Credit Classification: Supervised Machine Learning

Gabriel D'assumpção de Carvalho

2025-07-14

Libraries

```r
# install.packages("ggplot2")
# install.packages("plotly")
# install.packages("caTools")
# install.packages("e1071")
# install.packages("class")

# Graphics
library(plotly)
library(ggplot2)

# Data Manipulation
library(caTools)
library(dplyr)

# Machine Learning
library(e1071) # For SVM
library(class) # For KNN
library(caret) # For confusionMatrix

# Warning and message suppression
options(warn = -1)
suppressMessages(library(ggplot2))
suppressMessages(library(dplyr))
suppressMessages(library(caret))

set.seed(42)
```

Introduction

Exploratory Data Analysis (EDA)

Data Import

```r
# Load the dataset
df <- read.csv("/home/gabrieldadcarvalho/github/actuarial_seminar/data/original.csv")[, -1]
print(head(df))
```

```
##     income      age      loan default
## 1 66155.93 59.01702 8106.5321       0
## 2 34415.15 48.11715 6564.7450       0
## 3 57317.17 63.10805 8020.9533       0
## 4 42709.53 45.75197 6103.6423       0
## 5 66952.69 18.58434 8770.0992       1
## 6 24904.06 57.47161   15.4986       0
```

Data Preprocessing

```r
# Check statistics of the dataset
summary(df)
```

```
##      income             age                loan                default
```

```
##  Min.   :20014   Min.   :-52.42   Min.   :    1.378   Min.   :0.0000
##  1st Qu.:32796   1st Qu.: 28.99   1st Qu.: 1939.709   1st Qu.:0.0000
##  Median :45789   Median : 41.32   Median : 3974.719   Median :0.0000
##  Mean   :45332   Mean   : 40.81   Mean   : 4444.370   Mean   :0.1415
##  3rd Qu.:57791   3rd Qu.: 52.59   3rd Qu.: 6432.411   3rd Qu.:0.0000
##  Max.   :69996   Max.   : 63.97   Max.   :13766.051   Max.   :1.0000
##                  NA's   :3
```

```
print(df$age[df$age <= 0])
```

```
## [1] -28.21836 -52.42328 -36.49698        NA        NA        NA
```

Analyzing the statistics of the variables, we can see that the `age` variable has some tree NaN and negative values. Below i will converter the negative values to positive values, and for the NaN values, we discussed some imputation methods to handle these missing values.

```
# Convert age for positive values
df$age <- abs(df$age)
print(summary(df$age))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   18.06   29.03   41.35   40.92   52.59   63.97       3
```

