Loan Approval

Shaibu Abdullateef Topa (CST/20/COM/00591)

2025-03-08

#Load and inspect the dataset

```
# Load data manipulation library
library(dplyr)

# Read the loan data set (same folder as this R script)
loan_data <- read.csv("loan_approval.csv")
```

#Explore Dataset

```
# Show first few rows
head(loan_data)
```

```
##   loan_id no_of_dependents     education self_employed income_annum loan_amount
## 1       1                2      Graduate            No      9600000    29900000
## 2       2                0  Not Graduate           Yes      4100000    12200000
## 3       3                3      Graduate            No      9100000    29700000
## 4       4                3      Graduate            No      8200000    30700000
## 5       5                5  Not Graduate           Yes      9800000    24200000
## 6       6                0      Graduate           Yes      4800000    13500000
##   loan_term cibil_score residential_assets_value commercial_assets_value
## 1        12         778                  2400000                17600000
## 2         8         417                  2700000                 2200000
## 3        20         506                  7100000                 4500000
## 4         8         467                 18200000                 3300000
## 5        20         382                 12400000                 8200000
## 6        10         319                  6800000                 8300000
##   luxury_assets_value bank_asset_value loan_status
## 1            22700000          8000000    Approved
## 2             8800000          3300000    Rejected
## 3            33300000         12800000    Rejected
## 4            23300000          7900000    Rejected
## 5            29400000          5000000    Rejected
## 6            13700000          5100000    Rejected
```

```
# Show last few rows
tail(loan_data)
```

```
##      loan_id no_of_dependents     education self_employed income_annum
## 4264    4264                3      Graduate            No      5000000
## 4265    4265                5      Graduate           Yes      1000000
## 4266    4266                0  Not Graduate           Yes      3300000
## 4267    4267                2  Not Graduate            No      6500000
## 4268    4268                1  Not Graduate            No      4100000
## 4269    4269                1      Graduate            No      9200000
```

```
##      loan_amount loan_term cibil_score residential_assets_value
## 4264    12700000        14         865                  4700000
## 4265     2300000        12         317                  2800000
## 4266    11300000        20         559                  4200000
## 4267    23900000        18         457                  1200000
## 4268    12800000         8         780                  8200000
## 4269    29700000        10         607                 17800000
##      commercial_assets_value luxury_assets_value bank_asset_value loan_status
## 4264                 8100000            19500000          6300000    Approved
## 4265                  500000             3300000           800000    Rejected
## 4266                 2900000            11000000          1900000    Approved
## 4267                12400000            18100000          7300000    Rejected
## 4268                  700000            14100000          5800000    Approved
## 4269                11800000            35700000         12000000    Approved
```

```r
#Show the dataset dimension
dim(loan_data)
```

```
## [1] 4269   13
```

#Summary of dataset

```r
# Check dataset structure
str(loan_data)
```

```
## 'data.frame':    4269 obs. of  13 variables:
##  $ loan_id                 : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ no_of_dependents        : int  2 0 3 3 5 0 5 2 0 5 ...
##  $ education               : chr  " Graduate" " Not Graduate" " Graduate" " Graduate" ...
##  $ self_employed           : chr  " No" " Yes" " No" " No" ...
##  $ income_annum            : int  9600000 4100000 9100000 8200000 9800000 4800000 8700000 5700000 800
##  $ loan_amount             : int  29900000 12200000 29700000 30700000 24200000 13500000 33000000 1500
##  $ loan_term               : int  12 8 20 8 20 10 4 20 20 10 ...
##  $ cibil_score             : int  778 417 506 467 382 319 678 382 782 388 ...
##  $ residential_assets_value: int  2400000 2700000 7100000 18200000 12400000 6800000 22500000 1320000
##  $ commercial_assets_value : int  17600000 2200000 4500000 3300000 8200000 8300000 14800000 5700000 8
##  $ luxury_assets_value     : int  22700000 8800000 33300000 23300000 29400000 13700000 29200000 1180
##  $ bank_asset_value        : int  8000000 3300000 12800000 7900000 5000000 5100000 4300000 6000000 60
##  $ loan_status             : chr  " Approved" " Rejected" " Rejected" " Rejected" ...
```

```r
# Summary statistics of the loan dataset
summary(loan_data)
```

```
##     loan_id     no_of_dependents  education          self_employed
##  Min.   :   1   Min.   :0.000    Length:4269        Length:4269
##  1st Qu.:1068   1st Qu.:1.000    Class :character   Class :character
##  Median :2135   Median :3.000    Mode  :character   Mode  :character
##  Mean   :2135   Mean   :2.499
##  3rd Qu.:3202   3rd Qu.:4.000
##  Max.   :4269   Max.   :5.000
##   income_annum      loan_amount         loan_term     cibil_score
##  Min.   : 200000   Min.   :  300000   Min.   : 2.0   Min.   :300.0
##  1st Qu.:2700000   1st Qu.: 7700000   1st Qu.: 6.0   1st Qu.:453.0
##  Median :5100000   Median :14500000   Median :10.0   Median :600.0
##  Mean   :5059124   Mean   :15133450   Mean   :10.9   Mean   :599.9
##  3rd Qu.:7500000   3rd Qu.:21500000   3rd Qu.:16.0   3rd Qu.:748.0
##  Max.   :9900000   Max.   :39500000   Max.   :20.0   Max.   :900.0
```

```
##  residential_assets_value commercial_assets_value luxury_assets_value
##  Min.   : -100000         Min.   :       0        Min.   :  300000
##  1st Qu.: 2200000         1st Qu.: 1300000        1st Qu.: 7500000
##  Median : 5600000         Median : 3700000        Median :14600000
##  Mean   : 7472617         Mean   : 4973155        Mean   :15126306
##  3rd Qu.:11300000         3rd Qu.: 7600000        3rd Qu.:21700000
##  Max.   :29100000         Max.   :19400000        Max.   :39200000
##  bank_asset_value   loan_status
##  Min.   :       0   Length:4269
##  1st Qu.: 2300000   Class :character
##  Median : 4600000   Mode  :character
##  Mean   : 4976692
##  3rd Qu.: 7100000
##  Max.   :14700000
```

```r
# Check number of distinct values in the entire dataset
sapply(loan_data, function(x) length(unique(x)))
```

```
##                  loan_id         no_of_dependents                education
##                     4269                        6                        2
##            self_employed             income_annum              loan_amount
##                        2                       98                      378
##                loan_term              cibil_score residential_assets_value
##                       10                      601                      278
##  commercial_assets_value      luxury_assets_value         bank_asset_value
##                      188                      379                      146
##              loan_status
##                        2
```

#From the above, categorical feature :education, self_emplyed and loan_status each has 2 unique values respectively [Graduate,Not],[No,Yes],[Approved,Rejected]

#Data Cleaning

```r
# Check for duplicates
sum(duplicated(loan_data))
```

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

#Handle Missing Values (if any)

```r
# Count missing values in each column
colSums(is.na(loan_data))
```

