2. Data Preparation

2022-06-19

Contents


```r
# install.packages("caTools")
# install.packages("ROSE")

# Load caTools package for data partitioning
library(caTools)

# Load ROSE package for data balancing
library(ROSE)
```

```
## Loaded ROSE 0.0-4
```

```r
# Import our data and save it to variable creditdf
creditdf <- read.csv("Credit 2.csv")
```

```r
# Check the structure of the variables in the dataframe by using str() function
str(creditdf)
```

```
## 'data.frame':    1000 obs. of  11 variables:
##  $ loan_duration       : int  6 48 12 42 24 36 24 36 12 30 ...
##  $ credit_history      : chr  "critical" "good" "critical" "good" ...
##  $ purpose             : chr  "furniture" "furniture" "education" "furniture" ...
##  $ amount              : int  1169 5951 2096 7882 4870 9055 2835 6948 3059 5234 ...
##  $ percent_of_income   : int  4 2 2 2 3 2 3 2 2 4 ...
##  $ years_at_residence  : int  4 2 3 4 4 4 4 2 4 2 ...
```

```
##  $ age                 : int  67 22 49 45 53 35 53 35 61 28 ...
##  $ existing_loans_count: int  2 1 1 1 2 1 1 1 1 2 ...
##  $ dependents          : int  1 1 2 2 2 2 1 1 1 1 ...
##  $ phone               : chr  "yes" "no" "no" "no" ...
##  $ high_risk           : chr  "no" "yes" "no" "no" ...
```

label encoding

applying data encoding for machine learning models that may not work well with categorical variables. Therefore, after this step, the variable should be saved as a numeric variable with as.numeric() function.

revalue()

```
# revalue(column name, c("level name" = "label"))
```

revalue() function from plyr package

`credit_history`: critical < poor < good < very good < perfect with labels from 1 to 5

```
# install.packages("plyr")

# Load plyr package for data encoding
library(plyr)

# Apply label encoding to credit_history
unique(creditdf$credit_history)
```

label encoding `credit_history` column

```
## [1] "critical"  "good"      "poor"      "perfect"   "very good"
```

```
creditdf$credit_history <- revalue(creditdf$credit_history, c("critical" = "1", "poor" = "2", "good" =

# Save credit_history as a numerical variable
creditdf$credit_history <- as.numeric(creditdf$credit_history)
```

phone: yes = 1 and no = 0

```
# Apply label encoding to phone
creditdf$phone <- revalue(creditdf$phone, c("yes" = "1", "no" = "0"))
```

```r
# Save credit_history as a numerical variable
creditdf$phone <- as.numeric(creditdf$phone)

# Check the summary of the updated dataset
summary(creditdf)
```

label encoding phone column

```
##  loan_duration  credit_history    purpose              amount
##  Min.   : 4.0   Min.   :1.000   Length:1000        Min.   :  250
##  1st Qu.:12.0   1st Qu.:1.000   Class :character   1st Qu.: 1366
##  Median :18.0   Median :3.000   Mode  :character   Median : 2320
##  Mean   :20.9   Mean   :2.455                      Mean   : 3271
##  3rd Qu.:24.0   3rd Qu.:3.000                      3rd Qu.: 3972
##  Max.   :72.0   Max.   :5.000                      Max.   :18424
##  percent_of_income years_at_residence      age        existing_loans_count
##  Min.   :1.000     Min.   :1.000      Min.   :19.00   Min.   :1.000
##  1st Qu.:2.000     1st Qu.:2.000      1st Qu.:27.00   1st Qu.:1.000
##  Median :3.000     Median :3.000      Median :33.00   Median :1.000
##  Mean   :2.973     Mean   :2.845      Mean   :35.55   Mean   :1.407
##  3rd Qu.:4.000     3rd Qu.:4.000      3rd Qu.:42.00   3rd Qu.:2.000
##  Max.   :4.000     Max.   :4.000      Max.   :75.00   Max.   :4.000
##    dependents         phone         high_risk
##  Min.   :1.000   Min.   :0.000   Length:1000
##  1st Qu.:1.000   1st Qu.:0.000   Class :character
##  Median :1.000   Median :0.000   Mode  :character
##  Mean   :1.155   Mean   :0.404
##  3rd Qu.:1.000   3rd Qu.:1.000
##  Max.   :2.000   Max.   :1.000
```

one hot encoding

one_hot()

```r
# one_hot(as.data.table(dataset), cols = column name)
```

one_hot() function from mltools package

`the 1st argument:` one_hot() function works with data tables to process the datasets easily. Therefore, the dataset should be first saved as data.table by using as.data.table(dataset) function.