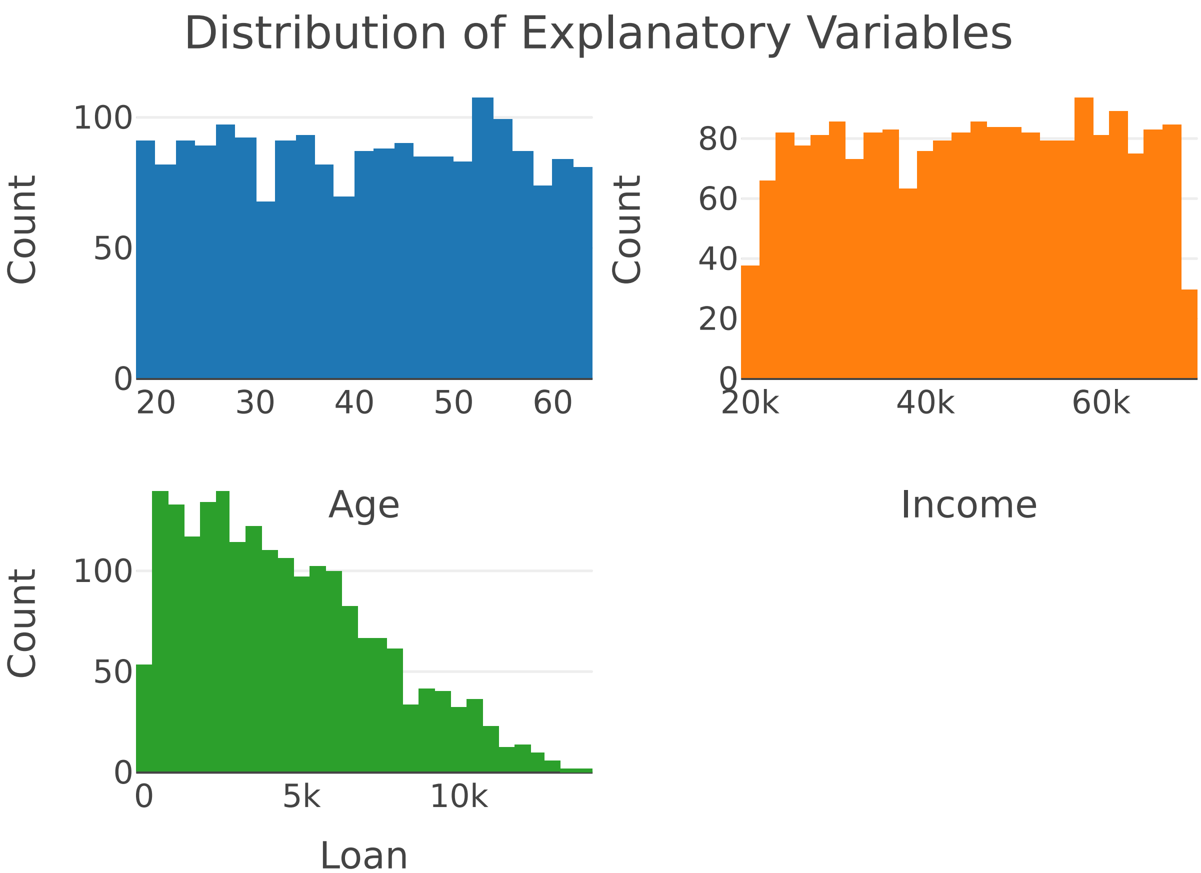
We can see that the variable `age` has a tree NaN values. We can apply some imputation methods to handle these missing values. For example:

- Mean imputation: Replace NaN values with the mean of the column.
- Median imputation: Replace NaN values with the median of the column.
- Linear Regression imputation: Use linear regression to predict missing values based on other variables.
- Regression imputation: Use regression models to predict missing values based on other variables.
- Interpolation: Use interpolation methods to estimate missing values based on surrounding data points.
- Exploratory Data Analysis (EDA): Analyze the data to understand the distribution and value intervals of the variables, and then apply one statistics to replace the NaN values.

Exploratory Data Analysis (EDA)

