Risk Analytics

Joshua

2023-05-03

Credit Risk Classification Dataset

Context This is Customer Transaction and Demographic related data , It holds Risky and Not Risky customer for specific banking products

Acknowledgements Thanks to Google Datasets search

Inspiration Your data will be in front of the world's largest data science community. What questions do you want to see answered?

This dataset help to find out weather customer is Credit Risky or Credit Worthy in Banking perspective

Q1 - What are the factors contributing to Credit Risky customer? Q2 - Behavior of Credit Worthy Customer?

Data location https://www.kaggle.com/datasets/praveengovi/credit-risk-classification-dataset?select=payment_data.csv (https://www.kaggle.com/datasets/praveengovi/credit-risk-classification-dataset?select=payment_data.csv)

Loading the tidyverse package

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.2.3
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```
## Warning: package 'tibble' was built under R version 4.2.3
```

```
## Warning: package 'dplyr' was built under R version 4.2.3
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.1     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.2     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
## ℹ Use the ]8;;http://conflicted.r-lib.org/conflicted package]8;; to force all conflicts to become errors
```

```
library(data.table)
```

```
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
library(ggcorrplot)
```

```
## Warning: package 'ggcorrplot' was built under R version 4.2.3
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.2.3
```

```
## Loading required package: lattice
```

```
## Warning: package 'lattice' was built under R version 4.2.3
```

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

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.2.3
```

```
## 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(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 4.2.3
```

```
##
## Attaching package: 'reshape2'
##
## The following objects are masked from 'package:data.table':
##
##     dcast, melt
##
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```r
library(ROSE)
```

```
## Warning: package 'ROSE' was built under R version 4.2.3
```

```
## Loaded ROSE 0.0-4
```

```r
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 4.2.3
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:randomForest':
##
##     combine
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(DALEX)
```

```
## Warning: package 'DALEX' was built under R version 4.2.3
```

```
## Welcome to DALEX (version: 2.4.3).
## Find examples and detailed introduction at: http://ema.drwhy.ai/
## Additional features will be available after installation of: ggpubr.
## Use 'install_dependencies()' to get all suggested dependencies
##
## Attaching package: 'DALEX'
##
## The following object is masked from 'package:dplyr':
##
##     explain
```

```r
library(randomForestExplainer)
```

```
## Warning: package 'randomForestExplainer' was built under R version 4.2.3
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

Loading the data

```
customer_data <- read_csv("~/Datasets/Personal Project - Risk Analyst/customer_data.csv")
```

```
## Rows: 1125 Columns: 13
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (13): label, id, fea_1, fea_2, fea_3, fea_4, fea_5, fea_6, fea_7, fea_8,...
##
## 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.
```

```
payment_data <- read_csv("~/Datasets/Personal Project - Risk Analyst/payment_data.csv")
```

```
## Rows: 8250 Columns: 12
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (2): update_date, report_date
## dbl (10): id, OVD_t1, OVD_t2, OVD_t3, OVD_sum, pay_normal, prod_code, prod_l...
##
## 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.
```

Preliminary look at customer_data

```
head(customer_data)
```

```
## # A tibble: 6 × 13
##   label       id fea_1 fea_2 fea_3  fea_4 fea_5 fea_6 fea_7 fea_8 fea_9 fea_10
##   <dbl>    <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1     1 54982665     5 1246.     3  77000     2    15     5   109     5 151300
## 2     0 59004779     4 1277      1 113000     2     8    -1   100     3 341759
## 3     0 58990862     7 1298      1 110000     2    11    -1   101     5  72001
## 4     1 58995168     7 1336.     1 151000     2    11     5   110     3  60084
## 5     0 54987320     7   NA      2  59000     2    11     5   108     4 450081
## 6     0 59005995     6 1217      3  56000     2     6    -1   100     3  60091
## # ℹ 1 more variable: fea_11 <dbl>
```

```
summary(customer_data)
```

```
##      label           id               fea_1          fea_2     
##  Min.   :0.0   Min.   :54982353   Min.   :1.000   Min.   :1116  
##  1st Qu.:0.0   1st Qu.:54990497   1st Qu.:4.000   1st Qu.:1244  
##  Median :0.0   Median :58989748   Median :5.000   Median :1282  
##  Mean   :0.2   Mean   :57836771   Mean   :5.483   Mean   :1284  
##  3rd Qu.:0.0   3rd Qu.:58997994   3rd Qu.:7.000   3rd Qu.:1314  
##  Max.   :1.0   Max.   :59006239   Max.   :7.000   Max.   :1481  
##                                                   NA's   :149   
##      fea_3          fea_4             fea_5          fea_6      
##  Min.   :1.000   Min.   :  15000   Min.   :1.000   Min.   : 3.00  
##  1st Qu.:1.000   1st Qu.:  72000   1st Qu.:2.000   1st Qu.: 8.00  
##  Median :3.000   Median : 102000   Median :2.000   Median :11.00  
##  Mean   :2.333   Mean   : 120884   Mean   :1.929   Mean   :10.87  
##  3rd Qu.:3.000   3rd Qu.: 139000   3rd Qu.:2.000   3rd Qu.:11.00  
##  Max.   :3.000   Max.   :1200000   Max.   :2.000   Max.   :16.00  
##                                                                   
##      fea_7           fea_8          fea_9           fea_10      
##  Min.   :-1.000   Min.   : 64.0   Min.   :1.000   Min.   : 60000  
##  1st Qu.: 5.000   1st Qu.: 90.0   1st Qu.:3.000   1st Qu.: 60044  
##  Median : 5.000   Median :105.0   Median :4.000   Median : 72000  
##  Mean   : 4.833   Mean   :100.8   Mean   :4.196   Mean   :164619  
##  3rd Qu.: 5.000   3rd Qu.:111.0   3rd Qu.:5.000   3rd Qu.:151307  
##  Max.   :10.000   Max.   :115.0   Max.   :5.000   Max.   :650070  
##                                                                   
##      fea_11     
##  Min.   :  1.0  
##  1st Qu.:  1.0  
##  Median :173.2  
##  Mean   :135.0  
##  3rd Qu.:202.5  
##  Max.   :707.1  
## 
```

Preliminary look at payment_data

```
head(payment_data)
```

```
## # A tibble: 6 × 12
##         id OVD_t1 OVD_t2 OVD_t3 OVD_sum pay_normal prod_code prod_limit
##      <dbl>  <dbl>  <dbl>  <dbl>   <dbl>      <dbl>     <dbl>      <dbl>
## 1 58987402      0      0      0       0          1        10      16500
## 2 58995151      0      0      0       0          1         5         NA
## 3 58997200      0      0      0       0          2         5         NA
## 4 54988608      0      0      0       0          3        10      37400
## 5 54987763      0      0      0       0          2        10         NA
## 6 59004828      0      0      0       0          3        10      88000
## # ℹ 4 more variables: update_date <chr>, new_balance <dbl>,
## #   highest_balance <dbl>, report_date <chr>
```

```
summary(payment_data)
```

```
##        id               OVD_t1            OVD_t2            OVD_t3       
##  Min.   :54982353   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000  
##  1st Qu.:54990497   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000  
##  Median :58989048   Median : 0.0000   Median : 0.0000   Median : 0.0000  
##  Mean   :57821730   Mean   : 0.2491   Mean   : 0.1272   Mean   : 0.3692  
##  3rd Qu.:58996551   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000  
##  Max.   :59006239   Max.   :34.0000   Max.   :34.0000   Max.   :35.0000  
##                                                                          
##     OVD_sum          pay_normal      prod_code        prod_limit      
##  Min.   :    0.0   Min.   : 0.00   Min.   : 0.000   Min.   :     1.1  
##  1st Qu.:    0.0   1st Qu.: 4.00   1st Qu.: 6.000   1st Qu.: 37400.0  
##  Median :    0.0   Median :11.00   Median :10.000   Median : 68200.0  
##  Mean   :  187.7   Mean   :14.53   Mean   : 8.232   Mean   : 85789.7  
##  3rd Qu.:    0.0   3rd Qu.:25.00   3rd Qu.:10.000   3rd Qu.:112200.0  
##  Max.   :31500.0   Max.   :36.00   Max.   :27.000   Max.   :660000.0  
##                                                     NA's   :6118      
##  update_date         new_balance        highest_balance     report_date       
##  Length:8250        Min.   :   -40303   Min.   :      501   Length:8250       
##  Class :character   1st Qu.:        0   1st Qu.:    23453   Class :character  
##  Mode  :character   Median :        0   Median :    44047   Mode  :character  
##                     Mean   :   105404   Mean   :   219203                     
##                     3rd Qu.:    24948   3rd Qu.:   100500                     
##                     Max.   :163211958   Max.   :180000500                     
##                                         NA's   :409                           
```

I see null values and also the date columns are the wrong data type.

Cleaning the data

Changing date types, but trying with a copy of the table.

