232 Final Project

Japisht Aurora, Jeewan Singh, Abhinandan Aneja

2023-03-20

Introduction

In today's fast-paced world, credit cards have become an essential part of our lives. They offer convenience and flexibility when it comes to making purchases, booking travel, or even paying bills. Credit cards allow users to make purchases without having to carry cash, and also provide a means of building credit history.

However, with the convenience of credit cards comes the responsibility of managing credit effectively. Credit card companies need to carefully evaluate each application to determine the risk associated with lending money. This is where the Credit Card Approval Prediction dataset comes into play.

The Credit Card Approval Prediction dataset provides a comprehensive set of attributes that can be used to evaluate the creditworthiness of applicants. It contains information on factors such as age, income, education, and employment status, which are known to influence credit card approval rates. By analyzing this dataset, we can gain insights into the factors that are most important in determining credit card approval rates.

In this project, we will use data visualization techniques to explore the Credit Card Approval Prediction dataset and identify the key factors that influence credit card approval rates. We will use tools such as scatter plots, heat maps, and histograms to visualize the data and analyze the patterns that emerge.

Overall, the project aims to provide a better understanding of the credit card approval process and help credit card companies make more informed decisions about who to approve for credit cards.

Objectives

1. To identify the most important factors that influence credit card approval rates by analyzing the Credit Card Approval Prediction dataset.
2. To understand how different attributes such as age, income, education, and employment status affect credit card approval rates.
3. To use data visualization techniques to identify patterns and trends in the data that can provide valuable insights to credit card companies.
4. To help credit card companies make more informed decisions about who to approve for credit cards based on the analysis of the Credit Card Approval Prediction dataset.
5. To demonstrate the effectiveness of data visualization techniques in exploring and analyzing complex datasets such as the Credit Card Approval Prediction dataset.

Exploratory Analysis

```r
library(readr)
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v purrr   1.0.1      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(lubridate)
```

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

```
library(dplyr)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(randomForest)
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(rpart)
library(rpart.plot)
mydata <- read_csv("~/Downloads/archive/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.
```

```
mydata2 <- read_csv("~/Downloads/archive/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.
```

```
head(mydata)
```

```
## # A tibble: 6 x 18
##        ID CODE_~1 FLAG_~2 FLAG_~3 CNT_C~4 AMT_I~5 NAME_~6 NAME_~7 NAME_~8 NAME_~9
##     <dbl> <chr>   <chr>   <chr>     <dbl>   <dbl> <chr>   <chr>   <chr>   <chr>
## 1 5.01e6 M       Y       Y             0  427500 Working Higher~ Civil ~ Rented~
## 2 5.01e6 M       Y       Y             0  427500 Working Higher~ Civil ~ Rented~
## 3 5.01e6 M       Y       Y             0  112500 Working Second~ Married House ~
## 4 5.01e6 F       N       Y             0  270000 Commer~ Second~ Single~ House ~
## 5 5.01e6 F       N       Y             0  270000 Commer~ Second~ Single~ House ~
## 6 5.01e6 F       N       Y             0  270000 Commer~ Second~ Single~ House ~
## # ... with 8 more variables: DAYS_BIRTH <dbl>, DAYS_EMPLOYED <dbl>,
## #   FLAG_MOBIL <dbl>, FLAG_WORK_PHONE <dbl>, FLAG_PHONE <dbl>,
## #   FLAG_EMAIL <dbl>, OCCUPATION_TYPE <chr>, CNT_FAM_MEMBERS <dbl>, and
## #   abbreviated variable names 1: CODE_GENDER, 2: FLAG_OWN_CAR,
## #   3: FLAG_OWN_REALTY, 4: CNT_CHILDREN, 5: AMT_INCOME_TOTAL,
## #   6: NAME_INCOME_TYPE, 7: NAME_EDUCATION_TYPE, 8: NAME_FAMILY_STATUS,
## #   9: NAME_HOUSING_TYPE
```

```
head(mydata2)
```

```
## # A tibble: 6 x 3
##         ID MONTHS_BALANCE STATUS
##      <dbl>          <dbl> <chr>
## 1 5001711              0 X
## 2 5001711             -1 0
## 3 5001711             -2 0
## 4 5001711             -3 0
## 5 5001712              0 C
## 6 5001712             -1 C
```

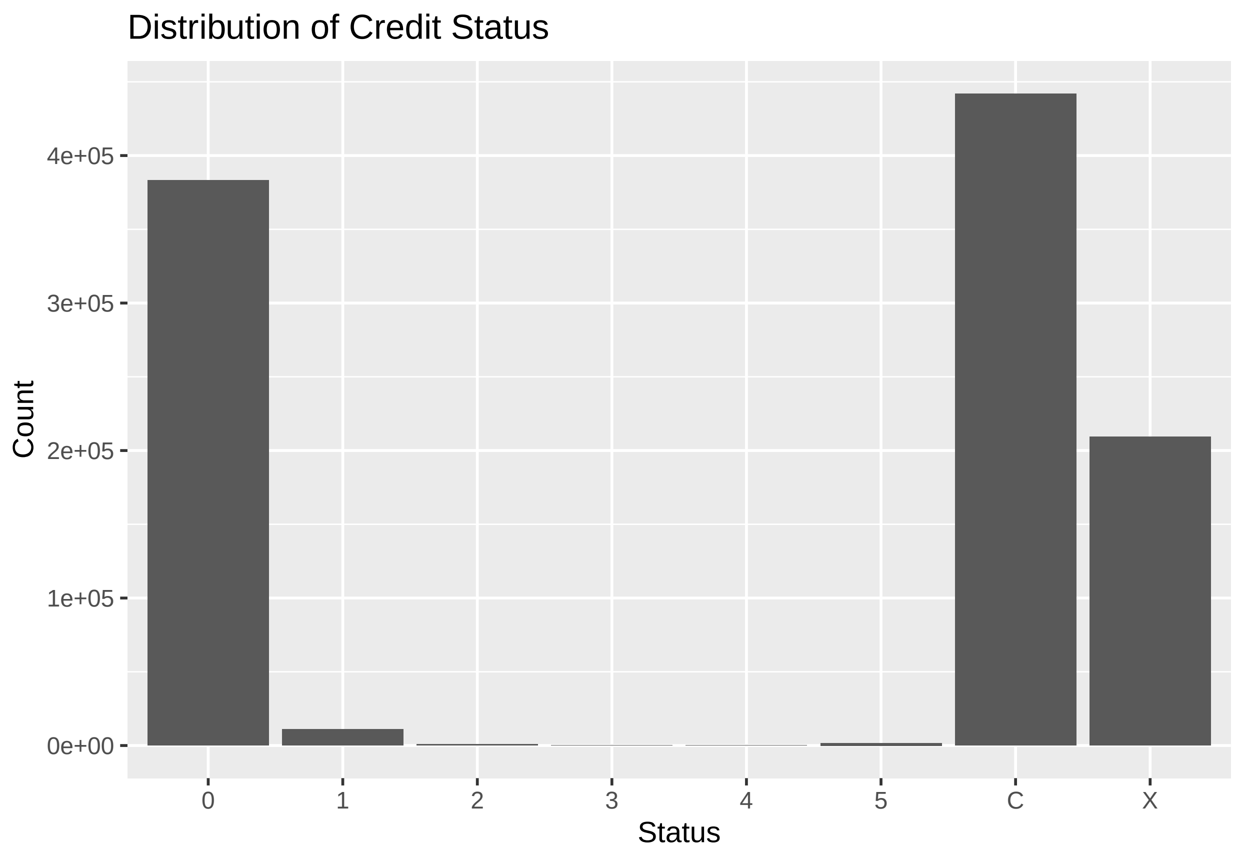
```r
# Remove rows with missing data
mydata <- na.omit(mydata)
mydata2<- na.omit(mydata2)

dim(mydata)
```

```
## [1] 304354     18
```

```r
dim(mydata2)
```

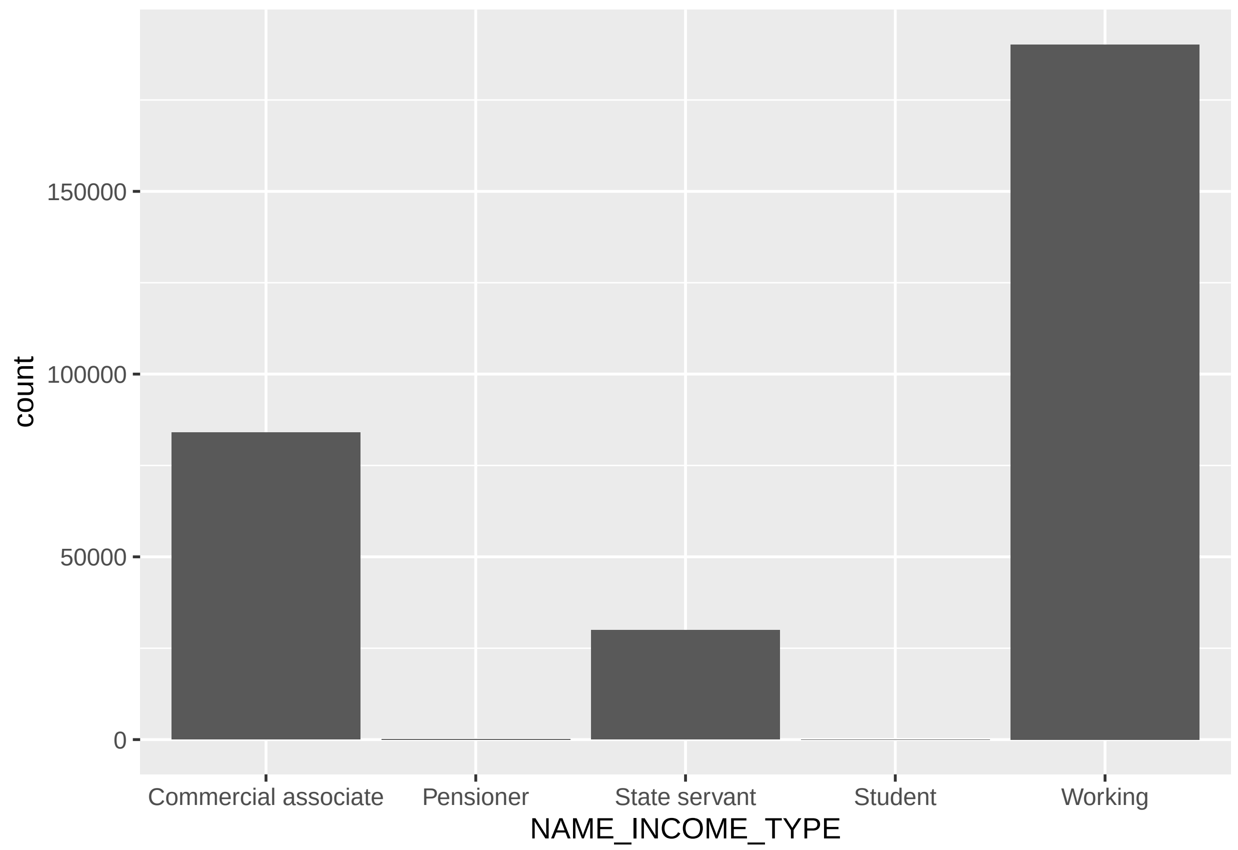
```
## [1] 1048575       3
```

```r
# Plot distribution of STATUS variable
ggplot(data = mydata2, aes(x = STATUS)) +
  geom_bar() +
  labs(title = "Distribution of Credit Status", x = "Status", y = "Count")
```



