.control.sector < 39162.5  to the right, improve=0.22332880, (0 missing)
##       Banking.orders1        < 73027    to the left,  improve=0.16682670, (0 missing)
##       Week.of.the.month      < 2.5      to the left,  improve=0.07796941, (0 missing)
##       Banking.orders2        < 63288.5  to the left,  improve=0.06952039, (0 missing)
##   Surrogate splits:
##       Banking.orders1        < 73027    to the left,  agree=0.741, adj=0.3, (0 split)
##       Banking.orders3        < 32811.5  to the left,  agree=0.741, adj=0.3, (0 split)
##       Week.of.the.month      < 1.5      to the right, agree=0.667, adj=0.1, (0 split)
##       Traffic.control.sector < 30195.5  to the right, agree=0.667, adj=0.1, (0 split)
##       Banking.orders2        < 106047.5 to the left,  agree=0.667, adj=0.1, (0 split)
```

```
##
## Node number 10: 17 observations
##   mean=278.518, MSE=1278.473
##
## Node number 11: 10 observations
##   mean=327.136, MSE=1885.736
```

```r
sum.rp1$variable.importance
```

```
##        Banking.orders2        Day.of.the.week        Banking.orders1
##             304545.955             110142.476              83650.426
## Traffic.control.sector          Fiscal.sector        Banking.orders3
##              38202.421              14882.618               4464.785
##      Week.of.the.month
##               1488.262
```

```r
## orders.no.df
rp_tree2 <- rpart(Total.orders ~ Week.of.the.month + Day.of.the.week + Fiscal.sector +
                      Traffic.control.sector + Banking.orders1 + Banking.orders2 +
                      Banking.orders3 , data = orders.no.training, method = "anova",
                    control = rpart.control(cp = 0))

sum.rp2 <- summary(rp_tree2)
```

```
## Call:
## rpart(formula = Total.orders ~ Week.of.the.month + Day.of.the.week +
##     Fiscal.sector + Traffic.control.sector + Banking.orders1 +
##     Banking.orders2 + Banking.orders3, data = orders.no.training,
##     method = "anova", control = rpart.control(cp = 0))
##   n= 40
##
##           CP nsplit rel error    xerror      xstd
## 1 0.39694414      0 1.0000000 1.0358096 0.3207339
## 2 0.13585650      1 0.6030559 1.0864794 0.3363646
## 3 0.05600028      2 0.4671994 0.9909525 0.3042911
## 4 0.00000000      3 0.4111991 0.9670868 0.3026505
##
## Variable importance
##        Day.of.the.week        Banking.orders2        Banking.orders1
##                     40                     25                     14
##      Week.of.the.month        Banking.orders3 Traffic.control.sector
##                      9                      9                      3
##
## Node number 1: 40 observations,    complexity param=0.3969441
##   mean=291.9835, MSE=4843.342
##   left son=2 (33 obs) right son=3 (7 obs)
##   Primary splits:
##       Day.of.the.week        < 2.5      to the right, improve=0.39694410, (0 missing)
##       Banking.orders2        < 95376    to the left,  improve=0.34428420, (0 missing)
##       Banking.orders1        < 67795.5  to the left,  improve=0.33028560, (0 missing)
##       Traffic.control.sector < 39162.5  to the right, improve=0.20008640, (0 missing)
##       Fiscal.sector          < 0.6115   to the left,  improve=0.05642081, (0 missing)
##   Surrogate splits:
```

```
##       Banking.orders2 < 102307.5 to the left,  agree=0.925, adj=0.571, (0 split)
##       Banking.orders3 < 10566.5  to the right, agree=0.850, adj=0.143, (0 split)
##
## Node number 2: 33 observations,    complexity param=0.1358565
##   mean=271.7892, MSE=2532.683
##   left son=4 (26 obs) right son=5 (7 obs)
##   Primary splits:
##       Banking.orders1        < 56115.5  to the left,  improve=0.3149132, (0 missing)
##       Fiscal.sector          < 19.69    to the left,  improve=0.1519487, (0 missing)
##       Banking.orders2        < 49190.5  to the left,  improve=0.1428460, (0 missing)
##       Traffic.control.sector < 39162.5  to the right, improve=0.1354718, (0 missing)
##       Week.of.the.month      < 2.5      to the left,  improve=0.1326699, (0 missing)
##   Surrogate splits:
##       Week.of.the.month      < 4.5      to the left,  agree=0.848, adj=0.286, (0 split)
##       Traffic.control.sector < 30347    to the right, agree=0.818, adj=0.143, (0 split)
##       Banking.orders3        < 33521    to the left,  agree=0.818, adj=0.143, (0 split)
##
## Node number 3: 7 observations
##   mean=387.1853, MSE=4750.529
##
## Node number 4: 26 observations,    complexity param=0.05600028
##   mean=257.1354, MSE=1635.838
##   left son=8 (12 obs) right son=9 (14 obs)
##   Primary splits:
##       Week.of.the.month < 2.5      to the left,  improve=0.25508310, (0 missing)
##       Banking.orders2   < 57948    to the left,  improve=0.17722760, (0 missing)
##       Banking.orders1   < 17270.5  to the left,  improve=0.08418994, (0 missing)
##       Day.of.the.week   < 3.5      to the right, improve=0.06656452, (0 missing)
##       Fiscal.sector     < 49.1615  to the left,  improve=0.04789683, (0 missing)
##   Surrogate splits:
##       Banking.orders2        < 57948    to the left,  agree=0.846, adj=0.667, (0 split)
##       Banking.orders3        < 16696.5  to the right, agree=0.692, adj=0.333, (0 split)
##       Day.of.the.week        < 3.5      to the right, agree=0.654, adj=0.250, (0 split)
##       Traffic.control.sector < 46809.5  to the left,  agree=0.615, adj=0.167, (0 split)
##       Banking.orders1        < 15715    to the left,  agree=0.615, adj=0.167, (0 split)
##
## Node number 5: 7 observations
##   mean=326.2173, MSE=2103.823
##
## Node number 8: 12 observations
##   mean=235.0714, MSE=165.9942
##
## Node number 9: 14 observations
##   mean=276.0474, MSE=2120.765
```