```
payment_data_copy <- copy(payment_data)
today <- Sys.Date()
payment_data_copy$update_date[is.na(payment_data_copy$update_date)] <- today
payment_data_copy$report_date[is.na(payment_data_copy$report_date)] <- today
head(payment_data_copy)
```

```
## # A tibble: 6 × 12
##         id OVD_t1 OVD_t2 OVD_t3 OVD_sum pay_normal prod_code prod_limit
##      <dbl>  <dbl>  <dbl>  <dbl>   <dbl>      <dbl>     <dbl>      <dbl>
## 1 58987402      0      0      0       0          1        10      16500
## 2 58995151      0      0      0       0          1         5         NA
## 3 58997200      0      0      0       0          2         5         NA
## 4 54988608      0      0      0       0          3        10      37400
## 5 54987763      0      0      0       0          2        10         NA
## 6 59004828      0      0      0       0          3        10      88000
## # ℹ 4 more variables: update_date <chr>, new_balance <dbl>,
## #   highest_balance <dbl>, report_date <chr>
```

It didn't work. I think I need to import it again so that the data is properly formatted.

```
payment_data <- read.csv("~/Datasets/Personal Project - Risk Analyst/payment_data.csv" ,
                         stringsAsFactors = FALSE,
                         colClasses = c("numeric", "numeric", "numeric", "numeric", "numeric", "numeric",
                                        "character", "numeric", "character", "numeric", "numeric", "character"),
                         col.names = c("id", "OVD_t1", "OVD_t2", "OVD_t3", "OVD_sum", "pay_normal",
                                       "prod_code", "prod_limit", "update_date", "new_balance", "highest_balance", "rep
ort_date"))
payment_data$update_date <- as.Date(payment_data$update_date, "%d/%m/%Y")
payment_data$report_date <- as.Date(payment_data$report_date, "%d/%m/%Y")
```

Renaming the label column to “credit_risk” for clarity, and credit_risk values 0 and 1 to low and high respectively.

```
customer_data <- customer_data %>%
  rename(credit_risk = label) %>%
  mutate(credit_risk = ifelse(credit_risk == 1, "high", "low"))
```

Checking for duplicates

```
customer_data %>%
  duplicated() %>%
  any()
```

```
## [1] FALSE
```

```
payment_data %>%
  duplicated() %>%
  any()
```

```
## [1] TRUE
```

Removing duplicate rows from payment_data

```
payment_data <- payment_data %>% distinct()
```

Checking for null values in payment_data

```
payment_data %>%
  summarise_all(~ sum(is.na(.)))
```

```
##   id OVD_t1 OVD_t2 OVD_t3 OVD_sum pay_normal prod_code prod_limit update_date
## 1  0      0      0      0       0          0         0       6040          21
##   new_balance highest_balance report_date
## 1           0             396        1091
```

```
customer_data %>%
  summarise_all(~ sum(is.na(.)))
```

```
## # A tibble: 1 × 13
##   credit_risk    id fea_1 fea_2 fea_3 fea_4 fea_5 fea_6 fea_7 fea_8 fea_9 fea_10
##         <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>  <int>
## 1           0     0     0   149     0     0     0     0     0     0     0      0
## # ℹ 1 more variable: fea_11 <int>
```

I want to show each column and the impact of the null values on those columns.

```r
data_profile <- function(df) {
  stats <- data.frame()
  for (col in names(df)) {
    n_missing <- sum(is.na(df[[col]]))
    if(n_missing == 0){
      missing_percent <- NA
    } else {
      missing_percent <- n_missing * 100 / nrow(df)
    }
    stats_row <- data.frame(Feature = col,
                            Unique_values = n_distinct(df[[col]]),
                            `Percentage of missing values` = missing_percent,
                            `Percentage of values in the biggest category` = max(table(df[[col]], useNA = "ifany")) * 1
00 / sum(!is.na(df[[col]])))
    stats <- rbind(stats, stats_row)
  }
print(stats)
}
```

```r
customer_stats <- data_profile(customer_data)
```

```
##        Feature Unique_values Percentage.of.missing.values
## 1  credit_risk             2                           NA
## 2           id          1125                           NA
## 3        fea_1             6                           NA
## 4        fea_2           159                     13.24444
## 5        fea_3             3                           NA
## 6        fea_4           229                           NA
## 7        fea_5             2                           NA
## 8        fea_6            10                           NA
## 9        fea_7            10                           NA
## 10       fea_8            52                           NA
## 11       fea_9             5                           NA
## 12      fea_10           280                           NA
## 13      fea_11           266                           NA
##    Percentage.of.values.in.the.biggest.category
## 1                                   80.00000000
## 2                                    0.08888889
## 3                                   42.31111111
## 4                                   15.26639344
## 5                                   60.80000000
## 6                                    3.02222222
## 7                                   92.88888889
## 8                                   41.33333333
## 9                                   61.24444444
## 10                                   8.71111111
## 11                                  46.31111111
## 12                                  11.37777778
## 13                                  36.17777778
```

```r
payment_stats <- data_profile(payment_data)
```

```
##           Feature Unique_values Percentage.of.missing.values
## 1              id          1125                           NA
## 2          OVD_t1            21                           NA
## 3          OVD_t2            16                           NA
## 4          OVD_t3            33                           NA
## 5         OVD_sum           393                           NA
## 6      pay_normal            37                           NA
## 7       prod_code            21                           NA
## 8      prod_limit           322                   74.0286800
## 9     update_date          3042                    0.2573845
## 10    new_balance          3939                           NA
## 11 highest_balance         5141                    4.8535360
## 12    report_date          1863                   13.3717367
##    Percentage.of.values.in.the.biggest.category
## 1                                     0.6741022
## 2                                    90.6483638
## 3                                    95.6122074
## 4                                    96.7520529
## 5                                    88.8956980
## 6                                    10.8836867
## 7                                    54.9577154
## 8                                   285.0401133
## 9                                     0.2703367
## 10                                   46.6111043
## 11                                    5.1011207
## 12                                   15.4357668
```

Since customer data has so few nulls, I will just replace all the null values it the column average on fea_2.

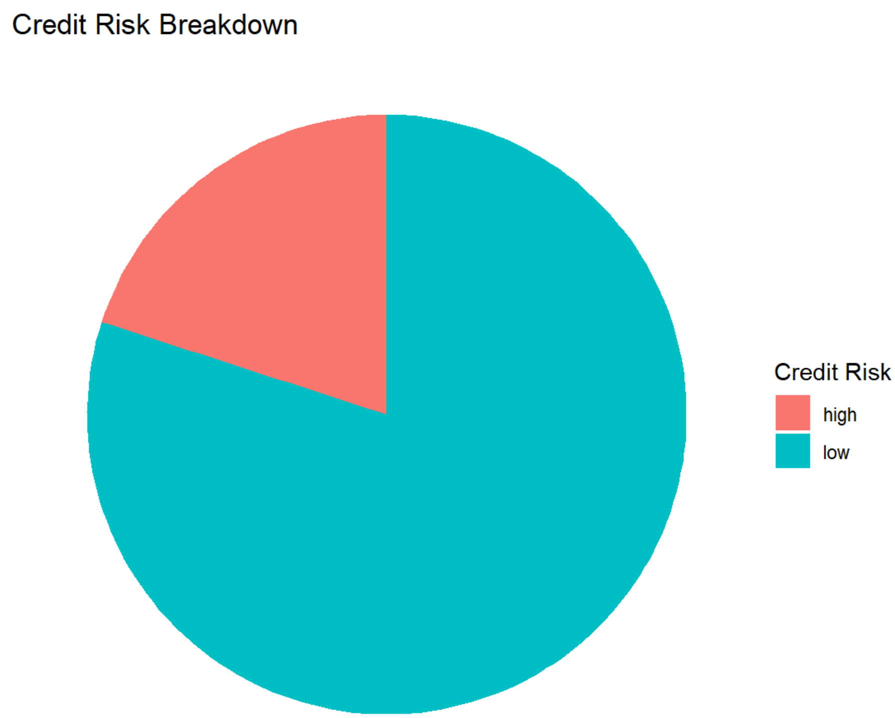
```
customer_data$fea_2[is.na(customer_data$fea_2)] <- mean(customer_data$fea_2, na.rm = TRUE)
```

Explore the Data

