Webster University
George Herbert Walker School of Business & Technology

DATA ANALYTICS

TERM PROJECT

R CODES

FALL 2021

CSDA 6010 DATA ANALYTICS PRACTICUM

Case 1: Predicting Corporate Bankruptcy

Case 2: Classifying Book Sales Market

Case 3: Classifying Home Equity Loan Applicants

Developed by: Chau Hai Phuong Nguyen

Instructor: Dr. Ali Ovlia

CONTENTS

CASE 1
PREDICTING CORPORATE BANKRUPTCY

1. SET ENVIRONMENTS

1.1. Load Mandy Nguyen's Customed Functions

```
source("https://raw.githubusercontent.com/mandyhpnguyen/Mandy-Functions/main/M.R-Funcs/general.R")
```

1.2. Load Required Packages

```
pkgs <- c(
  "beepr",
  # General pkgs:
  "tidyverse", "dplyr", "summarytools", "reshape2", "pastecs", "ROSE",
  # Analytics pkgs:
  "forecast", "zoo", "scorecard",
  # Evaluation pkgs:
  "caret", "DT", "lift", "gains", "gt", "cvms","tibble", "fourfoldplot",
  # Visualization pkgs:
  "RColorBrewer", "hrbrthemes", "knitr", "showtext", "sysfonts",
  "ggfortify", "ggplot2", "corrplot", "GGally", "viridis",
  "lattice", "grid", "cowplot", "Amelia"
)
suppressMessages(suppressWarnings(loadpkg(pkgs)))
```

1.3. Set fonts

```
# Load the main font used in the paper to plot charts for standardization

windowsFonts(cambria = windowsFont("Cambria"))
my.par <- function() {
  options(scipen = 999)
  par(mfrow = c(1, 1),
      family = "cambria",
      cex.main = 1.25, cex.lab = 1.25, cex.axis = 1.25)
}
```

1.4. Create Customed Color Scheme

```
pal <- c("cornflowerblue", "orange", "red2", "forestgreen", "mediumorchid1", "tomato3", "cadetblue1")
```

2. COLLECT DATA

```
bankruptcy      <-      read.csv("https://raw.githubusercontent.com/mandyhpnguyen/MS-Data-Analytics-
Datasets/main/Practicum/Bankruptcy.csv")
data <- bankruptcy
```

3. PRE-PROCESS DATA

3.1. Create Variable Objects and Data Set

```
cols                                                                                            <-
c('D','R1','R2','R3','R4','R5','R6','R7','R8','R9','R10','R11','R12','R13','R14','R15','R16','R17',
'R18','R19','R20','R21','R22','R23','R24')
vars                                                                                            <-
c('R1','R2','R3','R4','R5','R6','R7','R8','R9','R10','R11','R12','R13','R14','R15','R16','R17','R18
','R19','R20','R21','R22','R23','R24')
```

```
data2 <- data
data2$STATUS <- factor(ifelse(data$D == 0, "bankrupt", "healthy"),
                       levels = c("bankrupt", "healthy"))
```

3.2. Check for Missing and Null Values

3.2.1. Check Missing Values

```
sum(is.na(data))
sum(data[, 4:27] == 0) # 50
```

3.2.2. Check Null Values

```
zeroset <- data[data[, 4] == 0 |
                      data[, 5] == 0 |
                      data[, 6] == 0 |
                      data[, 7] == 0 |
                      data[, 8] == 0 |
                      data[, 9] == 0 |
                      data[, 10] == 0 |
                      data[, 11] == 0 |
                      data[, 12] == 0 |
                      data[, 13] == 0 |
                      data[, 14] == 0 |
                      data[, 15] == 0 |
                      data[, 16] == 0 |
                      data[, 17] == 0 |
                      data[, 18] == 0 |
                      data[, 19] == 0 |
                      data[, 20] == 0 |
                      data[, 21] == 0 |
                      data[, 22] == 0 |
                      data[, 23] == 0 |
                      data[, 24] == 0 |
                      data[, 25] == 0 |
                      data[, 26] == 0 |
                      data[, 27] == 0,
                ]

View(zeroset)

# Check number of rows with Null values
zeroset %>%
  group_by(YR) %>%
  filter(D == 1) %>%
  nrow()
```

4. ATTRIBUTES ANALYSIS

4.1. Statistical Measures

```
summary(data)
```

4.2. "D" Target Attribute

```
# Distribution of the data indicator in the data set
my.par() %>%  par(mfrow = c(1, 1),
                 mai = par("mai") * 1,
                 oma = c(0,0,0,0) + 0.1,
                 mar = c(4,2,2,0) + 0.1
                 )
D_p <- plot(as.factor(data$D), ylim = c(1, 75),
            xlab = "0: Bankrupt - 1: Healthy",
     col = c("cornflowerblue", "red2"))
text(D_p, y = table(data$D),
     labels = table(data$D),
     pos = 3, cex = 1)
```

4.3. "Year" Attribute

```
# Year Distribution
yr <- data[, 2:3]
table(yr$YR)

yrs <- function(i, j){
  for (i in i:j) {
    print(table(yr[yr$YR == i,]$D))
    i = i + 1
  }
}
yrs(70, 82)

# Plot Year Distribution
my.par() %>%  par(mfrow = c(1, 1),
                 mai = par("mai") * 1,
                 oma = c(0,0,0,0) + 0.1,
                 mar = c(4,2,2,0) + 0.1
                 )
yr_p <- plot(as.factor(data$YR), ylim = c(0, 30),
            xlab = "Year (YY)",
     col = pal)
text(yr_p, y = table(data$YR),
     labels = table(data$YR),
     pos = 3, cex = 1)
```

4.4. Ratios

```
# Generic Function to Create Box Plot

out.bp <- function(dataset) {
  my.par() %>% par(mfrow = c(2, 12),
                 mai = par("mai") * 1,
                 oma = c(0,0,0,0) + 0.1,
                 mar = c(0.5,2.5,5,0) + 0.1
                 )
  i = 2
  for (i in 2:length(colnames(dataset))) {
    boxplot(dataset[, i],
            main = colnames(dataset)[i],
            cex.main = 2, cex.axis = 2,
            col = "red2")
    i = i + 1
  }
}
# Plot All Ratio Attributes
out.bp(data[, -c(1:2)])
```

4.5. Correlation Matrix

```
# Compute Correlation

corr.df <- data[, cols]
corr.mat <- round(cor(corr.df),2)
testRes = cor.mtest(corr.mat, conf.level = 0.95)

# Plot Correlation Matrix
my.par()
corrplot(corr.mat, method = "color",
         type = "full", tl.col = "black")$corrPos -> corrp
text(corrp$x, corrp$y, round(corrp$corr, 2))
```

4.6. Plot Box

```
# Generic Function for Box Plot
bp <- function(i, ylab) {
  ggplot(data = data2, aes(fill = STATUS, y = data2[, i], x = STATUS)) +
    geom_boxplot(alpha = .5, notch = TRUE, notchwidth = .95) +
```

```
    geom_jitter(alpha = .25) +
    theme_classic() + labs(y = ylab) +
    scale_fill_manual(values = c("red2", "cornflowerblue")) +
    theme(text = element_text(size = 12, family = "cambria")) +
    theme(axis.ticks.y = element_blank(),
          axis.title.x = element_blank(),
          legend.position = "none")
}

# Box plot for R9, R10, R17, R20
bpR9 <- bp(12, "R9 (CURASS / CURDEBT)")
bpR10 <- bp(13, "R10 (CURASS / SALES)")
bpR17 <- bp(20, "R17 (INCDEP / ASSETS)")
bpR20 <- bp(21, "R20 (SALES / ASSETS)")
my.par()
cowplot::plot_grid(bpR9, bpR10, bpR17, bpR20, labels = "AUTO", nrow = 1, ncol = 4)
```

5. PARTITION

5.1. Original Dataset

```
# Randomize Data set
set.seed(2021)
index_ran <- sample(seq_len(nrow(data)), size = 1*nrow(data))
data_s <- data[index_ran,]

set.seed(2021)
index_s <- sample(seq_len(nrow(data_s)), size = 0.6*nrow(data_s))
train_s <- data_s[index_s, cols]
test_s <- data_s[-index_s, cols]

# summary(as.factor(train_s$D))
# summary(as.factor(test_s$D))
```

5.2. Normalized Dataset

```
# Normalize Function using Min-max
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}

# Create Normalized Dataset
data_n <- data
data_n[, 4:27] <- as.data.frame(lapply(data[, 4:27], normalize))

set.seed(2021)
index_n <- sample(seq_len(nrow(data_n)), size = 0.6*nrow(data_n))
train_n <- data_n[index_n, cols]
test_n <- data_n[-index_n, cols]

# summary(as.factor(train_n$D))
# summary(as.factor(test_n$D))
```

5.3. Reduced Dataset

```
data_r <- data[, c("D", "R9", "R10", "R17", "R20")]
str(data_r)
set.seed(2021)
index_r <- sample(seq_len(nrow(data_r)), size = 0.6*nrow(data_r))
train_r <- data_r[index_r, ]
test_r <- data_r[-index_r, ]

# summary(as.factor(train_r$D))
# summary(as.factor(test_r$D))
```

5.4. Summary of Data sets

```
# summary(as.factor(train_r$D))
```

```
# summary(as.factor(test_r$D))

## Set 1: Original Dataset & Testing Set
# data_r
# test_r

## Set 2: Original Training Dataset & Testing Set
# train_r
# test_r

## Set 3: Normalized Dataset & Normalized Testing Set
# data_r
# test_r

## Set 4: Normalized Training Dataset & Normalized Testing Set
# data_r
# test_r

## Set 5: Original Training Dataset & Testing Set
# train_r
# test_r

## Set 6: Reduced Training Dataset & Testing Set
# train_r
# test_r
```

6. PREDICTION

6.1. Logistic Regression

6.1.1. Logistic Regression Generic Functions

```
### Model + Evaluation

loreg_m <- function(X, Y){
  # Model
  if (!require("lattice")) install.packages("lattice")
  lg.reg <- glm(D ~ ., data = X, family = "binomial")
  print(summary(lg.reg))
    # Predict
  lg.reg.pred <- predict(lg.reg, Y)
    # Evaluate
  if (!require("caret")) install.packages("caret")
  library(caret)
  confusionMatrix(factor(ifelse(lg.reg.pred > 0.5, 1, 0)), factor(Y$D), positive = '0')
  # plot(lg.reg, which = 1:2)
}

### Stepwise Function

loreg_m_ic <- function(X, Z){
  lg.reg <- glm(D ~ ., data = X, family = "binomial")
  # step(lg.reg, direction = Z)
  summary(step(lg.reg, direction = Z))
}

# New Attributes Sets' Prediction & Evaluation
loreg_m_i <- function(X, Y, a, b, c){
  # 3 new models
  lg.reg_2 <- glm(D ~ ., data = X[, a], family = "binomial")
  lg.reg_3 <- glm(D ~ ., data = X[, b], family = "binomial")
  lg.reg_4 <- glm(D ~ ., data = X[, c], family = "binomial")
  summary(lg.reg_2)
  summary(lg.reg_3)
  summary(lg.reg_4)
  # Predict 3 new models
  lg.reg.pred_2 <- predict(lg.reg_2, Y)
  lg.reg.pred_3 <- predict(lg.reg_3, Y)
```

```
  lg.reg.pred_4 <- predict(lg.reg_4, Y)
  # Evaluate
  library(caret)
  print(confusionMatrix(factor(ifelse(lg.reg.pred_2 > 0.5, 1, 0)), factor(Y$D), positive = '0'))
  # plot(lg.reg_2, which = 1:2)
  print(confusionMatrix(factor(ifelse(lg.reg.pred_3 > 0.5, 1, 0)), factor(Y$D), positive = '0'))
  # plot(lg.reg_3, which = 1:2)
  print(confusionMatrix(factor(ifelse(lg.reg.pred_4 > 0.5, 1, 0)), factor(Y$D), positive = '0'))
  # plot(lg.reg_4, which = 1:2)
}
```