```
sum.rp2$variable.importance
```

```
##        Day.of.the.week        Banking.orders2        Banking.orders1
##              79613.731              51176.444              28128.168
##      Week.of.the.month        Banking.orders3 Traffic.control.sector
##              18369.134              18362.298               5568.187
```

```r
## orders.tx
rp_tree3 <- rpart(Total.orders ~ Week.of.the.month + Day.of.the.week + Fiscal.sector +
                     Traffic.control.sector + Banking.orders1 + Banking.orders2 +
                     Banking.orders3, data = orders.tx.training,
                   method = "anova", control = rpart.control(cp = 0))

sum.rp3 <- summary(rp_tree3)
```

```
## Call:
## rpart(formula = Total.orders ~ Week.of.the.month + Day.of.the.week +
##     Fiscal.sector + Traffic.control.sector + Banking.orders1 +
##     Banking.orders2 + Banking.orders3, data = orders.tx.training,
##     method = "anova", control = rpart.control(cp = 0))
##   n= 40
##
##           CP nsplit rel error    xerror      xstd
## 1 0.46268773      0 1.0000000 1.0675046 0.2681830
## 2 0.16562008      1 0.5373123 0.6927145 0.1612899
## 3 0.03896529      2 0.3716922 0.5451327 0.1154429
## 4 0.00000000      3 0.3327269 0.6006185 0.1259745
##
## Variable importance
##        Banking.orders2      Week.of.the.month          Fiscal.sector
##                     77                      8                      5
## Traffic.control.sector        Banking.orders3        Day.of.the.week
##                      5                      5                      1
##
## Node number 1: 40 observations,    complexity param=0.4626877
##   mean=298.0407, MSE=6675.325
##   left son=2 (33 obs) right son=3 (7 obs)
##   Primary splits:
##       Banking.orders2        < 112928     to the left,  improve=0.46268770, (0 missing)
##       Traffic.control.sector < 40191.5    to the right, improve=0.09812746, (0 missing)
##       Day.of.the.week        < -0.6555019 to the left,  improve=0.09320290, (0 missing)
##       Fiscal.sector          < 0.3039083  to the left,  improve=0.07192063, (0 missing)
##       Week.of.the.month      < 0.192028   to the left,  improve=0.04888122, (0 missing)
##
## Node number 2: 33 observations,    complexity param=0.1656201
##   mean=272.4447, MSE=3392.07
##   left son=4 (10 obs) right son=5 (23 obs)
##   Primary splits:
##       Banking.orders2        < 49278      to the left,  improve=0.39506330, (0 missing)
##       Banking.orders3        < -0.7468151 to the right, improve=0.13488800, (0 missing)
##       Week.of.the.month      < 0.2017352  to the left,  improve=0.11412260, (0 missing)
##       Fiscal.sector          < 0.8318296  to the right, improve=0.10506400, (0 missing)
##       Traffic.control.sector < 40191.5    to the right, improve=0.07165433, (0 missing)
##   Surrogate splits:
##       Week.of.the.month < -1.41675   to the left,  agree=0.788, adj=0.3, (0 split)
##       Fiscal.sector     < 1.079536   to the right, agree=0.758, adj=0.2, (0 split)
##       Banking.orders3   < 2.20419    to the right, agree=0.758, adj=0.2, (0 split)
##
## Node number 3: 7 observations
##   mean=418.7076, MSE=4504.436
```

```
##
## Node number 4: 10 observations
##   mean=216.9272, MSE=1049.591
##
## Node number 5: 23 observations,    complexity param=0.03896529
##   mean=296.5827, MSE=2487.812
##   left son=10 (14 obs) right son=11 (9 obs)
##   Primary splits:
##       Traffic.control.sector < 40191.5    to the right, improve=0.18182980, (0 missing)
##       Week.of.the.month      < 0.0209148  to the left,  improve=0.11246070, (0 missing)
##       Day.of.the.week        < 0.4206382  to the left,  improve=0.08137602, (0 missing)
##       Banking.orders2        < 63288.5    to the left,  improve=0.07978779, (0 missing)
##       Banking.orders3        < -0.6882132 to the right, improve=0.05134182, (0 missing)
##   Surrogate splits:
##       Week.of.the.month < 0.6656738  to the left,  agree=0.739, adj=0.333, (0 split)
##       Banking.orders2   < 67899.5    to the right, agree=0.739, adj=0.333, (0 split)
##       Day.of.the.week   < 0.3335598  to the left,  agree=0.696, adj=0.222, (0 split)
##       Fiscal.sector     < 0.07452339 to the left,  agree=0.696, adj=0.222, (0 split)
##       Banking.orders3   < -0.6882132 to the right, agree=0.652, adj=0.111, (0 split)
##
## Node number 10: 14 observations
##   mean=279.5298, MSE=1467.998
##
## Node number 11: 9 observations
##   mean=323.1094, MSE=2918.163
```

```r
sum.rp3$variable.importance
```

```
##        Banking.orders2      Week.of.the.month          Fiscal.sector
##             171234.429              16734.894              11156.596
## Traffic.control.sector        Banking.orders3        Day.of.the.week
##              10404.240              10000.569               2312.053
```

8. Provide an analysis of all 6 models (using their respective testing data sets), including Adjusted R-Squared and RMSE for train and test sets. Which of these models is the best? Why?