```
customer_data %>%
  count(credit_risk)
```

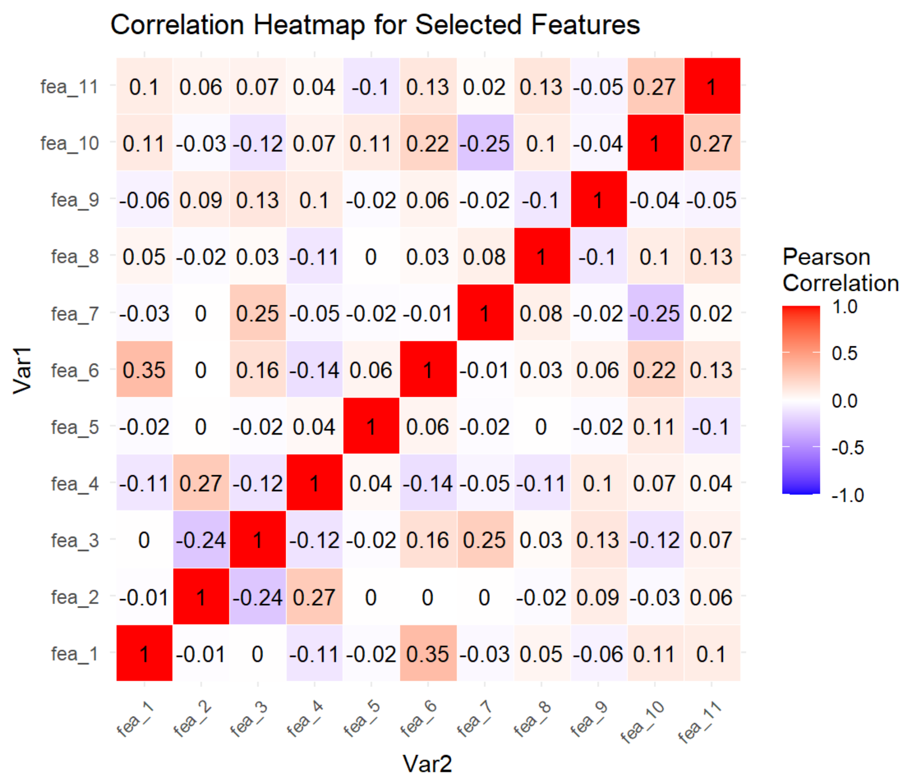
```
## # A tibble: 2 × 2
##   credit_risk     n
##   <chr>       <int>
## 1 high          225
## 2 low           900
```

```
customer_data %>%
  count(credit_risk) %>%
  ggplot(aes(x = "", y = n, fill = credit_risk)) +
  geom_bar(width = 1, stat = "identity") +
  coord_polar(theta = "y") +
  labs(title = "Credit Risk Breakdown", fill = "Credit Risk") +
  theme_void()
```



Correlation matrix heatmap

```
cust_data_subset <- customer_data[, c("fea_1", "fea_2", "fea_3", "fea_4", "fea_5", "fea_6", "fea_7", "fea_8", "fea_9",
"fea_10", "fea_11")]
cust_data_subset_corr <- cor(cust_data_subset)

cust_data_subset_melted <- melt(cust_data_subset_corr)

ggplot(cust_data_subset_melted, aes(Var2, Var1)) +
  geom_tile(aes(fill = value), colour = "white") +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0, limit = c(-1,1), space = "Lab", name="P
earson\nCorrelation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, size = 8, hjust = 1)) +
  coord_fixed() +
  geom_text(aes(Var2, Var1, label = round(value,2)), color = "black", size = 3.5) +
  labs(title = "Correlation Heatmap for Selected Features")
```



The average balance for customers with high and low credit risk

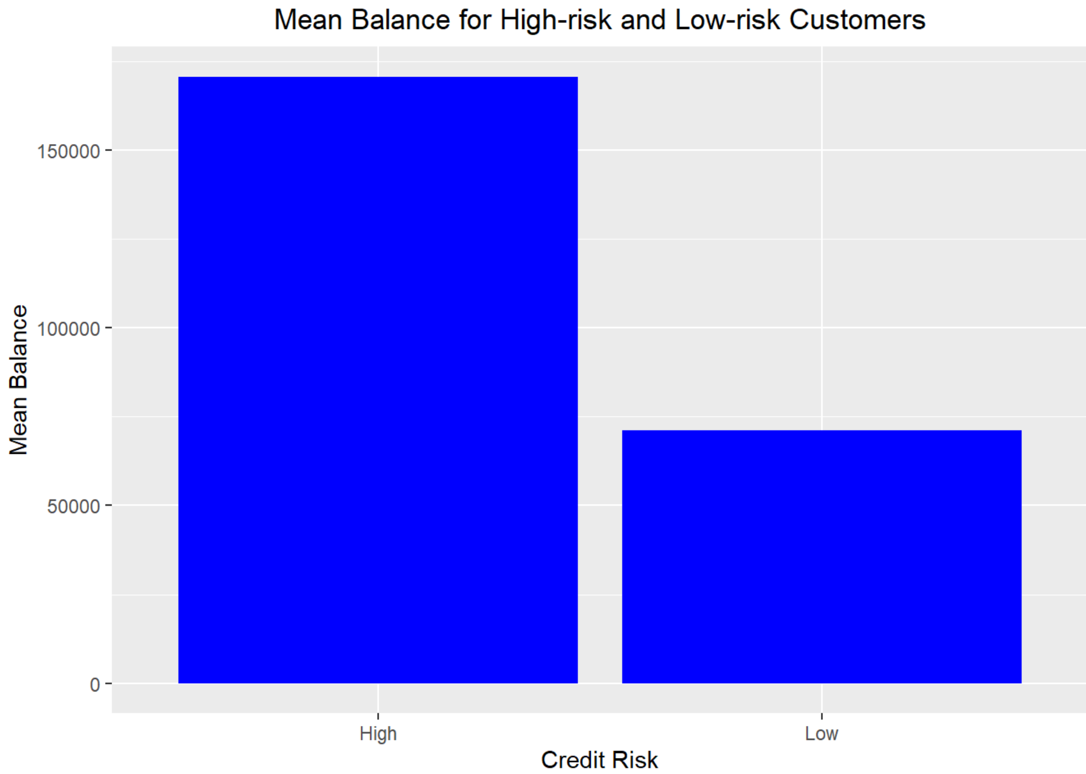
```
cust_bal_avg <- aggregate(new_balance ~ id, payment_data, mean)
cust_pymt_df <- merge(customer_data, cust_bal_avg, by = "id")
high_risk_mean_balance <- mean(cust_pymt_df$new_balance[cust_pymt_df$credit_risk == "high"])
low_risk_mean_balance <- mean(cust_pymt_df$new_balance[cust_pymt_df$credit_risk == "low"])
cat("Mean balance for high risk customers: ", high_risk_mean_balance, "\n")
```