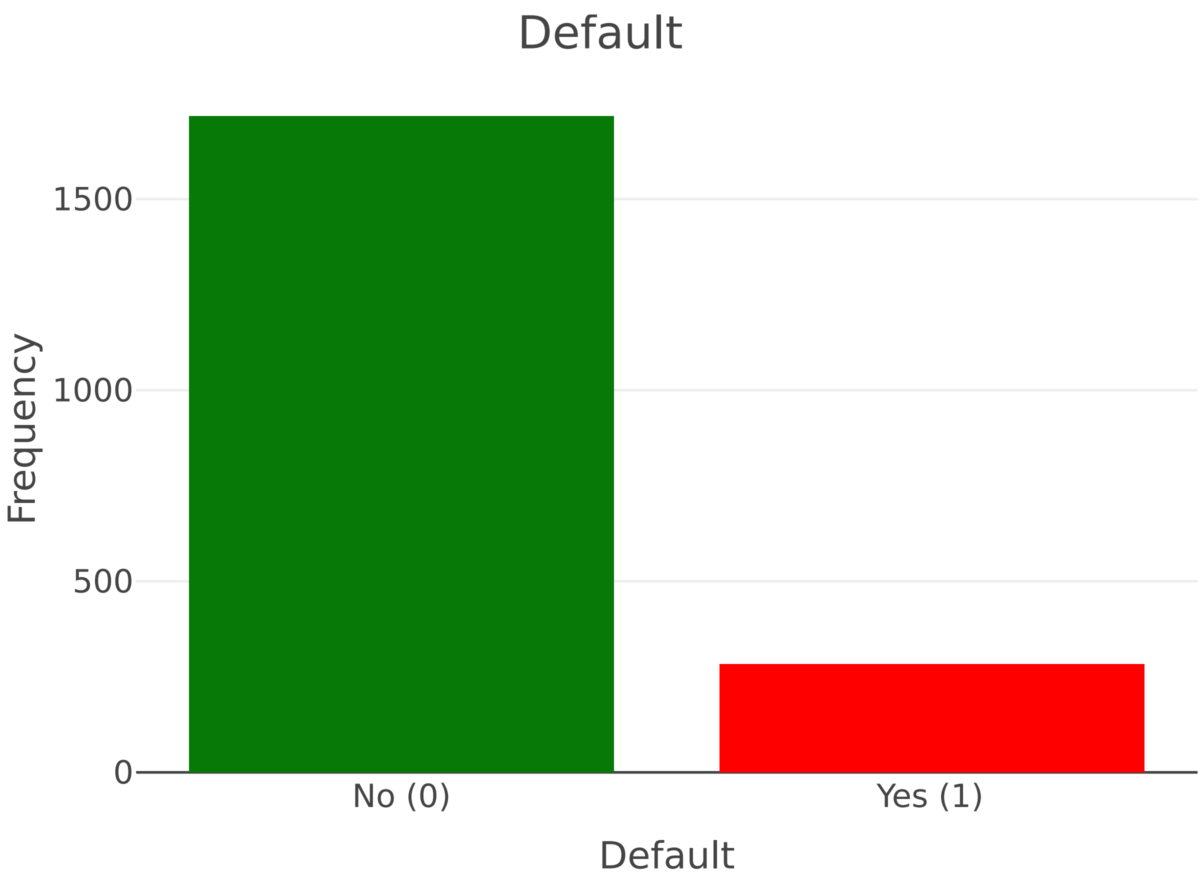
```
p1 <- plot_ly(df, x = ~age, type = "histogram", name = "Age") %>%
  layout(title = "Age", xaxis = list(title = "Age"), yaxis = list(title = "Count"), showlegend = FALSE)

p2 <- plot_ly(df, x = ~income, type = "histogram", name = "Income") %>%
  layout(title = "Income", xaxis = list(title = "Income"), yaxis = list(title = "Count"), showlegend = F

p3 <- plot_ly(df, x = ~loan, type = "histogram", name = "Loan") %>%
  layout(title = "Loan", xaxis = list(title = "Loan"), yaxis = list(title = "Count"), showlegend = FALS

subplot(p1, p2, p3,
  nrows = 2, margin = 0.07,
  titleX = TRUE, titleY = TRUE,
  shareX = FALSE, shareY = FALSE
) %>%
  layout(title = "Distribution of Explanatory Variables")
```

```
## `google-chrome` and `chromium-browser` were not found. Try setting the `CHROMOTE_CHROME` environment
```



The `age` and `income` variables have a similar uniform distribution, while the `loan` variable has an asymmetric positive distribution. This affirmation is confirmed by the summary statistics of the dataset and the above plots.

```
df %>%
  count(default) %>%
  plot_ly(
    x = ~ factor(default, levels = c(0, 1), labels = c("No (0)", "Yes (1)")),
    y = ~n,
    type = "bar",
    color = ~ factor(default),
    colors = c("#067906", "red"),
    name = "Default Variable"
  ) %>%
  layout(
    title = "Default",
    xaxis = list(title = "Default", type = "category"),
    yaxis = list(title = "Frequency"),
    showlegend = FALSE
  )
```




```r
corrPearson <- cor(df[, !names(df) %in% "default"], method = "pearson", use = "pairwise.complete.obs")
corrSpearman <- cor(df[, !names(df) %in% "default"], method = "spearman", use = "pairwise.complete.obs"
print(corrPearson)
```

```
##             income          age        loan
## income  1.00000000 -0.034000535 0.441116504
## age    -0.03400054  1.000000000 0.006440323
## loan    0.44111650  0.006440323 1.000000000
```

```r
print(corrSpearman)
```

```
##             income          age        loan
## income  1.00000000 -0.034486684 0.401601061
## age    -0.03448668  1.000000000 0.009835956
## loan    0.40160106  0.009835956 1.000000000
```