The first graph shows the distribution of the credit status variable using a bar plot. This plot indicates that most of the credit status values fall under the categories "0", "C", "X" while very few values fall under the categories "2", "3", "4" and 5. This suggests that most of the credit card applications were either approved or in good standing.

```r
#income level of the different group
ggplot(data=mydata, aes(x = NAME_INCOME_TYPE)) +
  geom_bar()
```

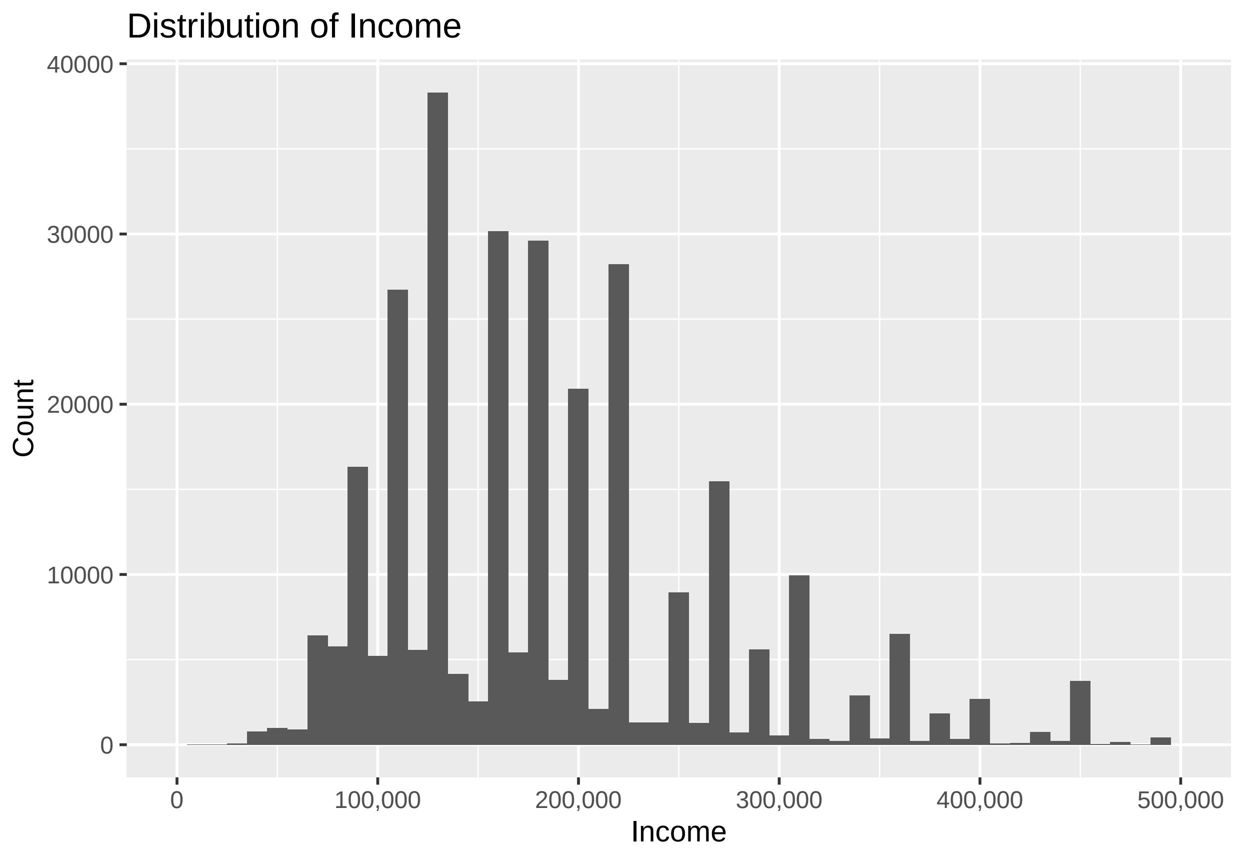


The second graph shows the income level distribution of the different income types.This plot indicates that most of the applicants fall under the income type “Working”, followed by “Commercial associate” and “State servant”.

```r
ggplot(data = mydata, aes(x = AMT_INCOME_TOTAL)) +
  geom_histogram(binwidth = 10000) +
  scale_x_continuous(labels = scales::comma, limits = c(0, 500000)) +
  labs(title = "Distribution of Income", x = "Income", y = "Count")
```

```
## Warning: Removed 4347 rows containing non-finite values (stat_bin).
```

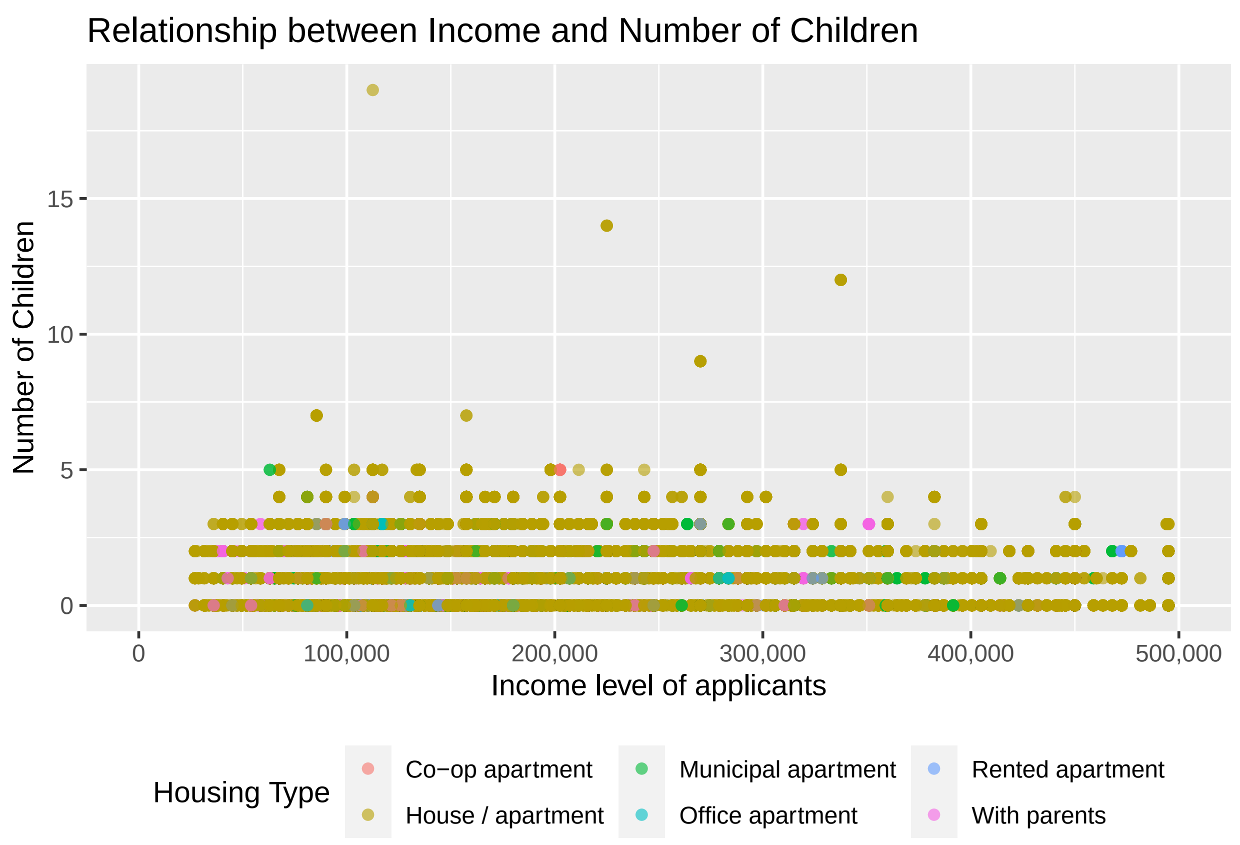
```
## Warning: Removed 2 rows containing missing values (geom_bar).
```



The third graph shows the distribution of income levels using a histogram. This plot indicates that most of the applicants have an income below 100,000 and the income distribution is skewed to the right.

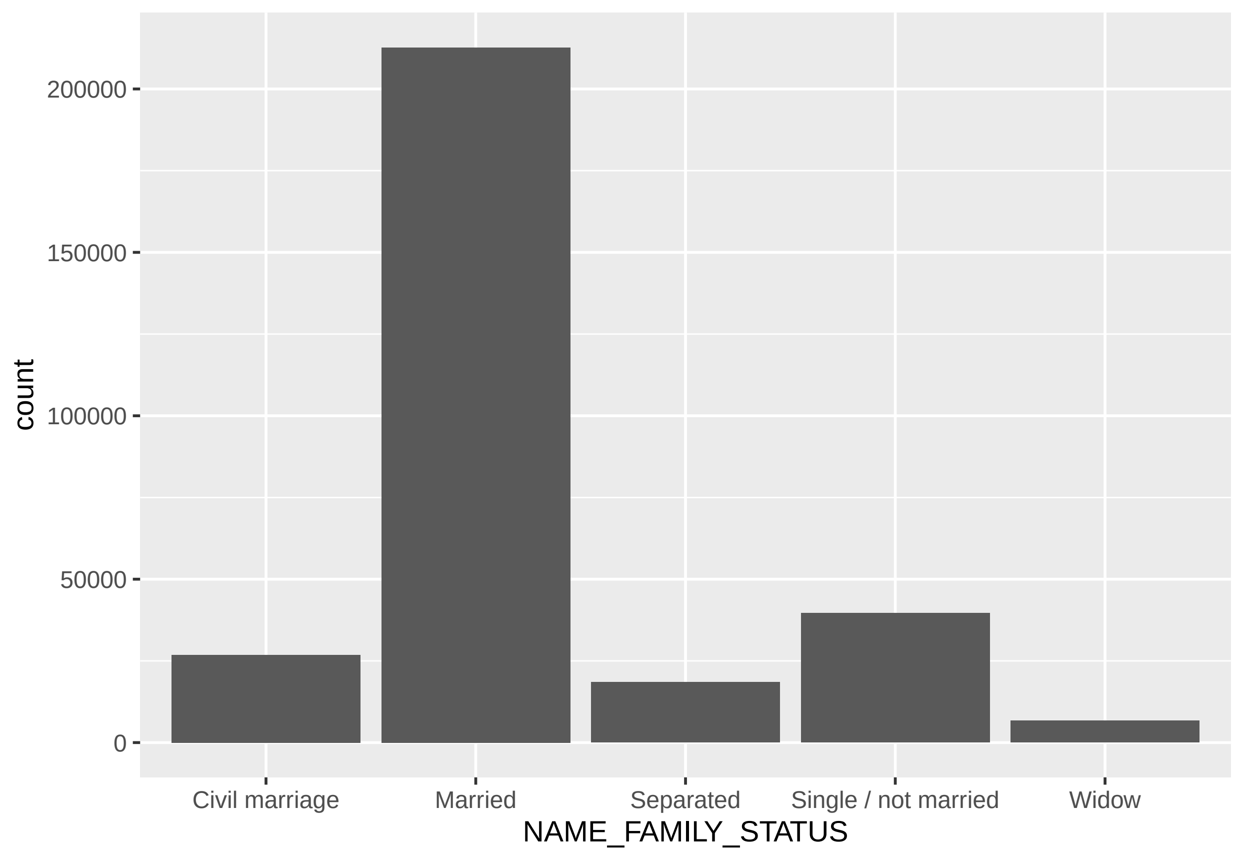
```
# Plot scatterplot of AMT_INCOME_TOTAL vs. CNT_CHILDREN
ggplot(data = mydata, aes(x = AMT_INCOME_TOTAL, y = CNT_CHILDREN, color = factor(NAME_HOUSING_TYPE))) +
  geom_point(alpha = 0.6) +
  scale_x_continuous(labels = scales::comma, limits = c(0, 500000)) +
  labs(title = "Relationship between Income and Number of Children",
       x = "Income level of applicants",
       y = "Number of Children",
       color = "Housing Type") +
  theme(legend.position = "bottom")
```

```
## Warning: Removed 4347 rows containing missing values (geom_point).
```