```
## Mean balance for high risk customers:  170785.7
```

```
cat("Mean balance for low risk customers: ", low_risk_mean_balance, "\n")
```

```
## Mean balance for low risk customers:  71209.11
```

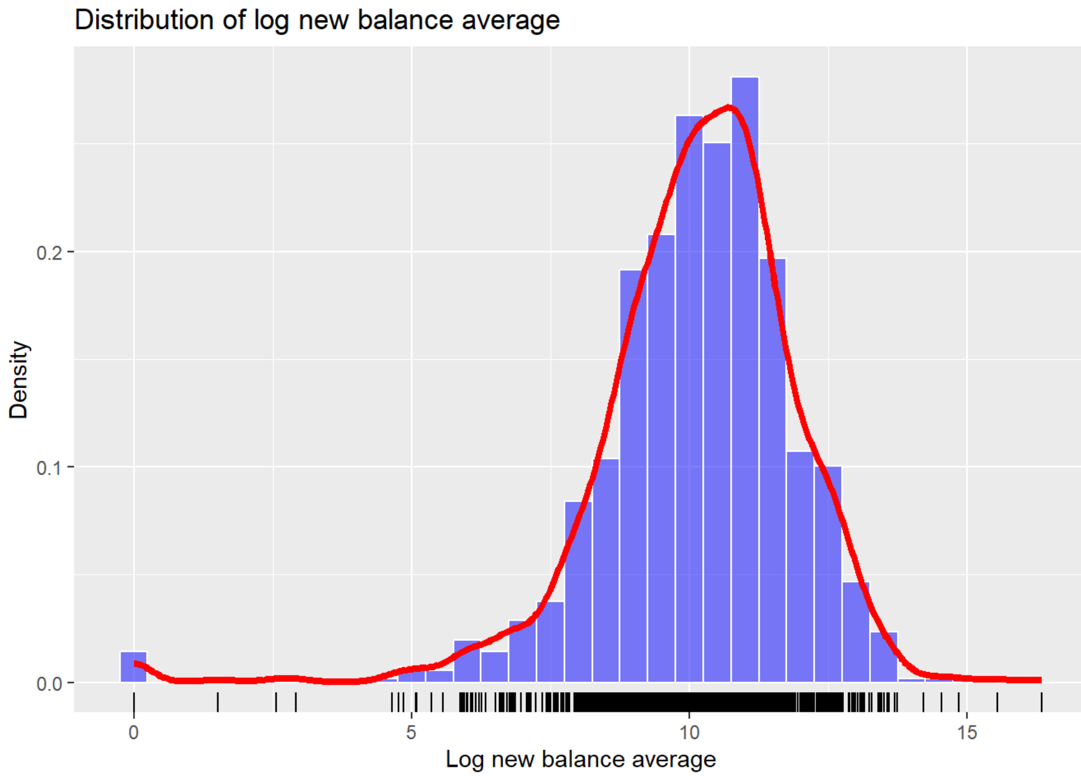
```
mean_balances <- data.frame(
  Risk = c("High", "Low"),
  Balance = c(high_risk_mean_balance, low_risk_mean_balance)
)
ggplot(mean_balances, aes(x = Risk, y = Balance)) +
  geom_bar(stat = "identity", fill = "blue") +
  ggtitle("Mean Balance for High-risk and Low-risk Customers") +
  xlab("Credit Risk") +
  ylab("Mean Balance") +
  theme(plot.title = element_text(hjust = 0.5))
```



The distribution of the new balances in the dataset

```
suppressWarnings({
  df <- payment_data %>% left_join(customer_data, by = "id")
df_new_bal_avg <- df %>%
    group_by(id) %>%
    summarise(new_bal_avg = mean(new_balance, na.rm = TRUE))
df_new_bal_avg <- na.omit(df_new_bal_avg)
min_new_bal_avg <- min(df_new_bal_avg$new_bal_avg)
max_new_bal_avg <- max(df_new_bal_avg$new_bal_avg)
cat("Minimum new balance avg in the dataset:", min_new_bal_avg, "\n")
cat("Maximum new balance avg in the dataset:", max_new_bal_avg, "\n")

ggplot(df_new_bal_avg, aes(x = log(new_bal_avg + 1))) +
  geom_histogram(aes(y = ..density..), binwidth = 0.5, color = "white", fill = "blue", alpha = 0.5) +
  geom_density(color = "red", size = 1.5) +
  geom_rug() +
  scale_x_continuous(limits = c(log(min_new_bal_avg + 1), log(max_new_bal_avg + 1))) +
  labs(title = "Distribution of log new balance average", x = "Log new balance average", y = "Density")
})
```

