Bank Loan Classification - HarvardX Capstone Project

Pradeep Kumar

19/06/2020

Contents

1 Executive Summary

As part of its customer acquisition efforts, **Bank of India** wants to run a campaign to convince more of its current customers to accept personal loan offers. In order to improve targeting quality, they want to find customers that are most likely to accept the personal loan offer. The dataset is from a previous campaign on 5,000 customers, 4,80 of them accepted. The metrics used to evaluate the models is **Classification Accuracy and F1-Score**; Although Accuracy is useful, we consider the F1-Score because the prediction class is unbalanced.
We have obtained an **F1-Score** of approximately **0.911** and Accuracy of **98.31%** for the best performing model.

2 Introduction

We use the dataset to solve the classification task. We go through the machine learning pipeline, starting with reading the dataset and exploring the data through plots and summaries. Then, we move to preprocess the data to standardize the data and check for any missing values. Later, we build models to classify the data. Finally, we evaluate the best models using the whole test dataset.

2.1 Objective of the project

The goal of this project is to train a machine learning model that classifies whether or not a customer will take a personal loan. Hence, the target variable is **Personal Loan**.

The metric used to evaluate the model's performance is the F1-Score. It is used because we have an unbalanced dataset. It tries to maximize both precision and recall.

2.2 Dataset

The dataset used is the Bank of India dataset of 5,000 customers.
The dataset is from a previous campaign on 5,000 customers run by the bank.

The **variables in the dataset** are described as follows:

```
1. id: Customer ID
2. age: Customer's age in completed years
3. experience: Number of years of professional experience
4. income: Annual income of the customer (in thousands)
5. zip: Home Address ZIP code.
6. family: The family size of the customer
7. credit_card_spend: Avg. spending on credit cards per month (in thousands)
8. education: Education Level. 1: Undergrad; 2: Graduate; 3: Advanced/Professional
9. mortgage: Value of house mortgage if any. (in thousands)
10. personal_loan: Did this customer accept the personal loan offered in the last campaign?
11. securities_account: Does the customer have securities account with the bank?
12. cd_account: Does the customer have a certificate of deposit (CD) account with the bank?
13. online: Does the customer use internet banking facilities?
14. credit_card: Does the customer use a credit card issued by Bank?
```

We read the dataset using either the local file, if available offline, or directly from the set up amazon s3 bucket.

```r
# Read data directly from my s3 bucket
raw_data<-read_csv("https://datasetbankofindia.s3.ap-south-1.amazonaws.com/Bank_of_India.csv")

# Read data from local file if you clone my repo
# raw_data <- read_csv("./dataset/Bank_of_India.csv")
```

First, To get familiar with the dataset, we look at the head of the dataset.

```
## # A tibble: 6 x 14
##      id   age experience income   zip family credit_card_spe~ education mortgage
##   <dbl> <dbl>      <dbl>  <dbl> <dbl>  <dbl>            <dbl>     <dbl>    <dbl>
## 1     1    25          1     49 91107      4              1.6         1        0
## 2     2    45         19     34 90089      3              1.5         1        0
## 3     3    39         15     11 94720      1              1           1        0
## 4     4    35          9    100 94112      1              2.7         2        0
## 5     5    35          8     45 91330      4              1           2        0
## 6     6    37         13     29 92121      4              0.4         2      155
## # ... with 5 more variables: personal_loan <dbl>, securities_account <dbl>,
```

```
## #   cd_account <dbl>, online <dbl>, credit_card <dbl>
```

We check if the dat has any missing values:

```r
# Check for missing values
colSums(is.na(raw_data))
```

```
##                  id                 age          experience              income 
##                   0                   0                   0                   0 
##                 zip              family   credit_card_spend           education 
##                   0                   0                   0                   0 
##            mortgage       personal_loan securities_account          cd_account 
##                   0                   0                   0                   0 
##              online         credit_card 
##                   0                   0
```

```r
# turns out: No missing value;
# If there were missing values do imputation or knn imputation
```