```r
# create the evaluation metrics function
eval_results <- function(actual, predicted, data, p) {
  SSE <- sum((predicted - actual)^2)
  SSR <- sum((actual - mean(actual))^2)
  SST <- SSE + SSR
  # R-Squared value
  R_square <- SSR / SST
  # RMSE
  ## n = Total sample size
  n = nrow(data)
  RMSE = sqrt(SSE/nrow(data))
  # Adjusted R-Squared value
  ## R_square = Sample R-Square
  ## p = Number of independent variable used in the model
  Adj.RSq <- abs(1 - (((1 - R_square) * (nrow(data)-1)) /
                      (nrow(data) - p - 1)))
```

```r
  # Model performance metrics
  data.frame(
    RMSE = RMSE,
    Adj.Rsquare = Adj.RSq
  )

}
```

```r
## orders.df(data1)

# `lm` models (model1)
# predicting and evaluating the model on train data
lm1_pred.train = predict(BE_model1, data = orders.training)
eval_results(actual = orders.training$Total.orders,
             predicted = lm1_pred.train, data = orders.training, p = 4)
```

```
##       RMSE Adj.Rsquare
## 1 36.18988   0.8684089
```

```r
# predicting and evaluating the model on test data
lm1_pred.test = predict(BE_model1, data = orders.testing)
eval_results(actual = orders.testing$Total.orders,
             predicted = lm1_pred.test, data = orders.testing, p = 4)
```

```
## Warning in predicted - actual: longer object length is not a multiple of shorter
## object length

##       RMSE Adj.Rsquare
## 1 179.9183   0.2064761
```

```r
# `rpart` models (model2)
# predicting and evaluating the model on train data
rp1_pred.train = predict(rp_tree1, data = orders.training)
eval_results(actual = orders.training$Total.orders,
             predicted = rp1_pred.train, data = orders.training, p = 7)
```

```
##       RMSE Adj.Rsquare
## 1 46.35864   0.7816368
```

```r
# predicting and evaluating the model on test data
rp1_pred.test = predict(rp_tree1, data = orders.testing)
eval_results(actual = orders.testing$Total.orders,
             predicted = rp1_pred.test, data = orders.testing, p = 7)
```

```
## Warning in predicted - actual: longer object length is not a multiple of shorter
## object length

##       RMSE Adj.Rsquare
## 1 159.2367   0.5355413
```

```r
## orders.no.df (data2)

# `lm` models (model3)
# predicting and evaluating the model on train data
lm2_pred.train = predict(BE_model2, data = orders.no.training)
eval_results(actual = orders.no.training$Total.orders,
             predicted = lm2_pred.train, data = orders.no.training, p = 2)
```

```
##       RMSE Adj.Rsquare
## 1 35.91711      0.7783
```

```r
# predicting and evaluating the model on test data
lm2_pred.test = predict(BE_model2, data = orders.no.testing)
eval_results(actual = orders.no.testing$Total.orders,
             predicted = lm2_pred.test, data = orders.no.testing, p = 2)
```

```
## Warning in predicted - actual: longer object length is not a multiple of shorter
## object length

##       RMSE Adj.Rsquare
## 1 173.4419  0.07132528
```

```r
# `rpart` models (model4)
# predicting and evaluating the model on train data
rp2_pred.train = predict(rp_tree2, data = orders.no.training)
eval_results(actual = orders.no.training$Total.orders,
             predicted = rp2_pred.train, data = orders.no.training, p = 7)
```

```
##      RMSE Adj.Rsquare
## 1 44.6271   0.6448773
```

```r
# predicting and evaluating the model on test data
rp2_pred.test = predict(rp_tree2, data = orders.no.testing)
eval_results(actual = orders.no.testing$Total.orders,
             predicted = rp2_pred.test, data = orders.no.testing, p = 7)
```

```
## Warning in predicted - actual: longer object length is not a multiple of shorter
## object length

##       RMSE Adj.Rsquare
## 1 161.5483   0.4044231
```

```r
## orders.tx (data3)

# `lm` models (model5)
# predicting and evaluating the model on train data
lm3_pred.train = predict(BE_model3, data = orders.tx.training)
eval_results(actual = orders.tx.training$Total.orders,
             predicted = lm3_pred.train, data = orders.tx.training, p = 1)
```

```
##       RMSE Adj.Rsquare
## 1 50.71905   0.7145123
```

```r
# predicting and evaluating the model on test data
lm3_pred.test = predict(BE_model3, data = orders.tx.testing)
eval_results(actual = orders.tx.testing$Total.orders,
             predicted = lm3_pred.test, data = orders.tx.testing, p = 1)
```

```
## Warning in predicted - actual: longer object length is not a multiple of shorter
## object length

##       RMSE Adj.Rsquare
## 1 125.9911  0.06637591
```

```r
# `rpart` models (model6)
# predicting and evaluating the model on train data
rp3_pred.train = predict(rp_tree3, data = orders.tx.training)
eval_results(actual = orders.tx.training$Total.orders,
             predicted = rp3_pred.train, data = orders.tx.training, p = 7)
```

```
##       RMSE Adj.Rsquare
## 1 47.12812   0.6957284
```

```r
# predicting and evaluating the model on test data
rp3_pred.test = predict(rp_tree3, data = orders.tx.testing)
eval_results(actual = orders.tx.testing$Total.orders,
             predicted = rp3_pred.test, data = orders.tx.testing, p = 1)
```

```
## Warning in predicted - actual: longer object length is not a multiple of shorter
## object length