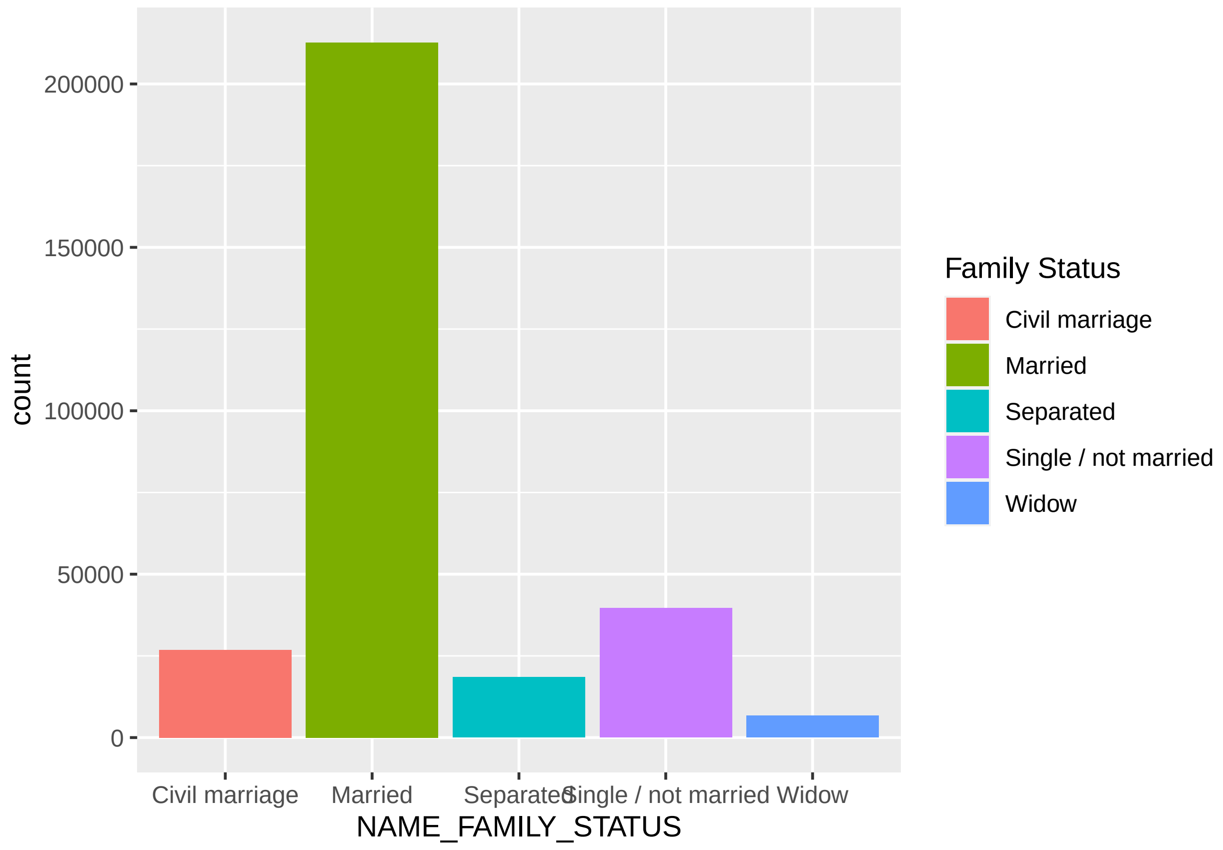
This plot shows the relationship between income and the number of children an applicant has, colored by their housing type. This plot indicates that applicants with higher income tend to have fewer children and that those who own a house tend to have higher income levels compared to other housing types.

```
ggplot(data = mydata, aes(x = NAME_FAMILY_STATUS)) +
  geom_bar()
```

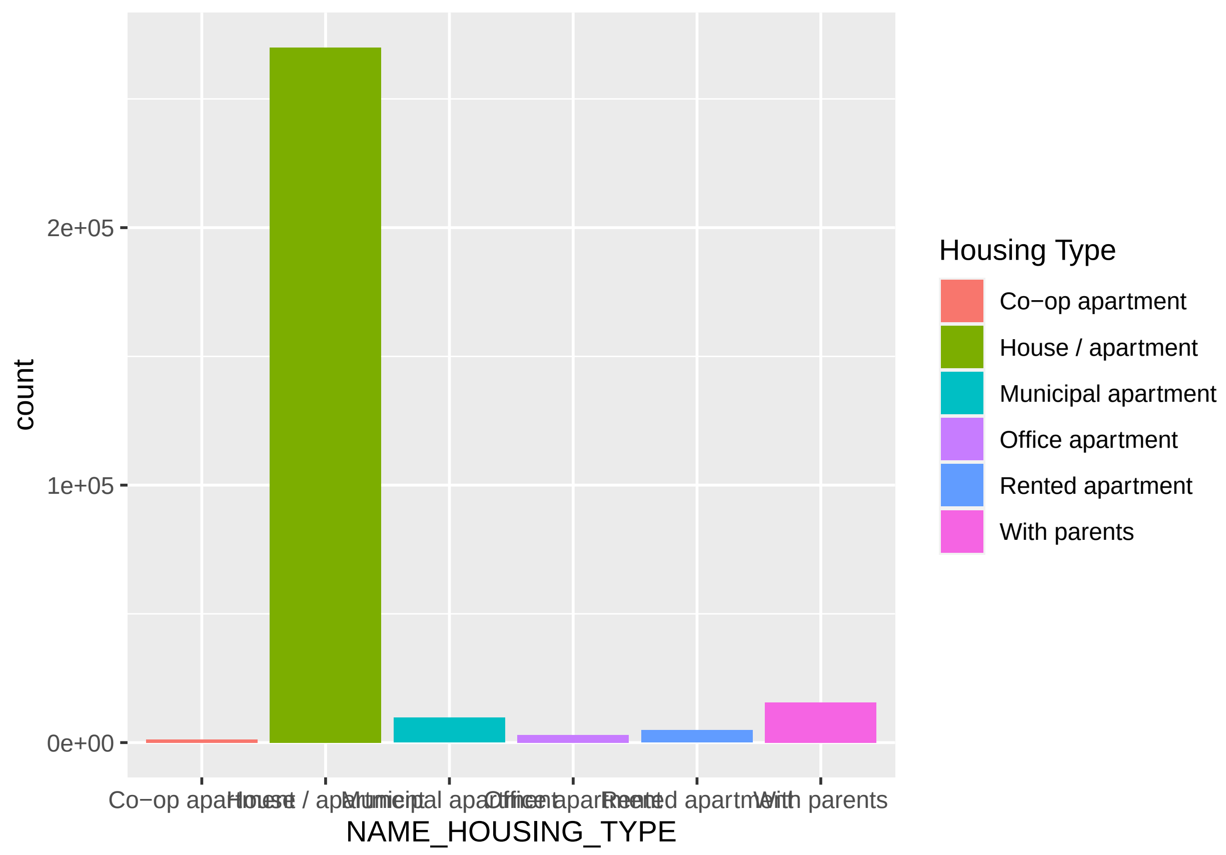


This plot indicates that most of the applicants are married or in a civil union, followed by single applicants.

```r
# Define a 6-color palette
my_palette <- c("#F8766D", "#7CAE00", "#00BFC4", "#C77CFF", "#619CFF", "#F564E3")

# Plot with colors and legend
ggplot(data = mydata, aes(x = NAME_FAMILY_STATUS, fill = NAME_FAMILY_STATUS)) +
  geom_bar() +
  scale_fill_manual(values = my_palette) +
  labs(fill = "Family Status")
```

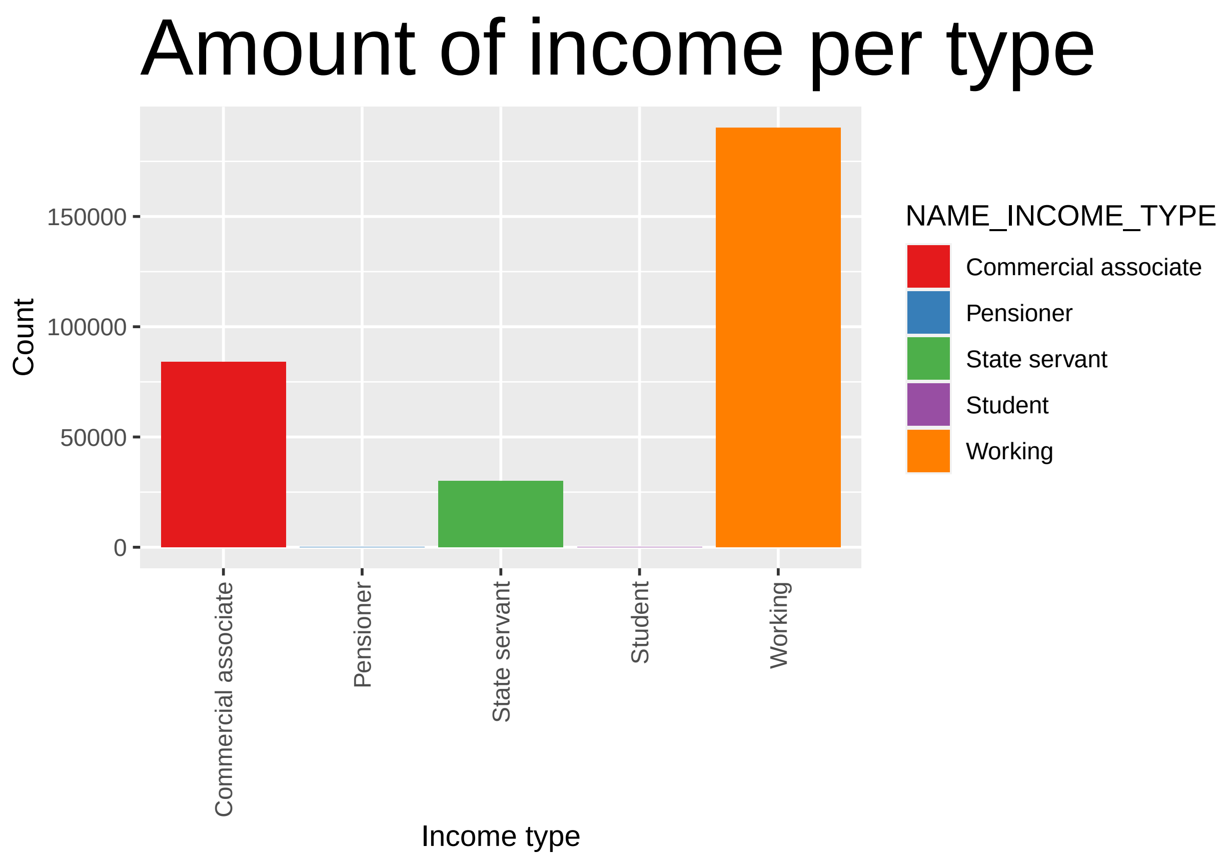


```r
# Plot with colors and legend
ggplot(data = mydata, aes(x = NAME_HOUSING_TYPE, fill = NAME_HOUSING_TYPE)) +
  geom_bar() +
  scale_fill_manual(values = my_palette) +
  labs(fill = "Housing Type")
```