Linear Regression For Imputation data

For imputation, will be used the Linear Regression algorithm, which is a simple and effective methods for handling missing values. The Linear Regression algorithm works by finding the relationship between the target variable and the other features to predict the missing values.

```r
# Get the missing values in the 'age' column
ageNan <- df[is.na(df$age), ]
print(ageNan)
```

```
##      income age     loan default
## 29 59417.81  NA 2082.626       0
```

```
## 31 48528.85  NA 6155.785       0
## 32 23526.30  NA 2862.010       0
```

```r
# Remove rows with NaN in 'age' for training and testing
dfNN <- df[!is.na(df$age), ]
print(summary(dfNN))
```

```
##      income           age             loan              default
##  Min.   :20014   Min.   :18.06   Min.   :    1.378   Min.   :0.0000
##  1st Qu.:32805   1st Qu.:29.03   1st Qu.: 1936.813   1st Qu.:0.0000
##  Median :45789   Median :41.35   Median : 3977.287   Median :0.0000
##  Mean   :45334   Mean   :40.92   Mean   : 4445.488   Mean   :0.1417
##  3rd Qu.:57788   3rd Qu.:52.59   3rd Qu.: 6440.861   3rd Qu.:0.0000
##  Max.   :69996   Max.   :63.97   Max.   :13766.051   Max.   :1.0000
```

Split the dataset into training and testing sets

```r
# Split the dataset into training and testing sets
split <- sample.split(dfNN$age, SplitRatio = 0.9)
train <- subset(dfNN, split == TRUE)
test <- subset(dfNN, split == FALSE)
```

Normalize the numeric columns

```r
# Select numeric columns, excluding 'default'
numeric_cols <- sapply(train, is.numeric)
cols_for_stats <- names(train)[numeric_cols & names(train) != "default"]

# Calculate statistics only for the selected columns
means <- colMeans(train[, cols_for_stats], na.rm = TRUE)
sds <- apply(train[, cols_for_stats], 2, sd, na.rm = TRUE)

# Z-score normalization
for (c in colnames(train[(numeric_cols & names(train) != "default")])) {
  if (is.numeric(train[[c]])) {
    train[[c]] <- (train[[c]] - means[c]) / sds[c]
    test[[c]] <- (test[[c]] - means[c]) / sds[c]
    ageNan[[c]] <- (ageNan[[c]] - means[c]) / sds[c]
    df[[c]] <- (df[[c]] - means[c]) / sds[c]
  }
}

print(summary(train))
```

```
##      income             age               loan             default
##  Min.   :-1.75045   Min.   :-1.72182   Min.   :-1.4516   Min.   :0.0000
##  1st Qu.:-0.88670   1st Qu.:-0.90248   1st Qu.:-0.8233   1st Qu.:0.0000
##  Median : 0.02956   Median : 0.03059   Median :-0.1530   Median :0.0000
##  Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000   Mean   :0.1408
##  3rd Qu.: 0.87580   3rd Qu.: 0.88516   3rd Qu.: 0.6644   3rd Qu.:0.0000
##  Max.   : 1.72849   Max.   : 1.73190   Max.   : 3.0500   Max.   :1.0000
```

```r
print(summary(test))
```

```
##      income             age                loan              default
```

```
##  Min.   :-1.6667   Min.   :-1.70294   Min.   :-1.39934   Min.   :0.00
##  1st Qu.:-0.7079   1st Qu.:-0.78352   1st Qu.:-0.72853   1st Qu.:0.00
##  Median : 0.1134   Median : 0.01340   Median :-0.14067   Median :0.00
##  Mean   : 0.1189   Mean   :-0.01624   Mean   : 0.01773   Mean   :0.15
##  3rd Qu.: 0.9400   3rd Qu.: 0.80136   3rd Qu.: 0.55354   3rd Qu.:0.00
##  Max.   : 1.7076   Max.   : 1.72538   Max.   : 2.94447   Max.   :1.00
```

```
print(summary(ageNan))
```

```
##      income               age           loan              default
##  Min.   :-1.50601   Min.   : NA   Min.   :-0.77097   Min.   :0
##  1st Qu.:-0.63586   1st Qu.: NA   1st Qu.:-0.64353   1st Qu.:0
##  Median : 0.23429   Median : NA   Median :-0.51608   Median :0
##  Mean   :-0.09317   Mean   :NaN   Mean   :-0.24198   Mean   :0
##  3rd Qu.: 0.61325   3rd Qu.: NA   3rd Qu.: 0.02251   3rd Qu.:0
##  Max.   : 0.99222   Max.   : NA   Max.   : 0.56111   Max.   :0
##                     NA's   :3
```

