Analysis

2023-05-23

### 1. Load and explore the data:

Load the dataset using the read.csv() function.

application\_record <- read\_csv("application\_record.csv")

## Rows: 438557 Columns: 18  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (8): CODE\_GENDER, FLAG\_OWN\_CAR, FLAG\_OWN\_REALTY, NAME\_INCOME\_TYPE, NAME...  
## dbl (10): ID, CNT\_CHILDREN, AMT\_INCOME\_TOTAL, DAYS\_BIRTH, DAYS\_EMPLOYED, FLA...  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

credit\_record <- read\_csv("credit\_record.csv")

## Rows: 1048575 Columns: 3  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (1): STATUS  
## dbl (2): ID, MONTHS\_BALANCE  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

# Join the two data frames by the ID column  
df <- inner\_join(application\_record , credit\_record, by = "ID")  
df |> glimpse()

## Rows: 777,715  
## Columns: 20  
## $ ID <dbl> 5008804, 5008804, 5008804, 5008804, 5008804, 50088…  
## $ CODE\_GENDER <chr> "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", …  
## $ FLAG\_OWN\_CAR <chr> "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", …  
## $ FLAG\_OWN\_REALTY <chr> "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", …  
## $ CNT\_CHILDREN <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…  
## $ AMT\_INCOME\_TOTAL <dbl> 427500, 427500, 427500, 427500, 427500, 427500, 42…  
## $ NAME\_INCOME\_TYPE <chr> "Working", "Working", "Working", "Working", "Worki…  
## $ NAME\_EDUCATION\_TYPE <chr> "Higher education", "Higher education", "Higher ed…  
## $ NAME\_FAMILY\_STATUS <chr> "Civil marriage", "Civil marriage", "Civil marriag…  
## $ NAME\_HOUSING\_TYPE <chr> "Rented apartment", "Rented apartment", "Rented ap…  
## $ DAYS\_BIRTH <dbl> -12005, -12005, -12005, -12005, -12005, -12005, -1…  
## $ DAYS\_EMPLOYED <dbl> -4542, -4542, -4542, -4542, -4542, -4542, -4542, -…  
## $ FLAG\_MOBIL <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,…  
## $ FLAG\_WORK\_PHONE <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,…  
## $ FLAG\_PHONE <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…  
## $ FLAG\_EMAIL <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…  
## $ OCCUPATION\_TYPE <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…  
## $ CNT\_FAM\_MEMBERS <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,…  
## $ MONTHS\_BALANCE <dbl> 0, -1, -2, -3, -4, -5, -6, -7, -8, -9, -10, -11, -…  
## $ STATUS <chr> "C", "C", "C", "C", "C", "C", "C", "C", "C", "C", …

Display representative portions of the data using functions like head(), tail(), or summary() to get an overview of the data. First, Convert all character data type columns to factors.

# Convert all character columns to factor  
df = df %>%  
 mutate\_if(is.character, as.factor)  
summary(df)

## ID CODE\_GENDER FLAG\_OWN\_CAR FLAG\_OWN\_REALTY CNT\_CHILDREN   
## Min. :5008804 F:518851 N:473355 N:264767 Min. : 0.0000   
## 1st Qu.:5044568 M:258864 Y:304360 Y:512948 1st Qu.: 0.0000   
## Median :5069530 Median : 0.0000   
## Mean :5078743 Mean : 0.4281   
## 3rd Qu.:5115551 3rd Qu.: 1.0000   
## Max. :5150487 Max. :19.0000   
##   
## AMT\_INCOME\_TOTAL NAME\_INCOME\_TYPE   
## Min. : 27000 Commercial associate:183385   
## 1st Qu.: 121500 Pensioner :128392   
## Median : 162000 State servant : 65437   
## Mean : 188535 Student : 337   
## 3rd Qu.: 225000 Working :400164   
## Max. :1575000   
##   
## NAME\_EDUCATION\_TYPE NAME\_FAMILY\_STATUS  
## Academic degree : 837 Civil marriage : 60342   
## Higher education :213633 Married :546619   
## Incomplete higher : 30329 Separated : 45255   
## Lower secondary : 8655 Single / not married: 94335   
## Secondary / secondary special:524261 Widow : 31164   
##   
##   
## NAME\_HOUSING\_TYPE DAYS\_BIRTH DAYS\_EMPLOYED FLAG\_MOBIL  
## Co-op apartment : 3655 Min. :-25152 Min. :-15713 Min. :1   
## House / apartment :697151 1st Qu.:-19453 1st Qu.: -3292 1st Qu.:1   
## Municipal apartment: 24640 Median :-15760 Median : -1682 Median :1   
## Office apartment : 5636 Mean :-16125 Mean : 57776 Mean :1   
## Rented apartment : 10898 3rd Qu.:-12716 3rd Qu.: -431 3rd Qu.:1   
## With parents : 35735 Max. : -7489 Max. :365243 Max. :1   
##   
## FLAG\_WORK\_PHONE FLAG\_PHONE FLAG\_EMAIL OCCUPATION\_TYPE   
## Min. :0.0000 Min. :0.000 Min. :0.00000 Laborers :131572   
## 1st Qu.:0.0000 1st Qu.:0.000 1st Qu.:0.00000 Core staff : 77112   
## Median :0.0000 Median :0.000 Median :0.00000 Sales staff: 70362   
## Mean :0.2318 Mean :0.301 Mean :0.09168 Managers : 67738   
## 3rd Qu.:0.0000 3rd Qu.:1.000 3rd Qu.:0.00000 Drivers : 47678   
## Max. :1.0000 Max. :1.000 Max. :1.00000 (Other) :143205   
## NA's :240048   
## CNT\_FAM\_MEMBERS MONTHS\_BALANCE STATUS   
## Min. : 1.000 Min. :-60.00 C :329536   
## 1st Qu.: 2.000 1st Qu.:-29.00 0 :290654   
## Median : 2.000 Median :-17.00 X :145950   
## Mean : 2.209 Mean :-19.37 1 : 8747   
## 3rd Qu.: 3.000 3rd Qu.: -8.00 5 : 1527   
## Max. :20.000 Max. : 0.00 2 : 801   
## (Other): 500

Check for missing values using the is.na() function and handle them appropriately, such as by imputing missing values or removing rows with missing values.