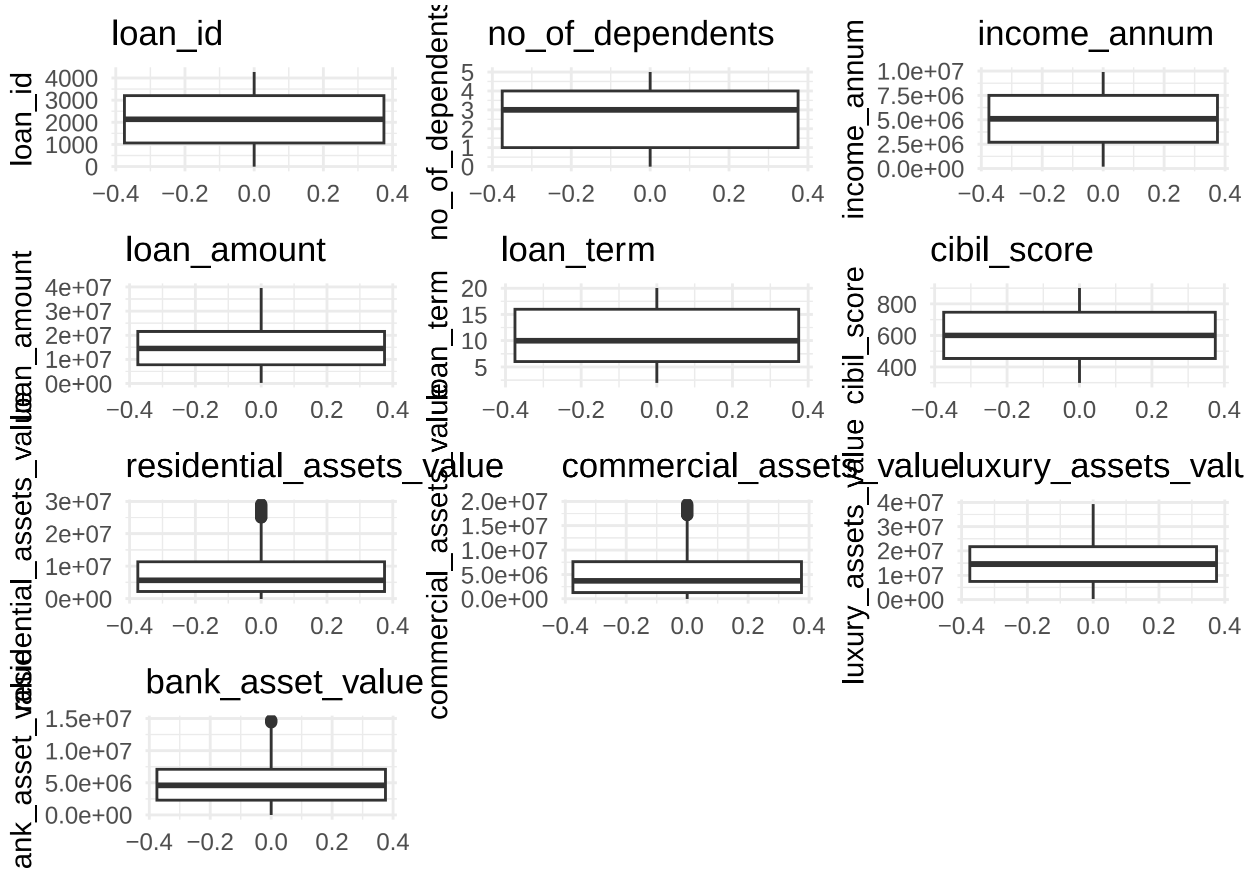
```
##                  loan_id         no_of_dependents                education
##                        0                        0                        0
##            self_employed             income_annum              loan_amount
##                        0                        0                        0
##                loan_term              cibil_score residential_assets_value
##                        0                        0                        0
##  commercial_assets_value      luxury_assets_value         bank_asset_value
##                        0                        0                        0
##              loan_status
##                        0
```

#Removed unnecessary spaces in column names and values

```r
names(loan_data) <- gsub(" ", "", names(loan_data))
```

#Outlier mining

```r
#Detecting Outlier by Visualization
# Load required libraries
library(ggplot2)
library(gridExtra)

# Create empty list to store plots
plot_list <- list()

# Create boxplot for each numeric column
num_cols <- sapply(loan_data, is.numeric)
numeric_data <- loan_data[, num_cols]

# Create individual boxplots
for(col in names(numeric_data)) {
  p <- ggplot(numeric_data, aes_string(y = col)) +
    geom_boxplot() +
    labs(title = col) +
    theme_minimal()

  plot_list[[col]] <- p
}

# Calculate grid dimensions (trying to approximate 4x4 layout)
n_plots <- length(plot_list)
n_rows <- ceiling(sqrt(n_plots))
n_cols <- ceiling(n_plots / n_rows)

# Arrange plots in a grid and display
grid.arrange(grobs = plot_list, nrow = n_rows, ncol = n_cols)
```



#Outliers detected in features bank_asset_value: This indicates that there are few applicants having more than 1,400,000 in their bank accounts residental_assets_value and commercial_assets_value: indicating there are few applicants having more value of residential and commercial assets

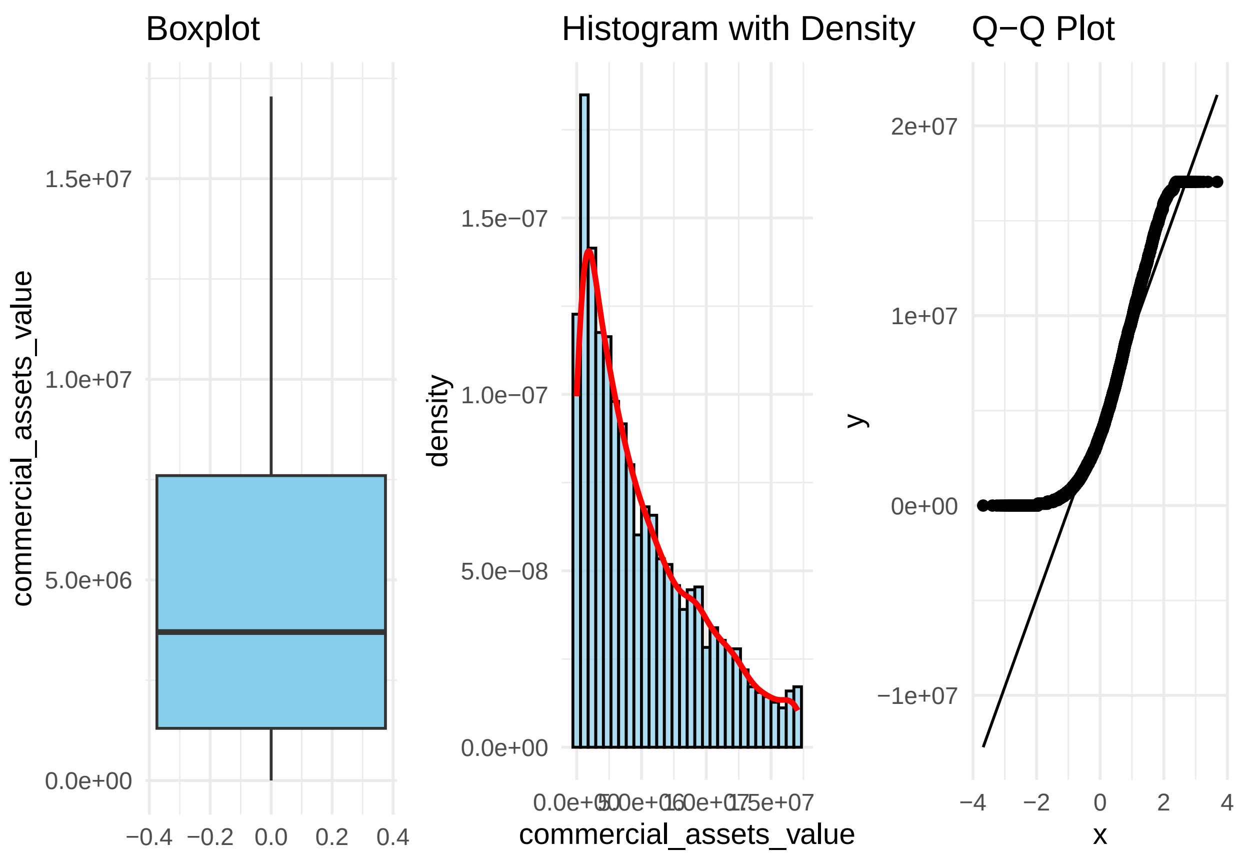
#Treating Outliers

```r
# Function to cap outliers
cap_outliers <- function(df, column, method = 'IQR') {
  if (method == 'IQR') {
    Q1 <- quantile(df[[column]], 0.25, na.rm = TRUE)
    Q3 <- quantile(df[[column]], 0.75, na.rm = TRUE)
    IQR_val <- Q3 - Q1
    lower_bound <- Q1 - 1.5 * IQR_val
    upper_bound <- Q3 + 1.5 * IQR_val
  } else if (method == 'zscore') {
    mean_val <- mean(df[[column]], na.rm = TRUE)
    std_val <- sd(df[[column]], na.rm = TRUE)
    lower_bound <- mean_val - 3 * std_val
    upper_bound <- mean_val + 3 * std_val
  }

  # Cap the values using pmin/pmax (R's equivalent to np.clip)
  df[[column]] <- pmin(pmax(df[[column]], lower_bound), upper_bound)
  return(df)
}

# Apply to all numerical columns
```

```r
num_cols <- sapply(loan_data, is.numeric)
for (col in names(loan_data[, num_cols])) {
  loan_data <- cap_outliers(loan_data, col, method = 'IQR')  # or 'zscore'
}

dim(loan_data)
```

```
## [1] 4269   13
```