##       RMSE Adj.Rsquare
## 1 127.1957  0.06417566
```

Ideally we used Adjusted R-square value to reliably guide the quality of the model. However, we are comparing regression models which have been transformed in different ways as well as used different sets of observations so we need additional indicator(s) to draw out the final conclusion.

Here, along with Adjusted R-square we can take RMSE into consideration as it provides an absolute measure of the fit. Hence, these two can be chosen as unbiased estimators.

Finally, after comparing both values for each model with respect to the data used, the best model appears to be **model2** which hassignificantly higher R.Adj-Sq and lower RMSE in train and test set among the others.

9. Using each of the regression models, how one unit change in Week of Month translates into the Total Orders prediction? (do not apply backward feature elimination for this part)

```r
# check for 1 unit change of `Week of the month` variable
# Add 1 unit to the vairable
## For orders.df
orders.train.wom <- orders.training
```

```r
orders.train.wom$Week.of.the.month <- orders.train.wom$Week.of.the.month + 1
# Multiple regression (model1)
lm1.wom <- lm(Total.orders ~ Week.of.the.month + Day.of.the.week + Fiscal.sector +
               Traffic.control.sector + Banking.orders1 + Banking.orders2 +
               Banking.orders3 , data = orders.train.wom)
sum.lm1.wom <- summary(lm1.wom)
sum.lm1.wom
```

```
##
## Call:
## lm(formula = Total.orders ~ Week.of.the.month + Day.of.the.week +
##     Fiscal.sector + Traffic.control.sector + Banking.orders1 +
##     Banking.orders2 + Banking.orders3, data = orders.train.wom)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -71.301 -20.436  -0.513  15.784 158.372
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            77.3852140 59.7038983   1.296   0.2037
## Week.of.the.month      -2.3166775  5.4173181  -0.428   0.6716
## Day.of.the.week         2.8510714  6.3711599   0.447   0.6574
## Fiscal.sector          -0.0081759  0.0466029  -0.175   0.8618
## Traffic.control.sector  0.0012391  0.0005645   2.195   0.0351 *
## Banking.orders1         0.0009910  0.0001688   5.872 1.26e-06 ***
## Banking.orders2         0.0014699  0.0002036   7.220 2.35e-08 ***
## Banking.orders3         0.0002238  0.0005714   0.392   0.6978
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.97 on 34 degrees of freedom
## Multiple R-squared:  0.8669, Adjusted R-squared:  0.8395
## F-statistic: 31.65 on 7 and 34 DF,  p-value: 4.201e-13
```

```r
# Regression Tree (model2)
rp1.wom <- rpart(Total.orders ~ Week.of.the.month + Day.of.the.week + Fiscal.sector +
                    Traffic.control.sector + Banking.orders1 + Banking.orders2 +
                    Banking.orders3 , data = orders.train.wom, method = "anova",
                  control = rpart.control(cp = 0))

summary(rp1.wom)$variable.importance
```

```
## Call:
## rpart(formula = Total.orders ~ Week.of.the.month + Day.of.the.week +
##     Fiscal.sector + Traffic.control.sector + Banking.orders1 +
##     Banking.orders2 + Banking.orders3, data = orders.train.wom,
##     method = "anova", control = rpart.control(cp = 0))
##   n= 42
##
##           CP nsplit rel error    xerror      xstd
## 1 0.62958576      0 1.0000000 1.0584634 0.3031883
## 2 0.11283240      1 0.3704142 0.7821784 0.3284132
```

```
## 3 0.03645882      2 0.2575818 0.6991950 0.2859467
## 4 0.00000000      3 0.2211230 0.8003769 0.3312558
##
## Variable importance
##       Banking.orders2        Day.of.the.week        Banking.orders1
##                    55                     20                     15
## Traffic.control.sector          Fiscal.sector        Banking.orders3
##                     7                      3                      1
##
## Node number 1: 42 observations,    complexity param=0.6295858
##   mean=311.7614, MSE=9719.131
##   left son=2 (35 obs) right son=3 (7 obs)
##   Primary splits:
##       Banking.orders2        < 131791.5 to the left,  improve=0.62958580, (0 missing)
##       Banking.orders1        < 43222.5  to the left,  improve=0.31675950, (0 missing)
##       Day.of.the.week        < 2.5      to the right, improve=0.22349710, (0 missing)
##       Traffic.control.sector < 57969.5  to the left,  improve=0.16111890, (0 missing)
##       Banking.orders3        < 29103    to the left,  improve=0.05848031, (0 missing)
##   Surrogate splits:
##       Day.of.the.week        < 2.5      to the right, agree=0.905, adj=0.429, (0 split)
##       Banking.orders1        < 166376   to the left,  agree=0.881, adj=0.286, (0 split)
##       Traffic.control.sector < 62705    to the left,  agree=0.857, adj=0.143, (0 split)
##
## Node number 2: 35 observations,    complexity param=0.1128324
##   mean=276.7784, MSE=3192.357
##   left son=4 (8 obs) right son=5 (27 obs)
##   Primary splits:
##       Banking.orders2   < 49878    to the left,  improve=0.41222190, (0 missing)
##       Fiscal.sector     < 17.6535  to the left,  improve=0.24677580, (0 missing)
##       Banking.orders1   < 73027    to the left,  improve=0.20401160, (0 missing)
##       Week.of.the.month < 3.5      to the left,  improve=0.13787330, (0 missing)
##       Banking.orders3   < 14483.5  to the right, improve=0.07311415, (0 missing)
##   Surrogate splits:
##       Banking.orders1 < 9604.5   to the left,  agree=0.8, adj=0.125, (0 split)
##
## Node number 3: 7 observations
##   mean=486.676, MSE=5638.843
##
## Node number 4: 8 observations
##   mean=210.1349, MSE=1274.986
##
## Node number 5: 27 observations,    complexity param=0.03645882
##   mean=296.5247, MSE=2054.593
##   left son=10 (17 obs) right son=11 (10 obs)
##   Primary splits:
##       Fiscal.sector          < 17.1935  to the left,  improve=0.26828090, (0 missing)
##       Traffic.control.sector < 39162.5  to the right, improve=0.22332880, (0 missing)
##       Banking.orders1        < 73027    to the left,  improve=0.16682670, (0 missing)
##       Week.of.the.month      < 3.5      to the left,  improve=0.07796941, (0 missing)
##       Banking.orders2        < 63288.5  to the left,  improve=0.06952039, (0 missing)
##   Surrogate splits:
##       Banking.orders1        < 73027    to the left,  agree=0.741, adj=0.3, (0 split)
##       Banking.orders3        < 32811.5  to the left,  agree=0.741, adj=0.3, (0 split)
##       Week.of.the.month      < 2.5      to the right, agree=0.667, adj=0.1, (0 split)
```