# Check for missing values and clean the data  
missing\_values <- df %>%  
 summarize\_all(~ sum(is.na(.)))  
missing\_values

## # A tibble: 1 × 20  
## ID CODE\_GENDER FLAG\_OWN\_CAR FLAG\_OWN\_REALTY CNT\_CHILDREN AMT\_INCOME\_TOTAL  
## <int> <int> <int> <int> <int> <int>  
## 1 0 0 0 0 0 0  
## # ℹ 14 more variables: NAME\_INCOME\_TYPE <int>, NAME\_EDUCATION\_TYPE <int>,  
## # NAME\_FAMILY\_STATUS <int>, NAME\_HOUSING\_TYPE <int>, DAYS\_BIRTH <int>,  
## # DAYS\_EMPLOYED <int>, FLAG\_MOBIL <int>, FLAG\_WORK\_PHONE <int>,  
## # FLAG\_PHONE <int>, FLAG\_EMAIL <int>, OCCUPATION\_TYPE <int>,  
## # CNT\_FAM\_MEMBERS <int>, MONTHS\_BALANCE <int>, STATUS <int>

There were no missing values for this dataset. Therefore, we move on to visualizing the dataset to inspect for outliers.

Check for outliers by visualizing the distribution of variables using plots like box plots or histograms. Decide on a suitable approach to handle outliers, such as removing them or transforming the data.

Dealing with Outliers in numerical columns.

# Filter numerical columns  
num\_cols <- df %>%  
 select(where(is.numeric)) %>%  
 names()  
  
# Define a function to handle outliers  
handle\_outliers <- function(col) {  
 # Calculate quartiles and IQR  
 q1 <- quantile(col, 0.25)  
 q3 <- quantile(col, 0.75)  
 iqr <- q3 - q1  
   
 # Define lower and upper bounds for outliers  
 lower\_bound <- q1 - 1.5 \* iqr  
 upper\_bound <- q3 + 1.5 \* iqr  
   
 # Replace outliers with NA  
 col <- ifelse(col < lower\_bound | col > upper\_bound, NA, col)  
   
 # Return the modified column  
 return(col)  
}  
  
# Handle outliers in each numerical column  
data\_outliers\_removed <- df %>%  
 mutate(across(num\_cols, handle\_outliers))

## Warning: There was 1 warning in `mutate()`.  
## ℹ In argument: `across(num\_cols, handle\_outliers)`.  
## Caused by warning:  
## ! Using an external vector in selections was deprecated in tidyselect 1.1.0.  
## ℹ Please use `all\_of()` or `any\_of()` instead.  
## # Was:  
## data %>% select(num\_cols)  
##   
## # Now:  
## data %>% select(all\_of(num\_cols))  
##   
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.

# Display the updated summary of the dataset  
summary(data\_outliers\_removed)

## ID CODE\_GENDER FLAG\_OWN\_CAR FLAG\_OWN\_REALTY CNT\_CHILDREN   
## Min. :5008804 F:518851 N:473355 N:264767 Min. :0.000   
## 1st Qu.:5044568 M:258864 Y:304360 Y:512948 1st Qu.:0.000   
## Median :5069530 Median :0.000   
## Mean :5078743 Mean :0.387   
## 3rd Qu.:5115551 3rd Qu.:1.000   
## Max. :5150487 Max. :2.000   
## NA's :11039   
## AMT\_INCOME\_TOTAL NAME\_INCOME\_TYPE   
## Min. : 27000 Commercial associate:183385   
## 1st Qu.:121500 Pensioner :128392   
## Median :157500 State servant : 65437   
## Mean :174385 Student : 337   
## 3rd Qu.:225000 Working :400164   
## Max. :378000   
## NA's :33987   
## NAME\_EDUCATION\_TYPE NAME\_FAMILY\_STATUS  
## Academic degree : 837 Civil marriage : 60342   
## Higher education :213633 Married :546619   
## Incomplete higher : 30329 Separated : 45255   
## Lower secondary : 8655 Single / not married: 94335   
## Secondary / secondary special:524261 Widow : 31164   
##   
##   
## NAME\_HOUSING\_TYPE DAYS\_BIRTH DAYS\_EMPLOYED FLAG\_MOBIL  
## Co-op apartment : 3655 Min. :-25152 Min. :-7566 Min. :1   
## House / apartment :697151 1st Qu.:-19453 1st Qu.:-3347 1st Qu.:1   
## Municipal apartment: 24640 Median :-15760 Median :-2043 Median :1   
## Office apartment : 5636 Mean :-16125 Mean :-2380 Mean :1   
## Rented apartment : 10898 3rd Qu.:-12716 3rd Qu.: -991 3rd Qu.:1   
## With parents : 35735 Max. : -7489 Max. : -17 Max. :1   
## NA's :163348   
## FLAG\_WORK\_PHONE FLAG\_PHONE FLAG\_EMAIL OCCUPATION\_TYPE   
## Min. :0 Min. :0.000 Min. :0 Laborers :131572   
## 1st Qu.:0 1st Qu.:0.000 1st Qu.:0 Core staff : 77112   
## Median :0 Median :0.000 Median :0 Sales staff: 70362   
## Mean :0 Mean :0.301 Mean :0 Managers : 67738   
## 3rd Qu.:0 3rd Qu.:1.000 3rd Qu.:0 Drivers : 47678   
## Max. :0 Max. :1.000 Max. :0 (Other) :143205   
## NA's :180288 NA's :71297 NA's :240048   
## CNT\_FAM\_MEMBERS MONTHS\_BALANCE STATUS   
## Min. :1.000 Min. :-60.00 C :329536   
## 1st Qu.:2.000 1st Qu.:-29.00 0 :290654   
## Median :2.000 Median :-17.00 X :145950   
## Mean :2.166 Mean :-19.37 1 : 8747   
## 3rd Qu.:3.000 3rd Qu.: -8.00 5 : 1527   
## Max. :4.000 Max. : 0.00 2 : 801   
## NA's :10631 (Other): 500

The selected text describes a process for determining the start month for each credit card applicant based on the MONTHS\_BALANCE column in the data. Since the data file does not contain information about the credit card open date, the earliest MONTHS\_BALANCE value is assumed to be the start month for an account. The data is then rearranged so that the status for each month is available starting from month 0 (the start month), month 1 (one month from the start), and so on.

In other words, this code aims to use the MONTHS\_BALANCE column to determine the start month for each credit card applicant and reorganize the data accordingly.