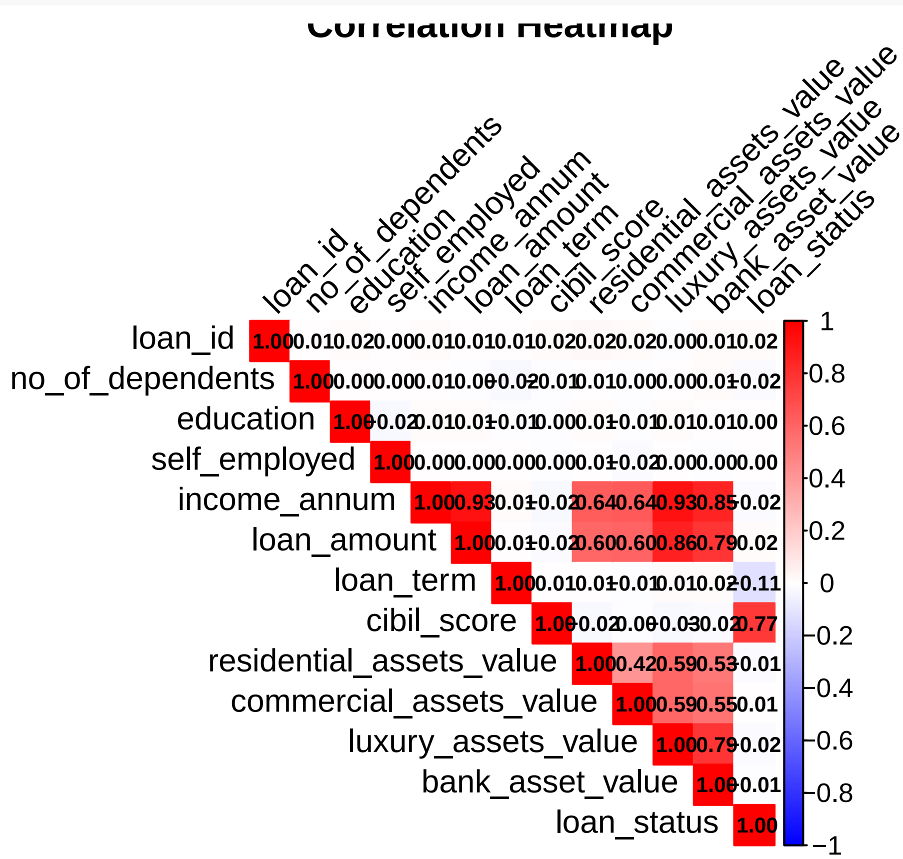
```r
# Load necessary libraries
library(ggplot2)
library(gridExtra)
library(ggpubr)

# Boxplot
p1 <- ggplot(loan_data, aes(y = commercial_assets_value)) +
  geom_boxplot(fill = "skyblue") +
  ggtitle("Boxplot") +
  theme_minimal()

# Histogram with density curve - using after_stat() instead of ..density..
p2 <- ggplot(loan_data, aes(x = commercial_assets_value)) +
  geom_histogram(aes(y = after_stat(density)), bins = 30, fill = "skyblue", color = "black", alpha = 0.
  geom_density(color = "red", linewidth = 1) +
  ggtitle("Histogram with Density") +
  theme_minimal()

# Q-Q plot
p3 <- ggplot(loan_data, aes(sample = commercial_assets_value)) +
  stat_qq() +
  stat_qq_line() +
  ggtitle("Q-Q Plot") +
  theme_minimal()

# Combine all plots
grid.arrange(p1, p2, p3, ncol = 3)
```



1 Exploratory Data Analysis (EDA)

```r
library(ggplot2)
library(reshape2)

# Correlation Between Features


# First, clean the spaces in character columns
loan_data_clean <- loan_data
char_cols <- sapply(loan_data, is.character)

for (col in names(loan_data)[char_cols]) {
  loan_data_clean[[col]] <- trimws(loan_data_clean[[col]])
}

# Make a copy for correlation
loan_data_corr <- loan_data_clean

# Encode binary categorical variables
binary_cats <- c('education', 'self_employed', 'loan_status')
encoding_map <- list(
  education = c('Not Graduate' = 0, 'Graduate' = 1),
  self_employed = c('No' = 0, 'Yes' = 1),
```

```
  loan_status = c('Rejected' = 0, 'Approved' = 1)
)

# Apply the encoding
for (col in binary_cats) {
  loan_data_corr[[col]] <- as.numeric(as.character(encoding_map[[col]][loan_data_corr[[col]]]))
}

# Create and plot correlation matrix
library(corrplot)
# Only include numeric columns
numeric_cols <- sapply(loan_data_corr, is.numeric)
corr_matrix <- cor(loan_data_corr[, numeric_cols], use = "complete.obs")

# Plot
corrplot(corr_matrix, method = "color", type = "upper",
         addCoef.col = "black", number.cex = 0.7,
         tl.col = "black", tl.srt = 45,
         col = colorRampPalette(c("blue", "white", "red"))(200),
         title = "Correlation Heatmap")
```



-The correlation analysis reveals key relationships between financial factors and loan approval

-Strong positive correlations exist between loan amount and income, luxury asset value and income, as well as bank asset value and income.

-Loan status has a high positive correlation with CIBIL score, indicating that credit history significantly impacts loan approval.

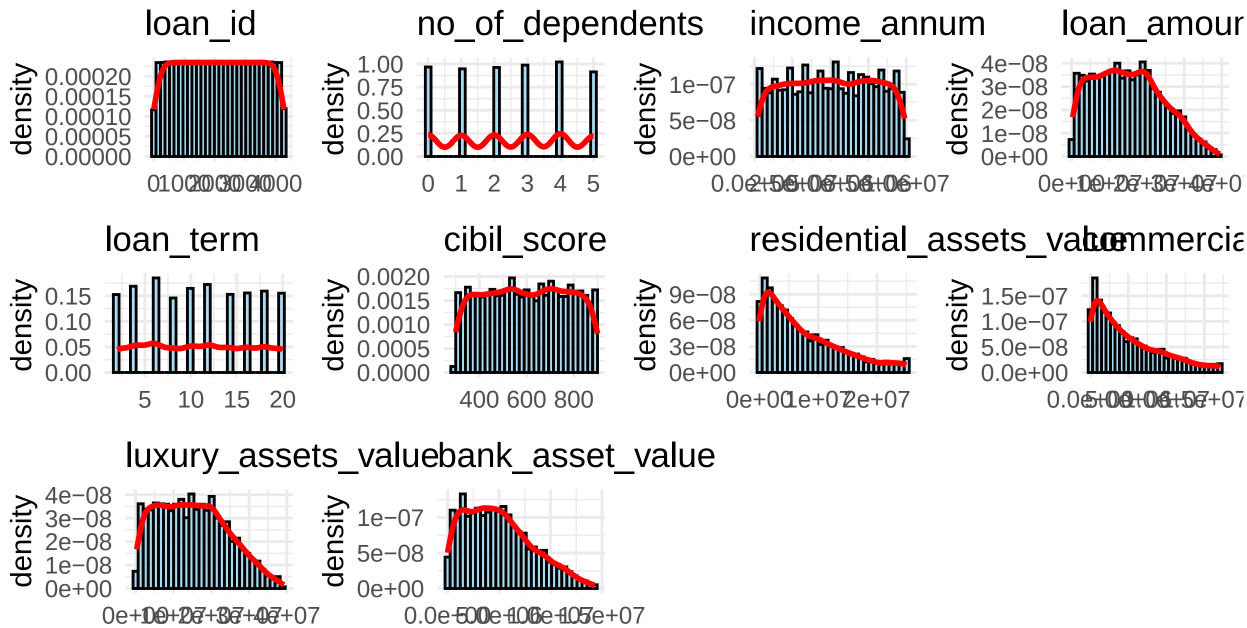
-Moderate positive correlations are observed between various asset values (residential, commercial, luxury) and loan amount.

-Loan status negatively correlates with bank, residential, and luxury asset values, as well as income and number of dependents.

-Loan approval is not strongly influenced by factors like education or commercial asset value.

-Applicants requesting longer loan terms tend to face more rejections, suggesting that longer repayment periods decrease approval chances.