6.1.2. Original Dataset

```
# Model + Evaluation
loreg_m(data_s[, cols], test_s) # 0.8491
# Improved Ratios
loreg_m_ic(data_s[, cols], "backward") # 0.8679
loreg_m_ic(data_s[, cols], "forward") # 0.8491
loreg_m_ic(data_s[, cols], "both") # 0.8679
a_s <- c("D", "R3", "R5", "R6", "R9", "R10", "R16", "R17", "R18", "R22", "R23", "R24")
b_s <- cols
c_s <- a_s
# Improved Performance with new ratios
loreg_m_i(data_s, test_s, a_s, b_s, c_s)
```

6.1.3. Original Training Dataset

```
# Model + Evaluation
loreg_m(train_s[, cols], test_s) # 0.6415
# Improved Ratios
loreg_m_ic(train_s[, cols], "backward") # 0.6604
loreg_m_ic(train_s[, cols], "forward") # 0.6415
loreg_m_ic(train_s[, cols], "both")
a_ts <- c("D", "R2", "R3", "R5", "R9", "R10", "R12", "R14", "R15", "R16", "R19", "R21")
b_ts <- cols
c_ts <- a_ts
# Improved Performance with new ratios
loreg_m_i(train_s, test_s, a_ts, b_ts, c_ts)
```

6.1.4. Normalized Dataset

```
# Model + Evaluation
loreg_m(data_n[, cols], test_n) # 0.9434
# Improved Ratios
loreg_m_ic(data_n[, cols], "backward")
loreg_m_ic(data_n[, cols], "forward")
loreg_m_ic(data_n[, cols], "both")
a_n <- c("D", "R3", "R5", "R6", "R9", "R10", "R16", "R17", "R18", "R22", "R23", "R24") # 0.9057
b_n <- cols
c_n <- a_n
# Improved Performance with new ratios
loreg_m_i(data_n, test_n, a_n, b_n, c_n)
```

6.1.5. Normalized Training Dataset

```
# Model + Evaluation
loreg_m(train_n[, cols], test_n) #0.6415
# Improved Ratios
loreg_m_ic(train_n[, cols], "backward")
loreg_m_ic(train_n[, cols], "forward")
loreg_m_ic(train_n[, cols], "both")
a_tn <- c("D", "R5", "R6", "R7", "R10", "R11", "R12", "R14", "R16", "R17", "R18", "R22", "R24") #
0.6415
b_tn <- cols
c_tn <- a_tn
# Improved Performance with new ratios
loreg_m_i(train_n, test_n, a_tn, b_tn, c_tn)
```

6.1.6. Reduced Dataset

```
loreg_m(data_r, test_r) # 0.9245
# Improved Ratios
loreg_m_ic(data_r, "backward")
loreg_m_ic(data_r, "forward")
loreg_m_ic(data_r, "both")
a_r <- c("D", "R9", "R10", "R17")
b_r <- c("D", "R9", "R10", "R17", "R20")
c_r <- a_r
# Improved Performance with new ratios
loreg_m_i(data_r, test_r, a_r, b_r, c_r) # 0.9245
```

6.1.7. Reduced Training Dataset

```
# Model + Evaluation
loreg_m(train_r, test_r) # 0.8868
# Improved Ratios
loreg_m_ic(train_r, "backward") # 0.9057
loreg_m_ic(train_r, "forward")
loreg_m_ic(train_r, "both")
a_tr <- c("D", "R9", "R10", "R17")
b_tr <- c("D", "R9", "R10", "R17", "R20")
c_tr <- a_r
# Improved Performance with new ratios
loreg_m_i(train_r, test_r, a_tr, b_tr, c_tr)
```

6.2. Neural Networks

6.2.1. Neural Networks Generic Functions

```
# Prediction & Plot Function

nn_f <- function(X, Y, i, j){
  if (!require("neuralnet")) install.packages("neuralnet")
  library(neuralnet)
  # Model
  nn <- neuralnet(D ~ .,
                      data = X,
                      hidden = i,
                      linear.output = FALSE)
  plot(nn)
  # Predict
  nn_pred <- compute(nn, Y[, -1])
  nn_pred_result <- nn_pred$net.result
  # Evaluate
  print(cor(nn_pred_result, Y$D))
  print(confusionMatrix(factor(ifelse(nn_pred_result > j, 1, 0)), factor(Y$D), positive = '0'))
}
```

6.2.2. Original Dataset

```
nn_f(data_s[, cols], test_s, 1, 0.5) # 0.9057
nn_f(data_s[, cols], test_s, 2, 0.5) # 0.8868
nn_f(data_s[, cols], test_s, 3, 0.5) # 0.9623
nn_f(data_s[, cols], test_s, c(1,2), 0.5) # 0.8868
nn_f(data_s[, cols], test_s, c(3,2), 0.5) # 0.9623
```

6.2.3. Original Training Dataset

```
nn_f(train_s[, cols], test_s, 1, 0.5) # 0.7547
nn_f(train_s[, cols], test_s, 2, 0.5) # 0.8113 (run twice)
nn_f(train_s[, cols], test_s, 3, 0.5) # 0.6981
nn_f(train_s[, cols], test_s, c(1,2), 0.5) # 0.6792
nn_f(train_s[, cols], test_s, c(3,2), 0.5) # 0.6415
```

6.2.4. Normalized Dataset

```
nn_f(data_n[, cols], test_n, 1, 0.5) # 0.9623
nn_f(data_n[, cols], test_n, 2, 0.5) # 0.9434
nn_f(data_n[, cols], test_n, 3, 0.5) # 0.9811
```

```
nn_f(data_n[, cols], test_n, c(1,2), 0.5) # 0.9623
nn_f(data_n[, cols], test_n, c(3,2), 0.5) # 0.9623
```

6.2.5. Normalized Training Dataset

```
nn_f(train_n[, cols], test_n, 1, 0.5) # 0.7736 (run 3 times)
nn_f(train_n[, cols], test_n, 2, 0.5) # 0.7358
nn_f(train_n[, cols], test_n, 3, 0.5) # 0.7358
nn_f(train_n[, cols], test_n, c(1,2), 0.5) # 0.7358
nn_f(train_n[, cols], test_n, c(3,2), 0.5) # 0.7547
```

6.2.6. Reduced Dataset

```
nn_f(data_r, test_r, 1, 0.5) # 0.9623
nn_f(data_r, test_r, 2, 0.5) # 0.9434
nn_f(data_r, test_r, 3, 0.5) # 0.9623
nn_f(data_r, test_r, c(1,2), 0.5) # 0.9623
nn_f(data_r, test_r, c(3,2), 0.5) # 0.9245
```

6.2.7. Reduced Training Dataset

```
nn_f(train_r, test_r, 1, 0.5) # 0.9245
nn_f(train_r, test_r, 2, 0.5) # 0.8679
nn_f(train_r, test_r, 3, 0.5) # 0.8113
nn_f(train_r, test_r, c(1,2), 0.5) # 0.9434
nn_f(train_r, test_r, c(3,2), 0.5) # 0.717
```

6.3. k-Nearest Neighbor

6.3.1. Calculate best k

```
numer_of_k <- sqrt(nrow(data_s))
numer_of_k # - 11.5
```

6.3.2. k-Nearest Neighbor Generic Functions

```
# Accuracy table and plot of k

knn_f <- function(X, Y, att, seq_i, seq_j, seq_step) {
  library(caret)
  library(FNN)
  library(class)
  library(gmodels)
  library(e1071)
  library(ggplot2)
  accuracy <- data.frame(k = seq(seq_i, seq_j, seq_step), accuracy = rep(0, 15))
  for(i in seq_i:seq_j) {
    knn.pred <- knn(X[, att],
                    Y[, att],
                    cl = X[, 'D'],
                    k = i)
    accuracy[i, 2] <- confusionMatrix(knn.pred, factor(Y[, 'D']), positive = '0')$overall[1]
  }
  print(accuracy)
  plot(accuracy$k, xlab = "Values of k",
       accuracy$accuracy,
       ylab = "Accuracies",
       main = "Values of k against their accuracies",
       type = 'l',
       cols = 'gray')
}

# Prediction and Evaluation of k = i

knn_e <- function(X, Y, att, i){
  library(caret)
  library(FNN)
```

```
  library(class)
  library(gmodels)
  library(e1071)
  library(ggplot2)
  knn_pred <- knn(X[, att],
                  Y[, att],
                  cl = X[, 'D'],
                  k = i)
  # Evaluate
  CrossTable(Y$D, knn_pred, prop.chisq = F)
  confusionMatrix(factor(knn_pred), factor(Y$D))
}
```

6.3.3. Original Dataset

```
# k table
knn_f(data_s, test_s, vars, 1, 15)
# Model & Predict & Evaluate
knn_e(data_s, test_s, vars, 11)
# Improve
knn_e(data_s, test_s, vars, 1)
knn_e(data_s, test_s, vars, 3)
```

6.3.4. Original Training Dataset

```
# k table
knn_f(train_s, test_s, vars, 1, 15)
# Model & Predict & Evaluate
knn_e(train_s, test_s, vars, 11)
# Improve
knn_e(train_s, test_s, vars, 3)
```

6.3.5. Normalized Dataset

```
# k table
knn_f(data_n, test_n, vars, 1, 15)
# Model & Predict & Evaluate
knn_e(data_n, test_n, vars, 11)
# Improve
knn_e(data_n, test_n, vars, 1)
knn_e(data_n, test_n, vars, 2)
```

6.3.6. Normalized Training Dataset

```
# k table
knn_f(train_n, test_n, vars, 1, 15)
# Model & Predict & Evaluate
knn_e(train_n, test_n, vars, 11)
# Improve
knn_e(train_n, test_n, vars, 1)
```

6.3.7. Reduced Dataset

```
# k table
knn_f(data_r, test_r, -1, 1, 15)
# Model & Predict & Evaluate
knn_e(data_r, test_r, -1, 11)
# Improve
knn_e(data_r, test_r, -1, 1)
knn_e(data_r, test_r, -1, 3)
```

6.3.8. Reduced Training Dataset

```
knn_f(train_r, test_r, -1, 1, 15)
# Model & Predict & Evaluate
knn_e(train_r, test_r, -1, 11)
# Improve
knn_e(train_r, test_r, -1, 4)
```

7. CLEAN ENVIRONMENT

```
rm(list = ls())
cat("\014")
```

CASE 2
CLASSIFYING BOOK SALES MARKET

The **Classifying Book Sales Market** with Charles Book Club data set requires two major classifications approaches including marketing expertise – RFM Analysis – and machine learning algorithms – Logistic Regression and K-Nearest Neighbor. For clarification and simplification, the analysis was written in one document with two separate R scripts for RFM Analysis and Machine Learning. Therefore, in this document, the R codes following for Case 2 are non-related despite similar naming methods. It is advised not to run the two sections at the same time in the same R environment.