```
## Minimum new balance avg in the dataset: -1666.4
## Maximum new balance avg in the dataset: 12475780
```

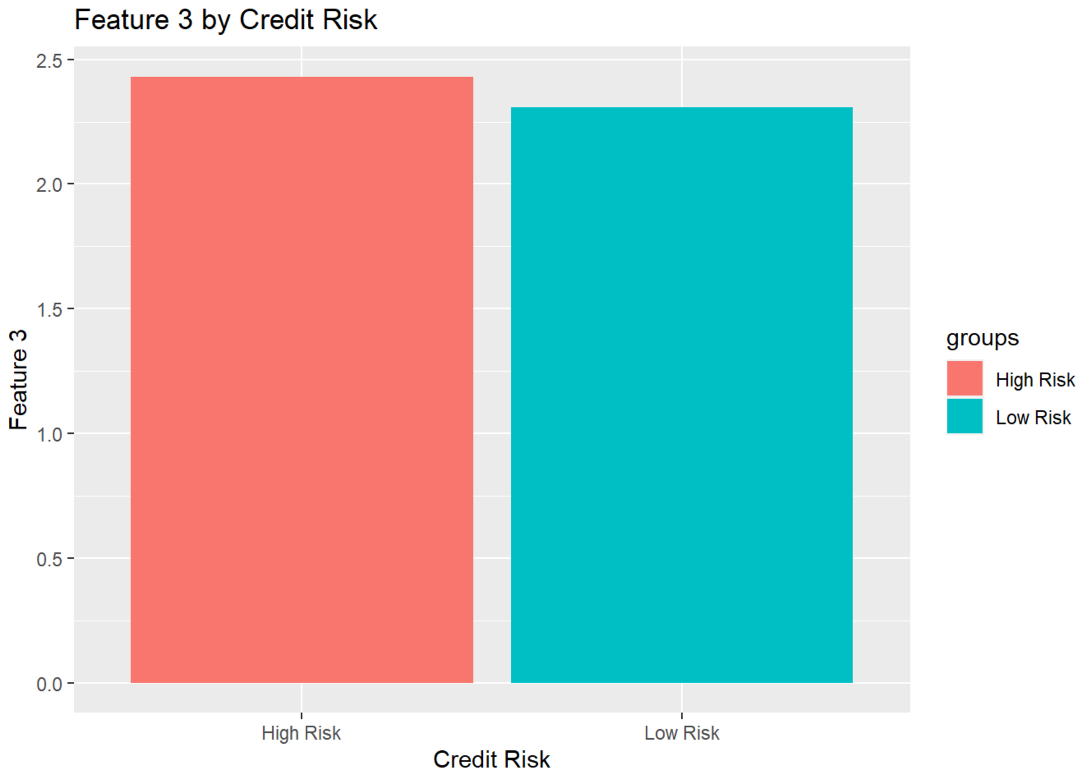


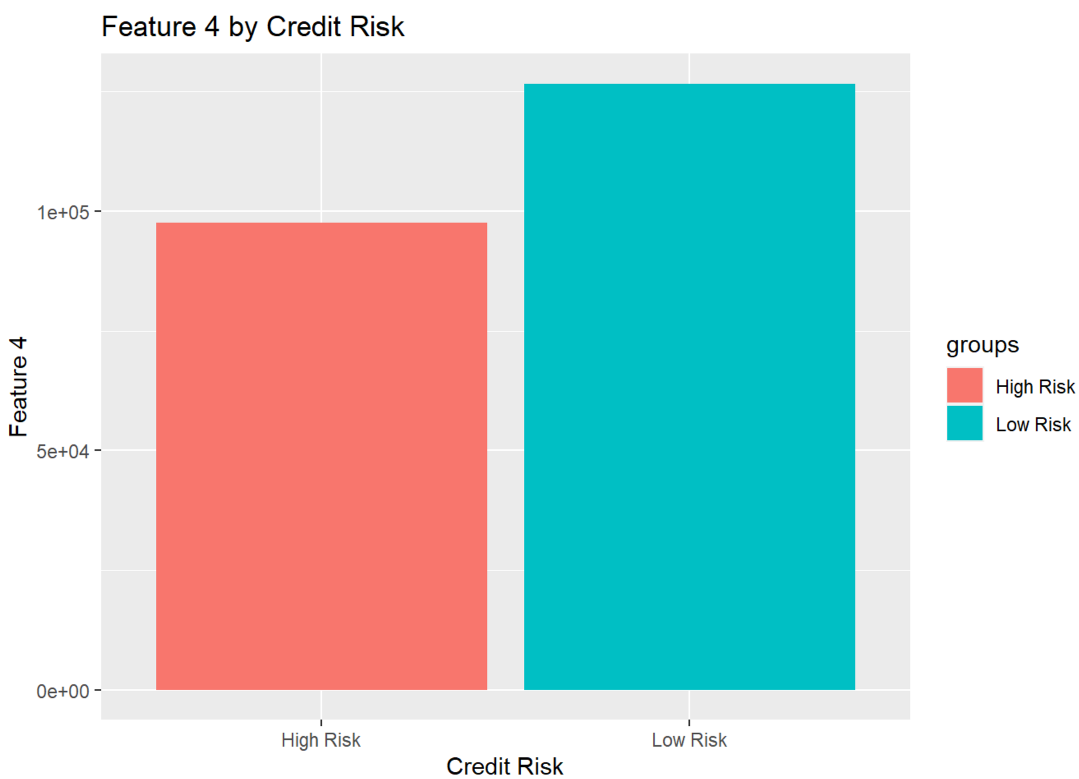
Calculate the mean for each demographic variable to check for correlation to high or low credit risk.

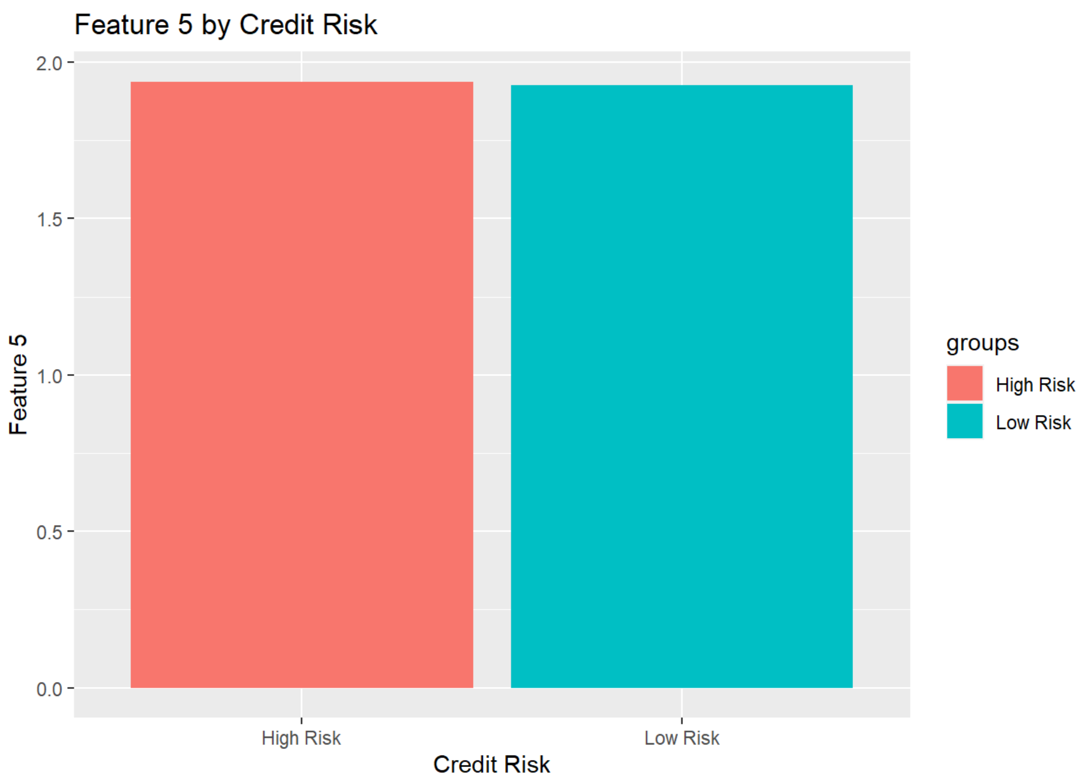
```
high_risk_customers <- customer_data[customer_data$credit_risk == "high", ]
low_risk_customers <- customer_data[customer_data$credit_risk == "low", ]
high_risk_means <- aggregate(high_risk_customers[, c("fea_1", "fea_2", "fea_3", "fea_4", "fea_5", "fea_6", "fea_7", "fe
a_8", "fea_9", "fea_10", "fea_11")],
                             by = list(high_risk_customers$credit_risk), FUN = mean)
low_risk_means <- aggregate(low_risk_customers[, c("fea_1", "fea_2", "fea_3", "fea_4", "fea_5", "fea_6", "fea_7", "fea_
8", "fea_9", "fea_10", "fea_11")],
                            by = list(low_risk_customers$credit_risk), FUN = mean)
generate_charts <- function(high_risk_customers, low_risk_customers, high_risk_means, low_risk_means) {
  for (fea in 1:11) {
    high_means <- high_risk_means[, fea + 1]
    low_means <- low_risk_means[, fea + 1]
    bp_data <- data.frame(means = c(high_means, low_means),
                          groups = rep(c("High Risk", "Low Risk"), each = length(high_means)))
    bp <- ggplot(bp_data, aes(x = groups, y = means, fill = groups)) +
      geom_bar(stat = "identity", position = "dodge") +
      ggtitle(paste("Feature", fea, "by Credit Risk")) +
      xlab("Credit Risk") +
      ylab(paste("Feature", fea))
    plot(bp)
  }
}