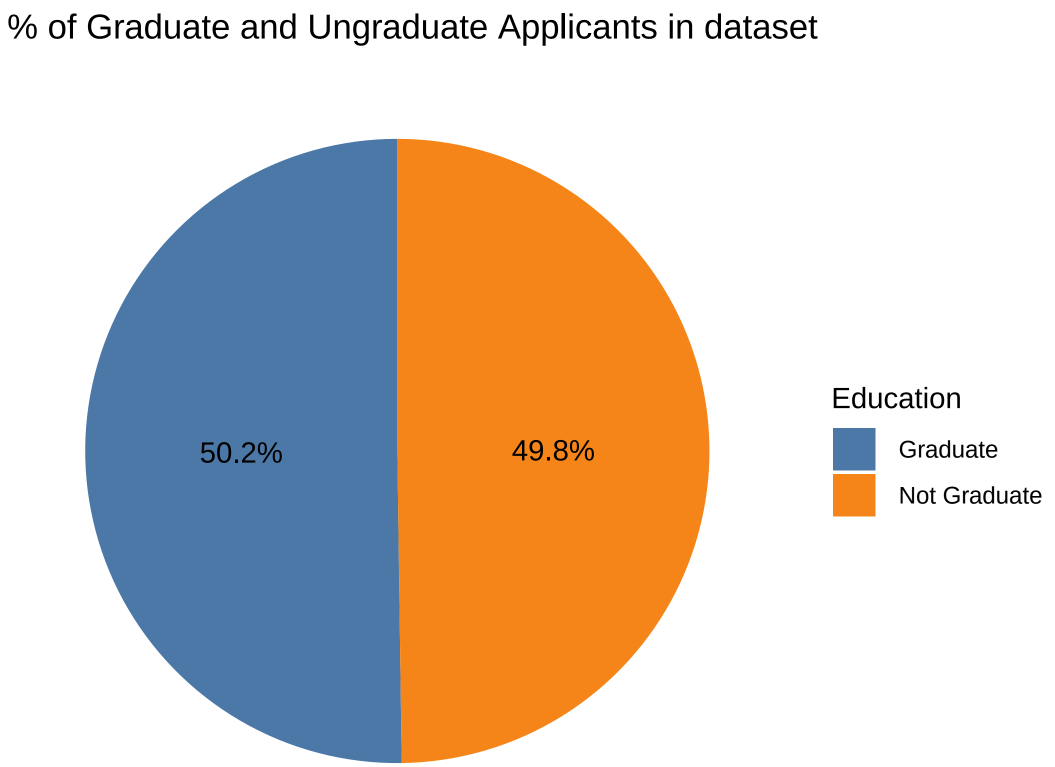
```
#DISTRIBUTION OF THE DATASET EACH FEATURE
# Load required libraries
library(ggplot2)
library(gridExtra)




# Select numeric columns
numeric_columns <- names(loan_data)[sapply(loan_data, is.numeric)]

# Create empty list to store plots
plot_list <- list()

# Create histogram with density curve for each numeric column
for (col in numeric_columns) {
  p <- ggplot(loan_data, aes_string(x = col)) +
    geom_histogram(aes(y = after_stat(density)),
                   bins = 30,
                   fill = "skyblue",
                   color = "black",
                   alpha = 0.7) +
    geom_density(color = "red", linewidth = 1) +
    ggtitle(col) +
    theme_minimal() +
    theme(axis.title.x = element_blank())

  plot_list[[col]] <- p
}

# Calculate grid dimensions (4x4 layout or whatever fits the number of columns)
n_plots <- length(plot_list)
n_rows <- ceiling(sqrt(n_plots))
n_cols <- ceiling(n_plots / n_rows)

# Arrange plots in a grid and display
grid.arrange(grobs = plot_list, nrow = 4, ncol = 4)
```



There are more applicants in the dataset they are either renting or living in other people space -More applicants are Approved in the dataset -There are more applicants having 4 no of dependents and less with 0 - 3 and 5 -Income annum majority lies b/w 0.4 and 0.6 -majority is in loan amount 1.1 -there are more applicants with loan term of 6 years -many applicants donot have any commercial asset -majority luxury asset value is 1.5 -More applicants have the bank asset value of 0.1 -dataset have almost equal educated and uneducated / self - employed and not self emplyed applicants

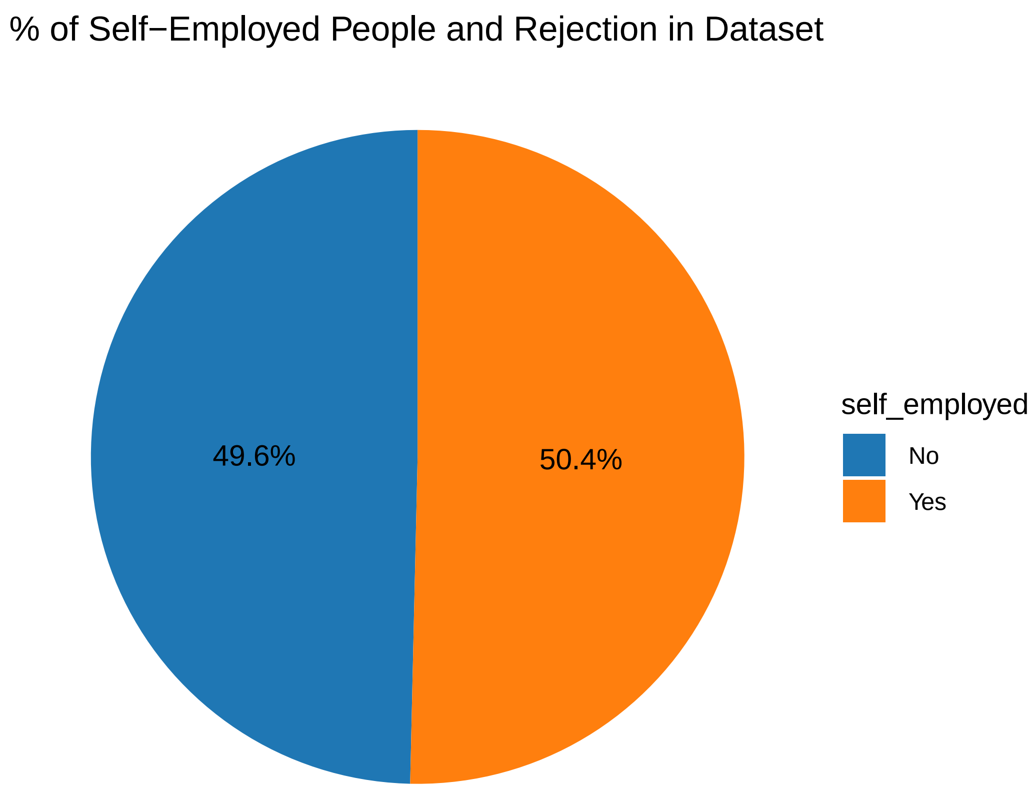
```
# % of Graduate and Ungraduate Applicants in dataset

# Load required libraries
library(ggplot2)


# Count the frequency of each education category
education_counts <- table(loan_data$education)
education_df <- as.data.frame(education_counts)
names(education_df) <- c("education", "count")

# Create the pie chart

ggplot(education_df, aes(x = "", y = count, fill = education)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar("y", start = 0) +
  # Add percentage labels
  geom_text(aes(label = paste0(round(count/sum(count)*100, 1), "%")),
            position = position_stack(vjust = 0.5)) +
  # Custom colors similar to Plotly T10
```

```
  scale_fill_manual(values = c("#4C78A8", "#F58518")) +
  # Customize appearance

  labs(title = "% of Graduate and Ungraduate Applicants in dataset",
       x = NULL, y = NULL, fill = "Education") +
  theme_minimal() +
  theme(axis.text = element_blank(),
        axis.ticks = element_blank(),
        panel.grid = element_blank())
```



