Credit Card: K-Means Clustering and Logistic Regression

Ruben Montano

2025-February

Data Upload

```
df_1 <- read.csv("application_record.csv")
df_2 <- read.csv("credit_record.csv")
```

Libraries

```
library(dplyr)
```

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

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

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

```
library(tidyr)
library(ggplot2)
library(cluster)
library(caTools)
library(nnet)
```

Data Preprocessing

Data set 2 (credit_record.csv) has multiple rows for the same ID number, this makes a little bit difficult if the two data sets are going to be merged by the ID number. In addition, the reading of the status are a little bit difficult when there are letters such as X and C and numbers from 1-5. These letters and numbers are mapped to a different value for easier readability. Moreover, new columns are added in order to reduce multiple rows using the ID number. The new columns added are latest status (latest_status), worst status (worst_status), number of months with no loans (num_no_loan_months), number of good months (num_good_months) and number of bad months (num_bad_months).

```r
# Convert STATUS to numeric values with shifting
df_2 <- df_2 %>%
  mutate(STATUS_NUM = case_when(
    STATUS == "X" ~ 0,  # No loan that month
    STATUS == "C" ~ 1,  # Paid off, no overdue
    STATUS == "0" ~ 2,  # 1-29 days overdue
    STATUS == "1" ~ 3,  # 30-59 days overdue
    STATUS == "2" ~ 4,  # 60-89 days overdue
    STATUS == "3" ~ 5,  # 90-119 days overdue
    STATUS == "4" ~ 6,  # 120-149 days overdue
    STATUS == "5" ~ 7,  # 150+ days overdue
    TRUE ~ NA_real_  # Handle unexpected values
  ))


# Aggregate credit history per ID
df_2_agg <- df_2 %>%
  group_by(ID) %>%
  summarize(
    num_months = n(),  # Total months recorded
    latest_status = STATUS_NUM[which.max(MONTHS_BALANCE)],  # Most recent status
    worst_status = max(STATUS_NUM, na.rm = TRUE),  # Worst recorded status
    num_no_loan_months = sum(STATUS_NUM == 0, na.rm = TRUE),  # Months with no loan
    num_good_months = sum(STATUS_NUM == 1, na.rm = TRUE),  # Months with full payment
    num_bad_months = sum(STATUS_NUM >= 3, na.rm = TRUE),  # Months overdue by 60+ days
  )
```

Merging Data

After data set 2 has been modified, then is merged with data set 1 by ID number.

```r
merged_df <- df_1 %>% left_join(df_2_agg, by = "ID")
```

Replacing NA values

Making sure number of months and number of good months NA values are replaced with 0, as the two columns will be used for clustering method.

```r
merged_df <- merged_df %>%
  mutate(
    num_months = replace_na(num_months,0),
    num_good_months = replace_na(num_good_months,0),
  )
```

Clustering with K-Means Method and Visualization

```r
X <- merged_df[, c(19, 23)] #Number of Months, Number of Good Months
```

```r
# Using elbow method
wcss = vector()
for (i in 1:10) wcss[i] = sum(kmeans(X, i)$withinss)
plot(x = 1:10,
     y = wcss,
     type = 'b',
     main = paste('The Elbow Method'),
     xlab = 'Number of clusters',
     ylab = 'WCSS')
```

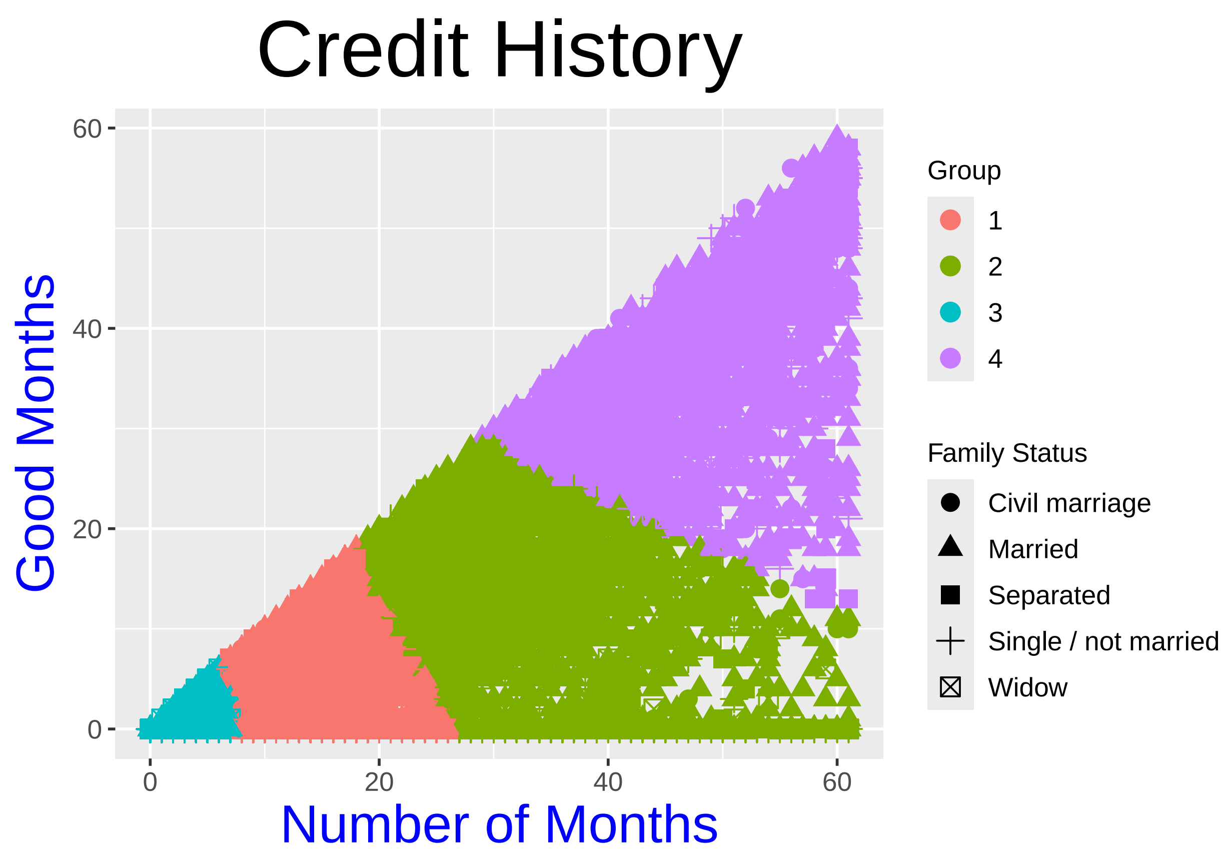

```r
set.seed(29)
kmeans = kmeans(x = X,
                centers = 4,
                iter.max = 100,
                nstart = 12)


# Visualizing the clusters using ggplot2
merged_df$Group <- as.factor(kmeans$cluster)
plot <- ggplot(dat = merged_df, aes(x = num_months , y = num_good_months)) +
  geom_point(aes(color = Group, shape = NAME_FAMILY_STATUS), size = 3) +
  xlab("Number of Months")+
  ylab("Good Months")+
  ggtitle("Credit History") +
  theme(axis.title.x = element_text(colour="Blue", size = 20),
```

```
        axis.title.y = element_text(colour = "Blue", size = 20),
        axis.text.x = element_text(size = 10),
        axis.text.y = element_text(size = 10),
        legend.title = element_text(size = 10),
        legend.text = element_text(size = 10),
        plot.title = element_text(colour = "Black", size = 30, hjust = 0.5))

plot$labels$shape = "Family Status"

plot
```