RFM Analysis

1. SET ENVIRONMENTS

1.1. Load Mandy's Functions

```
source("https://raw.githubusercontent.com/mandyhpnguyen/Mandy-Functions/main/M.R-Funcs/general.R")
```

1.2. Load Packages

```
pkgs <- c(
  "beepr",
  # General pkgs:
  "tidyverse", "dplyr", "summarytools", "reshape2", "pastecs", "ROSE",
  # Analytics pkgs:
  "forecast", "zoo", "scorecard",
  # Evaluation pkgs:
  "caret", "DT", "lift", "gains", "gt", "cvms","tibble", "fourfoldplot",
  # Visualization pkgs:
  "RColorBrewer", "hrbrthemes", "knitr", "showtext", "sysfonts",
  "ggfortify", "ggplot2", "corrplot", "GGally", "viridis",
  "lattice", "grid", "cowplot", "Amelia"
)
suppressMessages(suppressWarnings(loadpkg(pkgs)))
```

1.3. Set fonts

```
# Load the main font I used in my paper to plot charts for standardization
windowsFonts(cambria = windowsFont("Cambria"))
my.par <- function() {
  options(scipen = 999)
  par(mfrow = c(1, 1),
      family = "cambria",
      cex.main = 1.25, cex.lab = 1.25, cex.axis = 1.25)
}
```

2. COLLECT DATA

```
book <- read.csv("https://raw.githubusercontent.com/mandyhpnguyen/MS-Data-Analytics-
Datasets/main/Data%20Mining/CharlesBookClub.csv")
```

3. PRE-PROCESS DATA

3.1. Create Main Working Data Frame

```
cbc <- book
```

3.2. Compute RFM_score

```
cbc$RFM_score <- paste(cbc$Rcode, cbc$Fcode, cbc$Mcode)
cbc$RFM_score <- gsub(" ", "", cbc$RFM_score)
cbc$RFM_score <- as.factor(cbc$RFM_score)
cbc[1:10, "RFM_score"]
```

3.3. Factor Categorical Attributes

```
cbc[, "Gender"] <- factor(cbc[, "Gender"])
cbc[, "Yes_Florence"] <- factor(cbc[, "Yes_Florence"], levels = c(1, 0))
cbc[, "No_Florence"] <- factor(cbc[, "No_Florence"], levels = c(0, 1))

cbc[, "Mcode"] <- factor(cbc[, "Mcode"])
cbc[, "Rcode"] <- factor(cbc[, "Rcode"])
cbc[, "Fcode"] <- factor(cbc[, "Fcode"])
```

4. PARITION

```
set.seed(2021)
train.index <- sample(1:nrow(cbc), 0.7*nrow(cbc))
train <- cbc[train.index, ]
valid <- cbc[-train.index, ]
```

5. RFM ANALYSIS

5.1. Compute Response Rate

5.1.1. Plot Class Distribution of Target Variable

```
5.1.1.1. Generic Function to Create Bar Plot
# Fix height issue: https://stackoverflow.com/questions/16121903/r-barplot-y-axis-scale-too-short
rr_bar <- function(X, Z=""){
  barplot(height = X, space = .2,
          horiz = FALSE, density = 50, angle = 60,
          col = c("red2", "cornflowerblue"), border = NA,
          ylim = range(pretty(c(0, X))),
          xlab = 'Florence',
          ylab = 'Frequency of Response Rate',
          main = Z
  ) %>%
    text(0, y = X, labels = paste0(round(X*100,2), "%"),
         cex = 1, pos = 3)
}
```

5.1.2. Plot Class Distribution

```
my.par() %>% par(mfrow = c(1, 3))
rr_bar(prop.table(table(cbc$Florence)),
                   "Raw Set")
rr_bar(prop.table(table(train$Florence)),
                   "Training Set")
rr_bar(prop.table(table(valid$Florence)),
                   "Validation Set")
```

5.1.3. Response rate of Whole Data Set

```
toprr_cbc <- cbc %>%
  group_by(RFM_score) %>%
  summarise(Yes = length(Yes_Florence[Yes_Florence == '1']),
            No = length(No_Florence[No_Florence == '1']),
            Response_Rate = round(Yes / (Yes + No), 3),
            Count = length(RFM_score)
  ) %>%
  arrange(desc(Response_Rate)) %>%
  as.data.frame()
toprr_cbc

# Export to .CSV File
setwd("C:/One Drives/OneDrive - Webster University/Webster Classes/21F_CSDA 6010_Practicum")
options(scipen = 0, digits = 2)
toprr_cbc %>% write.csv("toprr_cbc.csv", row.names = TRUE)
```

5.1.4. Response Rate of Training

```
toprr_train <- train %>%
  group_by(RFM_score) %>%
  summarise(Yes = length(Yes_Florence[Yes_Florence == '1']),
            No = length(No_Florence[No_Florence == '1']),
            Response_Rate = round(Yes / (Yes + No), 3),
            Count = length(RFM_score)
  ) %>%
  arrange(desc(Response_Rate)) %>%
  as.data.frame()

# Export to .CSV File
setwd("C:/One Drives/OneDrive - Webster University/Webster Classes/21F_CSDA 6010_Practicum")
options(scipen = 0, digits = 2)
toprr_train %>% write.csv("toprr_train.csv", row.names = TRUE)
```

5.1.5. Response Rate of Validation

```
toprr_valid <- valid %>%
  group_by(RFM_score) %>%
  summarise(Yes = length(Yes_Florence[Yes_Florence == '1']),
            No = length(No_Florence[No_Florence == '1']),
            Response_Rate = round(Yes / (Yes + No), 3),
            Count = length(RFM_score)
  ) %>%
  arrange(desc(Response_Rate)) %>%
  as.data.frame()

# Export to .CSV File
setwd("C:/One Drives/OneDrive - Webster University/Webster Classes/21F_CSDA 6010_Practicum")
options(scipen = 0, digits = 2)
toprr_valid %>% write.csv("toprr_valid.csv", row.names = TRUE)
```

5.2. Evaluate with different cut-offs

5.2.1. Generic Functions

```
# More Comparison
RFM_emore <- function(cut_off, equation="") {
  cat(">> Valid >= Cut-off =", cut_off, "\n",
      ifelse(equation == "", "", paste0("> Equation: ", equation)), "\n\n")
  train_c <- toprr_train
  train_c$Response <- factor(ifelse(train_c$Response_Rate >= cut_off, "Yes", "No"))
  valid_c <- toprr_valid
  valid_c$Response_V <- factor(ifelse(valid_c$Response_Rate >= cut_off, "Yes", "No"))
  index1 <- valid_c$RFM_score
  valid_c$Response_T <- train_c[index1, ]$Response
  valid_c <- valid_c[, c(1, 4:7)]
  confusionMatrix(valid_c$Response_T, valid_c$Response_V)
}
```

```
# Less Comparison
RFM_eless <- function(cut_off, equation="") {
  cat(">> Valid <= Cut-off =", cut_off, "\n",
      ifelse(equation == "", "", paste0("> Equation: ", equation)), "\n\n")
  train_c <- toprr_train
  train_c$Response <- factor(ifelse(train_c$Response_Rate <= cut_off, "Yes", "No"))
  valid_c <- toprr_valid
  valid_c$Response_V <- factor(ifelse(valid_c$Response_Rate <= cut_off, "Yes", "No"))
  index1 <- valid_c$RFM_score
  valid_c$Response_T <- train_c[index1, ]$Response
  valid_c <- valid_c[, c(1, 4:7)]
  confusionMatrix(valid_c$Response_T, valid_c$Response_V)
}
```

5.2.3. Compute Mean of Response Rates

```
mean(toprr_cbc$Response_Rate)
mean(toprr_train$Response_Rate)
```

5.2.4. Compute More Comparison

```
more1 <- RFM_emore(mean(toprr_train$Response_Rate), "mean(toprr_train$Response_Rate)")
more2 <- RFM_emore(mean(toprr_train$Response_Rate)*2, "mean(toprr_train$Response_Rate)*2")
more3 <- RFM_emore(mean(toprr_cbc$Response_Rate), "mean(toprr_cbc$Response_Rate)")
more4 <- RFM_emore(mean(toprr_cbc$Response_Rate)*2, "mean(toprr_cbc$Response_Rate)*2")
```

5.2.5. Compute Less Comparison

```
less1 <- RFM_eless(mean(toprr_train$Response_Rate), "mean(toprr_train$Response_Rate)")
less2 <- RFM_eless(mean(toprr_train$Response_Rate)*2, "mean(toprr_train$Response_Rate)*2")
less3 <- RFM_eless(mean(toprr_cbc$Response_Rate), "mean(toprr_cbc$Response_Rate)")
less4 <- RFM_eless(mean(toprr_cbc$Response_Rate)*2, "mean(toprr_cbc$Response_Rate)*2")
```

5.2.6. Confusion Matrix Results

```
more1
more2
more3
more4
less1
less2
less3
less4
```

5.2.7. Plot Confusion Matrices

```
# Generic Function to Plot Confusion Matrices
rfm_cmp <- function(cm, title) {
  fourfoldplot(cm, color = c("cornflowerblue", "red2"),
               margin = 1, space = 0.3,
               conf.level = 0, main = title)
}

# Plot Confusion Matrices
my.par() %>% par(mfrow = c(2, 4),
                 mai = par("mai") * 1,
                 oma = c(0,0,0,0) + 0.1,
                 mar = c(2,4,2,0) + 0.1
                 )
rfm_cmp(more1$table, "A-1")
rfm_cmp(more2$table, "A-2")
rfm_cmp(more3$table, "A-3")
rfm_cmp(more4$table, "A-4")
rfm_cmp(less1$table, "A-5")
rfm_cmp(less2$table, "A-6")
rfm_cmp(less3$table, "A-7")
rfm_cmp(less4$table, "A-8")
```