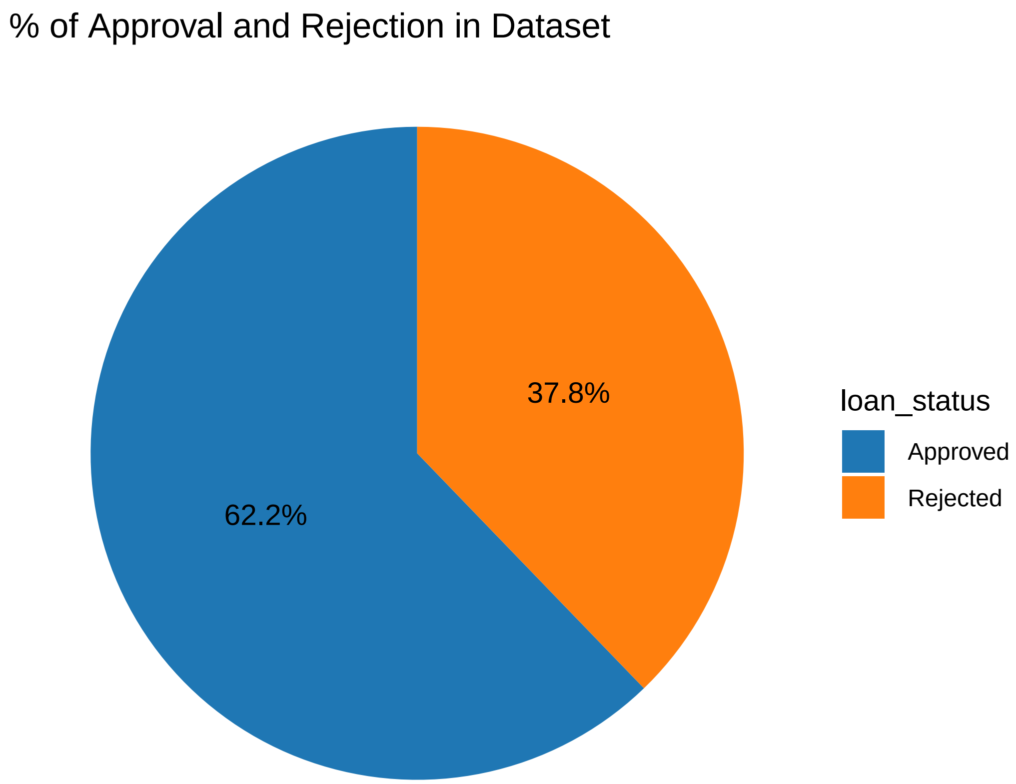

```
# % of self_employed people and rejection in dataset

# Load necessary library
library(ggplot2)

# Assuming loan_data is your data frame
# Summarize the data to get counts for each self_employed status
self_employed_counts <- aggregate(loan_id ~ self_employed, data = loan_data, FUN = length)

# Calculate the percentage for each self_employed status
self_employed_counts$percentage <- self_employed_counts$loan_id / sum(self_employed_counts$loan_id) * 1

# Create the pie chart with percentage labels
ggplot(self_employed_counts, aes(x = "", y = loan_id, fill = self_employed)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar(theta = "y") +
```

```
  labs(title = "% of Self-Employed People and Rejection in Dataset") +
  theme_void() +
  scale_fill_manual(values = c("#1f77b4", "#ff7f0e", "#2ca02c", "#d62728", "#9467bd", "#8c564b", "#e377
  geom_text(aes(label = paste0(round(percentage, 1), "%")),
            position = position_stack(vjust = 0.5))
```




```
# % Approval and Rejection in Dataset

# Load necessary library
library(ggplot2)

# Assuming loan_data is your data frame
# Summarize the data to get counts for each loan_status
loan_status_counts <- aggregate(loan_id ~ loan_status, data = loan_data, FUN = length)

# Calculate the percentage for each loan_status
loan_status_counts$percentage <- loan_status_counts$loan_id / sum(loan_status_counts$loan_id) * 100

# Create the pie chart with percentage labels
ggplot(loan_status_counts, aes(x = "", y = loan_id, fill = loan_status)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar(theta = "y") +
  labs(title = "% of Approval and Rejection in Dataset") +
  theme_void() +
  scale_fill_manual(values = c("#1f77b4", "#ff7f0e", "#2ca02c", "#d62728", "#9467bd", "#8c564b", "#e377
  geom_text(aes(label = paste0(round(percentage, 1), "%")),
```

```
        position = position_stack(vjust = 0.5))
```

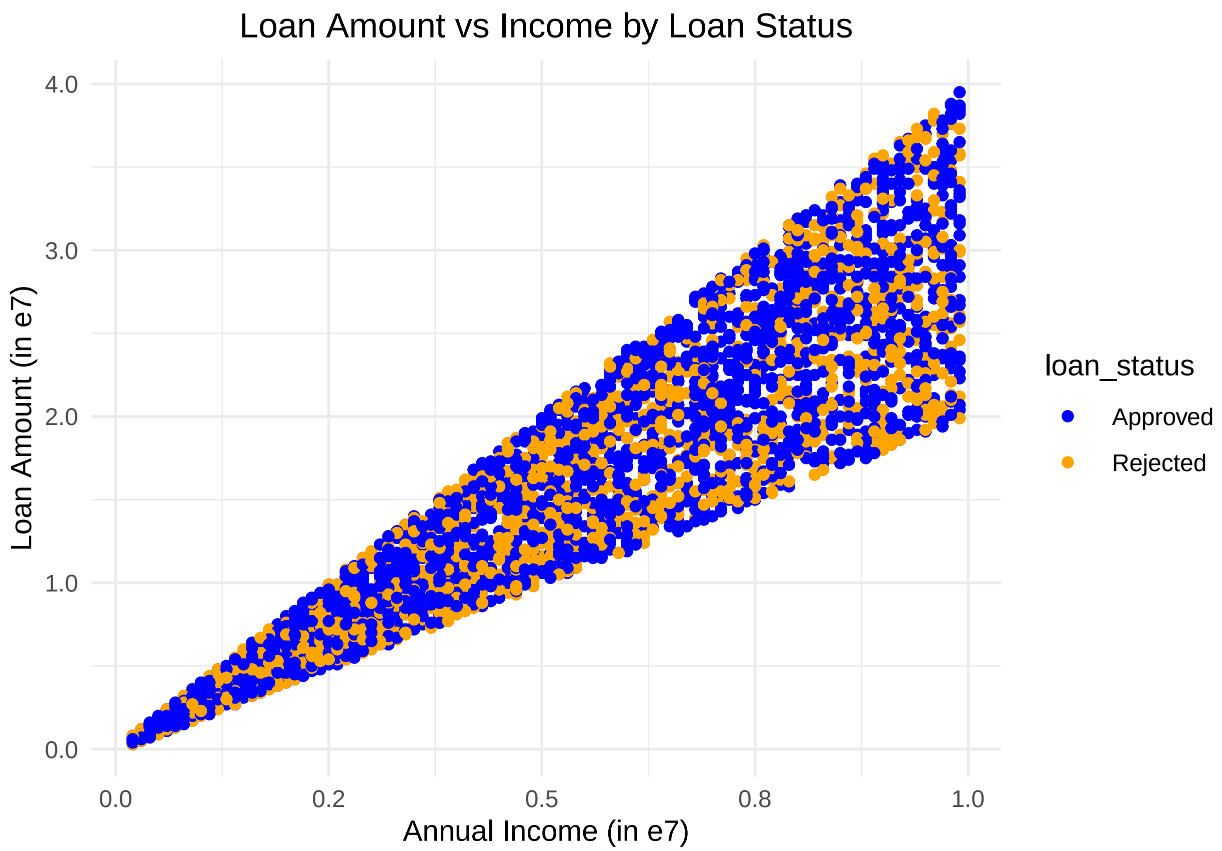


-sample has slightly more educated Applicants , and slightly more self_employed Applicants -dataset have more Approved loan status then Rejected

2 Visualizing NUMERIC FEATURES RELATIONSHIP(CORRELATION)

```
# Load necessary library
library(ggplot2)

# Define custom label functions for the axes
custom_labels <- function(x) sprintf("%.1f", x / 1e7)

# Create the scatter plot
ggplot(loan_data, aes(x = income_annum, y = loan_amount, color = loan_status)) +
  geom_point() +
  labs(
    x = "Annual Income (in e7)",
    y = "Loan Amount (in e7)",
    title = "Loan Amount vs Income by Loan Status"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_color_manual(values = c("blue", "orange", "green", "red", "purple", "brown", "pink", "gray", "ye
  scale_x_continuous(labels = custom_labels) +
```