# Call the function with the relevant variables
generate_charts(high_risk_customers, low_risk_customers, high_risk_means, low_risk_means)
```

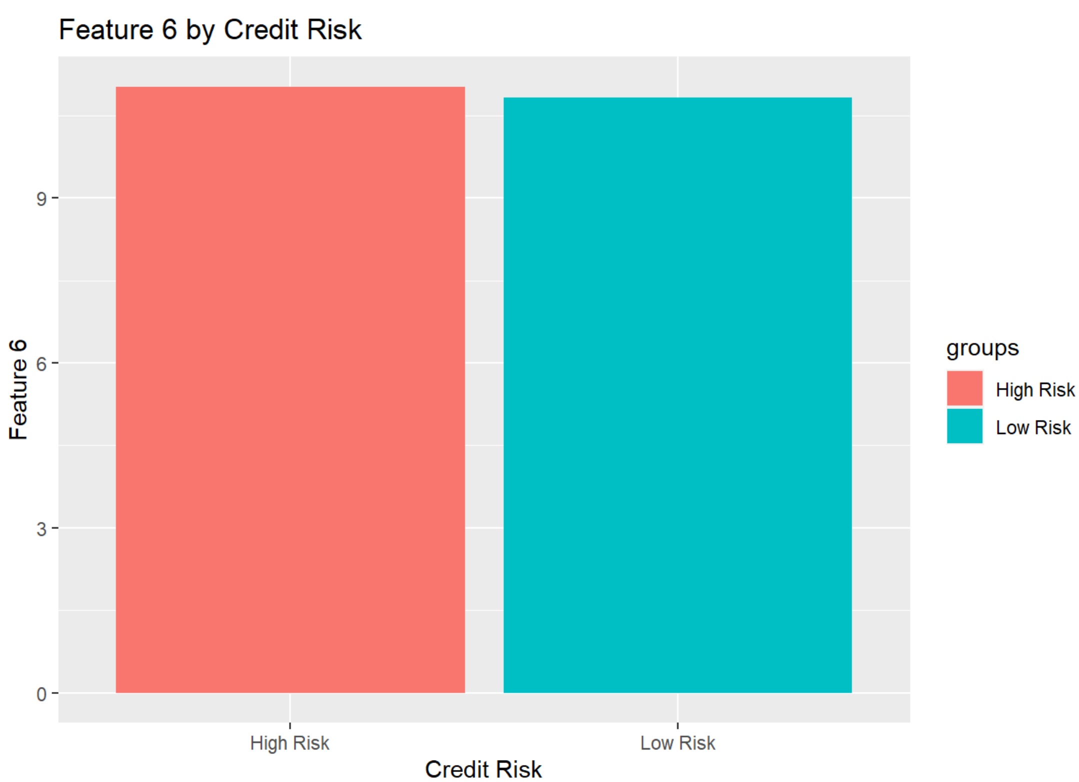

Feature 1 by Credit Risk
Feature 1
4
2
0
High Risk
Low Risk
Credit Risk
groups
High Risk
Low Risk


Feature 2 by Credit Risk
Feature 2
1000
500
0
High Risk
Low Risk
Credit Risk
groups
High Risk
Low Risk


Feature 3 by Credit Risk
2.5
2.0
1.5
1.0
0.5
0.0
Feature 3
High Risk
Low Risk
Credit Risk
groups
High Risk
Low Risk

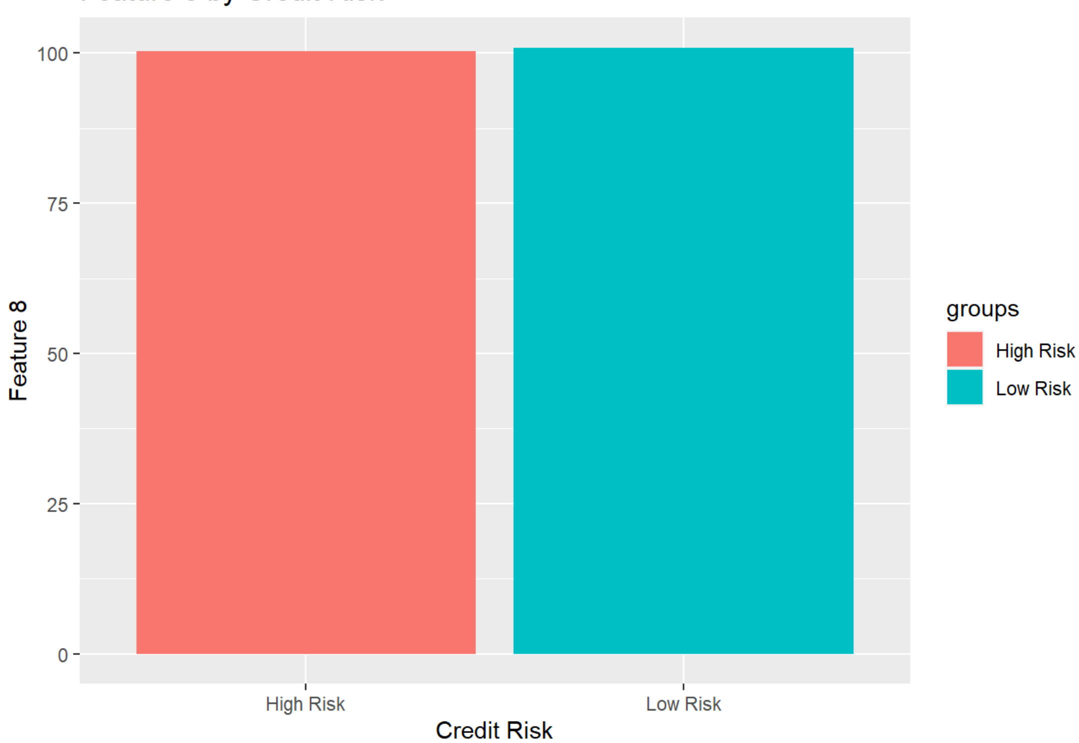

Feature 4 by Credit Risk
1e+05
5e+04
0e+00
Feature 4
High Risk
Low Risk
Credit Risk
groups
High Risk
Low Risk


Feature 5 by Credit Risk
2.0
1.5
1.0
0.5
0.0
Feature 5
High Risk
Low Risk
Credit Risk
groups
High Risk
Low Risk


Feature 6 by Credit Risk
9
6
3
0
Feature 6
High Risk
Low Risk
Credit Risk
groups
High Risk
Low Risk

Feature 7 by Credit Risk

Feature 8 by Credit Risk




Feature 9 by Credit Risk
4
3
2
1
0
Feature 9
High Risk
Low Risk
Credit Risk
groups
High Risk
Low Risk


Feature 10 by Credit Risk
150000
100000
50000
0
Feature 10
High Risk
Low Risk
Credit Risk
groups
High Risk
Low Risk

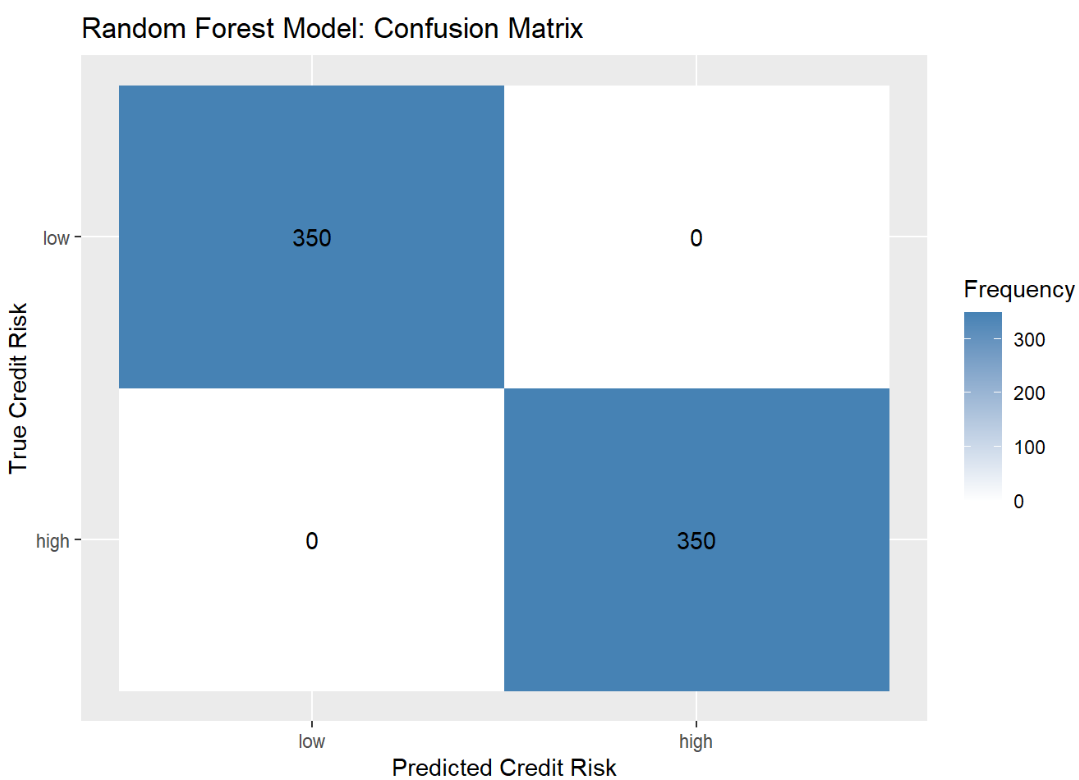
Fea_4 is the only demographic that shows a noticeable difference between the high risk and low risk customers.

Train a random forest model with the hyperperameters