Train the Linear Regression model

```
# Train a linear regression model to predict 'age'
model <- lm(age ~ ., data = train)
```

```
print(summary(model))
```

```
##
## Call:
## lm(formula = age ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.01295 -0.61196  0.01648  0.63902  2.08071
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.22180    0.02235   9.926  < 2e-16 ***
## income      -0.17646    0.02312  -7.634 3.67e-14 ***
## loan         0.29377    0.02499  11.757  < 2e-16 ***
## default     -1.57541    0.06457 -24.399  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8653 on 1793 degrees of freedom
## Multiple R-squared:  0.2524, Adjusted R-squared:  0.2512
## F-statistic: 201.8 on 3 and 1793 DF,  p-value: < 2.2e-16
```

Test the model

```
# Predict missing 'age' values in the test set
predictedAge <- predict(model, newdata = test)
```

```
# Mean Squared Error (MSE) for the predictions
mse <- 0
for (i in 1:length(predictedAge)) {
  mse = mse + (predictedAge[i] - test$age[i])^2
```

```r
}

mse = mse / length(predictedAge)
print(mse)
```

```
##       1
## 0.71681
```

Impute missing values in the original dataset

```r
# Impute missing values in the original dataset
ageNan$age <- predict(model, newdata = ageNan)

# Replace NaN values in the original dataset with the predicted values
df$age[is.na(df$age)] <- ageNan$age

# Reverse Z-score normalization for the imputed values
for (c in colnames(df[(numeric_cols & names(df) != "default")])) {
  if (is.numeric(df[[c]])) {
    df[[c]] <- (df[[c]] * sds[c]) + means[c]
    ageNan[[c]] <- (ageNan[[c]] * sds[c]) + means[c]
  }
}

print(head(ageNan))
```

```
##      income      age     loan default
## 29 59417.81 38.55642 2082.626       0
## 31 48528.85 45.53711 6155.785       0
## 32 23526.30 45.41288 2862.010       0
```

```r
print(summary(df))
```

```
##      income           age             loan              default
##  Min.   :20014   Min.   :18.06   Min.   :    1.378   Min.   :0.0000
##  1st Qu.:32796   1st Qu.:29.06   1st Qu.: 1939.709   1st Qu.:0.0000
##  Median :45789   Median :41.38   Median : 3974.719   Median :0.0000
##  Mean   :45332   Mean   :40.93   Mean   : 4444.370   Mean   :0.1415
##  3rd Qu.:57791   3rd Qu.:52.58   3rd Qu.: 6432.411   3rd Qu.:0.0000
##  Max.   :69996   Max.   :63.97   Max.   :13766.051   Max.   :1.0000
```