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
```

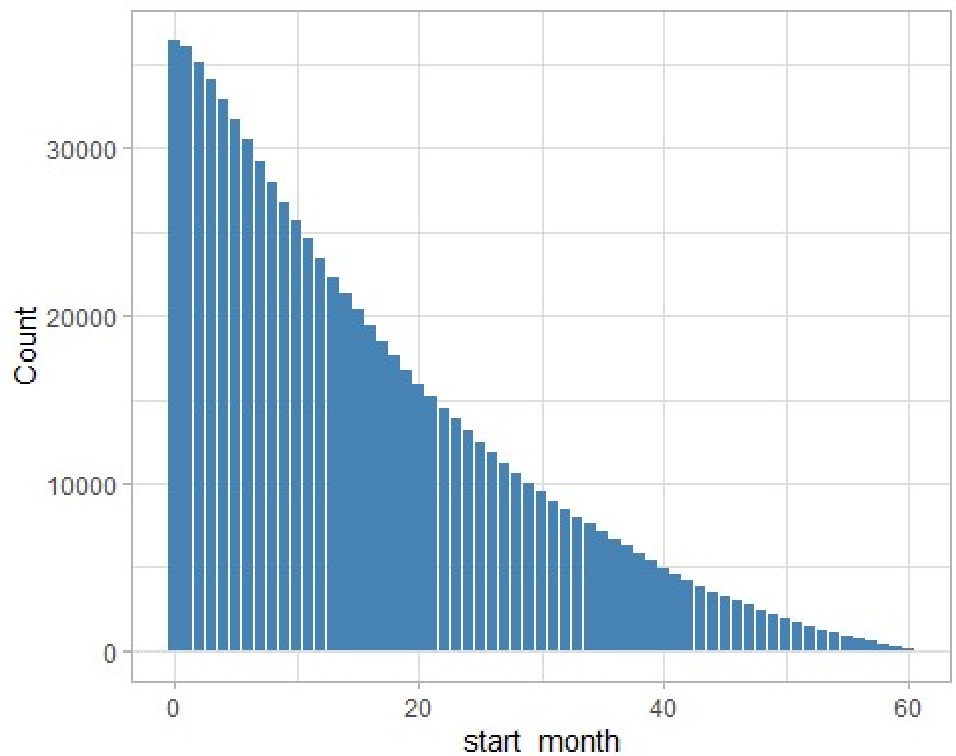
```
## 5 5008804          11 Oct-2018                          -4 C
## 6 5008804          10 Oct-2018                          -5 C
```

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()
```

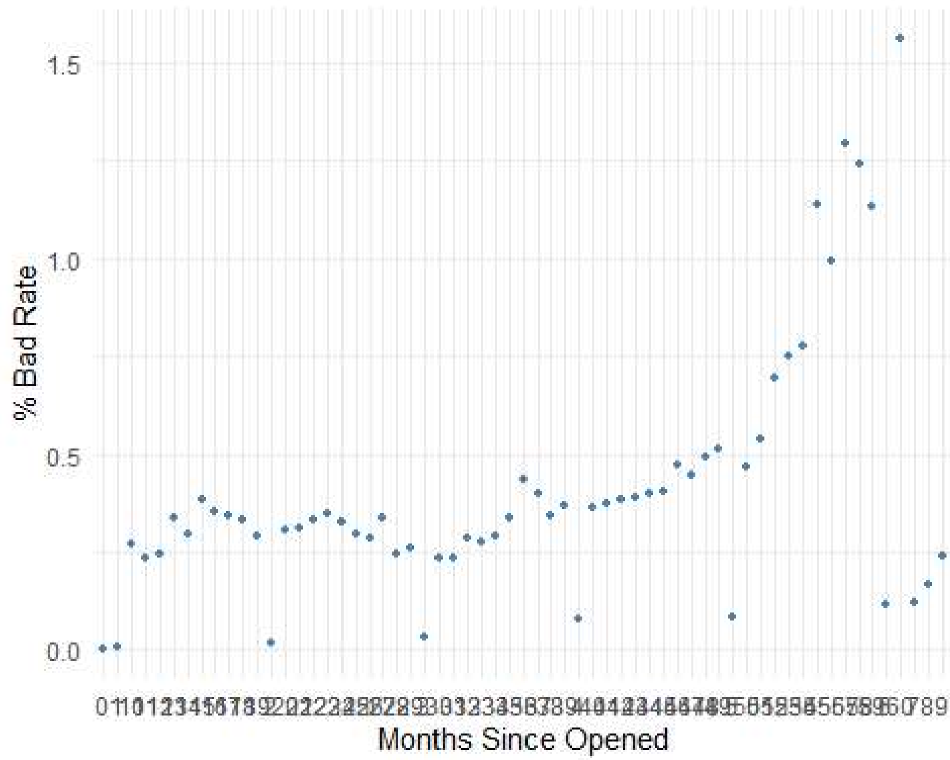


The goal is to calculate the percentage of bad rate for the entire portfolio across all account open months. This will help determine the period during which the overall bad rate remains stable. It is important to note that only a small number of credit card accounts were opened in the early months and 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
```

```r
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)
}
```

```r
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
```

2. 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%).

```r
# 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.

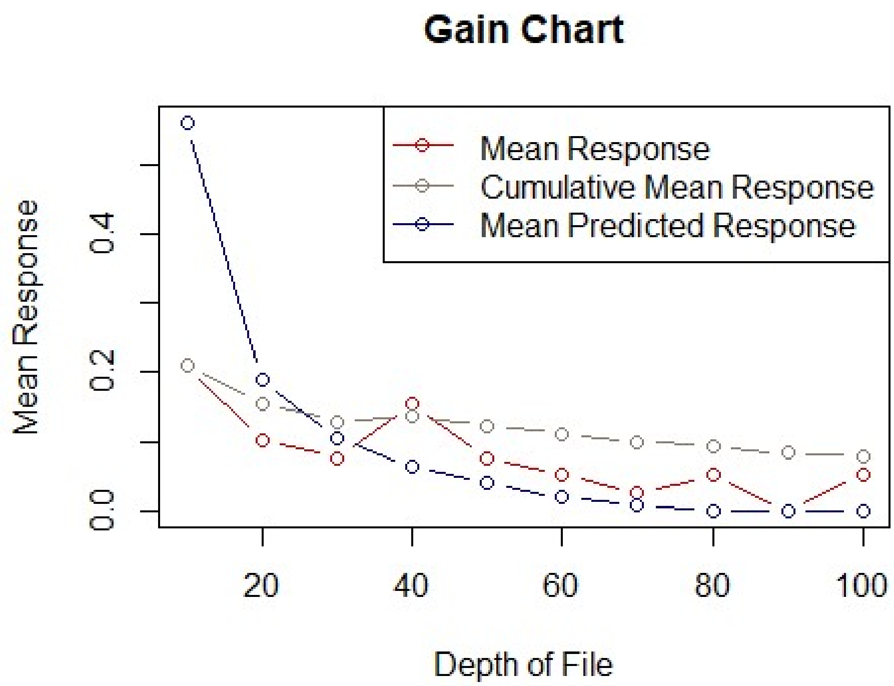
(II)Compare the predicted values with the actual loan approval status using appropriate evaluation metrics such as accuracy, precision, recall, or F1-score.

```
# 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.

5. 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.