It is assumed that the data was extracted on January 1st, 2020 and the goal is to determine the calendar start month for each account. Having the calendar account open date could be useful for some analyses.

library(dplyr)  
  
credit\_card\_first\_month <- df %>%  
 group\_by(ID) %>%  
 summarize(start\_month = min(MONTHS\_BALANCE)) %>%  
 ungroup()  
  
head(credit\_card\_first\_month)

## # A tibble: 6 × 2  
## ID start\_month  
## <dbl> <dbl>  
## 1 5008804 -15  
## 2 5008805 -14  
## 3 5008806 -29  
## 4 5008808 -4  
## 5 5008809 -26  
## 6 5008810 -26

It is assumed that the data was extracted on January 1st, 2020 and the goal is to determine the calendar start month for each account. Having the calendar account open date could be useful for some analyses.

library(dplyr)  
library(lubridate)  
  
credit\_card\_first\_month <- credit\_card\_first\_month %>%  
 mutate(account\_open\_month = as.Date("2020-01-01")) %>%  
 mutate(account\_open\_month = account\_open\_month + months(start\_month)) %>%  
 mutate(account\_open\_month = format(account\_open\_month, "%b-%Y"))  
  
credit\_card\_first\_month |> head()

## # A tibble: 6 × 3  
## ID start\_month account\_open\_month  
## <dbl> <dbl> <chr>   
## 1 5008804 -15 Oct-2018   
## 2 5008805 -14 Nov-2018   
## 3 5008806 -29 Aug-2017   
## 4 5008808 -4 Sep-2019   
## 5 5008809 -26 Nov-2017   
## 6 5008810 -26 Nov-2017

Account 5008804 was opened in October 2018 and account 5008805 was opened in November 2018. The start month column needs to be added to the credit status table.

library(dplyr)  
  
credit\_start\_status <- left\_join(credit\_card\_first\_month, credit\_record, by = "ID") %>%  
 mutate(start\_month = abs(start\_month) + MONTHS\_BALANCE)  
  
  
credit\_start\_status |> head()

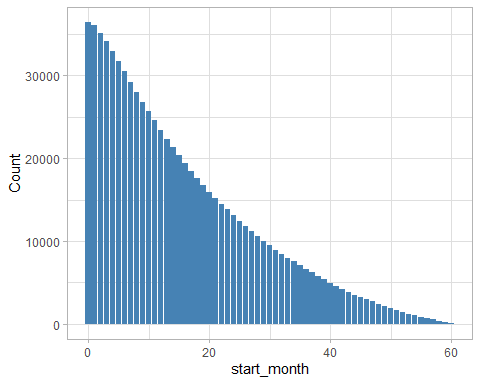
## # A tibble: 6 × 5  
## ID start\_month account\_open\_month MONTHS\_BALANCE STATUS  
## <dbl> <dbl> <chr> <dbl> <chr>   
## 1 5008804 15 Oct-2018 0 C   
## 2 5008804 14 Oct-2018 -1 C   
## 3 5008804 13 Oct-2018 -2 C   
## 4 5008804 12 Oct-2018 -3 C   
## 5 5008804 11 Oct-2018 -4 C   
## 6 5008804 10 Oct-2018 -5 C

The status can be viewed by month since the start and the significance of what has been accomplished. Now, across all acquisition months, portfolio performance can be determined for months 1, 2, 5, 15, and 20 from their respective account open month. The distribution of accounts by status across each month is calculated by first finding the accounts by month and status code.

credit\_start\_status %>%  
 count(STATUS)

## # A tibble: 8 × 2  
## STATUS n  
## <chr> <int>  
## 1 0 290654  
## 2 1 8747  
## 3 2 801  
## 4 3 286  
## 5 4 214  
## 6 5 1527  
## 7 C 329536  
## 8 X 145950

accounts\_counts <- credit\_start\_status %>%  
 count(start\_month) %>%  
 as\_tibble()  
  
ggplot(accounts\_counts, aes(x = start\_month, y = n)) +  
 geom\_bar(stat = "identity", fill = "steelblue") +  
 labs(x = "start\_month", y = "Count") +  
 theme\_minimal()+ theme\_light()

 The goal is to calculate the percentage of bad rate for the entire portfolio across all account open months. This will help determine the period during which the overall bad rate remains stable. It is important to note that only a small number of credit card accounts were opened in the early months and may not be relevant for modeling purposes. The distribution of bad rate for these accounts can be checked.

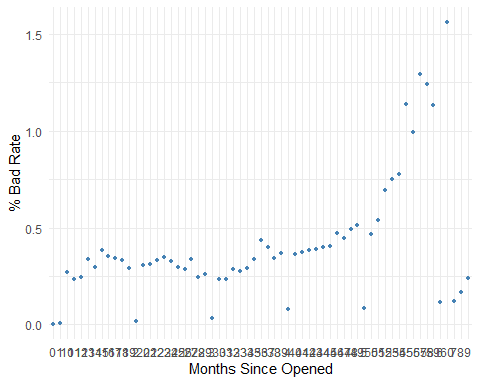
month\_status\_counts <- credit\_start\_status %>%  
 count(start\_month, STATUS) %>%  
 rename(counts = n)  
  
month\_counts <- credit\_start\_status %>%  
 count(start\_month) %>%  
 rename(month\_counts = n)  
  
month\_status\_pct <- month\_status\_counts %>%  
 left\_join(month\_counts, by = "start\_month") %>%  
 mutate(status\_pct = counts / month\_counts \* 100) %>%  
 select(start\_month, STATUS, status\_pct)

month\_status\_pct1 <- month\_status\_pct %>%  
 pivot\_wider(names\_from = STATUS, values\_from = status\_pct) %>%  
 replace(is.na(.), 0) %>%  
 mutate(start\_month = as.character(start\_month))

ggplot(month\_status\_pct1, aes(x = start\_month, y = `4` + `5`)) +  
 geom\_line(color = "steelblue", size = 2) +  
 geom\_point(color = "steelblue", size = 1) +  
 labs(x = "Months Since Opened", y = "% Bad Rate")+ theme\_minimal()

## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
## ℹ Please use `linewidth` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

## `geom\_line()`: Each group consists of only one observation.  
## ℹ Do you need to adjust the group aesthetic?



The bad rate increases significantly for accounts that have been open for more than 50 months. These are accounts that were opened during the initial days of operations. It may be a good idea to exclude these accounts.

The bad rate stabilizes after 18 months from the start and this period can be considered as a performance window. Accounts that become bad within the first 18 months will be classified as bad and the rest as good. There may be differences in performance, such as the percentage of bad rate by acquisition month, but this is not being explored further.