```
##       Traffic.control.sector < 30195.5  to the right, agree=0.667, adj=0.1, (0 split)
##       Banking.orders2        < 106047.5 to the left,  agree=0.667, adj=0.1, (0 split)
##
## Node number 10: 17 observations
##   mean=278.518, MSE=1278.473
##
## Node number 11: 10 observations
##   mean=327.136, MSE=1885.736
```

```
##        Banking.orders2        Day.of.the.week        Banking.orders1
##             304545.955             110142.476              83650.426
## Traffic.control.sector          Fiscal.sector        Banking.orders3
##              38202.421              14882.618               4464.785
##      Week.of.the.month
##               1488.262
```

```r
## For orders.no.df
orders.no.train.wom <- orders.no.training
orders.no.train.wom$Week.of.the.month <- orders.no.train.wom$Week.of.the.month + 1
# Multiple regression (model3)
lm2.wom <- lm(Total.orders ~ Week.of.the.month + Day.of.the.week + Fiscal.sector +
                   Traffic.control.sector + Banking.orders1 + Banking.orders2 +
                   Banking.orders3 , data = orders.no.train.wom)
sum.lm2.wom <- summary(lm2.wom)
sum.lm2.wom
```

```
##
## Call:
## lm(formula = Total.orders ~ Week.of.the.month + Day.of.the.week +
##     Fiscal.sector + Traffic.control.sector + Banking.orders1 +
##     Banking.orders2 + Banking.orders3, data = orders.no.train.wom)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -53.537 -19.652  -5.443  16.763 158.286
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             2.298e+02  7.140e+01   3.219  0.00295 **
## Week.of.the.month      -3.307e+00  5.606e+00  -0.590  0.55941
## Day.of.the.week        -1.015e+01  6.549e+00  -1.550  0.13109
## Fiscal.sector          -1.583e-02  3.838e-02  -0.412  0.68283
## Traffic.control.sector  1.052e-04  8.710e-04   0.121  0.90461
## Banking.orders1         1.038e-03  1.977e-04   5.249 9.62e-06 ***
## Banking.orders2         8.867e-04  2.800e-04   3.167  0.00338 **
## Banking.orders3        -2.266e-05  6.832e-04  -0.033  0.97375
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 38.29 on 32 degrees of freedom
## Multiple R-squared:  0.7579, Adjusted R-squared:  0.7049
## F-statistic: 14.31 on 7 and 32 DF,  p-value: 2.919e-08
```

```r
# Regression tree (model4)
rp2.wom <- rpart(Total.orders ~ Week.of.the.month + Day.of.the.week + Fiscal.sector +
                   Traffic.control.sector + Banking.orders1 + Banking.orders2 +
                   Banking.orders3 , data = orders.no.train.wom, method = "anova",
                 control = rpart.control(cp = 0))

summary(rp2.wom)$variable.importance
```

```
## Call:
## rpart(formula = Total.orders ~ Week.of.the.month + Day.of.the.week +
##     Fiscal.sector + Traffic.control.sector + Banking.orders1 +
##     Banking.orders2 + Banking.orders3, data = orders.no.train.wom,
##     method = "anova", control = rpart.control(cp = 0))
##   n= 40
##
##           CP nsplit rel error    xerror      xstd
## 1 0.39694414      0 1.0000000 1.0568745 0.3252909
## 2 0.13585650      1 0.6030559 0.8625687 0.2118007
## 3 0.05600028      2 0.4671994 0.8163117 0.2055881
## 4 0.00000000      3 0.4111991 0.7817814 0.2173838
##
## Variable importance
##        Day.of.the.week        Banking.orders2        Banking.orders1
##                     40                     25                     14
##      Week.of.the.month        Banking.orders3 Traffic.control.sector
##                      9                      9                      3
##
## Node number 1: 40 observations,    complexity param=0.3969441
##   mean=291.9835, MSE=4843.342
##   left son=2 (33 obs) right son=3 (7 obs)
##   Primary splits:
##       Day.of.the.week        < 2.5      to the right, improve=0.39694410, (0 missing)
##       Banking.orders2        < 95376    to the left,  improve=0.34428420, (0 missing)
##       Banking.orders1        < 67795.5  to the left,  improve=0.33028560, (0 missing)
##       Traffic.control.sector < 39162.5  to the right, improve=0.20008640, (0 missing)
##       Fiscal.sector          < 0.6115   to the left,  improve=0.05642081, (0 missing)
##   Surrogate splits:
##       Banking.orders2 < 102307.5 to the left,  agree=0.925, adj=0.571, (0 split)
##       Banking.orders3 < 10566.5  to the right, agree=0.850, adj=0.143, (0 split)
##
## Node number 2: 33 observations,    complexity param=0.1358565
##   mean=271.7892, MSE=2532.683
##   left son=4 (26 obs) right son=5 (7 obs)
##   Primary splits:
##       Banking.orders1        < 56115.5  to the left,  improve=0.3149132, (0 missing)
##       Fiscal.sector          < 19.69    to the left,  improve=0.1519487, (0 missing)
##       Banking.orders2        < 49190.5  to the left,  improve=0.1428460, (0 missing)
##       Traffic.control.sector < 39162.5  to the right, improve=0.1354718, (0 missing)
##       Week.of.the.month      < 3.5      to the left,  improve=0.1326699, (0 missing)
##   Surrogate splits:
##       Week.of.the.month      < 5.5      to the left,  agree=0.848, adj=0.286, (0 split)
##       Traffic.control.sector < 30347    to the right, agree=0.818, adj=0.143, (0 split)
##       Banking.orders3        < 33521    to the left,  agree=0.818, adj=0.143, (0 split)
```