Classification using Logistic Regression Method, Accuracy and Confusion Matrix

For the classification method, the following columns are used: amount of total income (AMT_INCOME_TOTAL [6]), total number of days since the person was born (DAYS_BIRH[11]), amount of days employed (DAYS_EMPLOYED[12]), total amount of family members if any (COUNT_FAM_MEMBERS[18]), number of months (num_good_months [19]) and number of good months (num_bad_months[23]) and group (Group [25]) which is the now target/output. The merged data was further modified by changing the negative values of days employed to positive and if the person is not currently employed then mark it as 0, on the other hand for DAYS_BIRTH changed all values to positive.

```
# Transforming the DAYS_EMPLOYED column
merged_df <- merged_df %>%
  mutate(DAYS_EMPLOYED = ifelse(DAYS_EMPLOYED < 0, abs(DAYS_EMPLOYED), 0))
```

```r
# Transforming the DAYS_Birth column
merged_df <- merged_df %>%
  mutate(DAYS_BIRTH = if (TRUE) abs(DAYS_BIRTH))


# Training Linear Regression Method
dataset <- merged_df[, c(6, 11, 12, 18, 19, 23, 25)]
set.seed(123)
split = sample.split(dataset$Group, SplitRatio = 0.75)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)


# Feature Scaling
training_set[-7] = scale(training_set[-7])
test_set[-7] = scale(test_set[-7])

# Fitting Logistic Regression to the Training set
classifier <- multinom(Group ~ ., data = training_set)
```

```
## # weights:  32 (21 variable)
## initial  value 455975.782380
## iter  10 value 15954.020399
## iter  20 value 6005.628640
## iter  30 value 679.982222
## iter  40 value 625.644225
## iter  50 value 616.535138
## iter  60 value 429.826941
## iter  70 value 62.110660
## iter  80 value 21.227429
## iter  90 value 13.390241
## iter 100 value 10.741195
## final  value 10.741195
## stopped after 100 iterations
```

```r
# Predicting the Test set results
prob_pred <- predict(classifier, newdata = test_set[-7], type = 'class')

# Create the confusion matrix
confusion_matrix <- table(test_set$Group, prob_pred)
print(confusion_matrix)
```

```
##    prob_pred
##          1      2      3      4
##   1   3783      1      0      0
##   2      0   2261      0      4
##   3      0      0 102295      0
##   4      0      0      0   1296
```

Calculating Accuracy, Recall and F-1 Score

```r
# Calculate overall accuracy
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)

# Calculate precision, recall, and F1-score for each class
n_classes <- nrow(confusion_matrix)
metrics <- data.frame(Class = 1:n_classes,
                      Precision = NA,
                      Recall = NA,
                      F1_Score = NA)

for(i in 1:n_classes) {
    # Precision
    precision <- confusion_matrix[i,i] / sum(confusion_matrix[,i])
    # Recall
    recall <- confusion_matrix[i,i] / sum(confusion_matrix[i,])
    # F1 Score
    f1 <- 2 * (precision * recall) / (precision + recall)

    metrics[i,2:4] <- c(precision, recall, f1)
}

# Print the metrics
print("Overall Accuracy:")
```

```
## [1] "Overall Accuracy:"
```

```r
print(accuracy)
```

```
## [1] 0.9999544
```

```r
print("\nPer-class metrics:")
```

```
## [1] "\nPer-class metrics:"
```

```r
print(metrics)
```

```
##   Class Precision    Recall  F1_Score
## 1     1 1.0000000 0.9997357 0.9998678
## 2     2 0.9995579 0.9982340 0.9988955
## 3     3 1.0000000 1.0000000 1.0000000
## 4     4 0.9969231 1.0000000 0.9984592
```