`the 2nd argument:` stores the nominal variables (column names) that should be encoded.

```r
# install.packages("mltools")
# install.packages("data.table")

# Load mltools package
library(mltools)

# Load data.table package
library(data.table)

# Apply one hot encoding
creditdf$purpose <- as.factor(creditdf$purpose)
creditdf <- one_hot(as.data.table(creditdf), cols = "purpose")

# Check the summary of the updated dataset
summary(creditdf)
```

applying one hot encoding to purpose variable

```
##  loan_duration  credit_history  purpose_business  purpose_car
##  Min.   : 4.0   Min.   :1.000   Min.   :0.000    Min.   :0.000
##  1st Qu.:12.0   1st Qu.:1.000   1st Qu.:0.000    1st Qu.:0.000
##  Median :18.0   Median :3.000   Median :0.000    Median :0.000
##  Mean   :20.9   Mean   :2.455   Mean   :0.097    Mean   :0.349
##  3rd Qu.:24.0   3rd Qu.:3.000   3rd Qu.:0.000    3rd Qu.:1.000
##  Max.   :72.0   Max.   :5.000   Max.   :1.000    Max.   :1.000
##  purpose_education purpose_furniture purpose_renovations     amount
##  Min.   :0.000     Min.   :0.000     Min.   :0.000       Min.   :  250
##  1st Qu.:0.000     1st Qu.:0.000     1st Qu.:0.000       1st Qu.: 1366
##  Median :0.000     Median :0.000     Median :0.000       Median : 2320
##  Mean   :0.059     Mean   :0.473     Mean   :0.022       Mean   : 3271
##  3rd Qu.:0.000     3rd Qu.:1.000     3rd Qu.:0.000       3rd Qu.: 3972
##  Max.   :1.000     Max.   :1.000     Max.   :1.000       Max.   :18424
##  percent_of_income years_at_residence      age        existing_loans_count
##  Min.   :1.000     Min.   :1.000      Min.   :19.00   Min.   :1.000
##  1st Qu.:2.000     1st Qu.:2.000      1st Qu.:27.00   1st Qu.:1.000
##  Median :3.000     Median :3.000      Median :33.00   Median :1.000
##  Mean   :2.973     Mean   :2.845      Mean   :35.55   Mean   :1.407
##  3rd Qu.:4.000     3rd Qu.:4.000      3rd Qu.:42.00   3rd Qu.:2.000
##  Max.   :4.000     Max.   :4.000      Max.   :75.00   Max.   :4.000
##    dependents        phone        high_risk
##  Min.   :1.000   Min.   :0.000   Length:1000
##  1st Qu.:1.000   1st Qu.:0.000   Class :character
##  Median :1.000   Median :0.000   Mode  :character
##  Mean   :1.155   Mean   :0.404
##  3rd Qu.:1.000   3rd Qu.:1.000
##  Max.   :2.000   Max.   :1.000
```

partition

partition the dataset into training and test sets

sample.split()

subset()

```r
# Set a seed of 10 by using set.seed() function
set.seed(10)

# Generate split vector to partition the data into training and test sets with training ratio of 0.70
split <- sample.split(creditdf$high_risk, SplitRatio = 0.7)

# Generate the training and test sets by subsetting the data records from actual dataset
training <- subset(creditdf, split == TRUE)

testing <- subset(creditdf, split == FALSE)
```

split the dataset into the training set (70%) and test set (30%)

data balancing

ovun.sample()

ovun.sample() function with method = "over", "both" or "under"

```r
# Apply oversampling technique
oversampled <- ovun.sample(high_risk ~ ., data = training, method = "over", p=0.4, seed=1)$data
```

balance training dataset: balance the data with `oversampling` technique so that the minority class accounts for approximately 40% of the training dataset

```r
# Apply both over and under sampling technique
bothsampled <- ovun.sample(high_risk ~ ., data = training, method = "both", p=0.4, seed=1)$data
```

try both undersampling and oversampling method by using ovun.sample() function with method = "both". Set the proportion of minority class as 0.4.

compare different training sets

Compare the distribution of high risk customers in the initial training set with the oversampled training set and both over and under sampled training set. Use table() and prob.table() functions.

```r
# Check the distribution of high risk customers in the initial training set
table(training$high_risk)
```