```
##
## Node number 3: 7 observations
##   mean=387.1853, MSE=4750.529
##
## Node number 4: 26 observations,    complexity param=0.05600028
##   mean=257.1354, MSE=1635.838
##   left son=8 (12 obs) right son=9 (14 obs)
##   Primary splits:
##       Week.of.the.month < 3.5      to the left,  improve=0.25508310, (0 missing)
##       Banking.orders2   < 57948    to the left,  improve=0.17722760, (0 missing)
##       Banking.orders1   < 17270.5  to the left,  improve=0.08418994, (0 missing)
##       Day.of.the.week   < 3.5      to the right, improve=0.06656452, (0 missing)
##       Fiscal.sector     < 49.1615  to the left,  improve=0.04789683, (0 missing)
##   Surrogate splits:
##       Banking.orders2        < 57948    to the left,  agree=0.846, adj=0.667, (0 split)
##       Banking.orders3        < 16696.5  to the right, agree=0.692, adj=0.333, (0 split)
##       Day.of.the.week        < 3.5      to the right, agree=0.654, adj=0.250, (0 split)
##       Traffic.control.sector < 46809.5  to the left,  agree=0.615, adj=0.167, (0 split)
##       Banking.orders1        < 15715    to the left,  agree=0.615, adj=0.167, (0 split)
##
## Node number 5: 7 observations
##   mean=326.2173, MSE=2103.823
##
## Node number 8: 12 observations
##   mean=235.0714, MSE=165.9942
##
## Node number 9: 14 observations
##   mean=276.0474, MSE=2120.765
```

```
##        Day.of.the.week        Banking.orders2        Banking.orders1
##              79613.731              51176.444              28128.168
##      Week.of.the.month        Banking.orders3 Traffic.control.sector
##              18369.134              18362.298               5568.187
```

```
## For orders.tx (model5)
orders.tx.train.wom <- orders.tx.training
orders.tx.train.wom$Week.of.the.month <- orders.tx.train.wom$Week.of.the.month + 1
# Multiple regression (model3)
lm3.wom <- lm(Total.orders ~ Week.of.the.month + Day.of.the.week + Fiscal.sector +
                Traffic.control.sector + Banking.orders1 + Banking.orders2 +
                Banking.orders3 , data = orders.tx.train.wom)
sum.lm3.wom <- summary(lm3.wom)
sum.lm3.wom
```

```
##
## Call:
## lm(formula = Total.orders ~ Week.of.the.month + Day.of.the.week +
##     Fiscal.sector + Traffic.control.sector + Banking.orders1 +
##     Banking.orders2 + Banking.orders3, data = orders.tx.train.wom)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -89.456 -20.517  -8.353  21.517 124.018
```

```
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             1.479e+02  3.578e+01   4.134  0.00024 ***
## Week.of.the.month       2.254e+00  9.388e+00   0.240  0.81178
## Day.of.the.week         1.532e+01  7.753e+00   1.976  0.05685 .
## Fiscal.sector           1.366e+01  9.523e+00   1.434  0.16130
## Traffic.control.sector  4.689e-04  7.137e-04   0.657  0.51585
## Banking.orders1        -4.127e+00  1.022e+01  -0.404  0.68907
## Banking.orders2         1.561e-03  2.237e-04   6.976 6.65e-08 ***
## Banking.orders3        -2.327e+00  8.009e+00  -0.291  0.77323
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 52.42 on 32 degrees of freedom
## Multiple R-squared:  0.6706, Adjusted R-squared:  0.5986
## F-statistic: 9.308 on 7 and 32 DF,  p-value: 3.03e-06
```

```
# Regression tree (model4)
rp3.wom <- rpart(Total.orders ~ Week.of.the.month + Day.of.the.week + Fiscal.sector +
                   Traffic.control.sector + Banking.orders1 + Banking.orders2 +
                   Banking.orders3 , data = orders.tx.train.wom, method = "anova",
                 control = rpart.control(cp = 0))

summary(rp3.wom)$variable.importance
```

```
## Call:
## rpart(formula = Total.orders ~ Week.of.the.month + Day.of.the.week +
##     Fiscal.sector + Traffic.control.sector + Banking.orders1 +
##     Banking.orders2 + Banking.orders3, data = orders.tx.train.wom,
##     method = "anova", control = rpart.control(cp = 0))
##   n= 40
##
##           CP nsplit rel error    xerror      xstd
## 1 0.46268773      0 1.0000000 1.0687247 0.2715006
## 2 0.16562008      1 0.5373123 0.8174614 0.2070264
## 3 0.03896529      2 0.3716922 0.6194195 0.1537639
## 4 0.00000000      3 0.3327269 0.6353943 0.1582827
##
## Variable importance
##        Banking.orders2      Week.of.the.month          Fiscal.sector
##                     77                      8                      5
## Traffic.control.sector        Banking.orders3        Day.of.the.week
##                      5                      5                      1
##
## Node number 1: 40 observations,    complexity param=0.4626877
##   mean=298.0407, MSE=6675.325
##   left son=2 (33 obs) right son=3 (7 obs)
##   Primary splits:
##       Banking.orders2        < 112928     to the left,  improve=0.46268770, (0 missing)
##       Traffic.control.sector < 40191.5    to the right, improve=0.09812746, (0 missing)
##       Day.of.the.week        < -0.6555019 to the left,  improve=0.09320290, (0 missing)
##       Fiscal.sector          < 0.3039083  to the left,  improve=0.07192063, (0 missing)
##       Week.of.the.month      < 1.192028   to the left,  improve=0.04888122, (0 missing)
```