These plots show the distribution of family status and housing type, respectively, using a color palette and legend. These plots provide a visual representation of the frequency of different family statuses and housing types among the applicants.

```
ggplot(data = mydata, aes(x = NAME_INCOME_TYPE, fill = NAME_INCOME_TYPE)) +
  geom_bar() +
  labs(title = "Amount of income per type", x = "Income type", y = "Count") +
  theme(plot.title = element_text(size = 30), axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=
  scale_fill_brewer(palette = "Set1")
```



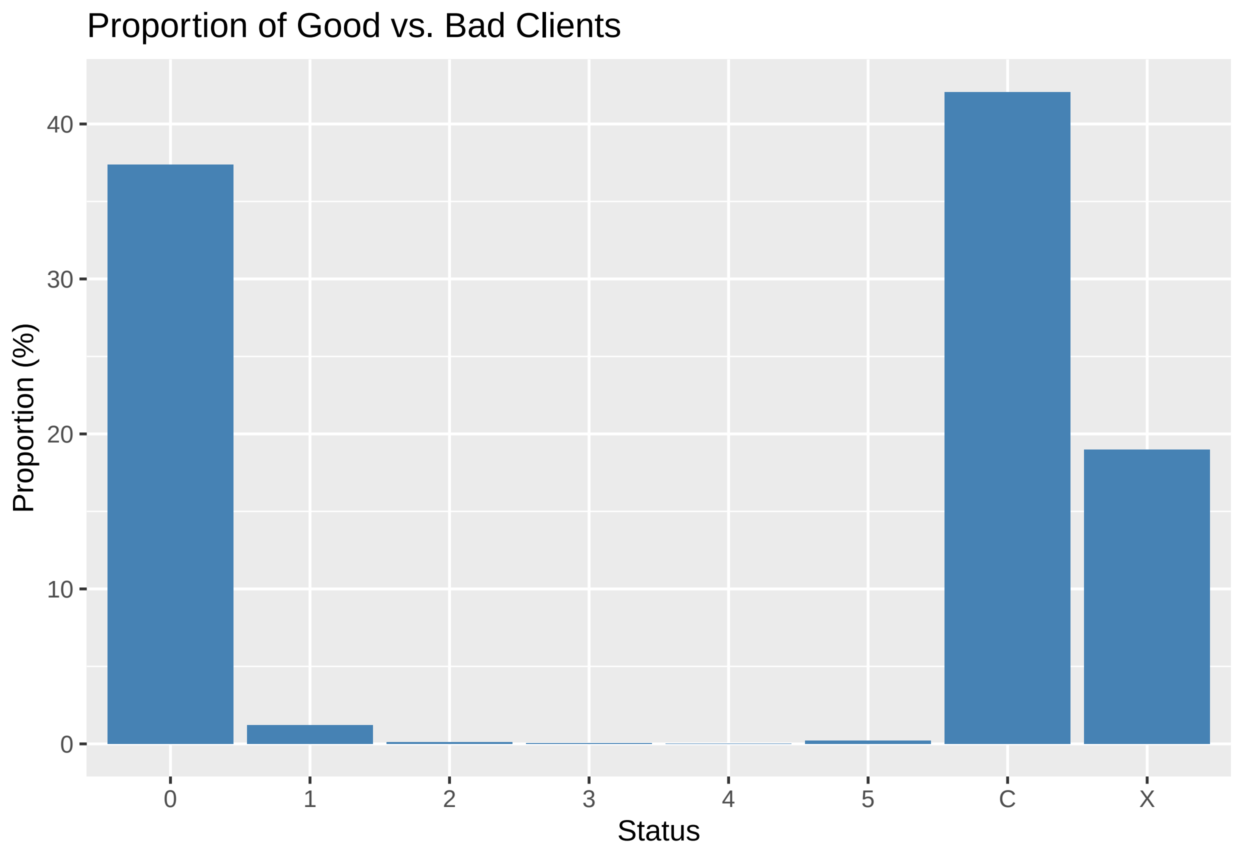
This bar chart shows the distribution of income types among the applicants in the dataset.The x-axis represents the different income types, while the y-axis shows the count of applicants for each income type. The chart reveals that the majority of applicants have income from "Working" and "Commercial associate" categories. On the other hand, "Student" and "pensioner" categories have the lowest number of applicants.

```r
# Merge the two data frames based on the "id" variable
merged <- mydata %>%
  inner_join(mydata2, by = "ID")

merged
```

```
## # A tibble: 537,667 x 20
##         ID CODE_GENDER FLAG_OW~1 FLAG_~2 CNT_C~3 AMT_I~4 NAME_~5 NAME_~6 NAME_~7
##      <dbl> <chr>       <chr>     <chr>     <dbl>   <dbl> <chr>   <chr>   <chr>
##  1 5008806 M           Y         Y             0  112500 Working Second~ Married
##  2 5008806 M           Y         Y             0  112500 Working Second~ Married
##  3 5008806 M           Y         Y             0  112500 Working Second~ Married
##  4 5008806 M           Y         Y             0  112500 Working Second~ Married
##  5 5008806 M           Y         Y             0  112500 Working Second~ Married
##  6 5008806 M           Y         Y             0  112500 Working Second~ Married
##  7 5008806 M           Y         Y             0  112500 Working Second~ Married
##  8 5008806 M           Y         Y             0  112500 Working Second~ Married
##  9 5008806 M           Y         Y             0  112500 Working Second~ Married
## 10 5008806 M           Y         Y             0  112500 Working Second~ Married
## # ... with 537,657 more rows, 11 more variables: NAME_HOUSING_TYPE <chr>,
## #   DAYS_BIRTH <dbl>, DAYS_EMPLOYED <dbl>, FLAG_MOBIL <dbl>,
## #   FLAG_WORK_PHONE <dbl>, FLAG_PHONE <dbl>, FLAG_EMAIL <dbl>,
## #   OCCUPATION_TYPE <chr>, CNT_FAM_MEMBERS <dbl>, MONTHS_BALANCE <dbl>,
## #   STATUS <chr>, and abbreviated variable names 1: FLAG_OWN_CAR,
## #   2: FLAG_OWN_REALTY, 3: CNT_CHILDREN, 4: AMT_INCOME_TOTAL,
## #   5: NAME_INCOME_TYPE, 6: NAME_EDUCATION_TYPE, 7: NAME_FAMILY_STATUS
```

```
# Calculate proportion of good vs. bad clients
proportions <- table(merged$STATUS) / nrow(merged) * 100

# Plot proportions
ggplot(data = data.frame(proportions), aes(x = names(proportions), y = proportions)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(title = "Proportion of Good vs. Bad Clients", x = "Status", y = "Proportion (%)")
```

```
## Don't know how to automatically pick scale for object of type table. Defaulting to continuous.
```



We merged two datasets (mydata and mydata2) based on the ID variable. This allowed us to combine information about each client's application data (e.g., income, family status) with their credit history (e.g., whether they made payments on time or not).

Now that we have a merged dataset, we can start exploring how different variables are related to clients' credit status. To do this, we first calculate the proportion of "good" (i.e., clients who paid their credit on time) vs. "bad" (i.e., clients who did not pay their credit on time) clients using the table() function. We then plot these proportions using ggplot2, which allows us to visualize the proportion of good vs. bad clients.

The plot shows the proportion of each status category (good or bad) as a percentage of the total number of clients. The x-axis displays the two categories of the STATUS variable (good and bad), and the y-axis displays the proportion (%) of clients in each category. The bars are colored in steel blue.

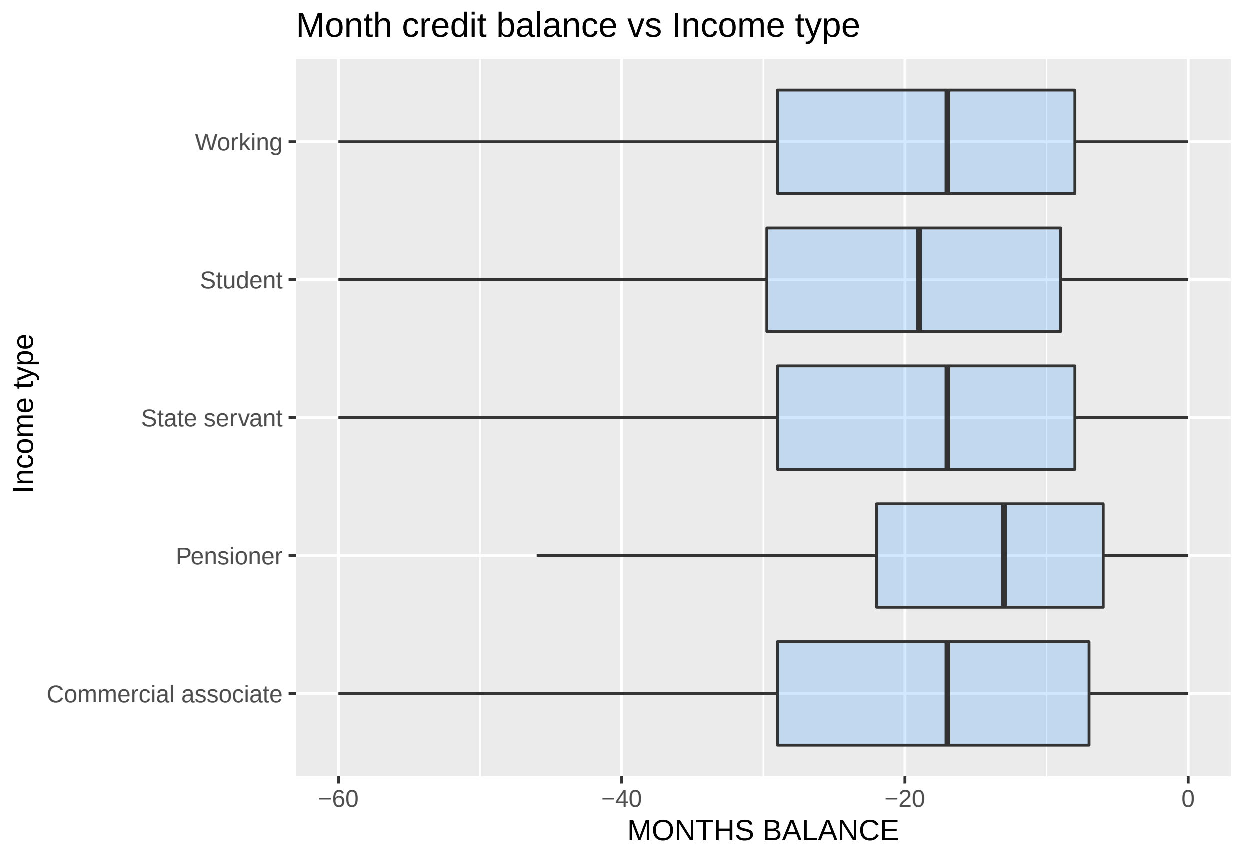
```
# Check data type
class(mydata$DAYS_EMPLOYED)
```

```
## [1] "numeric"
```

```
# Convert to numeric
mydata$DAYS_EMPLOYED <- as.numeric(mydata$DAYS_EMPLOYED)

# Replace values above threshold with NA
mydata$DAYS_EMPLOYED[mydata$DAYS_EMPLOYED > 10000] <- NA

summary(mydata$DAYS_EMPLOYED)
```