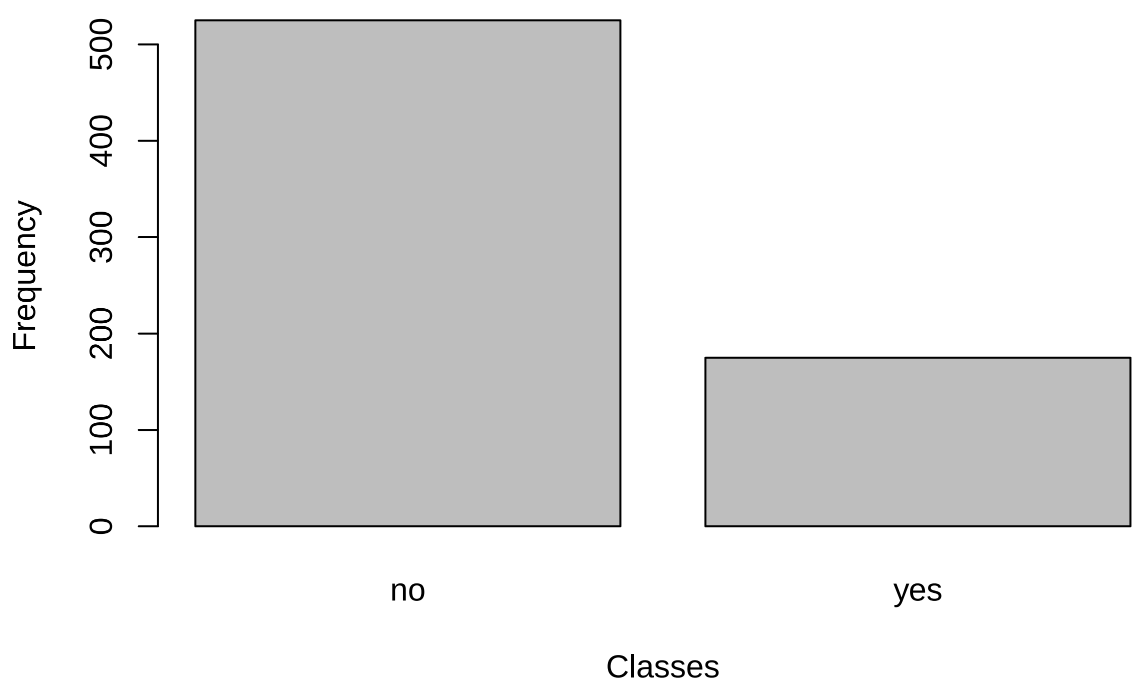
the initial training set

```
##
##  no yes
## 525 175
```

```r
# Check the proportion of high risk customers in the initial training set
prop.table(table(training$high_risk))
```

```
##
##   no  yes
## 0.75 0.25
```

```r
# Use barplot() function to plot the distribution of high risk customers
barplot(table(training$high_risk), xlab= "Classes", ylab="Frequency")
```



the `oversampled` training set

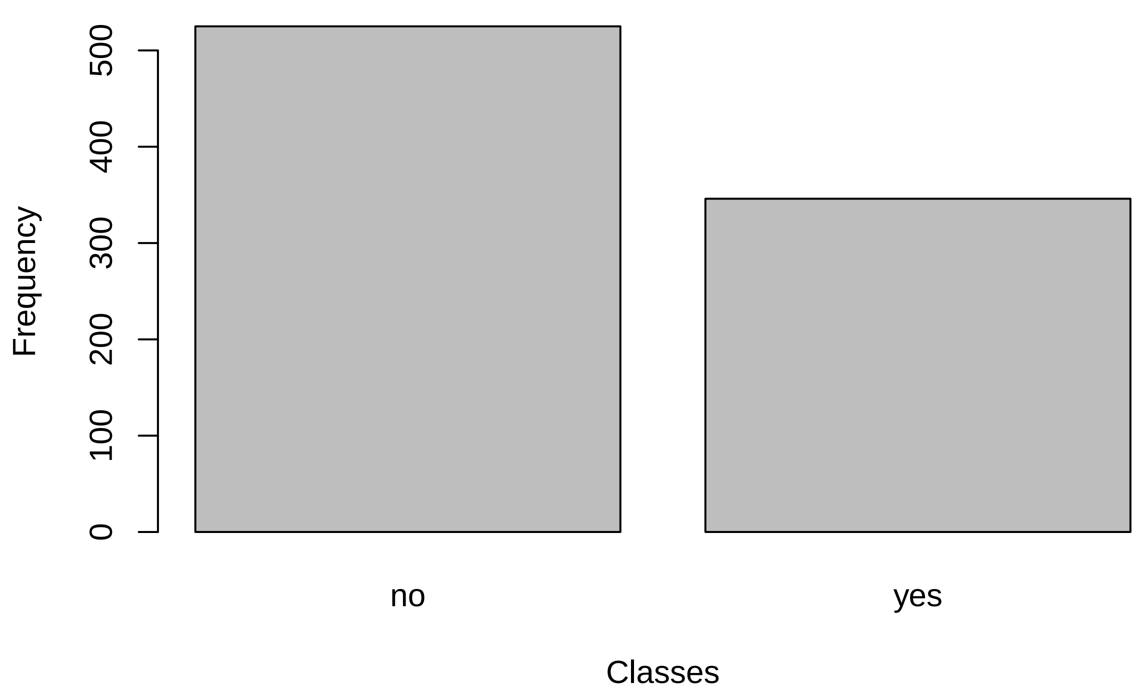
```r
# Check the distribution of high risk customers in the oversampled training set
table(oversampled$high_risk)
```

```
##
##  no yes
## 525 346
```

```r
# Check the proportion of high risk customers in the oversampled training set
prop.table(table(oversampled$high_risk))
```

```
##
##        no       yes
## 0.6027555 0.3972445
```

```r
# Plot the distribution by using barplot() function
barplot(table(oversampled$high_risk), xlab= "Classes", ylab="Frequency")
```



the `bothsampled` training set

```r
# Check the distribution of high risk customers in "bothsampled" training set
table(bothsampled$high_risk)
```

```
##
##  no yes
## 426 274
```

```r
# Check the proportion of high risk customers in bothsampled training set
prop.table(table(bothsampled$high_risk))
```

```
##
##        no       yes
## 0.6085714 0.3914286
```

```r
# Plot the distribution by using barplot() function
barplot(table(bothsampled$high_risk), xlab= "Classes", ylab="Frequency")
```