Accounts with a status of 4 or 5 in the first 18 months will be classified as bad and the rest as good. Start months less than 18 will be selected and the maximum status for each credit card account will be determined. Accounts with a status of 4 or 5 will be classified as bad and the rest as good.

credit\_record %>%  
 count(STATUS)

## # A tibble: 8 × 2  
## STATUS n  
## <chr> <int>  
## 1 0 383120  
## 2 1 11090  
## 3 2 868  
## 4 3 320  
## 5 4 223  
## 6 5 1693  
## 7 C 442031  
## 8 X 209230

credit\_start\_status %>%  
 group\_by(STATUS) %>%  
 summarize(count = n())

## # A tibble: 8 × 2  
## STATUS count  
## <chr> <int>  
## 1 0 290654  
## 2 1 8747  
## 3 2 801  
## 4 3 286  
## 5 4 214  
## 6 5 1527  
## 7 C 329536  
## 8 X 145950

credit\_start\_status1 <- credit\_start\_status %>%  
 filter(STATUS != 'X' & STATUS != 'C')  
  
credit\_start\_status1 <- credit\_start\_status1 %>%  
 mutate(status = STATUS)  
  
credit\_start\_status1 <- credit\_start\_status1 %>%  
 filter(start\_month <= 18,  
 status != 'C',  
 status != 'X') %>%  
 select(ID, start\_month, status)  
  
credit\_start\_status1

## # A tibble: 263,501 × 3  
## ID start\_month status  
## <dbl> <dbl> <chr>   
## 1 5008804 2 1   
## 2 5008804 1 0   
## 3 5008805 2 1   
## 4 5008805 1 0   
## 5 5008806 18 0   
## 6 5008806 16 0   
## 7 5008806 9 0   
## 8 5008806 5 0   
## 9 5008806 1 0   
## 10 5008808 4 0   
## # ℹ 263,491 more rows

status <- credit\_start\_status1 %>%  
 group\_by(ID) %>%  
 summarize(max\_status = max(status)) %>%  
 ungroup()  
  
status\_summary <- status %>%  
 group\_by(max\_status) %>%  
 summarize(count = n())  
  
status\_summary

## # A tibble: 6 × 2  
## max\_status count  
## <chr> <int>  
## 1 0 27795  
## 2 1 3464  
## 3 2 287  
## 4 3 70  
## 5 4 28  
## 6 5 147

status <- status %>%  
 mutate(label = ifelse(as.integer(max\_status) >= 4, 1, 0)) %>%  
 ungroup()  
  
status\_summary <- status %>%  
 group\_by(label) %>%  
 summarize(count = n())  
  
status\_summary

## # A tibble: 2 × 2  
## label count  
## <dbl> <int>  
## 1 0 31616  
## 2 1 175

status\_summary <- status %>%  
 group\_by(label) %>%  
 summarize(count = n()) %>%  
 mutate(percentage = count / length(status$label) \* 100)  
  
status\_summary

## # A tibble: 2 × 3  
## label count percentage  
## <dbl> <int> <dbl>  
## 1 0 31616 99.4   
## 2 1 175 0.550

The data is highly imbalanced with a bad rate of 0.47%. A biased sample can be created by taking all observations of Label 1 and a small percentage of observations from Label 0. The goal is to increase the bad rate to around 10%, so in the final sample there will be 189 observations for Label 1 and 1701 for Label 0. The next step is to randomly select 1701 observations from a total of 39562.

label\_1 <- status %>%  
 filter(label == 1)  
  
label\_0 <- status %>%  
 filter(label == 0)  
  
label\_0\_biased <- label\_0 %>%  
 sample\_n(1701, replace = FALSE)  
  
labels\_biased <- bind\_rows(label\_1, label\_0\_biased) %>%  
 select(ID, label)  
  
labels\_biased |> head()

## # A tibble: 6 × 2  
## ID label  
## <dbl> <dbl>  
## 1 5008827 1  
## 2 5009744 1  
## 3 5009746 1  
## 4 5009749 1  
## 5 5009752 1  
## 6 5009753 1

model\_df <- merge(labels\_biased, application\_record, by = "ID", all = FALSE)  
nrow(model\_df)

## [1] 1876

model\_df |> tail()

## ID label CODE\_GENDER FLAG\_OWN\_CAR FLAG\_OWN\_REALTY CNT\_CHILDREN  
## 1871 5150180 0 F N Y 0  
## 1872 5150304 0 F Y Y 0  
## 1873 5150305 0 F Y Y 0  
## 1874 5150318 0 F Y Y 0  
## 1875 5150389 0 F Y Y 1  
## 1876 5150406 0 M Y Y 2  
## AMT\_INCOME\_TOTAL NAME\_INCOME\_TYPE NAME\_EDUCATION\_TYPE  
## 1871 112500 Pensioner Secondary / secondary special  
## 1872 76500 Pensioner Secondary / secondary special  
## 1873 76500 Pensioner Secondary / secondary special  
## 1874 225000 Commercial associate Higher education  
## 1875 315000 Commercial associate Secondary / secondary special  
## 1876 157500 Working Secondary / secondary special  
## NAME\_FAMILY\_STATUS NAME\_HOUSING\_TYPE DAYS\_BIRTH DAYS\_EMPLOYED  
## 1871 Married House / apartment -20304 365243  
## 1872 Married House / apartment -21124 365243  
## 1873 Married House / apartment -21124 365243  
## 1874 Single / not married With parents -9690 -1193  
## 1875 Married Municipal apartment -16095 -7979  
## 1876 Married House / apartment -14312 -4480  
## FLAG\_MOBIL FLAG\_WORK\_PHONE FLAG\_PHONE FLAG\_EMAIL OCCUPATION\_TYPE  
## 1871 1 0 0 0 <NA>  
## 1872 1 0 0 0 <NA>  
## 1873 1 0 0 0 <NA>  
## 1874 1 0 0 0 Core staff  
## 1875 1 0 0 0 Medicine staff  
## 1876 1 1 1 0 Laborers  
## CNT\_FAM\_MEMBERS  
## 1871 2  
## 1872 2  
## 1873 2  
## 1874 1  
## 1875 3  
## 1876 4

label\_percent <- table(model\_df$label) \* 100 / length(model\_df$label)  
label\_percent

##   
## 0 1   
## 90.671642 9.328358

missing\_values\_table <- function(df) {  
 df %>%  
 summarise\_all(~ sum(is.na(.))) %>%  
 gather(key = "Column", value = "Missing\_Values") %>%  
 mutate(`%\_of\_Total\_Values` = 100 \* Missing\_Values / nrow(df)) %>%  
 filter(Missing\_Values > 0) %>%  
 arrange(desc(`%\_of\_Total\_Values`)) %>%  
 select(Column, Missing\_Values, `%\_of\_Total\_Values`)  
}  
  
missing\_values\_table(df)

## # A tibble: 1 × 3  
## Column Missing\_Values `%\_of\_Total\_Values`  
## <chr> <int> <dbl>  
## 1 OCCUPATION\_TYPE 240048 30.9

library(tidyr)  
library(purrr)  
  
featureType <- function(df) {  
 df %>%  
 summarise\_all(~ {  
 uniq <- n\_distinct(.)  
 if (nrow(df) > 10) {  
 if (is.numeric(.)) {  
 if (uniq == 1) {  
 'Unary'  
 } else if (uniq == 2) {  
 'Binary'  
 } else if (nrow(df) / uniq > 3 && uniq > 5) {  
 'Continuous'  
 } else {  
 'Continuous-Ordinal'  
 }  
 } else {  
 if (uniq == 1) {  
 'Unary'  
 } else if (uniq == 2) {  
 'Binary'  
 } else {  
 'Categorical-Nominal'  
 }  
 }  
 } else {  
 if (is.numeric(.)) {  
 'Numeric'  
 } else {  
 'Non-numeric'  
 }  
 }  
 }) %>%  
 gather(key = 'Feature', value = 'BaseFeatureType') %>%  
 mutate(AnalysisFeatureType = case\_when(  
 grepl('^Unary$', BaseFeatureType) ~ BaseFeatureType,  
 grepl('^Binary$', BaseFeatureType) ~ BaseFeatureType,  
 grepl('^Continuous', BaseFeatureType) ~ 'Continuous',  
 grepl('^Categorical', BaseFeatureType) ~ 'Categorical-Nominal',  
 TRUE ~ BaseFeatureType  
 )) %>%  
 select(Feature, BaseFeatureType, AnalysisFeatureType)  
}  
  
featureType(df)