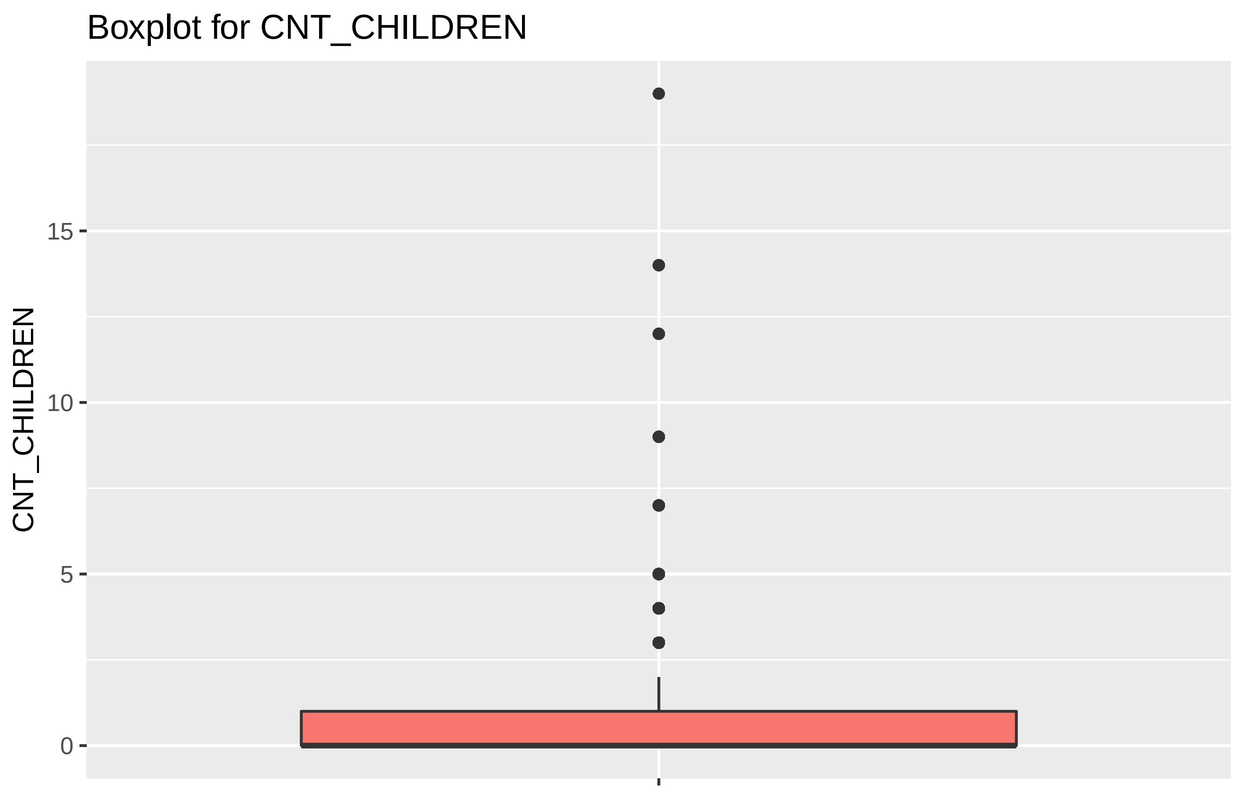
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -17531   -3508   -1917   -2622    -919     -12
```

```
ggplot(data = merged, aes(x = MONTHS_BALANCE , y = NAME_INCOME_TYPE)) +
  geom_boxplot(fill = "dodgerblue", alpha = 0.2) +
  labs(title = "Month credit balance vs Income type", x = "MONTHS BALANCE", y = " Income type")
```

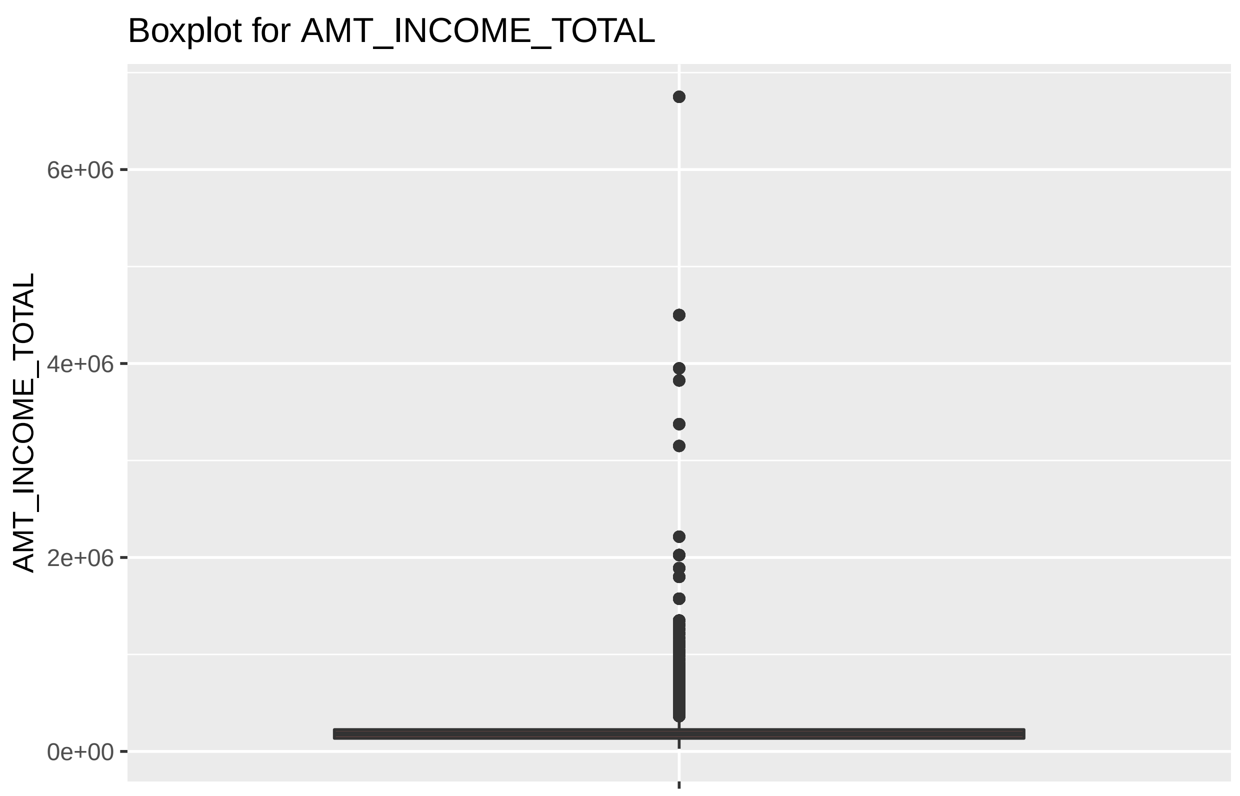


This boxplot to visualize the relationship between the months of credit balance and the income types of the clients.The plot suggests that the income types have different distributions of credit balance months, with some income types having a longer credit balance than others.

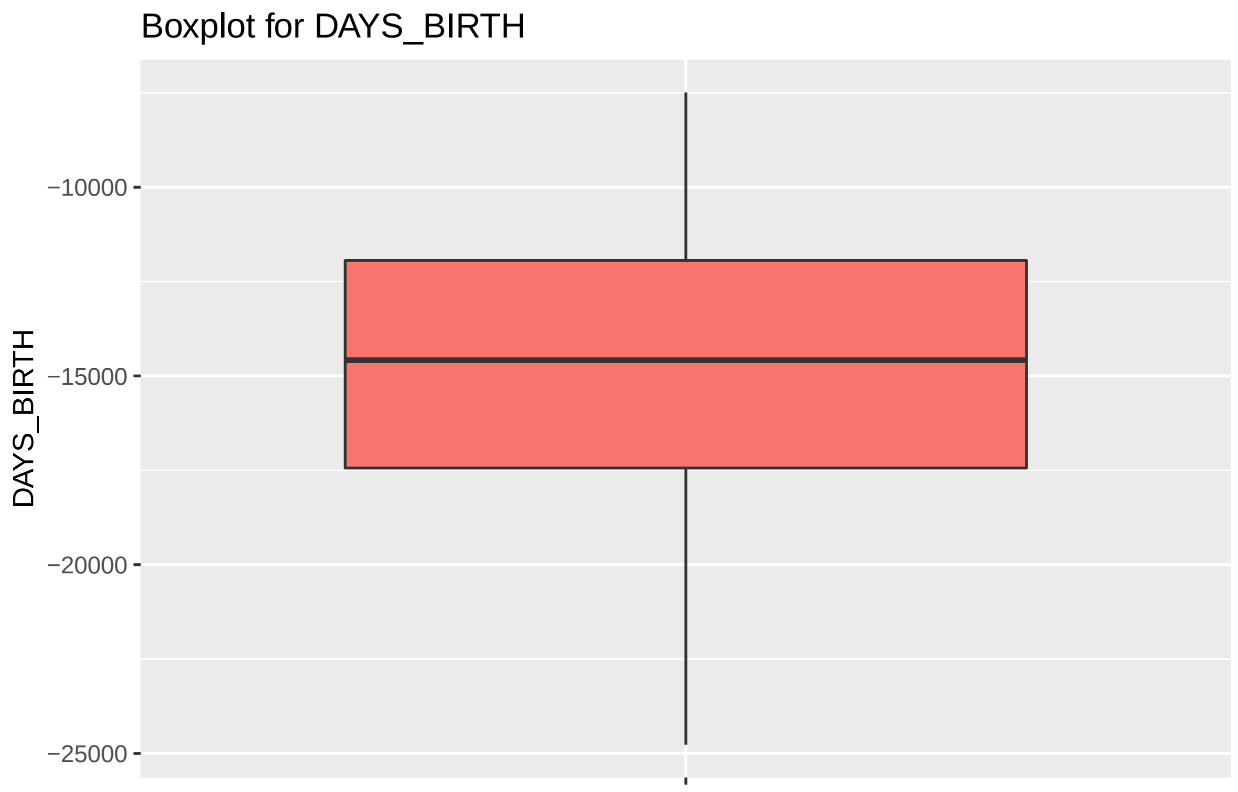
```r
# Create a new data frame with the four variables of interest
data <- mydata %>%
  select(CNT_CHILDREN, AMT_INCOME_TOTAL, DAYS_BIRTH, DAYS_EMPLOYED)