```
# Merge data and remove missing values
merged_data <- merge(customer_data, payment_data[, !(colnames(payment_data) %in% "prod_limit")], by = "id")
merged_data <- na.omit(merged_data)
merged_data$credit_risk <- factor(ifelse(merged_data$credit_risk == "low", "low", "high"), levels = c("low", "high"))
merged_data <- subset(merged_data, select = -prod_code)


# Undersample
merged_data_balanced <- downSample(x = merged_data, y = merged_data$credit_risk)

# Split data into training and testing sets
set.seed(123)
train_index <- createDataPartition(y = merged_data_balanced$credit_risk, p = 0.7, list = FALSE)
train_data_balanced <- merged_data_balanced[train_index,]
train_data_balanced$credit_risk <- as.factor(train_data_balanced$credit_risk)
test_data_balanced <- merged_data_balanced[-train_index,]

# Define hyperparameter tuning grid and control parameters
rf_params <- expand.grid(mtry = seq(2, ncol(train_data_balanced)-1, by = 1))
ntree_vals <- seq(50, 200, by = 50)
mtry_values <- c(2, 4, 6, 8)
tune_grid <- expand.grid(mtry = c(2, 3, 4, 5))
ctrl <- trainControl(method = "cv", number = 5, verboseIter = TRUE)

# Train random forest model with hyperparameter tuning
rf_model <- train(credit_risk ~ ., data = train_data_balanced, method = "rf", ntree = 50, tuneGrid = tune_grid, trContr
ol = ctrl)
```

```
## + Fold1: mtry=2
## - Fold1: mtry=2
## + Fold1: mtry=3
## - Fold1: mtry=3
## + Fold1: mtry=4
## - Fold1: mtry=4
## + Fold1: mtry=5
## - Fold1: mtry=5
## + Fold2: mtry=2
## - Fold2: mtry=2
## + Fold2: mtry=3
## - Fold2: mtry=3
## + Fold2: mtry=4
## - Fold2: mtry=4
## + Fold2: mtry=5
## - Fold2: mtry=5
## + Fold3: mtry=2
## - Fold3: mtry=2
## + Fold3: mtry=3
## - Fold3: mtry=3
## + Fold3: mtry=4
## - Fold3: mtry=4
## + Fold3: mtry=5
## - Fold3: mtry=5
## + Fold4: mtry=2
## - Fold4: mtry=2
## + Fold4: mtry=3
## - Fold4: mtry=3
## + Fold4: mtry=4
## - Fold4: mtry=4
## + Fold4: mtry=5
## - Fold4: mtry=5
## + Fold5: mtry=2
## - Fold5: mtry=2
## + Fold5: mtry=3
## - Fold5: mtry=3
## + Fold5: mtry=4
## - Fold5: mtry=4
## + Fold5: mtry=5
## - Fold5: mtry=5
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 2 on full training set
```

```
# Make predictions on test set
predictions <- predict(rf_model, newdata = test_data_balanced)

# Create confusion matrix
conf_mat <- confusionMatrix(predictions, test_data_balanced$credit_risk)
conf_mat_df <- as.data.frame.matrix(conf_mat$table)
conf_mat_df$Reference <- rownames(conf_mat_df)

# Reshape data for plotting
conf_mat_df <- melt(conf_mat_df, id.vars = "Reference", variable.name = "Prediction", value.name = "Freq")

# Plot confusion matrix
ggplot(conf_mat_df, aes(x = Prediction, y = Reference, fill = Freq)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "steelblue", guide = "colorbar") +
  geom_text(aes(label = Freq)) +
  labs(title = "Random Forest Model: Confusion Matrix", x = "Predicted Credit Risk", y = "True Credit Risk", fill = "Fr
equency")
```

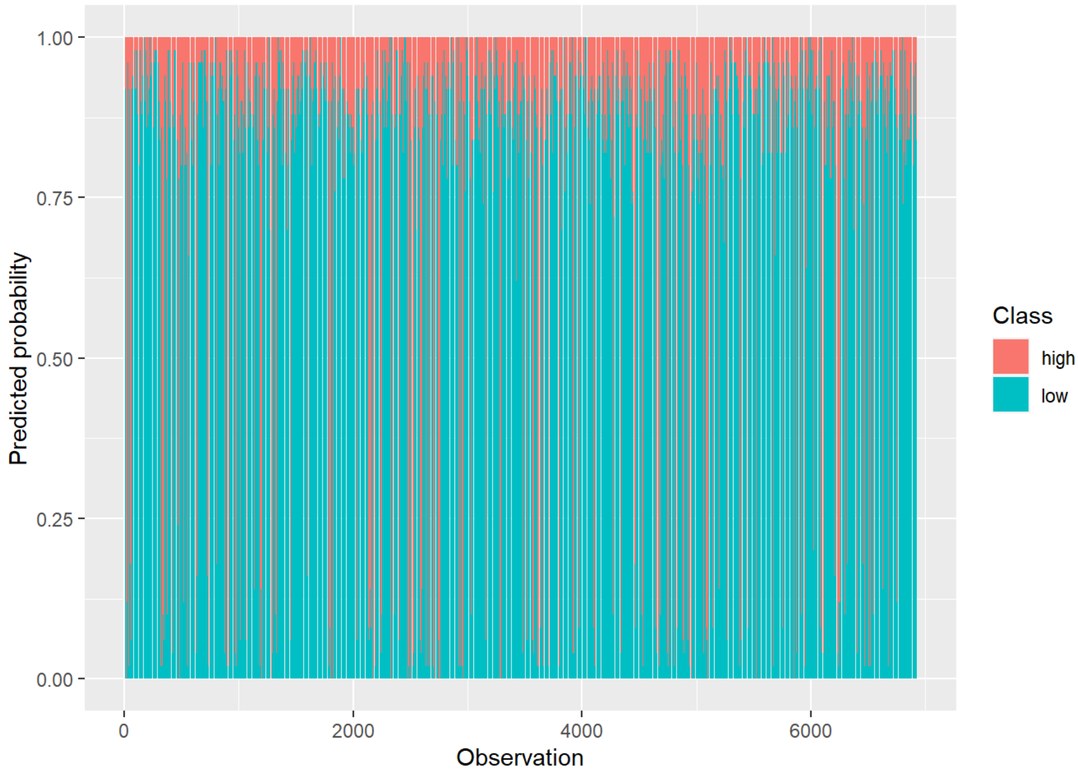


Make Predictions

```
# Convert credit_risk to a factor with the same levels as Class in the training data
merged_data$credit_risk <- factor(merged_data$credit_risk, levels = levels(train_data_balanced$Class))