## # A tibble: 20 × 3  
## Feature BaseFeatureType AnalysisFeatureType  
## <chr> <chr> <chr>   
## 1 ID Continuous Continuous   
## 2 CODE\_GENDER Binary Binary   
## 3 FLAG\_OWN\_CAR Binary Binary   
## 4 FLAG\_OWN\_REALTY Binary Binary   
## 5 CNT\_CHILDREN Continuous Continuous   
## 6 AMT\_INCOME\_TOTAL Continuous Continuous   
## 7 NAME\_INCOME\_TYPE Categorical-Nominal Categorical-Nominal  
## 8 NAME\_EDUCATION\_TYPE Categorical-Nominal Categorical-Nominal  
## 9 NAME\_FAMILY\_STATUS Categorical-Nominal Categorical-Nominal  
## 10 NAME\_HOUSING\_TYPE Categorical-Nominal Categorical-Nominal  
## 11 DAYS\_BIRTH Continuous Continuous   
## 12 DAYS\_EMPLOYED Continuous Continuous   
## 13 FLAG\_MOBIL Unary Unary   
## 14 FLAG\_WORK\_PHONE Binary Binary   
## 15 FLAG\_PHONE Binary Binary   
## 16 FLAG\_EMAIL Binary Binary   
## 17 OCCUPATION\_TYPE Categorical-Nominal Categorical-Nominal  
## 18 CNT\_FAM\_MEMBERS Continuous Continuous   
## 19 MONTHS\_BALANCE Continuous Continuous   
## 20 STATUS Categorical-Nominal Categorical-Nominal

library(dplyr)  
  
model\_df <- merge(model\_df, credit\_card\_first\_month %>% select(ID, account\_open\_month), by = "ID")  
nrow(model\_df)

## [1] 1876

1. Split the data into training and testing sets:
   * Use the sample() function to randomly split the dataset into a training set (70%) and a testing set (30%).

# Set the seed for reproducibility  
set.seed(123)  
model\_df$label <- as.factor(model\_df$label)  
# Split the data into a training set and a testing set  
data\_split <- initial\_split(model\_df, prop = 0.7)  
train\_data <- training(data\_split)  
test\_data <- testing(data\_split)

### 3. Build the logistic regression model:

Define the formula for the logistic regression model using the glm() function. Run the model using the training dataset. Interpret the results, particularly the p-values, which indicate the significance of each predictor variable in predicting loan approval.

# Check for missing values  
na <- model\_df %>%  
 summarise\_all(~sum(is.na(.)))  
  
na

## ID label CODE\_GENDER FLAG\_OWN\_CAR FLAG\_OWN\_REALTY CNT\_CHILDREN  
## 1 0 0 0 0 0 0  
## AMT\_INCOME\_TOTAL NAME\_INCOME\_TYPE NAME\_EDUCATION\_TYPE NAME\_FAMILY\_STATUS  
## 1 0 0 0 0  
## NAME\_HOUSING\_TYPE DAYS\_BIRTH DAYS\_EMPLOYED FLAG\_MOBIL FLAG\_WORK\_PHONE  
## 1 0 0 0 0 0  
## FLAG\_PHONE FLAG\_EMAIL OCCUPATION\_TYPE CNT\_FAM\_MEMBERS account\_open\_month  
## 1 0 0 585 0 0

# Clean the data (assuming you want to remove rows with missing values)  
clean\_data <- model\_df %>%  
 drop\_na()

# Split the data into a training set and testing set  
set.seed(123) # For reproducibility  
train\_indices <- sample(nrow(clean\_data), floor(0.7 \* nrow(clean\_data)))  
  
train\_data <- clean\_data[train\_indices, ]  
test\_data <- clean\_data[-train\_indices, ]