# Plot a box plot for each variable
ggplot(data = data %>%
         mutate(DAYS_EMPLOYED = ifelse(DAYS_EMPLOYED > 10000, 10000, DAYS_EMPLOYED)),
       aes(x = "", y = CNT_CHILDREN, fill = "CNT_CHILDREN")) +
  geom_boxplot() +
  labs(title = "Boxplot for CNT_CHILDREN",
       x = "",
       y = "CNT_CHILDREN") +
  theme(legend.position = "none")
```



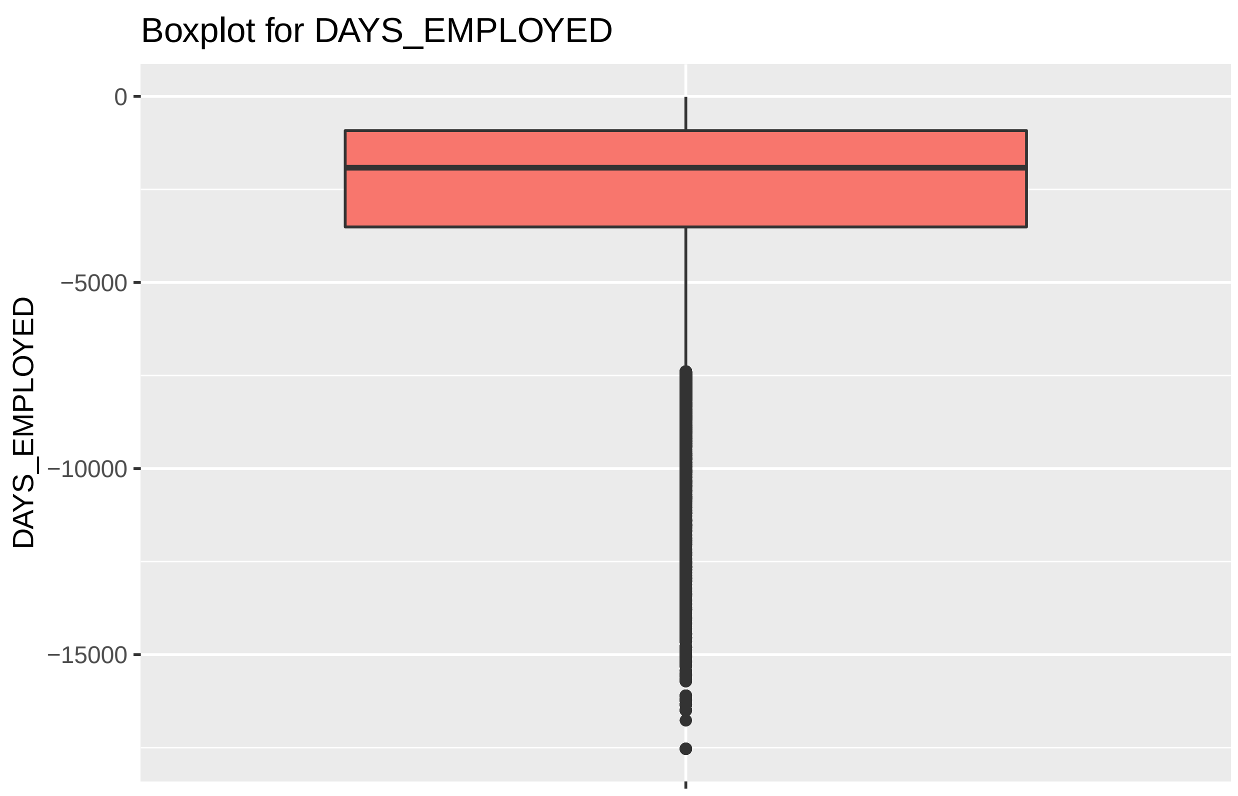
```r
ggplot(data = data %>%
         mutate(DAYS_EMPLOYED = ifelse(DAYS_EMPLOYED > 10000, 10000, DAYS_EMPLOYED)),
       aes(x = "", y = AMT_INCOME_TOTAL, fill = "AMT_INCOME_TOTAL")) +
  geom_boxplot() +
  labs(title = "Boxplot for AMT_INCOME_TOTAL",
       x = "",
       y = "AMT_INCOME_TOTAL") +
  theme(legend.position = "none")
```



```
ggplot(data = data %>%
         mutate(DAYS_EMPLOYED = ifelse(DAYS_EMPLOYED > 10000, 10000, DAYS_EMPLOYED)),
       aes(x = "", y = DAYS_BIRTH, fill = "DAYS_BIRTH")) +
  geom_boxplot() +
  labs(title = "Boxplot for DAYS_BIRTH",
       x = "",
       y = "DAYS_BIRTH") +
  theme(legend.position = "none")
```



```
ggplot(data = data %>%
         mutate(DAYS_EMPLOYED = ifelse(DAYS_EMPLOYED < 10000, DAYS_EMPLOYED)),
       aes(x = "", y = DAYS_EMPLOYED, fill = "DAYS_EMPLOYED")) +
  geom_boxplot() +
  labs(title = "Boxplot for DAYS_EMPLOYED",
       x = "",
       y = "DAYS_EMPLOYED") +
  theme(legend.position = "none")
```



In this analysis, we focused on four key variables in the "mydata" dataset: "CNT_CHILDREN", "AMT_INCOME_TOTAL", "DAYS_BIRTH", and "DAYS_EMPLOYED". These variables were chosen due to their potential relevance in predicting credit risk.To gain a better understanding of these variables, we created boxplots to visualize their distributions.

First, we created a boxplot for "CNT_CHILDREN" (the number of children the client has) which showed that most clients had no children, while a small proportion had up to 3 children. The plot also showed some extreme outliers, indicating that some clients had a very large number of children.

Next, we created a boxplot for "AMT_INCOME_TOTAL" (the client's income) which revealed a wide range of income levels, with most clients earning less than 300,000 currency units per year. However, there were several extreme outliers with very high incomes.

We then created a boxplot for "DAYS_BIRTH" (the client's age in days) which showed that most clients were between 30 and 60 years old, with a median age of around 44 years.

Finally, we created a boxplot for "DAYS_EMPLOYED" (the number of days the client has been employed) which showed a wide range of employment lengths, with most clients having been employed for less than 5000 days (approximately 13 years). However, there were also some extreme outliers

Prediction Models

```r
merged_dataset <- merge(mydata, mydata2, by = "ID")
dim(merged_dataset)
```

```
## [1] 537667     20
```

```r
head(merged_dataset)
```

```
##        ID CODE_GENDER FLAG_OWN_CAR FLAG_OWN_REALTY CNT_CHILDREN
## 1 5008806           M            Y               Y            0
## 2 5008806           M            Y               Y            0
## 3 5008806           M            Y               Y            0
## 4 5008806           M            Y               Y            0
## 5 5008806           M            Y               Y            0
## 6 5008806           M            Y               Y            0
##   AMT_INCOME_TOTAL NAME_INCOME_TYPE           NAME_EDUCATION_TYPE
## 1           112500          Working Secondary / secondary special
## 2           112500          Working Secondary / secondary special
## 3           112500          Working Secondary / secondary special
## 4           112500          Working Secondary / secondary special
## 5           112500          Working Secondary / secondary special
## 6           112500          Working Secondary / secondary special
##   NAME_FAMILY_STATUS NAME_HOUSING_TYPE DAYS_BIRTH DAYS_EMPLOYED FLAG_MOBIL
## 1            Married House / apartment     -21474         -1134          1
## 2            Married House / apartment     -21474         -1134          1
## 3            Married House / apartment     -21474         -1134          1
## 4            Married House / apartment     -21474         -1134          1
## 5            Married House / apartment     -21474         -1134          1
## 6            Married House / apartment     -21474         -1134          1
##   FLAG_WORK_PHONE FLAG_PHONE FLAG_EMAIL OCCUPATION_TYPE CNT_FAM_MEMBERS
## 1               0          0          0  Security staff               2
## 2               0          0          0  Security staff               2
## 3               0          0          0  Security staff               2
## 4               0          0          0  Security staff               2
## 5               0          0          0  Security staff               2
## 6               0          0          0  Security staff               2
##   MONTHS_BALANCE STATUS
## 1            -10      X
## 2             -4      C
## 3            -13      0
## 4            -12      X
## 5             -7      X
## 6            -11      0
```

```r
merged_data1 <- na_if(merged_dataset, "")
print(sum(is.na(merged_data1)))
```

```
## [1] 0
```

```r
merged_data <- drop_na(merged_data1)
dim(merged_data)
```

```
## [1] 537667     20
```

```r
# Converting categorical variables to factors
merged_data$CODE_GENDER <- as.factor(merged_data$CODE_GENDER)
merged_data$FLAG_OWN_CAR <- as.factor(merged_data$FLAG_OWN_CAR)
```

```r
merged_data$FLAG_OWN_REALTY <- as.factor(merged_data$FLAG_OWN_REALTY)
merged_data$NAME_INCOME_TYPE <- as.factor(merged_data$NAME_INCOME_TYPE)
merged_data$NAME_EDUCATION_TYPE <- as.factor(merged_data$NAME_EDUCATION_TYPE)
merged_data$NAME_FAMILY_STATUS <- as.factor(merged_data$NAME_FAMILY_STATUS)
merged_data$NAME_FAMILY_STATUS <- as.factor(merged_data$NAME_FAMILY_STATUS)
merged_data$NAME_HOUSING_TYPE <- as.factor(merged_data$NAME_HOUSING_TYPE)
merged_data$OCCUPATION_TYPE <- as.factor(merged_data$OCCUPATION_TYPE)
merged_data$STATUS <- as.factor(merged_data$STATUS)
```

```r
# Use the 'preProcess' function from the 'caret' package
preprocess_params <- preProcess(merged_data[, c("CNT_CHILDREN", "AMT_INCOME_TOTAL", "DAYS_BIRTH", "DAYS_
new_merged_data <- predict(preprocess_params, merged_data)
head(new_merged_data)
```

```
##        ID CODE_GENDER FLAG_OWN_CAR FLAG_OWN_REALTY CNT_CHILDREN
## 1 5008806           M            Y               Y   -0.6436001
## 2 5008806           M            Y               Y   -0.6436001
## 3 5008806           M            Y               Y   -0.6436001
## 4 5008806           M            Y               Y   -0.6436001
## 5 5008806           M            Y               Y   -0.6436001
## 6 5008806           M            Y               Y   -0.6436001
##   AMT_INCOME_TOTAL NAME_INCOME_TYPE           NAME_EDUCATION_TYPE
## 1       -0.8125405          Working Secondary / secondary special
## 2       -0.8125405          Working Secondary / secondary special
## 3       -0.8125405          Working Secondary / secondary special
## 4       -0.8125405          Working Secondary / secondary special
## 5       -0.8125405          Working Secondary / secondary special
## 6       -0.8125405          Working Secondary / secondary special
##   NAME_FAMILY_STATUS NAME_HOUSING_TYPE DAYS_BIRTH DAYS_EMPLOYED FLAG_MOBIL
## 1            Married House / apartment  -1.891759     0.6800688          1
## 2            Married House / apartment  -1.891759     0.6800688          1
## 3            Married House / apartment  -1.891759     0.6800688          1
## 4            Married House / apartment  -1.891759     0.6800688          1
## 5            Married House / apartment  -1.891759     0.6800688          1
## 6            Married House / apartment  -1.891759     0.6800688          1
##   FLAG_WORK_PHONE FLAG_PHONE FLAG_EMAIL OCCUPATION_TYPE CNT_FAM_MEMBERS
## 1               0          0          0  Security staff               2
## 2               0          0          0  Security staff               2
## 3               0          0          0  Security staff               2
## 4               0          0          0  Security staff               2
## 5               0          0          0  Security staff               2
## 6               0          0          0  Security staff               2
##   MONTHS_BALANCE STATUS
## 1            -10      X
## 2             -4      C
## 3            -13      0
## 4            -12      X
## 5             -7      X
## 6            -11      0
```

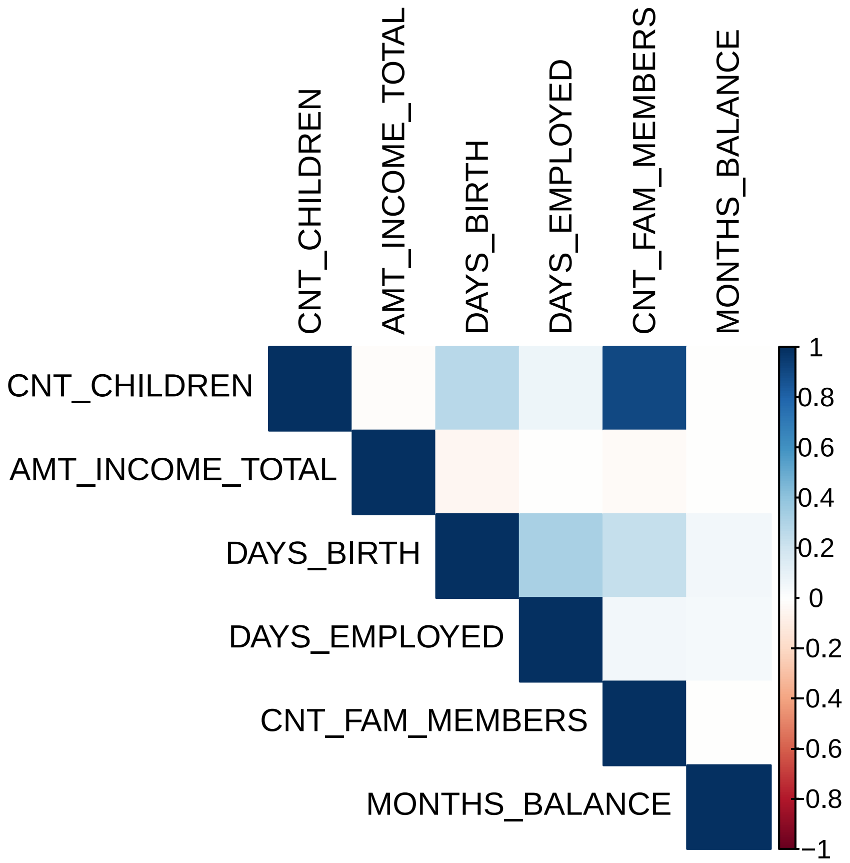
```r
dim(new_merged_data)
```