6. CLEAN ENVIRONMENT

```
rm(list = ls())
cat("\014")
```

MACHINE LEARNING

1. SET ENVIRONMENTS

1.1. Load Mandy's Functions

```
source("https://raw.githubusercontent.com/mandyhpnguyen/Mandy-Functions/main/M.R-Funcs/general.R")
```

1.2. Load Packages

```
pkgs <- c(
  "beepr",
  # General pkgs:
  "tidyverse", "dplyr", "summarytools", "reshape2", "pastecs", "ROSE",
  # Analytics pkgs:
  "forecast", "zoo", "scorecard",
  # Evaluation pkgs:
  "caret", "DT", "lift", "gains", "gt", "cvms","tibble", "fourfoldplot",
  # Visualization pkgs:
  "RColorBrewer", "hrbrthemes", "knitr", "showtext", "sysfonts",
  "ggfortify", "ggplot2", "corrplot", "GGally", "viridis",
  "lattice", "grid", "cowplot", "Amelia"
)
suppressMessages(suppressWarnings(loadpkg(pkgs)))
```

1.3. Set fonts

```
# Load the main font I used in my paper to plot charts for standardization
windowsFonts(cambria = windowsFont("Cambria"))
my.par <- function() {
  options(scipen = 999)
  par(mfrow = c(1, 1),
      family = "cambria",
      cex.main = 1.25, cex.lab = 1.25, cex.axis = 1.25)
}
```

1.4. Color Scheme

```
pal <- c("cornflowerblue", "orange", "red2", "forestgreen", "mediumorchid1", "tomato3", "cadetblue1")
```

2. COLLECT DATA

```
book <- read.csv("https://raw.githubusercontent.com/mandyhpnguyen/MS-Data-Analytics-Datasets/main/Data%20Mining/CharlesBookClub.csv")
```

3. Pre-Process Data

```
data <- book[, -c(1:2)]
```

3.1. Variables

```
var0 <- c(1:15, 17)
var1 <- c("Rcode", "Fcode", "Mcode", "FirstPurch", "Related.Purchase")
col0 <- c(1:17)
col1 <- c("Florence", "Rcode", "Fcode", "Mcode", "FirstPurch", "Related.Purchase")
```

3.2. Normalize Data

```
set.seed(2021)
options(scipen = 0)
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}
data.n <- data
data.n[, c(2:15, 17)] <- as.data.frame(lapply(data.n[, c(2:15, 17)], normalize))
```

4. Explore Variables

4.1. Target Variable

```
# Plot Distribution
my.par() %>%  par(mfrow = c(1, 1),
                 mai = par("mai") * 1,
                 oma = c(0,0,0,0) + 0.1,
                 mar = c(4,2,2,0) + 0.1
                 )
F_p <- plot(as.factor(book$Florence), ylim = c(0, 4000),
            xlab = "Florence", col = c("cornflowerblue", "red2"))
text(F_p, y = table(book$Florence),
     labels = table(book$Florence),
     pos = 3, cex = 1)
```

4.2. Attributes Summaries

```
# Export to .CSV File
setwd("C:/One Drives/OneDrive - Webster University/Webster Classes/21F_CSDA 6010_Practicum")
options(scipen = 0, digits = 2)
round(stat.desc(book), 0) %>% write.csv("book_descr.csv", row.names = TRUE)
```

4.3. Stepwise Regression

```
# Generic Function
loreg_m_ic <- function(X, Z){
  options(scipen = 0)
  set.seed(2021)
  lg.reg <- glm(Florence ~ ., data = X, family = "binomial")
  # step(lg.reg, direction = Z)
  summary(step(lg.reg, direction = Z))
}

# Fit Data
loreg_m_ic(data[, col0], "both")

# Set of Selected Variables
var2 <- c("Gender", "R", "F", "ChildBks", "YouthBks", "CookBks", "DoItYBks", "RefBks", "ArtBks",
"GeogBks", "ItalArt")
col2 <- c("Florence", "Gender", "R", "F", "ChildBks", "YouthBks", "CookBks", "DoItYBks", "RefBks",
"ArtBks", "GeogBks", "ItalArt")
```

5. Parition

5.1. Raw Data

```
set.seed(2021)
train.index <- sample(1:nrow(data), 0.7*nrow(data))
train <- data[train.index, ]
valid <- data[-train.index, ]

train.n <- data.n[train.index, ]
valid.n <- data.n[-train.index, ]
```

5.2. Balance with ROSE

```
train.rose <- ROSE(Florence ~ ., data = train)$data
train.n.rose <- ROSE(Florence ~ ., data = train.n)$data

table(train.rose$Florence)

# Plot New Distribution
my.par() %>% par(mfrow = c(1, 3),
                 mai = par("mai") * 1,
                 oma = c(0,0,0,0) + 0.1,
                 mar = c(4,4,2,0) + 0.1
                 )

plot(factor(train$Florence),
     ylim = c(0, 3000),
```

```
     xlab = "Florence",
     ylab = "Frequency",
     main = "ImBalanced Training Set",
     col = c("red2", "cornflowerblue")) %>%
text(0, y = table(train$Florence),
     labels = paste(table(train$Florence), "-",
                    round(prop.table(table(train$Florence))*100, 2),"%"),
     pos = 3, cex = 1)

plot(factor(train.rose$Florence),
     ylim = c(0, 3000),
     xlab = "Florence",
     ylab = "Frequency",
     main = "Balanced Training Set",
     col = c("red2", "cornflowerblue")) %>%
text(0, y = table(train.rose$Florence),
     labels = paste(table(train.rose$Florence), "-",
                    round(prop.table(table(train.rose$Florence))*100, 2),"%"),
     pos = 3, cex = 1)

plot(factor(valid$Florence),
     ylim = c(0, 3000),
     xlab = "Florence",
     ylab = "Frequency",
     main = "Validation Set",
     col = c("red2", "cornflowerblue")) %>%
text(0, y = table(valid$Florence),
     labels = paste(table(valid$Florence), "-",
                    round(prop.table(table(valid$Florence))*100, 2),"%"),
     pos = 3, cex = 1)
```

6. ANALYSIS

6.1. Logistic Regression (LR)

6.1.1. LR Data Sets

```
train.lr <- train
train.lr.rose <- train.rose
valid.lr <- valid

train.n.lr <- train.n
train.n.lr.rose <- train.n.rose
valid.n.lr <- valid.n
```

6.1.2. Generic Function for LR Fitting and Confusion Matrices Computing

```
loreg_m <- function(X, Y, cutoff){
  options(scipen = 0)
  set.seed(2021)
  # Model
  if (!require("lattice")) install.packages("lattice")
  lg.reg <- glm(Florence ~ ., data = X, family = "binomial")
  # print(summary(lg.reg))
    # Predict
  lg.reg.pred <- predict(lg.reg, Y)
    # Evaluate
  if (!require("caret")) install.packages("caret")
  library(caret)
  confusionMatrix(factor(ifelse(lg.reg.pred > cutoff, 1, 0)), factor(Y$Florence), positive = "0")
  # plot(lg.reg, which = 1:2)
}
```

6.1.3. Imbalanced Full

```
lr_cm_i0 <- loreg_m(train.n.lr[, col0], valid.n.lr[, col0], 0.5)
```

6.1.4. Balanced Full

```
lr_cm_b0 <- loreg_m(train.n.lr.rose[, col0], valid.n.lr[, col0], 0.5)

=> Two predictor variables are perfectly correlated.
```

6.1.5. Imbalanced Set 1

```
lr_cm_i1 <- loreg_m(train.n.lr[, col1], valid.n.lr[, col1], 0.5)
```

6.1.6. Balanced Set 1

```
lr_cm_b1 <- loreg_m(train.n.lr.rose[, col1], valid.n.lr[, col1], 0.5)
```

6.1.7. Imbalanced Set 2

```
lr_cm_i2 <- loreg_m(train.n.lr[, col2], valid.n.lr[, col2], 0.5)
```

6.1.8. Balanced Set 2

```
lr_cm_b2 <- loreg_m(train.n.lr.rose[, col2], valid.n.lr[, col2], 0.5)
```

6.1.9. LR Confusion Matrices

```
options(scipen = 0)
lr_cm_i0
lr_cm_i1
lr_cm_i2
lr_cm_b0
lr_cm_b1
lr_cm_b2
```

6.1.10. LR CM Plot

```
# Generic Function
lr_cmp <- function(cm, title) {
  fourfoldplot(cm, color = c("cornflowerblue", "red2"),
               margin = 1, space = 0.3,
               conf.level = 0, main = title)
}

# Plot
my.par() %>% par(mfrow = c(2, 3),
                 mai = par("mai") * 1,
                 oma = c(0,0,0,0) + 0.1,
                 mar = c(2,4,2,0) + 0.1
                 )
lr_cmp(lr_cm_i0$table, "Model B")
lr_cmp(lr_cm_i1$table, "Model C")
lr_cmp(lr_cm_i2$table, "Model D")
lr_cmp(lr_cm_b0$table, "Model E")
lr_cmp(lr_cm_b1$table, "Model F")
lr_cmp(lr_cm_b2$table, "Model G")
```

6.2. k-Nearest Neighbors

6.2.1. k-Nearest Neighbors (KNN) Data Sets

```
train.knn <- train
train.knn.rose <- train.rose
valid.knn <- valid
```

6.2.2. Optimal K

```
# Compute Optimal K
k_opt <- sqrt(nrow(train.knn))/2
k_opt

# Generic Function to automatically compute k's
knn_ks <- function(X, Y, att, seq_i=1, seq_j=k_opt, seq_step=1) {
  library(caret); library(FNN); library(class)
```

```
  library(gmodels); library(e1071); library(ggplot2)
  options(scipen = 0)
  set.seed(2021)
  accuracy <- data.frame(k = seq(seq_i, seq_j, seq_step), accuracy = rep(0, k_opt))
  for (i in seq_i:seq_j) {
    knn.pred <- knn(X[, att], Y[, att],
                    cl = X[, "Florence"], k = i)
    accuracy[i, 2] <- confusionMatrix(knn.pred,
                                      factor(Y[, "Florence"]),
                                      positive = '0')$overall[1]
  }
  accuracy
}
```

6.2.3. k's Accuracies

```
knn_i0 <- knn_ks(train.knn, valid.knn, var0) #5
knn_b0 <- knn_ks(train.knn.rose, valid.knn[, -1], var0) #21
knn_i1 <- knn_ks(train.knn[, col1], valid.knn[, col1], var1) #3
knn_b1 <- knn_ks(train.knn.rose[, col1], valid.knn[, col1], var1) #13
knn_i2 <- knn_ks(train.knn[, col2], valid.knn[, col2], var2) #9
knn_b2 <- knn_ks(train.rose[, col2], valid[, col2], var2) #9
```