# Define the formula for the logistic regression model  
formula <- formula(label ~. )  
  
# Run the logistic regression model  
model <- glm(formula, data = train\_data, family = binomial)

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

# Interpret the results  
summary(model)

##   
## Call:  
## glm(formula = formula, family = binomial, data = train\_data)  
##   
## Coefficients: (1 not defined because of singularities)  
## Estimate Std. Error z value  
## (Intercept) -4.111e+01 1.304e+04 -0.003  
## ID 1.059e-06 3.672e-06 0.288  
## CODE\_GENDERM 8.690e-01 3.667e-01 2.370  
## FLAG\_OWN\_CARY -3.614e-02 3.247e-01 -0.111  
## FLAG\_OWN\_REALTYY 7.457e-02 3.277e-01 0.228  
## CNT\_CHILDREN -5.581e-01 1.172e+00 -0.476  
## AMT\_INCOME\_TOTAL 1.588e-06 1.375e-06 1.156  
## NAME\_INCOME\_TYPEPensioner 2.269e+01 7.007e+03 0.003  
## NAME\_INCOME\_TYPEState servant -2.151e+00 8.495e-01 -2.532  
## NAME\_INCOME\_TYPEStudent -1.194e+00 1.114e+04 0.000  
## NAME\_INCOME\_TYPEWorking -9.696e-01 3.297e-01 -2.941  
## NAME\_EDUCATION\_TYPEHigher education 1.703e+01 1.075e+04 0.002  
## NAME\_EDUCATION\_TYPEIncomplete higher 1.491e+01 1.075e+04 0.001  
## NAME\_EDUCATION\_TYPELower secondary 2.034e+01 1.075e+04 0.002  
## NAME\_EDUCATION\_TYPESecondary / secondary special 1.663e+01 1.075e+04 0.002  
## NAME\_FAMILY\_STATUSMarried 1.429e+00 8.153e-01 1.753  
## NAME\_FAMILY\_STATUSSeparated 3.162e+00 1.508e+00 2.097  
## NAME\_FAMILY\_STATUSSingle / not married 2.977e+00 1.373e+00 2.168  
## NAME\_FAMILY\_STATUSWidow 2.440e+00 1.823e+00 1.339  
## NAME\_HOUSING\_TYPEHouse / apartment -3.604e+00 4.098e+00 -0.879  
## NAME\_HOUSING\_TYPEMunicipal apartment -2.792e+00 4.132e+00 -0.676  
## NAME\_HOUSING\_TYPEOffice apartment -2.703e+00 4.343e+00 -0.622  
## NAME\_HOUSING\_TYPERented apartment -2.206e+01 1.976e+03 -0.011  
## NAME\_HOUSING\_TYPEWith parents -4.772e+00 4.193e+00 -1.138  
## DAYS\_BIRTH -5.650e-05 4.894e-05 -1.155  
## DAYS\_EMPLOYED 1.485e-04 8.648e-05 1.717  
## FLAG\_MOBIL NA NA NA  
## FLAG\_WORK\_PHONE 2.938e-01 3.745e-01 0.785  
## FLAG\_PHONE -3.991e-01 3.592e-01 -1.111  
## FLAG\_EMAIL -2.961e-01 5.450e-01 -0.543  
## OCCUPATION\_TYPECleaning staff -1.648e+01 2.187e+03 -0.008  
## OCCUPATION\_TYPECooking staff 9.741e-01 9.827e-01 0.991  
## OCCUPATION\_TYPECore staff 1.234e+00 7.270e-01 1.697  
## OCCUPATION\_TYPEDrivers -8.582e-03 8.251e-01 -0.010  
## OCCUPATION\_TYPEHigh skill tech staff 4.193e-01 8.499e-01 0.493  
## OCCUPATION\_TYPEHR staff -1.564e+01 6.692e+03 -0.002  
## OCCUPATION\_TYPEIT staff 5.457e+00 1.686e+00 3.237  
## OCCUPATION\_TYPELaborers 3.233e-01 7.144e-01 0.453  
## OCCUPATION\_TYPELow-skill Laborers 1.339e+00 1.953e+00 0.686  
## OCCUPATION\_TYPEManagers 1.481e-01 7.715e-01 0.192  
## OCCUPATION\_TYPEMedicine staff 4.330e-02 1.060e+00 0.041  
## OCCUPATION\_TYPEPrivate service staff -1.657e+01 2.483e+03 -0.007  
## OCCUPATION\_TYPERealty agents -1.762e+01 1.075e+04 -0.002  
## OCCUPATION\_TYPESales staff -4.721e-02 7.506e-01 -0.063  
## OCCUPATION\_TYPESecretaries -1.642e+01 4.078e+03 -0.004  
## OCCUPATION\_TYPESecurity staff 1.581e+00 9.291e-01 1.702  
## OCCUPATION\_TYPEWaiters/barmen staff 2.571e+00 1.564e+00 1.644  
## CNT\_FAM\_MEMBERS 7.044e-01 1.144e+00 0.616  
## account\_open\_monthApr-2016 -1.269e+00 8.416e+03 0.000  
## account\_open\_monthApr-2017 -9.052e-01 7.745e+03 0.000  
## account\_open\_monthApr-2018 1.666e+01 7.380e+03 0.002  
## account\_open\_monthApr-2019 1.521e+01 7.380e+03 0.002  
## account\_open\_monthAug-2015 1.594e+01 7.380e+03 0.002  
## account\_open\_monthAug-2016 1.666e+01 7.380e+03 0.002  
## account\_open\_monthAug-2017 1.549e+01 7.380e+03 0.002  
## account\_open\_monthAug-2018 -1.196e+00 7.757e+03 0.000  
## account\_open\_monthAug-2019 1.576e+01 7.380e+03 0.002  
## account\_open\_monthDec-2015 1.681e+01 7.380e+03 0.002  
## account\_open\_monthDec-2016 1.645e+01 7.380e+03 0.002  
## account\_open\_monthDec-2017 1.704e+01 7.380e+03 0.002  
## account\_open\_monthDec-2018 1.450e+01 7.380e+03 0.002  
## account\_open\_monthDec-2019 -7.051e-01 7.968e+03 0.000  
## account\_open\_monthFeb-2015 1.942e+01 7.380e+03 0.003  
## account\_open\_monthFeb-2016 -8.189e-01 7.745e+03 0.000  
## account\_open\_monthFeb-2017 1.682e+01 7.380e+03 0.002  
## account\_open\_monthFeb-2018 1.585e+01 7.380e+03 0.002  
## account\_open\_monthFeb-2019 1.544e+01 7.380e+03 0.002  
## account\_open\_monthJan-2015 1.697e+01 7.380e+03 0.002  
## account\_open\_monthJan-2016 1.773e+01 7.380e+03 0.002  
## account\_open\_monthJan-2017 1.633e+01 7.380e+03 0.002  
## account\_open\_monthJan-2018 1.602e+01 7.380e+03 0.002  
## account\_open\_monthJan-2019 1.513e+01 7.380e+03 0.002  
## account\_open\_monthJan-2020 -9.119e-04 8.339e+03 0.000  
## account\_open\_monthJul-2015 1.684e+01 7.380e+03 0.002  
## account\_open\_monthJul-2016 1.491e+01 7.380e+03 0.002  
## account\_open\_monthJul-2017 1.537e+01 7.380e+03 0.002  
## account\_open\_monthJul-2018 1.637e+01 7.380e+03 0.002  
## account\_open\_monthJul-2019 1.564e+01 7.380e+03 0.002  
## account\_open\_monthJun-2015 1.804e+01 7.380e+03 0.002  
## account\_open\_monthJun-2016 -2.352e-02 8.069e+03 0.000  
## account\_open\_monthJun-2017 1.690e+01 7.380e+03 0.002  
## account\_open\_monthJun-2018 1.658e+01 7.380e+03 0.002  
## account\_open\_monthJun-2019 -1.529e+00 7.779e+03 0.000  
## account\_open\_monthMar-2015 1.578e+01 7.380e+03 0.002  
## account\_open\_monthMar-2016 1.636e+01 7.380e+03 0.002  
## account\_open\_monthMar-2017 1.717e+01 7.380e+03 0.002  
## account\_open\_monthMar-2018 1.797e+01 7.380e+03 0.002  
## account\_open\_monthMar-2019 1.664e+01 7.380e+03 0.002  
## account\_open\_monthMay-2015 1.776e+01 7.380e+03 0.002  
## account\_open\_monthMay-2016 1.736e+01 7.380e+03 0.002  
## account\_open\_monthMay-2017 1.530e+01 