# Rename credit_risk to Class
names(merged_data)[names(merged_data) == "credit_risk"] <- "Class"

# Use the rf_model to make predictions on merged_data
merged_predictions <- predict(rf_model, newdata = merged_data, type = "prob")
```

Visualize Predictions

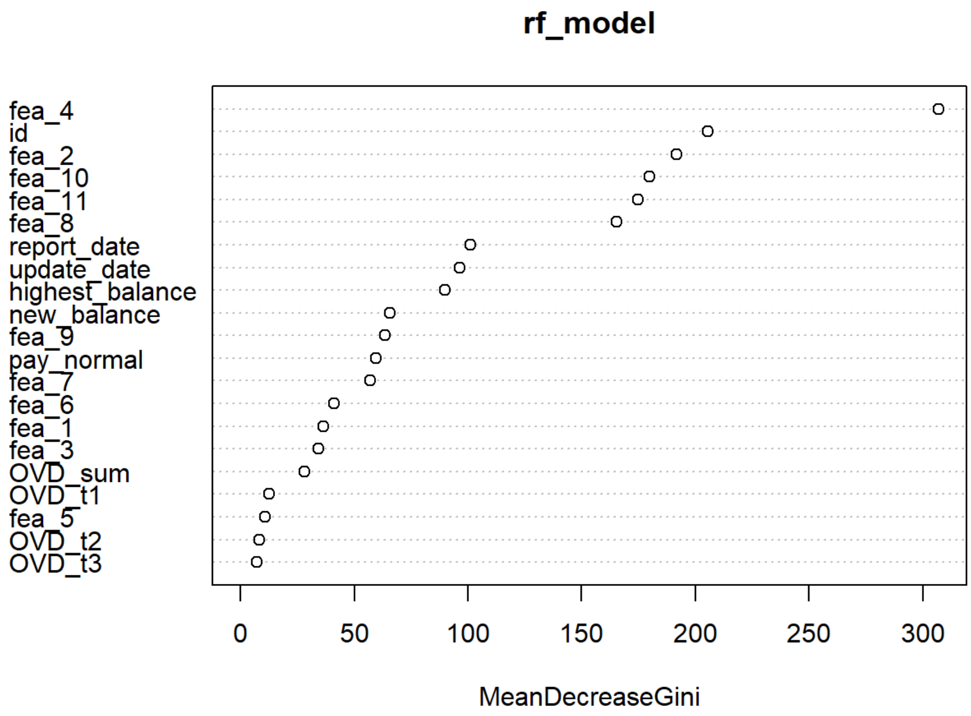
```
# Create a data frame with the predicted probabilities
prob_df <- data.frame(low = merged_predictions[,1], high = merged_predictions[,2])

# Add a column with the observation number
prob_df$obs <- 1:nrow(prob_df)

# Convert the data frame to long format
prob_df_long <- tidyr::gather(prob_df, key = "class", value = "probability", -obs)

# Create the stacked bar chart
ggplot(prob_df_long, aes(x = obs, y = probability, fill = class)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c("#F8766D", "#00BFC4")) +
  labs(x = "Observation", y = "Predicted probability", fill = "Class")
```



The model predicts high risk customers with good success.

The following answers Q1 - What are the factors contributing to Credit Risky customer?

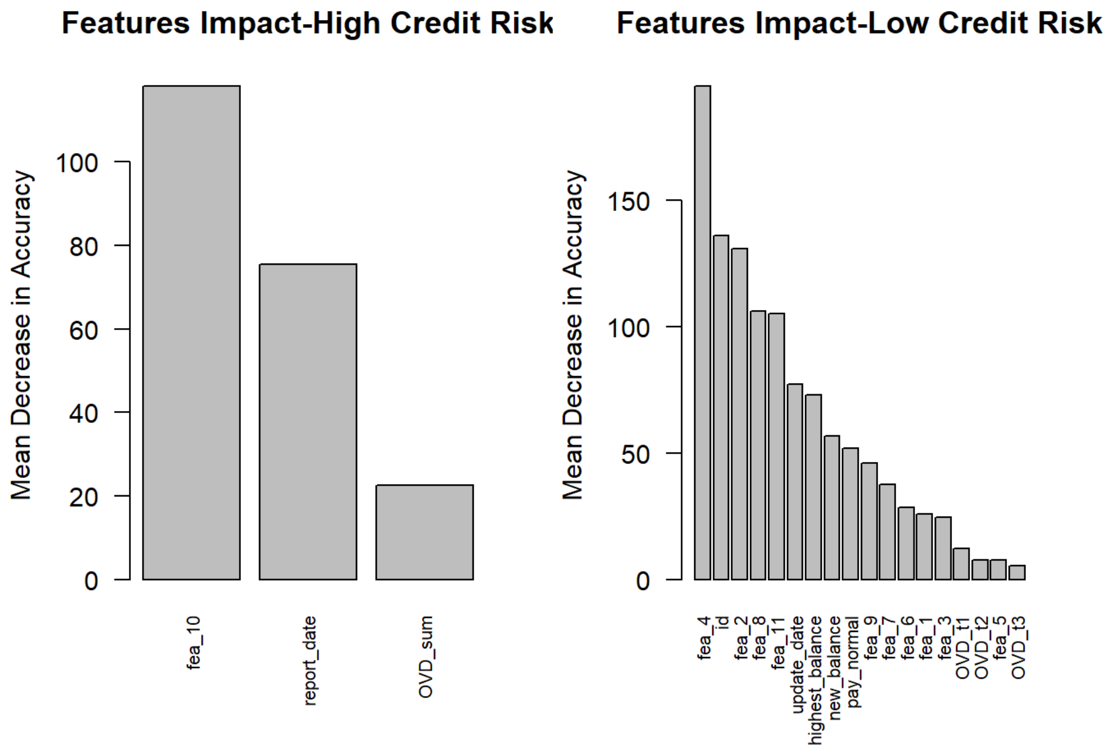
```
# Fit a random forest model to the data
rf_model <- randomForest(Class ~ ., data = merged_data)

# Create a variable importance plot
varImpPlot(rf_model)
```



MeanDecreaseGini is a measure of variable importance. It lists the features by impact on the credit risk. Though it doesn't differentiate high and low risk.

This differentiates the variable importance on hgih and low risk.

```r
varImpPlot_v2 <- function(data) {
  merged_data <- merge(customer_data, payment_data[, !(colnames(payment_data) %in% "prod_limit")], by = "id")
  merged_data <- na.omit(merged_data)
  merged_data <- merged_data[!apply(merged_data, 2, function(x) any(is.infinite(x))),]  # remove infinite values
  merged_data$credit_risk <- factor(ifelse(merged_data$credit_risk == "low", "low", "high"), levels = c("low", "high"))
  merged_data <- subset(merged_data, select = -prod_code)

  # split data into training and testing sets
  set.seed(123)
  train_index <- sample(nrow(merged_data), floor(0.7 * nrow(merged_data)))
  train_data <- merged_data[train_index, ]
  test_data <- merged_data[-train_index, ]


  # train the random forest model
  rf_model <- randomForest(credit_risk ~ ., data = train_data, ntree = 500, mtry = 3, importance = TRUE)

  # calculate feature importance
  imp <- importance(rf_model, scale = FALSE)

  # create data frame with feature importance
  imp_df <- data.frame(Feature = row.names(imp),
                       Overall_Importance = imp[, "MeanDecreaseGini"],
                       stringsAsFactors = FALSE)

  # order by overall importance
  imp_df <- imp_df[order(-imp_df$Overall_Importance), ]

  imp_high <- imp_df[train_data$credit_risk == "high", ]
  imp_low <- imp_df[train_data$credit_risk == "low", ]
  imp_high <- imp_high[order(-imp_high$Overall_Importance), ]
  imp_low <- imp_low[order(-imp_low$Overall_Importance), ]
  imp_low <- imp_low[!is.na(imp_low$Feature),]
  imp_high <- imp_high[!is.na(imp_high$Feature),]



# set plot size
options(repr.plot.width=12, repr.plot.height=6)


# plot feature importance for high and low credit risk
par(mfrow = c(1, 2))

# plot feature importance for high credit risk
barplot(imp_high$Overall_Importance, names.arg = imp_high$Feature, ylab = "Mean Decrease in Accuracy",
        main = "Features Impact-High Credit Risk", las = 2, cex.names = 0.7, width = 0.5)

# plot feature importance for low credit risk
barplot(imp_low$Overall_Importance, names.arg = imp_low$Feature, ylab = "Mean Decrease in Accuracy",
        main = "Features Impact-Low Credit Risk", las = 2, cex.names = 0.7, width = 0.5)

}

varImpPlot_v2(credit_data)
```


Features Impact-High Credit Risk
Mean Decrease in Accuracy
0
20
40
60
80
100
fea_10
report_date
OVD_sum
Features Impact-Low Credit Risk
Mean Decrease in Accuracy
0
50
100
150
fea_4
id
fea_2
fea_8
fea_11
update_date
highest_balance
new_balance
pay_normal
fea_9
fea_7
fea_6
fea_1
fea_3
OVD_t1
OVD_t2
fea_5
OVD_t3