6.2.4. Plot k's Accuracies

```
my.par() %>% par(mai = par("mai") * 1,
                 oma = c(0,0,0,0) + 0.1,
                 mar = c(4,4,0,0) + 0.1
                 )
plot(knn_i0, bty = "n", type = "n",
     xlim = c(0, 30), ylim = c(0.5, 1.05),
     xlab = "k", ylab = "Accuracy",
     yaxt = "n")
axis(2, at = seq(0.5, 1.0, 0.1),
     labels = format(seq(0.5, 1.0, 0.1), digits = 2))
lines(knn_i0, type = "b", pch = 18, lwd = 2, col = pal[1])
lines(knn_i1, type = "b", pch = 18, lwd = 2, col = pal[2])
lines(knn_i2, type = "b", pch = 18, lwd = 2, col = pal[3])
lines(knn_b0, type = "b", pch = 18, lwd = 2, col = pal[4])
lines(knn_b1, type = "b", pch = 18, lwd = 2, col = pal[5])
lines(knn_b2, type = "b", pch = 18, lwd = 2, col = pal[6])
legend("top", bty = "n",
       legend = c("H", "I", "J", "K", "L", "M"),
       title = "MODELS", cex = 1.5,
       col = pal[1:6], horiz = TRUE,
       lty = 1, lwd = 2, pch = 18
       )
```

6.2.5. KNN Confusion Matrices

```
# Generic Function
knn_e <- function(X, Y, att, i){
  library(caret); library(FNN); library(class)
  library(gmodels); library(e1071); library(ggplot2)
  options(scipen = 0)
  set.seed(2021)
  knn_pred <- knn(X[, att], Y[, att], cl = X[, "Florence"], k = i)
  confusionMatrix(factor(knn_pred),
                  factor(Y$Florence),
                  positive = "0")
}

# Compute Confusion Matrices
knn_e_i0 <- knn_e(train.knn, valid.knn, col0, 5) #5
knn_e_b0 <- knn_e(train.knn.rose, valid.knn, col0, 21) #21
knn_e_i1 <- knn_e(train.knn[, col1], valid.knn[, col1], var1, 3) #3
knn_e_b1 <- knn_e(train.knn.rose[, col1], valid.knn[, col1], var1, 13) #13
knn_e_i2 <- knn_e(train.knn[, col2], valid.knn[, col2], var2, 9) #9
knn_e_b2 <- knn_e(train.rose[, col2], valid[, col2], var2, 9) #9
```

```
knn_e_i0
knn_e_i1
knn_e_i2
knn_e_b0
knn_e_b1
knn_e_b2
```

6.2.6. KNN Confusion Matrices Plot

```
# Generic Function
knn_cmp <- function(cm, title) {
  fourfoldplot(cm, color = c("cornflowerblue", "red2"),
               margin = 1, space = 0.3,
               conf.level = 0, main = title)
}

# Plot
my.par() %>% par(mfrow = c(2, 3),
                 mai = par("mai") * 1,
                 oma = c(0,0,0,0) + 0.1,
                 mar = c(2,4,2,0) + 0.1
                 )
knn_cmp(knn_e_i0$table, "Model H: k = 5")
knn_cmp(knn_e_i1$table, "Model I: k = 21")
knn_cmp(knn_e_i2$table, "Model J: k = 3")
knn_cmp(knn_e_b0$table, "Model K: k = 13")
knn_cmp(knn_e_b1$table, "Model L: k = 9")
knn_cmp(knn_e_b2$table, "Model M: k = 9")
```

7. Clean Environment

```
rm(list = ls())
cat("\014")
```

CASE 3
CLASSIFYING HOME EQUITY LOAN APPLICANTS

1. SET ENVIRONMENTS

1.1. Load Mandy Nguyen's Customed Functions

```
source("https://raw.githubusercontent.com/mandyhpnguyen/Mandy-Functions/main/M.R-Funcs/general.R")
```

1.2. Load Required Packages

```
pkgs <- c(
  "beepr",
  # General pkgs:
  "tidyverse", "dplyr", "summarytools", "reshape2", "pastecs", "ROSE",
  # Analytics pkgs:
  "forecast", "zoo", "scorecard",
  # Evaluation pkgs:
  "caret", "DT", "lift", "gains", "gt", "cvms","tibble", "fourfoldplot",
  # Visualization pkgs:
  "RColorBrewer", "hrbrthemes", "knitr", "showtext", "sysfonts",
  "ggfortify", "ggplot2", "corrplot", "GGally", "viridis",
  "lattice", "grid", "cowplot", "Amelia"
)
suppressMessages(suppressWarnings(loadpkg(pkgs)))
```

1.3. Set fonts

```
# Load the main font used in the paper to plot charts for standardization

windowsFonts(cambria = windowsFont("Cambria"))
my.par <- function() {
  options(scipen = 999)
  par(mfrow = c(1, 1),
      family = "cambria",
      cex.main = 1.25, cex.lab = 1.25, cex.axis = 1.25)
}
```

1.4. Create Customed Color Scheme

```
pal <- c("cornflowerblue", "orange", "red2", "forestgreen", "mediumorchid1", "tomato3", "cadetblue1")
```

2. COLLECT DATA

2.1. Load Data Set Stored Online in My Github Repository

```
hmeq         <-         read.csv("https://raw.githubusercontent.com/mandyhpnguyen/MS-Data-Analytics-Datasets/main/Practicum/hmeq.csv")
```

2.2. Overview Data

```
View(hmeq)
str(hmeq)
datatable(hmeq)
```

2.3. Export Detailed Statistic Summary to .CSV File

```
setwd("~/~")
options(scipen = 0, digits = 2)
round(stat.desc(hmeq), 0) %>% write.csv("hmeq_descr.csv", row.names = TRUE)
```

3. DATA CLEANING AND HARMONIZATION

3.1. Create Working Data Frame and Character Version of Target Attribute

```
data <- hmeq
data$STATUS <- factor(ifelse(data$BAD == 1, "defaulted", "paid"),
                      levels = c("defaulted", "paid"),)
str(data)
View(data)
```

3.2. Check Missing Values

```
# Total Missing Values Cells
sum(is.na(data))
sum(apply(data, 1, anyNA))

# Missing Value Distribution
data.na <- data[apply(data, 1, anyNA), ]
table(data.na$BAD)

# Total number of Missing Values of Observations
count(data.na)

# Percentage of Missing Value
round((count(data.na)/count(data))*100, 2)

# Remaining Data
count(data) - count(data.na)
round(((count(data) - count(data.na))/count(data))*100, 2)

# Bar Plot of Missing Value Distribution
my.par()
na.aggr <- data.frame(sum = c("Defaulted", "Paid", "Total Rows", "Total Cells"),
                 count = c(880, 1565, 2445, 4740))

barplot(height = na.aggr$count,
        ylim = c(0, 5100), cex.lab = 0.75,
        names = na.aggr$sum,
        col = c("red2", "cornflowerblue", "forestgreen", "orange")) %>%
  text(0, y = na.aggr$count,
       labels = na.aggr$count,
       pos = 3, cex = 1)

# Heat Map of Missing Value
my.par()
missmap(data,
        col = c("red2", "cornflowerblue"),
        main = "",
        legend = FALSE,
        x.cex = 0.8, y.cex = 0.8,
        gap.xaxis = 1, x.las = 2
        )
#  Preference  for  Reference:  http://www.sthda.com/english/wiki/add-legends-to-plots-in-r-software-
the-easiest-way

# Create new data frame without missing value

data.omit <- na.omit(data)
str(data.omit)

# Explort new data frame with no missing values to .CSV file
setwd("C:/One Drives/OneDrive - Webster University/Webster Classes/21F_CSDA 6010_Practicum")
options(scipen = 0, digits = 2)
data.omit %>% write.csv("hmeq_omit.csv", row.names = TRUE)
```

3.3. Check Typographical Errors

```
# Plot Categorical Attributes
my.par() %>% par(mfrow = c(1, 1),
```

```
                 mai = par("mai") * 1,
                 oma = c(0,0,0,0) + 0.1,
                 mar = c(2,2,2,0) + 0.1
                 )

## REASON Attribute
reason.bp <- plot(as.factor(data$REASON),
                  ylim = c(0, 4200),
                  main = "",
                  col = c("red2", "cornflowerblue", "forestgreen"))
text(x = reason.bp, y = table(as.factor(data$REASON)),
     labels = table(as.factor(data$REASON)),
     pos = 3, cex = 1)

my.par() %>% par(mfrow = c(1, 1),
                 mai = par("mai") * 1,
                 oma = c(0,0,0,0) + 0.1,
                 mar = c(0,0,0,0) + 0.1
                 )
pie(table(data$REASON),
    labels = paste(c("","DebtCon", "HomeImp"),
                   round(prop.table(table(data$REASON)), 2)*100, "%"),
    col = c("red2", "cornflowerblue", "forestgreen")
    )

## JOB Attribute
my.par() %>% par(mfrow = c(1, 1),
                 mai = par("mai") * 1,
                 oma = c(0,0,0,0) + 0.1,
                 mar = c(2,5,2,0) + 0.1
                 )
job.bp <- plot(as.factor(data$JOB),
                  ylim = c(0, 2500),
                  main = "",
                  col = pal)
text(x = job.bp, y = table(as.factor(data$JOB)),
     labels = table(as.factor(data$JOB)),
     pos = 3, cex = 1)

my.par() %>% par(mfrow = c(1, 1),
                 mai = par("mai") * 1,
                 oma = c(0,0,0,0) + 0.1,
                 mar = c(0,0,0,0) + 0.1
                 )
pie(table(data$JOB),
    labels = paste(c("", "Mgr", "Office", "Other", "ProfExe", "Sales", "Self"),
                   round(prop.table(table(data$JOB)), 2)*100, "%"),
    col = pal
    )
```