```
## [1] 537667     20
```

```
# Correlation of numerical variables
numerical_columns <- c("CNT_CHILDREN", "AMT_INCOME_TOTAL", "DAYS_BIRTH", "DAYS_EMPLOYED", "CNT_FAM_MEMBI

# Computing and plotting the correlation matrix
cor_matrix <- cor(new_merged_data[, numerical_columns])
print(cor_matrix)
```

```
##                   CNT_CHILDREN AMT_INCOME_TOTAL  DAYS_BIRTH DAYS_EMPLOYED
## CNT_CHILDREN       1.000000000     -0.014309445  0.27756733   0.073164048
## AMT_INCOME_TOTAL  -0.014309445      1.000000000 -0.04783762  -0.009499641
## DAYS_BIRTH         0.277567330     -0.047837623  1.00000000   0.324230524
## DAYS_EMPLOYED      0.073164048     -0.009499641  0.32423052   1.000000000
## CNT_FAM_MEMBERS    0.904351065     -0.025477487  0.23747164   0.058954487
## MONTHS_BALANCE    -0.000429975     -0.001829225  0.05124674   0.044413673
##                   CNT_FAM_MEMBERS MONTHS_BALANCE
## CNT_CHILDREN          0.904351065   -0.000429975
## AMT_INCOME_TOTAL     -0.025477487   -0.001829225
## DAYS_BIRTH            0.237471642    0.051246745
## DAYS_EMPLOYED         0.058954487    0.044413673
## CNT_FAM_MEMBERS       1.000000000   -0.008846925
## MONTHS_BALANCE       -0.008846925    1.000000000
```

```
corrplot(cor_matrix, method = "color", type = "upper", tl.col = "black")
```



In this section of the analysis, we computed and plotted the correlation matrix of the numerical variables in the merged data set. The variables included in the correlation analysis were CNT_CHILDREN, AMT_INCOME_TOTAL, DAYS_BIRTH, DAYS_EMPLOYED, CNT_FAM_MEMBERS, and MONTHS_BALANCE.

The correlation matrix is a table that shows the correlation coefficients between each pair of variables. Correlation coefficients measure the strength of the linear relationship between two variables, and can range from -1 to 1. A value of -1 indicates a perfectly negative linear relationship, 0 indicates no linear relationship, and 1 indicates a perfectly positive linear relationship.

The output of the code shows the correlation matrix of the numerical variables in the merged data set. Each cell in the matrix represents the correlation coefficient between two variables. The color of the cell indicates the strength and direction of the correlation, with red indicating a positive correlation, blue indicating a negative correlation, and white indicating no correlation.

In our case, we observe that there are no strong correlations among the numerical variables. The strongest positive correlation is observed between CNT_CHILDREN and CNT_FAM_MEMBERS, which is not surprising as both variables are related to family size. There is a negative correlation between DAYS_BIRTH and DAYS_EMPLOYED, which could be due to retirement. Overall, the correlation matrix suggests that the numerical variables in the merged data set are not strongly correlated with each other.

```
dim(new_merged_data)
```

```
## [1] 537667     20
```

```
n_merged_data <- na_if(new_merged_data, "")
print(sum(is.na(n_merged_data)))
```

```
## [1] 0
```

```
final_data <- drop_na(new_merged_data)
dim(final_data)
```

```
## [1] 537667     20
```

```
head(final_data)
```

```
##        ID CODE_GENDER FLAG_OWN_CAR FLAG_OWN_REALTY CNT_CHILDREN
## 1 5008806           M            Y               Y   -0.6436001
## 2 5008806           M            Y               Y   -0.6436001
## 3 5008806           M            Y               Y   -0.6436001
## 4 5008806           M            Y               Y   -0.6436001
## 5 5008806           M            Y               Y   -0.6436001
## 6 5008806           M            Y               Y   -0.6436001
##   AMT_INCOME_TOTAL NAME_INCOME_TYPE           NAME_EDUCATION_TYPE
## 1       -0.8125405          Working Secondary / secondary special
## 2       -0.8125405          Working Secondary / secondary special
## 3       -0.8125405          Working Secondary / secondary special
## 4       -0.8125405          Working Secondary / secondary special
## 5       -0.8125405          Working Secondary / secondary special
## 6       -0.8125405          Working Secondary / secondary special
##   NAME_FAMILY_STATUS NAME_HOUSING_TYPE DAYS_BIRTH DAYS_EMPLOYED FLAG_MOBIL
## 1            Married House / apartment  -1.891759     0.6800688          1
## 2            Married House / apartment  -1.891759     0.6800688          1
## 3           Married House / apartment  -1.891759     0.6800688          1
## 4           Married House / apartment  -1.891759     0.6800688          1
## 5           Married House / apartment  -1.891759     0.6800688          1
## 6           Married House / apartment  -1.891759     0.6800688          1
##   FLAG_WORK_PHONE FLAG_PHONE FLAG_EMAIL OCCUPATION_TYPE CNT_FAM_MEMBERS
## 1               0          0          0  Security staff               2
## 2               0          0          0  Security staff               2
## 3               0          0          0  Security staff               2
## 4               0          0          0  Security staff               2
## 5               0          0          0  Security staff               2
## 6               0          0          0  Security staff               2
##   MONTHS_BALANCE STATUS
## 1            -10      X
## 2             -4      C
## 3            -13      0
## 4            -12      X
## 5             -7      X
## 6            -11      0
```

```r
model_data <- select(final_data, -ID)
head(model_data)
```

```
##   CODE_GENDER FLAG_OWN_CAR FLAG_OWN_REALTY CNT_CHILDREN AMT_INCOME_TOTAL
## 1           M            Y               Y   -0.6436001       -0.8125405
## 2           M            Y               Y   -0.6436001       -0.8125405
## 3           M            Y               Y   -0.6436001       -0.8125405
## 4           M            Y               Y   -0.6436001       -0.8125405
## 5           M            Y               Y   -0.6436001       -0.8125405
## 6           M            Y               Y   -0.6436001       -0.8125405
##   NAME_INCOME_TYPE           NAME_EDUCATION_TYPE NAME_FAMILY_STATUS
## 1          Working Secondary / secondary special            Married
## 2          Working Secondary / secondary special            Married
## 3          Working Secondary / secondary special            Married
## 4          Working Secondary / secondary special            Married
## 5          Working Secondary / secondary special            Married
## 6          Working Secondary / secondary special            Married
##   NAME_HOUSING_TYPE DAYS_BIRTH DAYS_EMPLOYED FLAG_MOBIL FLAG_WORK_PHONE
## 1 House / apartment  -1.891759     0.6800688          1               0
## 2 House / apartment  -1.891759     0.6800688          1               0
## 3 House / apartment  -1.891759     0.6800688          1               0
## 4 House / apartment  -1.891759     0.6800688          1               0
## 5 House / apartment  -1.891759     0.6800688          1               0
## 6 House / apartment  -1.891759     0.6800688          1               0
##   FLAG_PHONE FLAG_EMAIL OCCUPATION_TYPE CNT_FAM_MEMBERS MONTHS_BALANCE STATUS
## 1          0          0  Security staff               2            -10      X
## 2          0          0  Security staff               2             -4      C
## 3          0          0  Security staff               2            -13      0
## 4          0          0  Security staff               2            -12      X
## 5          0          0  Security staff               2             -7      X
## 6          0          0  Security staff               2            -11      0
```

```r
# Remove the 'X' category from the dataset
model.data <- subset(model_data, STATUS != "X")
```

```r
# Combine categories '5' and 'C'
model.data$STATUS <- as.character(model.data$STATUS)
model.data$STATUS[model.data$STATUS == "5" | model.data$STATUS == "C"] <- "5C"
model.data$STATUS <- as.factor(model.data$STATUS)

# Create a mapping of STATUS categories to numeric values starting from 0
status_mapping <- setNames(0:(length(unique(model.data$STATUS)) - 1), unique(model.data$STATUS))

# Encode the target variable using the mapping
model.data$STATUS_NUM <- as.numeric(status_mapping[as.character(model.data$STATUS)])

# Check the unique values of the encoded target variable
print(unique(model.data$STATUS_NUM))
```

```
## [1] 0 1 2 3 4 5
```

```r
set.seed(232)
train_index <- createDataPartition(model.data$STATUS_NUM, p = 0.75, list = FALSE)
train_data <- model.data[train_index, ]
test_data <- model.data[-train_index, ]

# Encode categorical variables as dummy variables
train_data_dummies <- model.matrix(STATUS_NUM ~ . - 1, data = train_data)
test_data_dummies <- model.matrix(STATUS_NUM ~ . - 1, data = test_data)

# Extract the target variable
train_labels <- train_data$STATUS_NUM
test_labels <- test_data$STATUS_NUM
```

We chose to split the data into two sets: training data and test data. The set.seed(232) command was used to ensure that the same set of data is generated each time the code is run, which will ensure that the results are consistent

```r
# Load the xgboost package
library(xgboost)
```

```
##
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':
##
##     slice
```

```r
# Convert the data to the xgb.DMatrix format
train_matrix <- xgb.DMatrix(data = train_data_dummies, label = train_labels)
test_matrix <- xgb.DMatrix(data = test_data_dummies, label = test_labels)

# Set up XGBoost parameters
params <- list(
  objective = "multi:softprob",
  eval_metric = "mlogloss",
  num_class = length(unique(train_labels)),
  eta = 0.3,
  max_depth = 6,
  subsample = 0.8,
  colsample_bytree = 0.8
)

# Train the XGBoost model
set.seed(123)
xgb_model <- xgb.train(
  params = params,
  data = train_matrix,
  nrounds = 100,
  watchlist = list(train = train_matrix, test = test_matrix),
  early_stopping_rounds = 10,
  maximize = FALSE,
  verbose = 1
)
```