```
##
## Node number 2: 33 observations,    complexity param=0.1656201
##   mean=272.4447, MSE=3392.07
##   left son=4 (10 obs) right son=5 (23 obs)
##   Primary splits:
##       Banking.orders2        < 49278      to the left,  improve=0.39506330, (0 missing)
##       Banking.orders3        < -0.7468151 to the right, improve=0.13488800, (0 missing)
##       Week.of.the.month      < 1.201735   to the left,  improve=0.11412260, (0 missing)
##       Fiscal.sector          < 0.8318296  to the right, improve=0.10506400, (0 missing)
##       Traffic.control.sector < 40191.5    to the right, improve=0.07165433, (0 missing)
##   Surrogate splits:
##       Week.of.the.month < -0.4167505 to the left,  agree=0.788, adj=0.3, (0 split)
##       Fiscal.sector     < 1.079536   to the right, agree=0.758, adj=0.2, (0 split)
##       Banking.orders3   < 2.20419    to the right, agree=0.758, adj=0.2, (0 split)
##
## Node number 3: 7 observations
##   mean=418.7076, MSE=4504.436
##
## Node number 4: 10 observations
##   mean=216.9272, MSE=1049.591
##
## Node number 5: 23 observations,    complexity param=0.03896529
##   mean=296.5827, MSE=2487.812
##   left son=10 (14 obs) right son=11 (9 obs)
##   Primary splits:
##       Traffic.control.sector < 40191.5    to the right, improve=0.18182980, (0 missing)
##       Week.of.the.month      < 1.020915   to the left,  improve=0.11246070, (0 missing)
##       Day.of.the.week        < 0.4206382  to the left,  improve=0.08137602, (0 missing)
##       Banking.orders2        < 63288.5    to the left,  improve=0.07978779, (0 missing)
##       Banking.orders3        < -0.6882132 to the right, improve=0.05134182, (0 missing)
##   Surrogate splits:
##       Week.of.the.month < 1.665674   to the left,  agree=0.739, adj=0.333, (0 split)
##       Banking.orders2   < 67899.5    to the right, agree=0.739, adj=0.333, (0 split)
##       Day.of.the.week   < 0.3335598  to the left,  agree=0.696, adj=0.222, (0 split)
##       Fiscal.sector     < 0.07452339 to the left,  agree=0.696, adj=0.222, (0 split)
##       Banking.orders3   < -0.6882132 to the right, agree=0.652, adj=0.111, (0 split)
##
## Node number 10: 14 observations
##   mean=279.5298, MSE=1467.998
##
## Node number 11: 9 observations
##   mean=323.1094, MSE=2918.163
```

```
##        Banking.orders2      Week.of.the.month          Fiscal.sector
##             171234.429              16734.894              11156.596
## Traffic.control.sector        Banking.orders3        Day.of.the.week
##              10404.240              10000.569               2312.053
```

After comparing all these model with respect to their standard models for all three data, its quite evident that statistical values are identical (i.e. pvalues, Adj R-square, residuals, important variables, etc.). Therefore, it is fair to say that the unit change in `Week of the month` does not affect the prediction for the response variable.

10. For each of the predictions, calculate the 95% prediction interval for the Total Orders. (Exclude

Regression Trees)

```r
# 95% confidence intervals for predictions of Multiple regression models

# orders.df (model1)
pred.model1 <- predict(BE_model1, newdata = orders.testing,
                       interval = "confidence")
predictions1 <- cbind(pred.model1, "actual"=orders.testing$Total.orders)
predictions1[1:5,]
```

```
##         fit      lwr      upr  actual
## 10 282.9629 262.5410 303.3847 248.428
## 55 256.4698 232.2527 280.6869 213.509
## 38 362.1057 326.9813 397.2301 333.359
## 48 269.2404 248.5567 289.9242 244.235
## 51 380.1353 347.8976 412.3729 342.606
```

```r
# orders.no.df (model2)
pred.model2 <- predict(BE_model2, newdata = orders.no.testing,
                       interval = "confidence")
predictions2 <- cbind(pred.model2, "actual"=orders.no.testing$Total.orders)
predictions2[1:5,]
```

```
##         fit      lwr      upr  actual
## 13 284.4429 271.9323 296.9535 308.178
## 38 367.6027 338.5032 396.7023 333.359
## 54 216.0684 192.5785 239.5583 202.022
## 9  300.3885 281.9000 318.8769 344.291
## 19 374.6300 340.8834 408.3766 404.380
```

```r
# orders.tx.df (model3)
pred.model3 <- predict(BE_model3, newdata = orders.tx.testing,
                       interval = "confidence")
predictions3 <- cbind(pred.model3, "actual"=orders.tx.testing$Total.orders)
predictions3[1:5,]
```