3.4. Check Outliers

```
# Generic Function for Combined Box Plot
out.bp <- function(dataset) {
  my.par() %>% par(mfrow = c(1, 10),
                 mai = par("mai") * 1,
                 oma = c(0,0,0,0) + 0.1,
                 mar = c(0.5,2.5,5,0) + 0.1
                 )
  i = 2
  for (i in 2:length(colnames(dataset))) {
    boxplot(dataset[, i],
            main = colnames(dataset)[i],
            cex.main = 2, cex.axis = 2,
            col = "red3")
    i = i + 1
  }
}
```

```
# Plot Raw Dataset
my.par()
out.bp(data[, -c(5, 6, 14)])

# Plot No Missing Value Set
my.par()
out.bp(data.omit[, -c(5, 6, 14)])

# Remove Outliers
## Generic Function to All Remove Outliers
outliers_remover <- function(a){
  df <- a
  aa <- c()
  count <- 1
  for (i in 1:ncol(df)) {
    if (is.numeric(df[,i])) {
      Q3 <- quantile(df[,i], 0.75, na.rm = TRUE)
      Q1 <- quantile(df[,i], 0.25, na.rm = TRUE)
      IQR <- Q3 - Q1  #IQR(df[,i])
      upper <- Q3 + 1.5 * IQR
      lower <- Q1 - 1.5 * IQR
      for (j in 1:nrow(df)) {
        if (is.na(df[j,i]) == TRUE) {
          next
        }
        else if (df[j,i] > upper | df[j,i] < lower) {
          aa[count] <- j
          count <- count + 1
        }
      }
    }
  }
  df <- df[-aa,]
}

## Remove Outliers in all Numeric Attributes
data.na.clean <- data
data.na.clean <- outliers_remover(data.na.clean)
data.clean <- data.omit
data.clean <- outliers_remover(data.clean)

## Check Variables
dim(data.na.clean)
dim(data.clean)

## Plot new data sets with no outliers
my.par()
out.bp(data.na.clean[, -c(5, 6, 14)])

out.bp(data.clean[, -c(5, 6, 14)])

# Clear Outliers Problem
dim(data.clean)

table(data.omit$DEROG)
table(data.clean$DEROG)
table(data.omit$DELINQ)
table(data.clean$DELINQ)
table(data.omit$NINQ)
table(data.clean$NINQ)
table(data.omit$CLNO)
table(data.clean$CLNO)
length(table(data.omit$CLNO))
length(table(data.clean$CLNO))

# Solution to Outliers
data.omit.group <- data.omit
data.omit.group$DEROG[data.omit.group$DEROG > 2] <- 2
data.omit.group$DELINQ[data.omit.group$DELINQ > 2] <- 2
```

```
data.omit.group$NINQ[data.omit.group$NINQ > 3] <- 3

table(data.omit.group$DEROG)
table(data.omit.group$DELINQ)
table(data.omit.group$NINQ)
data.omit.group.clean <- outliers_remover(data.omit.group)
str(data.omit.group.clean)

out.bp(data.omit.group.clean[, -c(5, 6, 14)])

table(data.omit.group.clean$DEROG)
table(data.omit.group.clean$DELINQ)
table(data.omit.group.clean$NINQ)

out.bp(data.omit.group[, -c(5, 6, 14)])

# Export new clean data set stats with no missing value and outliers
setwd("~")
options(scipen = 0, digits = 2)
round(stat.desc(data.omit.group), 0) %>% write.csv("hmeq_omit_group_stats.csv", row.names = TRUE)
```

3.5. Clean Data Set

```
# Replace Empty Spaces

## Raw Data
data$REASON[data$REASON == ""] <- "Other"
data$REASON <- factor(data$REASON)
data$JOB[data$JOB == ""] <- "Other"
data$JOB <- factor(data$JOB)

data$.REASON <- ifelse(data$REASON == "DebtCon", 1,
                       ifelse(data$REASON == "HomeImp", 2, 3))

data$.JOB <- ifelse(data$JOB == "Other", 1,
                    ifelse(data$JOB == "ProfExe", 2,
                           ifelse(data$JOB == "Office", 3,
                                  ifelse(data$JOB == "Mgr", 4,
                                         ifelse(data$JOB == "Self", 5, 6
                                                )))))

## No Missing Value Data
data.omit$REASON[data.omit$REASON == ""] <- "Other"
data.omit$REASON <- factor(data.omit$REASON)
data.omit$JOB[data.omit$JOB == ""] <- "Other"
data.omit$JOB <- factor(data.omit$JOB)

data.omit$.REASON <- ifelse(data.omit$REASON == "DebtCon", 1,
                       ifelse(data.omit$REASON == "HomeImp", 2, 3))

data.omit$.JOB <- ifelse(data.omit$JOB == "Other", 1,
                    ifelse(data.omit$JOB == "ProfExe", 2,
                           ifelse(data.omit$JOB == "Office", 3,
                                  ifelse(data.omit$JOB == "Mgr", 4,
                                         ifelse(data.omit$JOB == "Self", 5, 6
                                                )))))

## Grouped No Misisng Value Data
data.omit.group$REASON[data.omit.group$REASON == ""] <- "Other"
data.omit.group$REASON <- factor(data.omit.group$REASON)
data.omit.group$JOB[data.omit.group$JOB == ""] <- "Other"
data.omit.group$JOB <- factor(data.omit.group$JOB)

data.omit.group$.REASON <- ifelse(data.omit.group$REASON == "DebtCon", 1,
                       ifelse(data.omit.group$REASON == "HomeImp", 2, 3))

data.omit.group$.JOB <- ifelse(data.omit.group$JOB == "Other", 1,
```

```
                    ifelse(data.omit.group$JOB == "ProfExe", 2,
                           ifelse(data.omit.group$JOB == "Office", 3,
                                  ifelse(data.omit.group$JOB == "Mgr", 4,
                                         ifelse(data.omit.group$JOB == "Self", 5, 6
                                                       )))))

# Export Final Data Set

## No Missing Data
setwd("~")
options(scipen = 0, digits = 2)
data.omit %>% write.csv("hmeq_omit_final.csv", row.names = TRUE)

# Grouped No Missing Data
setwd("C:/One Drives/OneDrive - Webster University/Webster Classes/21F_CSDA 6010_Practicum")
options(scipen = 0, digits = 2)
data.omit.group %>% write.csv("hmeq_omit_group_final.csv", row.names = TRUE)
```

4. ATTRIBUTE EXPLORATION

```
df <- data.omit.group
str(df)
summary(df)
```

4.1. Target Attribute

```
my.par() %>% par(mfrow = c(1, 2),
                 mai = par("mai") * 1,
                 oma = c(0,0,0,0) + 0.1,
                 mar = c(2,4,0,0) + 0.1
                 )

stt.bp.raw <- plot(data$STATUS,
                   ylim = c(0, 5500),
                   ylab = "Frequency in Raw Dataset",
                   main = "",
                   col = c("red2", "cornflowerblue"))
text(x = stt.bp.raw, y = table(data$STATUS),
     labels = paste(table(data$STATUS), "-",
                    round(prop.table(table(data$STATUS))*100, 2),"%"),
     pos = 3, cex = 1)
# text(x = stt.bp.raw, y = c(1400, 4970),
#      labels = paste(round(prop.table(table(data$STATUS))*100, 2),"%"),
#      pos = 3, cex = 1)

stt.bp.clean <- plot(df$STATUS,
     ylim = c(0, 5500),
     ylab = "Frequency in Clean Dataset",
     main = "",
     col = c("red2", "cornflowerblue"))
text(x = stt.bp.clean, y = table(df$STATUS),
     labels = paste(table(df$STATUS), "-",
                    round(prop.table(table(df$STATUS))*100, 2),"%"),
     pos = 3, cex = 1)
# text(x = stt.bp.clean, y = c(300, 2400),
#      labels = paste(round(prop.table(table(df$STATUS))*100, 2),"%"),
#      pos = 3, cex = 1)
```

4.2. Predictor Attributes

4.2.1.Target against Categorical Attributes

```
# Function to Create Box Plot of Numeric Attributes
pred.box_f <- function(i) {
  explore.vars <- c(2:4, 15:16, 7:13)
  j = explore.vars[i]
  ggplot(df, aes(fill = STATUS,
             y = df[, j],
             x = STATUS)) +
```

```
  ggtitle(paste(colnames(df)[j], "against STATUS")) +
  geom_boxplot() + theme_minimal() +
  scale_fill_manual(values = c("red2", "cornflowerblue")) +
  theme(text = element_text(size = 12, family = "cambria")) +
  theme(axis.title.y = element_blank(),
        axis.ticks.y = element_blank(),
        axis.title.x = element_blank(),
        legend.position = "none")
}

# Plot Attributes
box.LOAN <- pred.box_f(1)
box.MORTDUE <- pred.box_f(2)
box.VALUE <- pred.box_f(3)
box.REASON <- pred.box_f(4)
box.JOB <- pred.box_f(5)
box.YOJ <- pred.box_f(6)
box.DEROG <- pred.box_f(7)
box.DELINQ <- pred.box_f(8)
box.CLAGE <- pred.box_f(9)
box.NINQ <- pred.box_f(10)
box.CLNO <- pred.box_f(11)
box.DEBTINC <- pred.box_f(12)

# Combine Box Plots
my.par()
cowplot::plot_grid(box.LOAN, box.MORTDUE, box.VALUE, box.REASON,
                   box.JOB, box.YOJ, box.DEBTINC, box.DEROG,
                   box.DELINQ, box.CLAGE, box.NINQ, box.CLNO
                   )
```

4.2.2. Target against Numerical Attributes

```
# Function to Create Bar Plot of Numeric Categories
rm(i) %>% suppressWarning()
pred.bar_f <- function(df, i) {
  ggplot(df, aes(fill = STATUS, x = df[, i])) +
  labs(title = colnames(df)[i], y = "Percentage") +
  geom_bar(position = "fill") +
  theme_classic() +
  scale_fill_manual(values = c("red2", "cornflowerblue")) +
  theme(text = element_text(size = 12, family = "cambria")) +
  theme(axis.title.x = element_blank(),
        legend.position = "top")
}

# Plot Attributes
bar.REASON <- pred.bar_f(df, 5)
bar.JOB <- pred.bar_f(df, 6)
bar.DEROG <- pred.bar_f(df, 8)
bar.DELINQ <- pred.bar_f(df, 9)
bar.NINQ <- pred.bar_f(df, 11)

# Combine Bar Plots
cowplot::plot_grid(bar.DEROG, bar.DELINQ, bar.NINQ) %>% suppressWarnings()
cowplot::plot_grid(bar.REASON, bar.JOB) %>% suppressWarnings()
```

4.3. Select Varibles

4.3.1. Correlation Matrix

```
# Calculate Correlation between Attributes
corr.var <- c(1:4, 7:13, 15, 16)
corr.df <- df[, corr.var]
corr.mat <- round(cor(corr.df),2)
testRes = cor.mtest(corr.mat, conf.level = 0.95)

# Plot Correlation
my.par()
```

```
corrplot(corr.mat, method = "color",
         type = "full", tl.col = "black")$corrPos -> corrp
text(corrp$x, corrp$y, round(corrp$corr, 2))
```

4.3.2. Stepwise Regression

```
loreg_m_ic <- function(X, Z){
  options(scipen = 0)
  set.seed(2021)
  lg.reg <- glm(STATUS ~ ., data = X, family = "binomial")
  # step(lg.reg, direction = Z)
  summary(step(lg.reg, direction = Z))
}



options(scipen = 0)
loreg_m_ic(data.omit[, -c(1, 15, 16)], "both")
```

4.3.3. Information Value

```
# library(scorecard)
iv.df <- as.data.frame(iv(df[, -c(5:6, 14)],
                          y = "BAD", positive = "BAD|1")) %>% arrange()
print(iv.df)

my.par()
barplot(height = iv.df$info_value, names = iv.df$variable,
        ylim = c(0, 0.9), col = pal) %>%
  text(0, y = iv.df$info_value,
       labels = round(iv.df$info_value, 6),
       pos = 3, cex = 1) %>% suppressWarnings()
```