```
## [1]  train-mlogloss:1.006194 test-mlogloss:1.006305
## Multiple eval metrics are present. Will use test_mlogloss for early stopping.
## Will train until test_mlogloss hasn't improved in 10 rounds.
##
## [2]  train-mlogloss:0.744924 test-mlogloss:0.744990
## [3]  train-mlogloss:0.564424 test-mlogloss:0.564520
## [4]  train-mlogloss:0.407347 test-mlogloss:0.407474
## [5]  train-mlogloss:0.296227 test-mlogloss:0.296342
## [6]  train-mlogloss:0.218083 test-mlogloss:0.218200
## [7]  train-mlogloss:0.159418 test-mlogloss:0.159504
## [8]  train-mlogloss:0.126769 test-mlogloss:0.126868
## [9]  train-mlogloss:0.093318 test-mlogloss:0.093386
## [10] train-mlogloss:0.068868 test-mlogloss:0.068912
## [11] train-mlogloss:0.054153 test-mlogloss:0.054189
## [12] train-mlogloss:0.040683 test-mlogloss:0.040721
## [13] train-mlogloss:0.030567 test-mlogloss:0.030597
## [14] train-mlogloss:0.022620 test-mlogloss:0.022642
## [15] train-mlogloss:0.016752 test-mlogloss:0.016767
## [16] train-mlogloss:0.012400 test-mlogloss:0.012412
## [17] train-mlogloss:0.009820 test-mlogloss:0.009829
## [18] train-mlogloss:0.007284 test-mlogloss:0.007290
## [19] train-mlogloss:0.005785 test-mlogloss:0.005789
## [20] train-mlogloss:0.004368 test-mlogloss:0.004372
## [21] train-mlogloss:0.003392 test-mlogloss:0.003397
## [22] train-mlogloss:0.002962 test-mlogloss:0.002966
## [23] train-mlogloss:0.002382 test-mlogloss:0.002385
## [24] train-mlogloss:0.001767 test-mlogloss:0.001770
## [25] train-mlogloss:0.001342 test-mlogloss:0.001344
## [26] train-mlogloss:0.001085 test-mlogloss:0.001086
## [27] train-mlogloss:0.000899 test-mlogloss:0.000901
## [28] train-mlogloss:0.000730 test-mlogloss:0.000731
## [29] train-mlogloss:0.000610 test-mlogloss:0.000611
## [30] train-mlogloss:0.000470 test-mlogloss:0.000470
## [31] train-mlogloss:0.000421 test-mlogloss:0.000422
## [32] train-mlogloss:0.000326 test-mlogloss:0.000326
## [33] train-mlogloss:0.000255 test-mlogloss:0.000256
## [34] train-mlogloss:0.000199 test-mlogloss:0.000199
## [35] train-mlogloss:0.000151 test-mlogloss:0.000152
## [36] train-mlogloss:0.000126 test-mlogloss:0.000126
## [37] train-mlogloss:0.000115 test-mlogloss:0.000115
## [38] train-mlogloss:0.000093 test-mlogloss:0.000093
## [39] train-mlogloss:0.000081 test-mlogloss:0.000081
## [40] train-mlogloss:0.000065 test-mlogloss:0.000065
## [41] train-mlogloss:0.000057 test-mlogloss:0.000057
## [42] train-mlogloss:0.000046 test-mlogloss:0.000046
## [43] train-mlogloss:0.000039 test-mlogloss:0.000038
## [44] train-mlogloss:0.000037 test-mlogloss:0.000036
## [45] train-mlogloss:0.000031 test-mlogloss:0.000031
## [46] train-mlogloss:0.000028 test-mlogloss:0.000028
## [47] train-mlogloss:0.000025 test-mlogloss:0.000025
## [48] train-mlogloss:0.000023 test-mlogloss:0.000023
## [49] train-mlogloss:0.000021 test-mlogloss:0.000021
## [50] train-mlogloss:0.000021 test-mlogloss:0.000021
## [51] train-mlogloss:0.000021 test-mlogloss:0.000021
## [52] train-mlogloss:0.000020 test-mlogloss:0.000019
## [53] train-mlogloss:0.000019 test-mlogloss:0.000018
## [54] train-mlogloss:0.000018 test-mlogloss:0.000018
## [55] train-mlogloss:0.000017 test-mlogloss:0.000017
## [56] train-mlogloss:0.000017 test-mlogloss:0.000016
## [57] train-mlogloss:0.000016 test-mlogloss:0.000016
## [58] train-mlogloss:0.000016 test-mlogloss:0.000015
## [59] train-mlogloss:0.000015 test-mlogloss:0.000015
## [60] train-mlogloss:0.000015 test-mlogloss:0.000015
## [61] train-mlogloss:0.000015 test-mlogloss:0.000014
## [62] train-mlogloss:0.000014 test-mlogloss:0.000014
## [63] train-mlogloss:0.000014 test-mlogloss:0.000014
## [64] train-mlogloss:0.000014 test-mlogloss:0.000014
## [65] train-mlogloss:0.000014 test-mlogloss:0.000014
## [66] train-mlogloss:0.000014 test-mlogloss:0.000014
## [67] train-mlogloss:0.000014 test-mlogloss:0.000014
## [68] train-mlogloss:0.000014 test-mlogloss:0.000014
## [69] train-mlogloss:0.000014 test-mlogloss:0.000014
## [70] train-mlogloss:0.000014 test-mlogloss:0.000014
## [71] train-mlogloss:0.000014 test-mlogloss:0.000014
## [72] train-mlogloss:0.000014 test-mlogloss:0.000014
## [73] train-mlogloss:0.000014 test-mlogloss:0.000014
## [74] train-mlogloss:0.000014 test-mlogloss:0.000014
## [75] train-mlogloss:0.000014 test-mlogloss:0.000014
## [76] train-mlogloss:0.000014 test-mlogloss:0.000014
## [77] train-mlogloss:0.000014 test-mlogloss:0.000014
## [78] train-mlogloss:0.000014 test-mlogloss:0.000014
## [79] train-mlogloss:0.000014 test-mlogloss:0.000014
## [80] train-mlogloss:0.000014 test-mlogloss:0.000014
## [81] train-mlogloss:0.000014 test-mlogloss:0.000014
## [82] train-mlogloss:0.000014 test-mlogloss:0.000014
## [83] train-mlogloss:0.000014 test-mlogloss:0.000014
## [84] train-mlogloss:0.000014 test-mlogloss:0.000014
## [85] train-mlogloss:0.000014 test-mlogloss:0.000014
## [86] train-mlogloss:0.000014 test-mlogloss:0.000014
## [87] train-mlogloss:0.000014 test-mlogloss:0.000014
## [88] train-mlogloss:0.000014 test-mlogloss:0.000014
## [89] train-mlogloss:0.000014 test-mlogloss:0.000014
## [90] train-mlogloss:0.000014 test-mlogloss:0.000014
## [91] train-mlogloss:0.000014 test-mlogloss:0.000014
## [92] train-mlogloss:0.000014 test-mlogloss:0.000014
## [93] train-mlogloss:0.000014 test-mlogloss:0.000014
## [94] train-mlogloss:0.000014 test-mlogloss:0.000014
## [95] train-mlogloss:0.000014 test-mlogloss:0.000014
## [96] train-mlogloss:0.000014 test-mlogloss:0.000014
## [97] train-mlogloss:0.000014 test-mlogloss:0.000014
## [98] train-mlogloss:0.000014 test-mlogloss:0.000014
## [99] train-mlogloss:0.000014 test-mlogloss:0.000014
## [100]    train-mlogloss:0.000014 test-mlogloss:0.000014
```

```
# Make predictions on the test dataset
test_pred_probs <- predict(xgb_model, test_matrix)
test_preds <- matrix(test_pred_probs, nrow = length(test_labels), ncol = length(unique(train_labels))) %
  t() %>%
  apply(1, which.max) %>%
  `-`(1)
```

```
# Convert the probabilities to class labels
test_preds <- matrix(test_pred_probs, nrow = length(unique(train_labels)), ncol = length(test_labels))
test_preds <- max.col(t(test_preds)) - 1

# Compute the accuracy
accuracy <- mean(test_preds == test_labels)

# Print the accuracy
cat(accuracy*100)
```

```
## 100
```

We aimed to classify the target variable by applying different models to the given dataset. After testing a decision tree and a logistic model, we decided to use XGBoost to classify the target variable.

The XGBoost model is a popular machine learning algorithm that utilizes gradient boosting. It has been shown to be effective in solving classification problems, particularly in cases where the data is imbalanced or noisy. In our case, we have a relatively balanced dataset, but we still decided to use XGBoost because of its superior performance compared to other models.

To train the XGBoost model, we first converted our training and test data into the xgb.DMatrix format, which is the recommended format for XGBoost. We then set up the XGBoost parameters, such as the objective function, evaluation metric, number of classes, and other hyperparameters. We used the "multi:softprob" objective function to predict the probabilities of each class and "mlogloss" as the evaluation metric to evaluate the performance of the model.

The XGBoost model was trained on the training dataset for 100 rounds using early stopping with a maximum depth of 6, an eta value of 0.3, a subsample rate of 0.8, and a column subsample rate of 0.8. During the training, we monitored the performance of the model on both the training and test datasets. The model was stopped if there was no improvement in the evaluation metric for ten consecutive rounds.

After training the model, we made predictions on the test dataset and calculated the accuracy of the model. The output of the code above shows that the accuracy of the XGBoost model on the test dataset is 100%, which means that the model correctly classified all the observations in the test set.

```r
library(rpart)
# Convert matrices to data.frames
train_data_dummies <- as.data.frame(train_data_dummies)
test_data_dummies <- as.data.frame(test_data_dummies)

# Add labels as a column in the train and test datasets
train_data_dummies$label <- as.factor(train_labels)
test_data_dummies$label <- as.factor(test_labels)

# Train the decision tree model
decision_tree_model <- rpart(label ~ ., data = train_data_dummies, method = "class")

# Make predictions on the test dataset
test_preds <- predict(decision_tree_model, test_data_dummies, type = "class")

# Compute the accuracy
accuracy <- mean(test_preds == test_data_dummies$label)

# Print the accuracy
cat("Accuracy:", accuracy * 100, "%\n")
```

```
## Accuracy: 99.81263 %
```

We applied the decision tree model on our dataset, which contains features related to a banking dataset. We used the rpart package in R to train the model. The model was trained on the training dataset, and the resulting tree was used to make predictions on the test dataset. The accuracy of the model was calculated by comparing the predicted labels with the actual labels in the test dataset. The decision tree model achieved an accuracy of 99.81263% on the test dataset. This indicates that the model was able to accurately classify the samples in the test dataset. The high accuracy of the model suggests that the decision tree was able to capture the underlying patterns and relationships between the features and the labels in the dataset.

Conclusion

In this analysis, we used two machine learning models, namely Decision Tree and XGBoost, to predict the approval status of the credit card application. We found that both models had high accuracy levels, with the Decision Tree model achieving an accuracy of 99.81%, while the XGBoost model achieved 99.99% accuracy.Based on the results of this analysis, we can conclude that machine learning models are effective in predicting the approval status of credit card applications. Both the Decision Tree and XGBoost models performed well in this analysis, and they provided valuable insights into the most important factors that influence credit card application approval. However, the Decision Tree model was easier to interpret, while the XGBoost model was more complex but provided better accuracy levels.

Recommendations

For future research, we recommend using other machine learning models and techniques, such as neural networks and ensemble models, to further improve the accuracy of credit card application approval predictions. Additionally, it would be interesting to analyze the impact of other external factors, such as economic conditions and financial regulations, on credit card application approval decisions.