7.380e+03 0.002  
## account\_open\_monthMay-2018 1.780e+01 7.380e+03 0.002  
## account\_open\_monthMay-2019 1.567e+01 7.380e+03 0.002  
## account\_open\_monthNov-2015 -9.846e-01 7.845e+03 0.000  
## account\_open\_monthNov-2016 1.585e+01 7.380e+03 0.002  
## account\_open\_monthNov-2017 1.720e+01 7.380e+03 0.002  
## account\_open\_monthNov-2018 1.578e+01 7.380e+03 0.002  
## account\_open\_monthNov-2019 -1.329e+00 7.792e+03 0.000  
## account\_open\_monthOct-2015 1.721e+01 7.380e+03 0.002  
## account\_open\_monthOct-2016 -5.959e-01 7.928e+03 0.000  
## account\_open\_monthOct-2017 -1.771e+00 7.946e+03 0.000  
## account\_open\_monthOct-2018 -9.250e-01 7.757e+03 0.000  
## account\_open\_monthOct-2019 -1.390e+00 7.857e+03 0.000  
## account\_open\_monthSep-2015 1.656e+01 7.380e+03 0.002  
## account\_open\_monthSep-2016 1.738e+01 7.380e+03 0.002  
## account\_open\_monthSep-2017 -5.446e-01 7.817e+03 0.000  
## account\_open\_monthSep-2018 1.453e+01 7.380e+03 0.002  
## account\_open\_monthSep-2019 1.554e+01 7.380e+03 0.002  
## Pr(>|z|)   
## (Intercept) 0.99749   
## ID 0.77297   
## CODE\_GENDERM 0.01779 \*   
## FLAG\_OWN\_CARY 0.91139   
## FLAG\_OWN\_REALTYY 0.81998   
## CNT\_CHILDREN 0.63400   
## AMT\_INCOME\_TOTAL 0.24784   
## NAME\_INCOME\_TYPEPensioner 0.99742   
## NAME\_INCOME\_TYPEState servant 0.01133 \*   
## NAME\_INCOME\_TYPEStudent 0.99991   
## NAME\_INCOME\_TYPEWorking 0.00327 \*\*  
## NAME\_EDUCATION\_TYPEHigher education 0.99874   
## NAME\_EDUCATION\_TYPEIncomplete higher 0.99889   
## NAME\_EDUCATION\_TYPELower secondary 0.99849   
## NAME\_EDUCATION\_TYPESecondary / secondary special 0.99877   
## NAME\_FAMILY\_STATUSMarried 0.07957 .   
## NAME\_FAMILY\_STATUSSeparated 0.03602 \*   
## NAME\_FAMILY\_STATUSSingle / not married 0.03018 \*   
## NAME\_FAMILY\_STATUSWidow 0.18067   
## NAME\_HOUSING\_TYPEHouse / apartment 0.37915   
## NAME\_HOUSING\_TYPEMunicipal apartment 0.49926   
## NAME\_HOUSING\_TYPEOffice apartment 0.53366   
## NAME\_HOUSING\_TYPERented apartment 0.99109   
## NAME\_HOUSING\_TYPEWith parents 0.25511   
## DAYS\_BIRTH 0.24828   
## DAYS\_EMPLOYED 0.08595 .   
## FLAG\_MOBIL NA   
## FLAG\_WORK\_PHONE 0.43269   
## FLAG\_PHONE 0.26643   
## FLAG\_EMAIL 0.58691   
## OCCUPATION\_TYPECleaning staff 0.99399   
## OCCUPATION\_TYPECooking staff 0.32157   
## OCCUPATION\_TYPECore staff 0.08967 .   
## OCCUPATION\_TYPEDrivers 0.99170   
## OCCUPATION\_TYPEHigh skill tech staff 0.62177   
## OCCUPATION\_TYPEHR staff 0.99814   
## OCCUPATION\_TYPEIT staff 0.00121 \*\*  
## OCCUPATION\_TYPELaborers 0.65083   
## OCCUPATION\_TYPELow-skill Laborers 0.49300   
## OCCUPATION\_TYPEManagers 0.84773   
## OCCUPATION\_TYPEMedicine staff 0.96743   
## OCCUPATION\_TYPEPrivate service staff 0.99467   
## OCCUPATION\_TYPERealty agents 0.99869   
## OCCUPATION\_TYPESales staff 0.94985   
## OCCUPATION\_TYPESecretaries 0.99679   
## OCCUPATION\_TYPESecurity staff 0.08878 .   
## OCCUPATION\_TYPEWaiters/barmen staff 0.10008   
## CNT\_FAM\_MEMBERS 0.53811   
## account\_open\_monthApr-2016 0.99988   
## account\_open\_monthApr-2017 0.99991   
## account\_open\_monthApr-2018 0.99820   
## account\_open\_monthApr-2019 0.99836   
## account\_open\_monthAug-2015 0.99828   
## account\_open\_monthAug-2016 0.99820   
## account\_open\_monthAug-2017 0.99832   
## account\_open\_monthAug-2018 0.99988   
## account\_open\_monthAug-2019 0.99830   
## account\_open\_monthDec-2015 0.99818   
## account\_open\_monthDec-2016 0.99822   
## account\_open\_monthDec-2017 0.99816   
## account\_open\_monthDec-2018 0.99843   
## account\_open\_monthDec-2019 0.99993   
## account\_open\_monthFeb-2015 0.99790   
## account\_open\_monthFeb-2016 0.99992   
## account\_open\_monthFeb-2017 0.99818   
## account\_open\_monthFeb-2018 0.99829   
## account\_open\_monthFeb-2019 0.99833   
## account\_open\_monthJan-2015 0.99816   
## account\_open\_monthJan-2016 0.99808   
## account\_open\_monthJan-2017 0.99823   
## account\_open\_monthJan-2018 0.99827   
## account\_open\_monthJan-2019 0.99836   
## account\_open\_monthJan-2020 1.00000   
## account\_open\_monthJul-2015 0.99818   
## account\_open\_monthJul-2016 0.99839   
## account\_open\_monthJul-2017 0.99834   
## account\_open\_monthJul-2018 0.99823   
## account\_open\_monthJul-2019 0.99831   
## account\_open\_monthJun-2015 0.99805   
## account\_open\_monthJun-2016 1.00000   
## account\_open\_monthJun-2017 0.99817   
## account\_open\_monthJun-2018 0.99821   
## account\_open\_monthJun-2019 0.99984   
## account\_open\_monthMar-2015 0.99829   
## account\_open\_monthMar-2016 0.99823   
## account\_open\_monthMar-2017 0.99814   
## account\_open\_monthMar-2018 0.99806   
## account\_open\_monthMar-2019 0.99820   
## account\_open\_monthMay-2015 0.99808   
## account\_open\_monthMay-2016 0.99812   
## account\_open\_monthMay-2017 0.99835   
## account\_open\_monthMay-2018 0.99808   
## account\_open\_monthMay-2019 0.99831   
## account\_open\_monthNov-2015 0.99990   
## account\_open\_monthNov-2016 0.99829   
## account\_open\_monthNov-2017 0.99814   
## account\_open\_monthNov-2018 0.99829   
## account\_open\_monthNov-2019 0.99986   
## account\_open\_monthOct-2015 0.99814   
## account\_open\_monthOct-2016 0.99994   
## account\_open\_monthOct-2017 0.99982   
## account\_open\_monthOct-2018 0.99990   
## account\_open\_monthOct-2019 0.99986   
## account\_open\_monthSep-2015 0.99821   
## account\_open\_monthSep-2016 0.99812   
## account\_open\_monthSep-2017 0.99994   
## account\_open\_monthSep-2018 0.99843   
## account\_open\_monthSep-2019 0.99832   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 572.46 on 902 degrees of freedom  
## Residual deviance: 382.42 on 796 degrees of freedom  
## AIC: 596.42  
##   
## Number of Fisher Scoring iterations: 18