4.3.4. Variables in Use

```
var1 <- c("STATUS", "LOAN", "CLNO", "YOJ", "DELINQ", "DEROG", "NINQ")
col1 <- c("LOAN", "CLNO", "YOJ", "DELINQ", "DEROG", "NINQ")
var2 <- c("STATUS", "CLAGE", "DEBTINC", "DELINQ", "DEROG", "NINQ")
col2 <- c("CLAGE", "DEBTINC", "DELINQ", "DEROG", "NINQ")
var <- c(2:4, 7:16)
col <- c(2:4, 7:13, 15:16)

var.bad <- c(1:4, 7:13, 15:16)
col.bad <- c(2:4, 7:13, 15:16)

col.rose.k <- c(1:3, 6:12, 14:15)
```

5. PARTITION

5.1. Training and Validation

```
set.seed(2021)
index <- sample(c(1:dim(df)[1]), dim(df)[1]*0.7)
train <- df[index, ]
valid <- df[-index, ]

table(train$STATUS)
prop.table(table(train$STATUS))
table(valid$STATUS)
prop.table(table(valid$STATUS))
```

5.2. Balance Training with ROSE

```
train.rose <- ROSE(STATUS ~ ., data = train[, -1])$data
```

```
table(train.rose$STATUS)
prop.table(table(train.rose$STATUS))
```

5.3. Plot STATUS distribution of 3 sets

```
my.par() %>% par(mfrow = c(1, 3),
                 mai = par("mai") * 1,
                 oma = c(0,0,0,0) + 0.1,
                 mar = c(2,4,4,0) + 0.1
                 )
plot(train$STATUS,
     ylim = c(0, 2500),
     ylab = "Frequency",
     main = "ImBalanced Training Set",
     col = c("red2", "cornflowerblue")) %>%
text(0, y = table(train$STATUS),
     labels = paste(table(train$STATUS), "-",
                    round(prop.table(table(train$STATUS))*100, 2),"%"),
     pos = 3, cex = 1)

plot(train.rose$STATUS,
     ylim = c(0, 2500),
     ylab = "Frequency",
     main = "Balanced Training Set",
     col = c("red2", "cornflowerblue")) %>%
text(0, y = table(train.rose$STATUS),
     labels = paste(table(train.rose$STATUS), "-",
                    round(prop.table(table(train.rose$STATUS))*100, 2),"%"),
     pos = 3, cex = 1)

plot(valid$STATUS,
     ylim = c(0, 2500),
     ylab = "Frequency",
     main = "Validation Set",
     col = c("red2", "cornflowerblue")) %>%
text(0, y = table(valid$STATUS),
     labels = paste(table(valid$STATUS), "-",
                    round(prop.table(table(valid$STATUS))*100, 2),"%"),
     pos = 3, cex = 1)
```

6. CLASSIFICATIONS

6.1. Logistic Regression

6.1.1. Logistic Regression (lr) Data Sets

```
train.lr <- train
train.lr.rose <- train.rose
valid.lr <- valid
```

6.1.2. Generic Function

```
loreg_m <- function(X, Y, cutoff){
  options(scipen = 0)
  set.seed(2021)
  # Model
  if (!require("lattice")) install.packages("lattice")
  lg.reg <- glm(STATUS ~ ., data = X, family = "binomial")
  # print(summary(lg.reg))
    # Predict
  lg.reg.pred <- predict(lg.reg, Y)
    # Evaluate
  if (!require("caret")) install.packages("caret")
  library(caret)
  confusionMatrix(factor(ifelse(lg.reg.pred  >  cutoff,  "paid",  "defaulted")),  factor(Y$STATUS),
positive = 'paid')
  # plot(lg.reg, which = 1:2)
}
```

6.1.3. Imbalanced All

```
loreg_m(train.lr, valid.lr, 0.5)
```

6.1.4. Balanced All

```
loreg_m(train.lr.rose, valid.lr, 0.5)
```

6.1.5. Imblanced Set 1

```
loreg_m(train.lr[, var1], valid.lr[, var1], 0.5)
```

6.1.6. Blanced Set 1

```
loreg_m(train.lr.rose[, var1], valid.lr[, var1], 0.5)
```

6.1.7. Imbalanced Set 2

```
loreg_m(train.lr[, var2], valid.lr[, var2], 0.5)
```

6.1.8. Balanced Set 2

```
loreg_m(train.lr.rose[, var2], valid.lr[, var2], 0.5)
```

6.1.9. Four Fold Plots of Confusion Matrix

```
lr_cmp <- function(cm, title) {
  fourfoldplot(cm, color = c("cornflowerblue", "red2"),
               margin = 1, space = 0.3,
               conf.level = 0, main = title)
}
my.par() %>% par(mfrow = c(2, 3),
                 mai = par("mai") * 1,
                 oma = c(0,0,0,0) + 0.1,
                 mar = c(2,4,2,0) + 0.1
                 )
lr_cmp(loreg_m(train.lr, valid.lr, 0.5)$table, "Model A")
lr_cmp(loreg_m(train.lr.rose, valid.lr, 0.5)$table, "Model B")
lr_cmp(loreg_m(train.lr[, var1], valid.lr[, var1], 0.5)$table, "Model C")
lr_cmp(loreg_m(train.lr.rose[, var1], valid.lr[, var1], 0.5)$table, "Model D")
lr_cmp(loreg_m(train.lr[, var2], valid.lr[, var2], 0.5)$table, "Model E")
lr_cmp(loreg_m(train.lr.rose[, var2], valid.lr[, var2], 0.5)$table, "Model F")
```

6.2. k-Nearest Neighbors

6.2.1. k-Nearest Neighbors (knn) Data Sets

```
train.knn <- train
train.knn.rose <- train.rose
valid.knn <- valid
```

6.2.2. Optimal K

```
k_opt <- sqrt(nrow(train.knn))/2
k_opt

knn_ks <- function(X, Y, att, seq_i=1, seq_j=k_opt, seq_step=1) {
  library(caret); library(FNN); library(class)
  library(gmodels); library(e1071); library(ggplot2)
  options(scipen = 0)
  set.seed(2021)
  accuracy <- data.frame(k = seq(seq_i, seq_j, seq_step), accuracy = rep(0, k_opt))
  for (i in seq_i:seq_j) {
    knn.pred <- knn(X[, att], Y[, att],
                    cl = X[, 'STATUS'], k = i)
    accuracy[i, 2] <- confusionMatrix(knn.pred,
                                      factor(Y[, 'STATUS']),
```

```
                                      positive = 'paid')$overall[1]
  }
  accuracy
}

knn_i0 <- knn_ks(train.knn, valid.knn, col) %>% suppressWarnings()
knn_b0 <- knn_ks(train.knn.rose, valid.knn[, -1], col.rose.k) %>% suppressWarnings()
knn_i1 <- knn_ks(train.knn[, var1], valid.knn[, var1], col1) %>% suppressWarnings()
knn_b1 <- knn_ks(train.knn.rose[, var1], valid.knn[, var1], col1) %>% suppressWarnings()
knn_i2 <- knn_ks(train.knn[, var2], valid.knn[, var2], col2) %>% suppressWarnings()
knn_b2 <- knn_ks(train.rose[, var2], valid[, var2], col2) %>% suppressWarnings()

my.par() %>% par(mai = par("mai") * 1,
                 oma = c(0,0,0,0) + 0.1,
                 mar = c(4,4,0,0) + 0.1
                 )
plot(knn_i0, bty = "n", type = "n",
     xlim = c(0, 25), ylim = c(0.5, 1.05),
     xlab = "k", ylab = "Accuracy",
     yaxt = "n")
axis(2, at = seq(0.5, 1.0, 0.1),
     labels = format(seq(0.5, 1.0, 0.1), digits = 2))
lines(knn_i0, type = "b", pch = 18, lwd = 2, col = pal[1])
lines(knn_i1, type = "b", pch = 18, lwd = 2, col = pal[2])
lines(knn_i2, type = "b", pch = 18, lwd = 2, col = pal[3])
lines(knn_b0, type = "b", pch = 18, lwd = 2, col = pal[4])
lines(knn_b1, type = "b", pch = 18, lwd = 2, col = pal[5])
lines(knn_b2, type = "b", pch = 18, lwd = 2, col = pal[6])
legend("top", bty = "n",
       legend = c("G", "H", "I", "J", "K", "L"),
       title = "MODELS", cex = 1.5,
       col = pal[1:6], horiz = TRUE,
       lty = 1, lwd = 2, pch = 18
       )
```

6.2.3. Generic Functions

```
knn_f <- function(X, Y, att, seq_i=1, seq_j=k_opt, seq_step=1) {
  library(caret); library(FNN); library(class)
  library(gmodels); library(e1071); library(ggplot2)
  options(scipen = 0)
  set.seed(2021)
  accuracy <- data.frame(k = seq(seq_i, seq_j, seq_step), accuracy = rep(0, k_opt))
  for (i in seq_i:seq_j) {
    knn.pred <- knn(X[, att], Y[, att], cl = X[, 'STATUS'], k = i)
    accuracy[i, 2] <- confusionMatrix(knn.pred, factor(Y[, 'STATUS']),
                                      positive = 'paid')$overall[1]
  }
  print(accuracy)
  plot(accuracy$k, xlab = "Values of k",
       accuracy$accuracy,
       ylab = "Accuracies",
       main = "Values of k against their accuracies",
       type = 'l',
       col = 'gray')
}

knn_e <- function(X, Y, att, i){
  library(caret); library(FNN); library(class)
  library(gmodels); library(e1071); library(ggplot2)
  options(scipen = 0)
  set.seed(2021)
  knn_pred <- knn(X[, att], Y[, att], cl = X[, 'STATUS'], k = i)
  confusionMatrix(factor(knn_pred,
                         levels = c("paid", "defaulted")),
                  factor(Y$STATUS,
                         levels = c("paid", "defaulted")),
                  positive = "paid")
}
```

```
knn_ks <- function(X, Y, att, seq_i=1, seq_j=k_opt, seq_step=1) {
  library(caret); library(FNN); library(class)
  library(gmodels); library(e1071); library(ggplot2)
  options(scipen = 0)
  set.seed(2021)
  accuracy <- data.frame(k = seq(seq_i, seq_j, seq_step), accuracy = rep(0, k_opt))
  for (i in seq_i:seq_j) {
    knn.pred <- knn(X[, att], Y[, att], cl = X[, 'STATUS'], k = i)
    accuracy[i, 2] <- confusionMatrix(knn.pred, factor(Y[, 'STATUS']),
                                      positive = 'paid')$overall[1]
  }
  accuracy
}
```

6.2.4. Imbalanced All

```
my.par()
knn_f(train.knn, valid.knn, col)
#5

knn_e(train.knn, valid.knn, col, 5)
```

6.2.5. Balanced All

```
my.par()
knn_f(train.knn.rose, valid.knn[, -1], col.rose.k) %>% suppressWarnings()
# 25

knn_e(train.knn.rose, valid.knn[, -1], col.rose.k, 25)
```