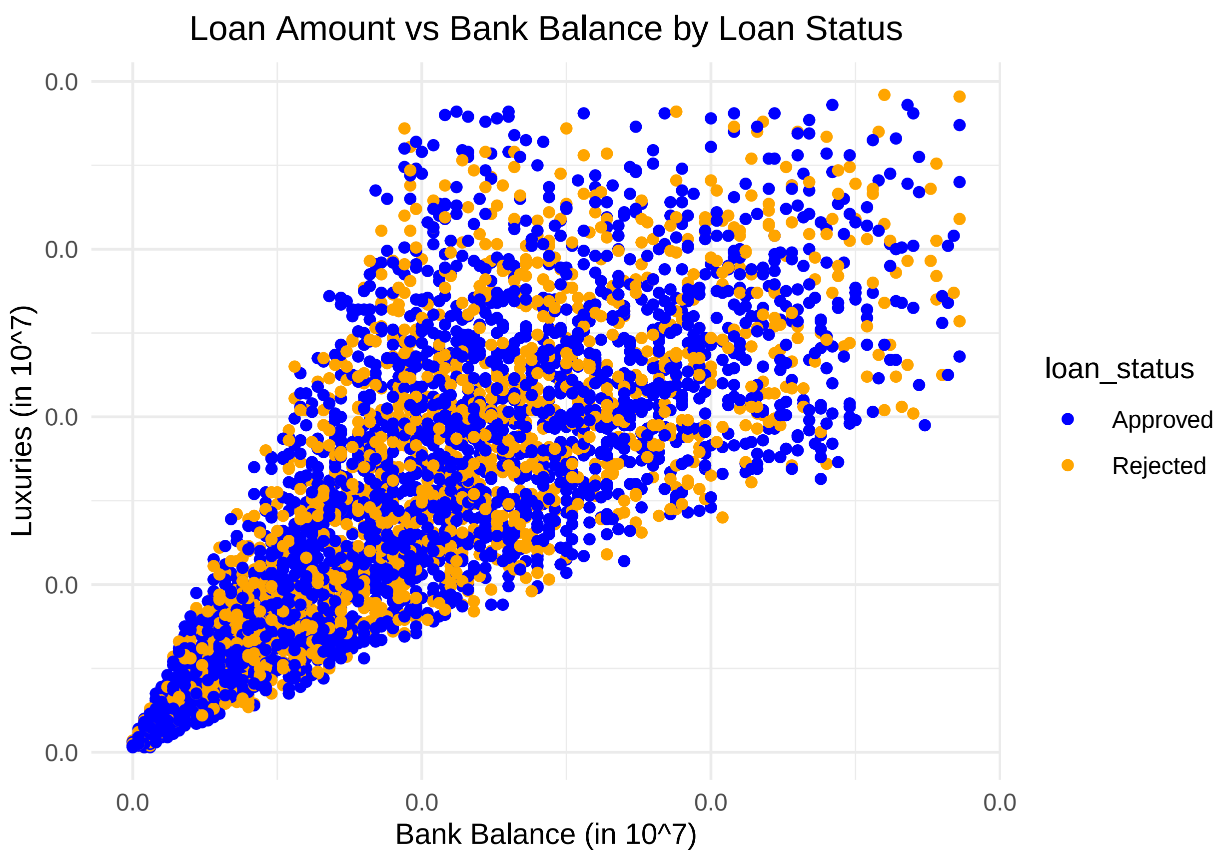
```
  scale_y_continuous(labels = custom_labels)
```



-Applicants with high income tends to take apply for high loan amounts

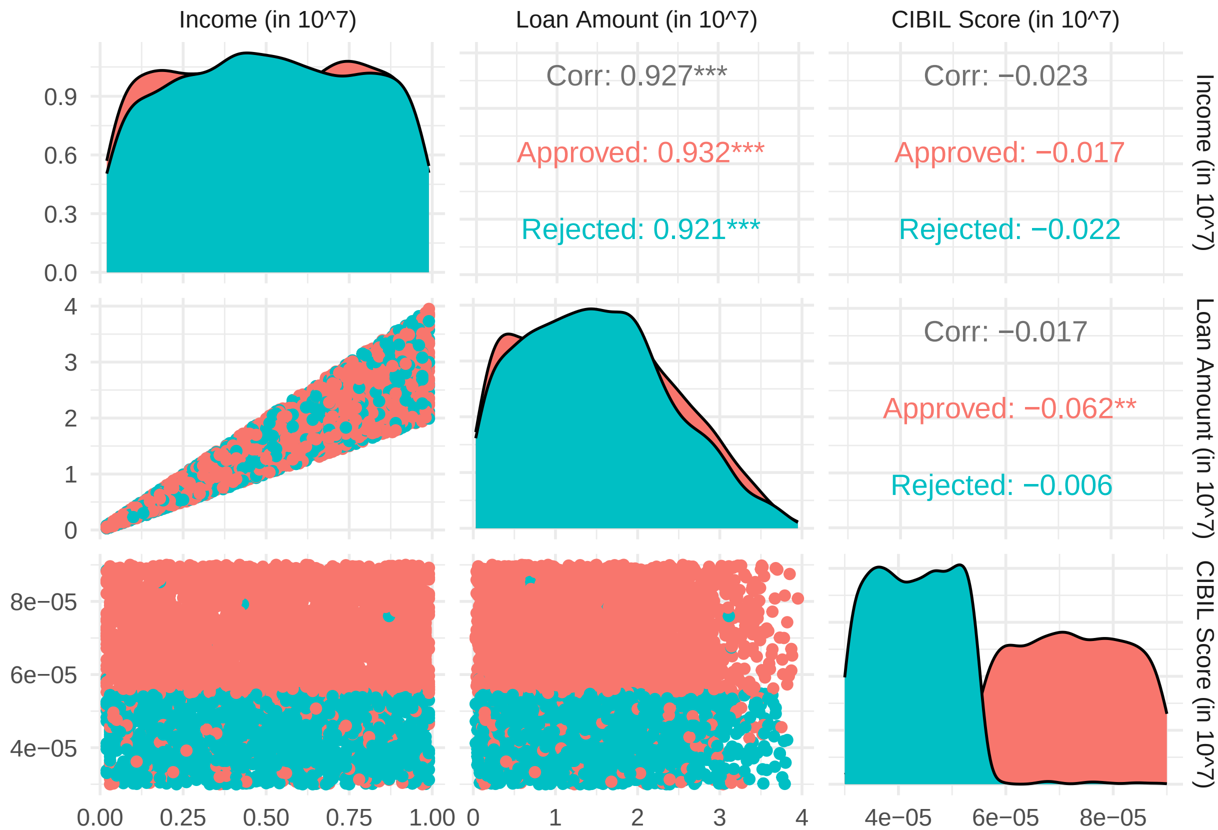
-Loan Amount vs Bank Balance by Loan Status

```
# Load necessary library
library(ggplot2)

# Scale the data by dividing by 10^7
loan_data_scaled <- loan_data
loan_data_scaled$bank_asset_value_scaled <- loan_data_scaled$bank_asset_value / 1e7
loan_data_scaled$luxury_assets_value_scaled <- loan_data_scaled$luxury_assets_value / 1e7

# Create the scatter plot
ggplot(loan_data_scaled, aes(x = bank_asset_value_scaled, y = luxury_assets_value_scaled, color = loan_s
  geom_point() +
  labs(
    x = "Bank Balance (in 10^7)",
    y = "Luxuries (in 10^7)",
    title = "Loan Amount vs Bank Balance by Loan Status"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_color_manual(values = c("blue", "orange", "green", "red", "purple", "brown", "pink", "gray", "ye
  scale_x_continuous(labels = custom_labels) +
  scale_y_continuous(labels = custom_labels)
```



-Applicants with more balance in their accounts tend to buy high value luxury items