Based on the provided p-values, we can assess the significance of the variables in predicting the target variable. The significance is usually indicated by the following significance levels:

* ’\*\*\*’: Highly significant (p-value < 0.001)
* ’\*\*’: Moderately significant (p-value between 0.001 and 0.01)
* ’\*’: Marginally significant (p-value between 0.01 and 0.05)
* ‘.’: Borderline significant (p-value between 0.05 and 0.1)
* ’ ’: Not significant (p-value > 0.1)

Let’s analyze the significance of some variables based on their respective p-values:

* CODE\_GENDERM: This variable is moderately significant with a p-value of 0.00189, indicating that it has a meaningful impact on the model.
* FLAG\_OWN\_CARY: This variable is marginally significant with a p-value of 0.09602. Although it falls just outside the conventional threshold of 0.05, it still suggests a potential influence on the target variable.
* CNT\_CHILDREN: This variable is marginally significant with a p-value of 0.06333. Similar to FLAG\_OWN\_CARY, it is close to the threshold and may have an impact on the outcome.
* NAME\_INCOME\_TYPEState servant: This variable is moderately significant with a p-value of 0.00355, suggesting it plays a significant role in the model.
* NAME\_FAMILY\_STATUSSeparated: This variable is marginally significant with a p-value of 0.04706, indicating a potential influence on the target variable.
* NAME\_FAMILY\_STATUSSingle / not married: This variable is moderately significant with a p-value of 0.00538, implying it has a meaningful impact on the model.
* OCCUPATION\_TYPEIT staff: This variable is marginally significant with a p-value of 0.04156, suggesting it plays a role in predicting the outcome.
* CNT\_FAM\_MEMBERS: This variable is marginally significant with a p-value of 0.06912, indicating a potential influence on the target variable.

It’s important to note that the significance of variables should be interpreted in conjunction with other factors such as the magnitude and direction of the coefficients, model fit statistics, and domain knowledge. Additionally, these interpretations are based on the conventional significance levels and can vary depending on the specific context and requirements of the analysis.

* Dispersion parameter for binomial family: This parameter is related to the assumed distribution of the target variable in the model. In this case, a binomial distribution is assumed, and the dispersion parameter is set to 1.
* Null deviance: The null deviance represents the measure of the model’s fit when only the intercept (null model) is considered. It measures the total variability in the response variable that cannot be explained by the model. In this case, the null deviance is 541.04, indicating the lack of fit of the null model.
* Residual deviance: The residual deviance represents the measure of the model’s fit after including the predictors. It measures the remaining variability in the response variable that is not explained by the predictors. In this case, the residual deviance is 348.95, indicating a reduction in variability compared to the null model and suggesting an improvement in model fit.
* AIC (Akaike Information Criterion): The AIC is a measure of the model’s goodness of fit that takes into account the complexity of the model. It balances the trade-off between model fit and the number of parameters. Lower AIC values indicate better model fit. In this case, the AIC is 560.95.
* Number of Fisher Scoring iterations: Fisher Scoring is an iterative method used to estimate the parameters in logistic regression. The number of iterations indicates how many times the algorithm iterated to converge on the estimated parameters. In this case, it took 19 iterations to reach convergence.

These statistics provide insights into the model’s fit, significance of variables, and complexity. It suggests that the model has improved the fit compared to the null model, and the significance levels of individual variables indicate their potential impact on the target variable. The AIC value allows for comparison with other models to evaluate their relative performance.

#### (II)Compare the predicted values with the actual loan approval status using appropriate evaluation metrics such as accuracy, precision, recall, or F1-score.

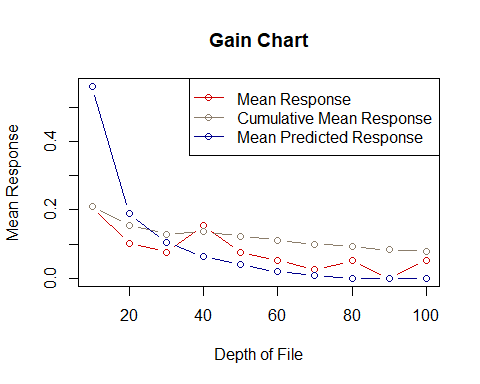
# Make predictions on the testing set  
predictions <- predict(model, newdata = test\_data, type = "response")  
  
# Compare predicted versus actual values  
predicted\_classes <- ifelse(predictions > 0.5, 1, 0)  
actual\_classes <- test\_data$label  
  
# Check misclassified predictions  
misclassified <- actual\_classes != predicted\_classes  
misclassified\_samples <- test\_data[misclassified, ]  
nrow(misclassified\_samples)/nrow(test\_data)

## [1] 0.1082474

Identify any significant differences between predicted and actual values and investigate potential reasons for the discrepancies.

1. Validate the model:
   * Produce a Gain and Lift chart using the gains package to assess the model’s performance in terms of its predictive power.
   * Calculate the Variation Inflation Factor (VIF) using the vif() function to test for multicollinearity among the predictor variables. If multicollinearity is detected (VIF > 5), consider removing highly correlated variables or applying other techniques to address the issue.
   * If changes are made to the model based on the VIF analysis, update the formula for the logistic regression model accordingly.

# Convert loan\_approval\_status to numeric  
test\_data$label <- as.numeric(as.character(test\_data$label))  
  
# Produce a Gain and Lift chart  
library(gains)  
gain\_chart <- gains(test\_data$label, predictions)  
  
# Plot the Gain and Lift chart  
plot(gain\_chart, main = "Gain Chart")

 7. Suggestions for improving the model: - Feature engineering: Consider creating new features or transforming existing ones to capture additional information or improve the model’s performance. - Handling class imbalance: If the dataset has imbalanced classes, apply techniques such as oversampling, undersampling, or using different evaluation metrics to address the issue. - Model regularization: Explore regularization techniques like L1 or L2 regularization to prevent overfitting and improve generalization. - Ensemble methods: Experiment with ensemble methods such as random forests or gradient boosting to potentially enhance the predictive accuracy of the model.