6.2.6. Imbalanced Set 1

```
my.par()
knn_f(train.knn[, var1], valid.knn[, var1], col1) %>% suppressWarnings()
# 5

knn_e(train.knn[, var1], valid.knn[var1], col1, 5)
```

6.2.7. Balanced Set 1

```
my.par()
knn_f(train.knn.rose[, var1], valid.knn[, var1], col1) %>% suppressWarnings()
# 1

knn_e(train.knn.rose[, var1], valid.knn[var1], col1, 1)
```

6.2.8. Imbalanced Set 2

```
my.par()
knn_f(train.knn[, var2], valid.knn[, var2], col2)
# 3

knn_e(train.knn[, var2], valid.knn[var2], col2, 3)
```

6.2.9. Balanced Set 2

```
my.par()
knn_f(train.rose[, var2], valid[, var2], col2) %>% suppressWarnings()
# 19

knn_e(train.rose[, var2], valid[var2], col2, 19)
```

6.2.10. Four Fold Plots of Confusion Matrix

```
knn_cmp <- function(cm, title) {
  fourfoldplot(cm, color = c("cornflowerblue", "red2"),
               margin = 1, space = 0.3,
               conf.level = 0, main = title)
```

```
}
my.par() %>% par(mfrow = c(2, 3),
                 mai = par("mai") * 1,
                 oma = c(0,0,0,0) + 0.1,
                 mar = c(2,4,2,0) + 0.1
                 )
knn_cmp(knn_e(train.knn, valid.knn, col, 5)$table, "Model G: k = 5")
knn_cmp(knn_e(train.knn[, var1], valid.knn[var1], col1, 5)$table, "Model H: k = 5")
knn_cmp(knn_e(train.knn[, var2], valid.knn[var2], col2, 3)$table, "Model I: k = 3")
knn_cmp(knn_e(train.knn.rose, valid.knn[, -1], col.rose.k, 25)$table, "Model J: k = 25")
knn_cmp(knn_e(train.knn.rose[, var1], valid.knn[var1], col1, 1)$table, "Model K: k = 1")
knn_cmp(knn_e(train.rose[, var2], valid[var2], col2, 19)$table, "Model L: k = 19")
```

6.3. Neural Networks

6.3.1. Neural Networks (nn) Data sets

```
train.nn <- train
train.nn.rose <- train.rose
valid.nn <- valid
```

6.3.2. Generic Function

```
set.seed(2021)
options(scipen = 0)
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}
train.nn[, c(2:4, 7:13, 15:16)] <- as.data.frame(lapply(train.nn[, c(2:4, 7:13, 15:16)], normalize))

train.nn.rose.bad <- train.nn.rose
train.nn.rose.bad$BAD <- ifelse(train.nn.rose$STATUS == "paid", 0, 1)
train.nn.rose.bad <- train.nn.rose.bad[, c(16, 1:15)]

train.nn.rose.bad[, c(2:4, 7:13, 15:16)] <- as.data.frame(lapply(train.nn.rose.bad[, c(2:4, 7:13,
15:16)], normalize))
```

6.3.3. Imbalanced All

```
nn_if <- function(x, y) {
  var.bad <- c(1:4, 7:13, 15:16)
  col.bad <- c(2:4, 7:13, 15:16)
  if (!require("neuralnet")) install.packages("neuralnet")
  library(neuralnet)
  # Model
  set.seed(2021)
  options(scipen = 0)
  nn <- neuralnet(BAD ~ .,
                    data = train.nn[, var.bad],
                    hidden = x,
                    linear.output = FALSE)
  # plot(nn)
  # Predict
  nn_pred <- neuralnet::compute(nn, valid.nn[, col.bad])
  nn_pred_result <- nn_pred$net.result

  # Evaluate
  confusionMatrix(factor(ifelse(nn_pred_result > y, "paid", "defaulted"),
                       levels = c("paid", "defaulted")),
                  factor(valid.nn$STATUS, levels = c("paid", "defaulted"))) %>% print()
  # if (!require("beepr")) install.packages("beepr"); library(beepr)
  # beep(2)
}

cm_nn1 <- nn_if(1, 0.5)
```

6.3.4. Balanced All

```
nn_bf <- function(x, y) {
```

```
  var.bad <- c(1:4, 7:13, 15:16)
  col.bad <- c(2:4, 7:13, 15:16)
  if (!require("neuralnet")) install.packages("neuralnet")
  library(neuralnet)
  set.seed(2021)
  options(scipen = 0)
  # Model
  nn <- neuralnet(BAD ~ .,
                  data = train.nn.rose.bad[, var.bad],
                  hidden = x,
                  linear.output = FALSE)
  # plot(nn)
  # Predict
  nn_pred <- neuralnet::compute(nn, valid.nn[, col.bad])
  nn_pred_result <- nn_pred$net.result

  # Evaluate
  confusionMatrix(factor(ifelse(nn_pred_result > y, "paid", "defaulted"),
                         levels = c("paid", "defaulted")),
                  factor(valid.nn$STATUS, levels = c("paid", "defaulted"))) %>% print()
  # if (!require("beepr")) install.packages("beepr"); library(beepr)
  # beep(2)
}

cm_nn4 <- nn_bf(2, 0.5)
```

6.3.5. Imbalaned Set 1

```
nn_i1 <- function(x, y) {
  var.bad <- c("BAD", "LOAN", "CLNO", "YOJ", "DELINQ", "DEROG", "NINQ")
  col.bad <- c("LOAN", "CLNO", "YOJ", "DELINQ", "DEROG", "NINQ")
  if (!require("neuralnet")) install.packages("neuralnet")
  library(neuralnet)
  nn <- neuralnet(BAD ~ .,
                  data = train.nn[, var.bad],
                  hidden = x,
                  linear.output = FALSE)
  # plot(nn)
  # Predict
  options(scipen = 0)
  set.seed(2021)
  nn_pred <- neuralnet::compute(nn, valid.nn[, col.bad])
  nn_pred_result <- nn_pred$net.result

  # Evaluate
  confusionMatrix(factor(ifelse(nn_pred_result > y, "paid", "defaulted"),
                         levels = c("paid", "defaulted")),
                  factor(valid.nn$STATUS, levels = c("paid", "defaulted"))) %>% print()
  # if (!require("beepr")) install.packages("beepr"); library(beepr)
  # beep(2)
}

cm_nn2 <- nn_i1(c(1, 1), 0.5)
```

6.3.6. Balanced Set 1

```
nn_b1 <- function(x, y) {
  var.bad <- c("BAD", "LOAN", "CLNO", "YOJ", "DELINQ", "DEROG", "NINQ")
  col.bad <- c("LOAN", "CLNO", "YOJ", "DELINQ", "DEROG", "NINQ")
  if (!require("neuralnet")) install.packages("neuralnet")
  library(neuralnet)
  nn <- neuralnet(BAD ~ .,
                  data = train.nn.rose.bad[, var.bad],
                  hidden = x,
                  linear.output = FALSE)
  # plot(nn)
  # Predict
  options(scipen = 0)
  set.seed(2021)
```

```
  nn_pred <- neuralnet::compute(nn, valid.nn[, col.bad])
  nn_pred_result <- nn_pred$net.result

  # Evaluate
  confusionMatrix(factor(ifelse(nn_pred_result > y, "paid", "defaulted"),
                         levels = c("paid", "defaulted")),
                  factor(valid.nn$STATUS, levels = c("paid", "defaulted"))) %>% print()
  # if (!require("beepr")) install.packages("beepr"); library(beepr)
  # beep(2)
}

cm_nn5 <- nn_b1(c(1, 2), 0.5)
```

6.3.7. Imbalanced Set 2

```
nn_i2 <- function(x, y) {
  var.bad <- c("BAD", "CLAGE", "DEBTINC", "DELINQ", "DEROG", "NINQ")
  col.bad <- c("CLAGE", "DEBTINC", "DELINQ", "DEROG", "NINQ")
  if (!require("neuralnet")) install.packages("neuralnet")
  library(neuralnet)
  nn <- neuralnet(BAD ~ .,
                  data = train.nn[, var.bad],
                  hidden = x,
                  linear.output = FALSE)
  # plot(nn)
  # Predict
  options(scipen = 0)
  set.seed(2021)
  nn_pred <- neuralnet::compute(nn, valid.nn[, col.bad])
  nn_pred_result <- nn_pred$net.result

  # Evaluate
  confusionMatrix(factor(ifelse(nn_pred_result > y, "paid", "defaulted"),
                         levels = c("paid", "defaulted")),
                  factor(valid.nn$STATUS, levels = c("paid", "defaulted"))) %>% print()
  # if (!require("beepr")) install.packages("beepr"); library(beepr)
  # beep(2)
}

cm_nn3 <- nn_i2(2, 0.5)
```

6.3.8. Balanced Set 2

```
nn_b2 <- function(x, y) {
  var.bad <- c("BAD", "CLAGE", "DEBTINC", "DELINQ", "DEROG", "NINQ")
  col.bad <- c("CLAGE", "DEBTINC", "DELINQ", "DEROG", "NINQ")
  if (!require("neuralnet")) install.packages("neuralnet")
  library(neuralnet)
  nn <- neuralnet(BAD ~ .,
                  data = train.nn.rose.bad[, var.bad],
                  hidden = x,
                  linear.output = FALSE)
  # plot(nn)
  # Predict
  options(scipen = 0)
  set.seed(2021)
  nn_pred <- neuralnet::compute(nn, valid.nn[, col.bad])
  nn_pred_result <- nn_pred$net.result

  # Evaluate
  confusionMatrix(factor(ifelse(nn_pred_result > y, "paid", "defaulted"),
                         levels = c("paid", "defaulted")),
                  factor(valid.nn$STATUS, levels = c("paid", "defaulted"))) %>% print()
  # if (!require("beepr")) install.packages("beepr"); library(beepr)
  # beep(2)
}

cm_nn6 <- nn_b2(c(1, 2), 0.5)
```

6.3.9. Four Fold Plots of Confusion Matrix

```
nn_cmp <- function(cm, title) {
  fourfoldplot(cm, color = c("cornflowerblue", "red2"),
               margin = 1, space = 0.3,
               conf.level = 0, main = title)
}
my.par() %>% par(mfrow = c(2, 3),
                 mai = par("mai") * 1,
                 oma = c(0,0,0,0) + 0.1,
                 mar = c(2,4,2,0) + 0.1
                 )
nn_cmp(cm_nn1$table, "Model M: (1) - cutoff:0.5")
nn_cmp(cm_nn2$table, "Model N: (1,1) - cutoff:0.5")
nn_cmp(cm_nn3$table, "Model O: (2) - cutoff:0.5")
nn_cmp(cm_nn4$table, "Model P: (1) - cutoff:0.5")
nn_cmp(cm_nn5$table, "Model Q: (1,2) - cutoff:0.5")
nn_cmp(cm_nn6$table, "Model R: (1,2) - cutoff:0.5")
```

7. CLEAN ENVIRONMENT

```
rm(list = ls())
cat("\014")
```