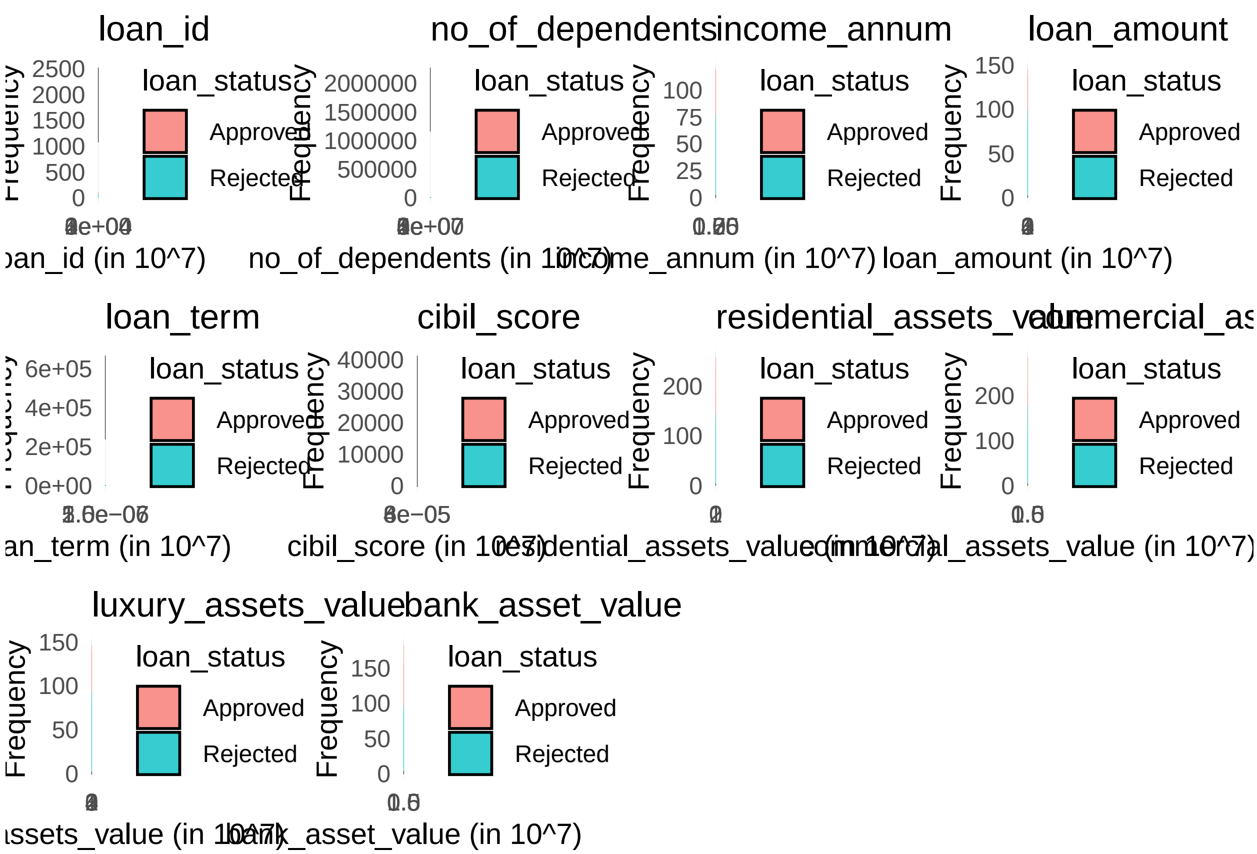
```
# Load necessary libraries
library(GGally)
library(ggplot2)

# Scale the data by dividing by 10^7
loan_data_scaled$income_annum_scaled <- loan_data_scaled$income_annum / 1e7
loan_data_scaled$loan_amount_scaled <- loan_data_scaled$loan_amount / 1e7
loan_data_scaled$cibil_score_scaled <- loan_data_scaled$cibil_score / 1e7

# Create the pair plot
ggpairs(
  loan_data_scaled,
  columns = c('income_annum_scaled', 'loan_amount_scaled', 'cibil_score_scaled'),
  aes(color = loan_status),
  columnLabels = c('Income (in 10^7)', 'Loan Amount (in 10^7)', 'CIBIL Score (in 10^7)')
) +
  theme_minimal()
```



-No relation between cibil score and income annum and loan amount

```r
# ANALYZING THE FEATURE HAVING THE HIGH CHANCE OF LOAN APPROVAL
# Load necessary libraries
library(ggplot2)
library(gridExtra)

# Select numeric columns
numeric_columns <- sapply(loan_data, is.numeric)
numeric_data <- loan_data[, numeric_columns]

# Scale the numeric data by dividing by 10^7
scaled_data <- as.data.frame(lapply(numeric_data, function(x) x / 1e7))

# Add the loan_status column back to the scaled data
scaled_data$loan_status <- loan_data$loan_status

# Create a list to store the plots
plots <- list()

# Create histograms for each numeric column
for (column in names(scaled_data)[-ncol(scaled_data)]) {
  p <- ggplot(scaled_data, aes_string(x = column, fill = "loan_status")) +
    geom_histogram(bins = 30, alpha = 0.7, position = "identity") +
    geom_density(alpha = 0.3) +
    labs(title = column, x = paste(column, "(in 10^7)"), y = "Frequency") +
    theme_minimal()
```

```
  plots[[column]] <- p
}

# Arrange the plots in a grid
grid.arrange(grobs = plots, ncol = 4)
```



- As the cibil_score increases the Approval of loan status has been seen
- INDICATING applicants having a good credit history and loan replayment tends to have higher chances of loan approval

3 MODELLING

```
# Feature Selection

# Load necessary libraries
library(dplyr)

# Drop unnecessary columns
# Drop 'loan_id' column - Not needed for analysis
loan_data <- loan_data %>% select(-loan_id)
```

#Checking Class Imbalance

```
# Class distribution
# ggplot(loan_data, aes(x=factor(loan_status), fill=factor(loan_status))) +
#   geom_bar(alpha=0.7) +
```

```r
#   theme_minimal() +
#   ggtitle("Loan Approval Distribution") +
#   scale_fill_manual(values = c("red", "green"), labels = c("Rejected", "Approved")) +
#   labs(x = "Loan Status", fill = "Status")
```

#KNN can be affected by imbalance because it tends to favor the majority class. #This means the model might predict 0 (Rejected) too often, leading to #poor recall for 1 (Approved).

```r
# library(themis)
# library(recipes)
#
# # Apply SMOTE using recipes (tidymodels framework)
# loan_data$education <- as.integer(factor(loan_data$education))
# loan_data$self_employed <- as.integer(factor(loan_data$self_employed))
#
# # Now apply SMOTE
# rec <- recipe(loan_status ~ ., data = loan_data) %>%
#   step_smote(loan_status, over_ratio = 1) %>%
#   prep() %>%
#   bake(new_data = NULL)
#
# table(rec$loan_status)
#
# loan_data <- rec
```

#Verify Balance

```r
# ggplot(loan_data, aes(x = factor(loan_status), fill = factor(loan_status))) +
#   geom_bar(alpha = 0.7) +
#   theme_minimal() +
#   ggtitle("Loan Approval Distribution After SMOTE") +
#   scale_fill_manual(values = c("red", "green"), labels = c("Rejected", "Approved")) +
#   labs(x = "Loan Status", fill = "Status")
```

```r
#Convert Categorical Variables - Since KNN works best with numerical data, convert categorical variable

loan_data$education <- as.factor(loan_data$education)

loan_data$self_employed <- as.factor(loan_data$self_employed)
```

```r
#Normalize Numerical Features

#Formula
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}

# Normalize relevant numerical columns
loan_data[, c("no_of_dependents", "income_annum", "loan_amount", "loan_term",
              "cibil_score", "residential_assets_value", "commercial_assets_value", "luxury_assets_valu
  apply(loan_data[, c("no_of_dependents", "income_annum", "loan_amount", "loan_term",
                      "cibil_score", "residential_assets_value", "commercial_assets_value", "luxury_ass
        2, normalize)

#Show first rows
```

```r
head(loan_data)
```

```
##   no_of_dependents     education self_employed income_annum loan_amount
## 1              0.4      Graduate            No    0.9690722   0.7551020
## 2              0.0  Not Graduate           Yes    0.4020619   0.3035714
## 3              0.6      Graduate            No    0.9175258   0.7500000
## 4              0.6      Graduate            No    0.8247423   0.7755102
## 5              1.0  Not Graduate           Yes    0.9896907   0.6096939
## 6              0.0      Graduate           Yes    0.4742268   0.3367347
##   loan_term cibil_score residential_assets_value commercial_assets_value
## 1 0.5555556  0.79666667                0.0998004               1.0000000
## 2 0.3333333  0.19500000                0.1117764               0.1290323
## 3 1.0000000  0.34333333                0.2874251               0.2639296
## 4 0.3333333  0.27833333                0.7305389               0.1935484
## 5 1.0000000  0.13666667                0.4990020               0.4809384
## 6 0.4444444  0.03166667                0.2754491               0.4868035
##   luxury_assets_value bank_asset_value loan_status
## 1           0.5758355        0.5594406    Approved
## 2           0.2185090        0.2307692    Rejected
## 3           0.8483290        0.8951049    Rejected
## 4           0.5912596        0.5524476    Rejected
## 5           0.7480720        0.3496503    Rejected
## 6           0.3444730        0.3566434    Rejected
```