```
##         fit      lwr      upr  actual
## 7  249.7961 228.9440 270.6482 263.043
## 31 380.1677 353.0845 407.2508 298.459
## 48 310.1697 293.2175 327.1218 244.235
## 32 253.3963 233.0937 273.6989 323.603
## 57 315.4103 298.1526 332.6680 286.412
```

From the predictions, we can observe that the intervals estimated are reasonably reliable. Moreover, a certain percentage of our resulting confidence intervals does contain the values of reponse variable.

11. compare this method of forecasting with the time series forecasting methods we have reviewed previously, what is the main difference between them?

Regression forecasting vs Time series forecasting Essentially, regression leading to predictions that are referred to a definitive and specific statement about when and where an event will occur whereas time-series forecast provides a probabilistic statement, usually over a longer time scale requiring leading indicators for estimation. Having said that, the primary difference between regression forecasting and time forecasting is the choice of variables. Diagnostically speaking, regression involves checking the significance of independent variables (relationship with each other and with the dependent variable) and how well it can explain the changes in response variables. While time forecasting focuses largely on estimating future values of the response variable regardless of their relationship with the explanatory variables. In regression, we can extrapolate already built regression model to new subjects not being in the training sample and predict the outcome. However, in forecasting, we usually look at subject's historical data to build model and then predict certain outcome in future based on the same model. Let's talk about the example of Forecasting Bank orders, before building the regression model we eliminate few explanatory variables (such as Non.urgent, Urgent, TypeA, TypeB, TypeC) as they created multi-collinearity which plummets the statistical significance of those independent variable and forecasts a redundant trend for future predictions. If we were to build a time-series forecasting model from the same data, we certainly would not drop those variables and as higher the number of predictors increase the accuracy of forecasting.

12. In this exercise we used regular Multiple Regression, however there is a link function which is suitable to model the count data (e.g. counts of total orders), what is that link function and what the resulting generalized linear model called?

```
link.model <- glm(Total.orders ~ Week.of.the.month + Day.of.the.week + Fiscal.sector +
                    Traffic.control.sector + Banking.orders1 + Banking.orders2 +
                    Banking.orders3,  data=orders.df, family = poisson(link = "log"))
```

```
## Warning in dpois(y, mu, log = TRUE): non-integer x = 539.577000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 224.675000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 129.412000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 317.120000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 210.517000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 207.364000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 263.043000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 248.958000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 344.291000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 248.428000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 281.420000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 243.568000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 308.178000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 363.402000
```

```
## Warning in dpois(y, mu, log = TRUE): non-integer x = 336.872000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 246.992000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 308.880000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 233.126000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 404.380000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 298.560000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 229.249000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 236.304000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 297.174000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 409.401000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 231.035000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 238.826000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 235.598000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 242.112000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 490.790000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 289.657000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 298.459000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 323.603000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 616.453000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 346.035000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 307.645000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 253.847000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 530.944000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 333.359000
```

```
## Warning in dpois(y, mu, log = TRUE): non-integer x = 306.356000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 416.830000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 415.187000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 268.002000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 234.503000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 234.724000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 230.064000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 357.394000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 259.246000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 244.235000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 402.607000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 255.061000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 342.606000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 268.640000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 188.601000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 202.022000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 213.509000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 316.849000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 286.412000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 303.447000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 304.950000

## Warning in dpois(y, mu, log = TRUE): non-integer x = 331.900000
```

```
sum.link <- summary(link.model)
sum.link
```

```
##
## Call:
## glm(formula = Total.orders ~ Week.of.the.month + Day.of.the.week +
##     Fiscal.sector + Traffic.control.sector + Banking.orders1 +
##     Banking.orders2 + Banking.orders3, family = poisson(link = "log"),
##     data = orders.df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.3576  -1.1239  -0.1057   1.0295   9.0387
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)             5.093e+00  6.508e-02  78.260  < 2e-16 ***
## Week.of.the.month       7.913e-04  6.953e-03   0.114    0.909
## Day.of.the.week        -2.778e-03  7.212e-03  -0.385    0.700
## Fiscal.sector          -4.863e-05  4.427e-05  -1.098    0.272
## Traffic.control.sector  2.932e-06  6.921e-07   4.236 2.27e-05 ***
## Banking.orders1         2.420e-06  1.851e-07  13.078  < 2e-16 ***
## Banking.orders2         4.181e-06  2.462e-07  16.977  < 2e-16 ***
## Banking.orders3         9.164e-07  6.776e-07   1.352    0.176
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 1451.00  on 59  degrees of freedom
## Residual deviance:  245.45  on 52  degrees of freedom
## AIC: Inf
##
## Number of Fisher Scoring iterations: 4
```

Link functions are powerful when it comes to problem solving using linear models on non-normal data.Here, we use a log link for modeling. So a log link isn't the same as a log transformation. The transformation changes the raw data. The link function doesn't touch the raw data, instead you can think of it as a transformation of the model for the mean of the raw data. We fitted a Gaussian (=Normal distributed errors) with a log link. The models are fitted via Maximum Likelihood estimation; thus optimal properties of the estimators

Such models can also be called as **Log-Linear model**.

13. Here for simplicity, we first perform the preprocessing and then split the data into train and test, what is a better way to go about this and why?

We did preprocessing before splitting the data because the identical transformations are applied to both the training and test partitions of the data set. Otherwise the test makes no sense.

In production or generally, what will will have is only the historic data, i.e. samples we have seen before. So, we will use the statistics of the historic data. But if you calculate the statistics before the split, test data will affect your preprocessing and it will stop being representative of the real world. So, you have to preprocess the data after the split. If you apply processing before splitting, you may cause information leakage.

Conversely, we can run some basic stationary tests. If we want to do variable reduction, we would do that before run my models, then we can use PCA. Once we have identified important features, that is when we do our splitting and modeling.