#Training the KNN Model #Split Data into Training & Testing Sets #Using Hold Out Estimation

```r
# Load required library
library(caTools)

set.seed(64)

# Split the data (80% train, 20% test)
split <- sample.split(loan_data$loan_status, SplitRatio = 0.8)
train_data <- subset(loan_data, split == TRUE)
test_data <- subset(loan_data, split == FALSE)

# Check split result
table(train_data$loan_status)
```

```
##
##  Approved  Rejected
##      2125      1290
```

```r
table(test_data$loan_status)
```

```
##
##  Approved  Rejected
##       531       323
```

```r
#Prepare Data for KNN
#Remove categorical variables and separate labels (target variable):

# Load KNN library
library(class)

# Remove categorical columns for KNN (excluding target variable)
```

```r
train_features <- train_data[, sapply(train_data, is.numeric)]
test_features <- test_data[, sapply(test_data, is.numeric)]

# Extract target labels
train_labels <- train_data$loan_status
test_labels <- test_data$loan_status

table(train_labels)
```

```
## train_labels
##  Approved  Rejected
##      2125      1290
```

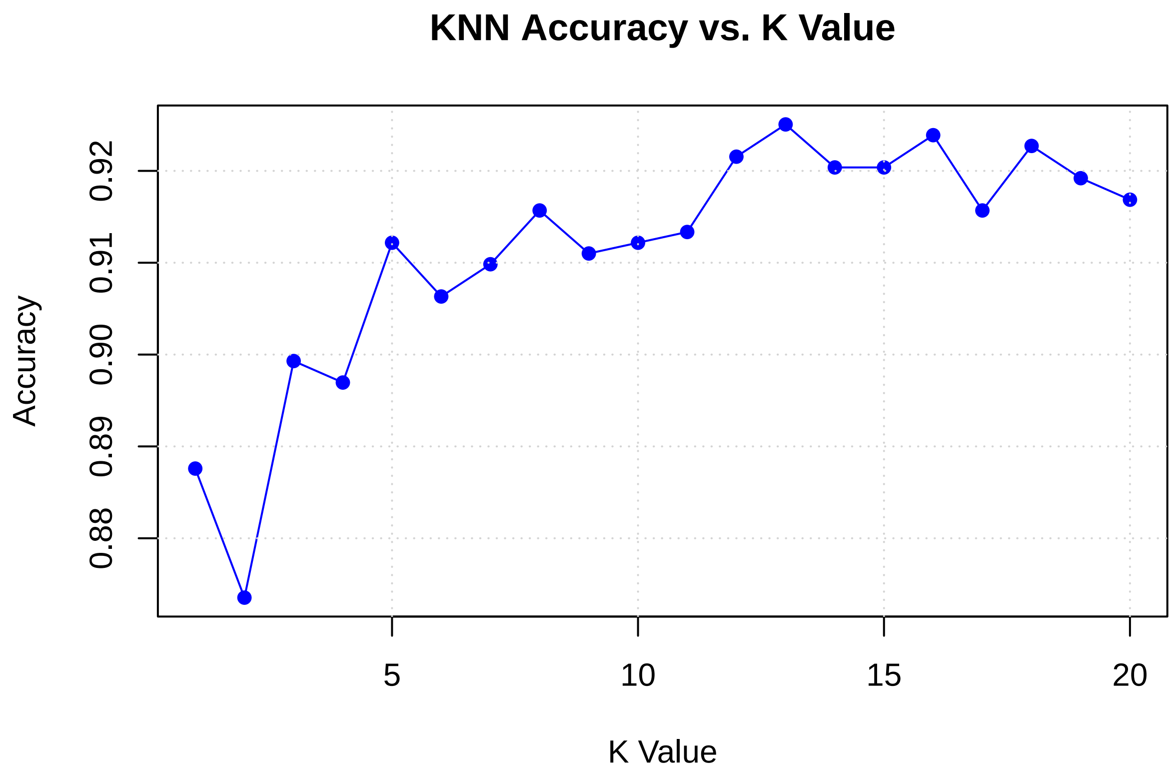
```r
table(test_labels)
```

```
## test_labels
##  Approved  Rejected
##       531       323
```

```r
#Check best K value

# Load necessary libraries
library(class)    # For KNN
library(caret)    # For confusion matrix

# Define a sequence of k values to test
k_values <- 1:20

# Initialize vector to store accuracy values
accuracy_scores <- numeric(length(k_values))

# Loop through k values
for (i in k_values) {
  # Train KNN model
  knn_pred <- knn(train = train_features, test = test_features, cl = train_labels, k = i)

  # Compute accuracy
  conf_matrix <- confusionMatrix(factor(knn_pred, levels = unique(train_labels)),
                                 factor(test_labels, levels = unique(train_labels)))
  accuracy_scores[i] <- conf_matrix$overall["Accuracy"]
}

# Find the best k value
best_k <- k_values[which.max(accuracy_scores)]
cat("Best k:", best_k, "with Accuracy:", max(accuracy_scores), "\n")
```

```
## Best k: 13 with Accuracy: 0.9250585
```

```r
# Plot Accuracy vs. k
plot(k_values, accuracy_scores, type = "o", col = "blue", pch = 16, xlab = "K Value", ylab = "Accuracy"
     main = "KNN Accuracy vs. K Value")
grid()
```



#Train KNN Model

```
# Load necessary libraries
library(class)  # For KNN
library(caret)  # For confusionMatrix
library(e1071)  # Required for confusionMatrix

# Set seed for reproducibility
set.seed(65)

# Define a function to compute Euclidean distance
euclidean_distance <- function(x1, x2) {
  sqrt(sum((x1 - x2)^2))
}

# Standardize the features to ensure fair distance computation
train_features_scaled <- scale(train_features)
test_features_scaled <- scale(test_features)

# Define K value
k_value <- 13

# Train KNN model with Euclidean distance
#Since knn() in the class package already defaults to Euclidean distance, this ensures that all feature

knn_pred <- knn(train = train_features_scaled,
                test = test_features_scaled,
```

```r
                cl = train_labels,
                k = k_value)

# Convert test labels and predictions to factors with the same levels
test_labels <- factor(test_labels, levels = unique(train_labels))
knn_pred <- factor(knn_pred, levels = unique(train_labels))

# Create confusion matrix
conf_matrix <- confusionMatrix(knn_pred, test_labels)

# Print full confusion matrix
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Approved Rejected
##   Approved       490       37
##   Rejected        41      286
##
##                Accuracy : 0.9087
##                  95% CI : (0.8873, 0.9271)
##     No Information Rate : 0.6218
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8063
##
##  Mcnemar's Test P-Value : 0.7341
##
##             Sensitivity : 0.9228
##             Specificity : 0.8854
##          Pos Pred Value : 0.9298
##          Neg Pred Value : 0.8746
##              Prevalence : 0.6218
##          Detection Rate : 0.5738
##    Detection Prevalence : 0.6171
##       Balanced Accuracy : 0.9041
##
##        'Positive' Class : Approved
##
```

```r
# Extract Accuracy
accuracy <- conf_matrix$overall["Accuracy"]
print(paste("Accuracy:", round(accuracy, 4)))
```

```
## [1] "Accuracy: 0.9087"
```

```r
# Extract Precision, Recall, and F1-score
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
f1_score <- conf_matrix$byClass["F1"]

print(paste("Precision:", round(precision, 4)))
```

```
## [1] "Precision: 0.9298"
```

```r
print(paste("Recall:", round(recall, 4)))
```

```
## [1] "Recall: 0.9228"
```

```r
print(paste("F1 Score:", round(f1_score, 4)))
```

```
## [1] "F1 Score: 0.9263"
```

4 Detailed Model Performance Insights:

-Accuracy: 0.9087 - The model correctly predicts loan approval status for 90.87% of cases.

-Precision: 0.9298 - When the model predicts loan approval, it's correct 92.98% of the time.

-Recall: 0.9228 - The model correctly identifies 92.28% of all actual approved loans.

-F1-score: 0.9263 - Indicates a good balance between precision and recall.

-Kappa: 0.8063 - Indicates a very good agreement.

5 CONCLUSION

The K-Nearest Neighbors (KNN) model developed for loan approval prediction demonstrates strong performance and reliability:

1. High Accuracy: With an accuracy of 90.87%, the model shows excellent overall predictive capability.
2. Balanced Performance: High precision (92.98%) and recall (92.28%) indicate the model's effectiveness in both approving worthy candidates and identifying potential defaults.
3. Robust Discrimination: A Balanced Accuracy of 0.9041 suggests the model's strong ability to distinguish between approved and rejected loan applications.
4. Key Factors: The analysis highlighted CIBIL score, income, and loan amount as crucial factors in loan approval decisions.
5. Practical Applicability: The model's performance suggests it could be a valuable tool in assisting loan approval decisions in real-world scenarios.