We confirm that there are **no missing values(NAs)**. Hence, we do not need to remove or impute missing values.

```r
# Summary Statistics of the dataset
summary(raw_data)
```

```
##        id            age          experience       income            zip       
##  Min.   :   1   Min.   :23.00   Min.   :-3.0   Min.   :  8.00   Min.   : 9307  
##  1st Qu.:1251   1st Qu.:35.00   1st Qu.:10.0   1st Qu.: 39.00   1st Qu.:91911  
##  Median :2500   Median :45.00   Median :20.0   Median : 64.00   Median :93437  
##  Mean   :2500   Mean   :45.34   Mean   :20.1   Mean   : 73.77   Mean   :93153  
##  3rd Qu.:3750   3rd Qu.:55.00   3rd Qu.:30.0   3rd Qu.: 98.00   3rd Qu.:94608  
##  Max.   :5000   Max.   :67.00   Max.   :43.0   Max.   :224.00   Max.   :96651  
##      family      credit_card_spend   education        mortgage    
##  Min.   :1.000   Min.   : 0.000    Min.   :1.000   Min.   :  0.0  
##  1st Qu.:1.000   1st Qu.: 0.700    1st Qu.:1.000   1st Qu.:  0.0  
##  Median :2.000   Median : 1.500    Median :2.000   Median :  0.0  
##  Mean   :2.396   Mean   : 1.938    Mean   :1.881   Mean   : 56.5  
##  3rd Qu.:3.000   3rd Qu.: 2.500    3rd Qu.:3.000   3rd Qu.:101.0  
##  Max.   :4.000   Max.   :10.000    Max.   :3.000   Max.   :635.0  
##  personal_loan   securities_account   cd_account         online      
##  Min.   :0.000   Min.   :0.0000     Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:0.000   1st Qu.:0.0000     1st Qu.:0.0000   1st Qu.:0.0000  
##  Median :0.000   Median :0.0000     Median :0.0000   Median :1.0000  
##  Mean   :0.096   Mean   :0.1044     Mean   :0.0604   Mean   :0.5968  
##  3rd Qu.:0.000   3rd Qu.:0.0000     3rd Qu.:0.0000   3rd Qu.:1.0000  
##  Max.   :1.000   Max.   :1.0000     Max.   :1.0000   Max.   :1.0000  
##   credit_card   
##  Min.   :0.000  
##  1st Qu.:0.000  
##  Median :0.000  
##  Mean   :0.294  
##  3rd Qu.:1.000  
##  Max.   :1.000
```

From the structure of the dataset, we see that all the columns are interpreted as **numeric**. We need to change the types of some variables to **categorical(factor)**.

```
## tibble [5,000 x 14] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
```

```
##  $ id                : num [1:5000] 1 2 3 4 5 6 7 8 9 10 ...
##  $ age               : num [1:5000] 25 45 39 35 35 37 53 50 35 34 ...
##  $ experience        : num [1:5000] 1 19 15 9 8 13 27 24 10 9 ...
##  $ income            : num [1:5000] 49 34 11 100 45 29 72 22 81 180 ...
##  $ zip               : num [1:5000] 91107 90089 94720 94112 91330 ...
##  $ family            : num [1:5000] 4 3 1 1 4 4 2 1 3 1 ...
##  $ credit_card_spend : num [1:5000] 1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
##  $ education         : num [1:5000] 1 1 1 2 2 2 2 3 2 3 ...
##  $ mortgage          : num [1:5000] 0 0 0 0 0 155 0 0 104 0 ...
##  $ personal_loan     : num [1:5000] 0 0 0 0 0 0 0 0 0 1 ...
##  $ securities_account: num [1:5000] 1 1 0 0 0 0 0 0 0 0 ...
##  $ cd_account        : num [1:5000] 0 0 0 0 0 0 0 0 0 0 ...
##  $ online            : num [1:5000] 0 0 0 0 0 1 1 0 1 0 ...
##  $ credit_card       : num [1:5000] 0 0 0 0 1 0 0 1 0 0 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   id = col_double(),
##   ..   age = col_double(),
##   ..   experience = col_double(),
##   ..   income = col_double(),
##   ..   zip = col_double(),
##   ..   family = col_double(),
##   ..   credit_card_spend = col_double(),
##   ..   education = col_double(),
##   ..   mortgage = col_double(),
##   ..   personal_loan = col_double(),
##   ..   securities_account = col_double(),
##   ..   cd_account = col_double(),
##   ..   online = col_double(),
##   ..   credit_card = col_double()
##   .. )
```

2.3 Preliminary Data Cleaning

We know that Id and ZIP code are not valuable information when it comes to being useful for the classification task. Hence, we have deleted both variables from the dataset.

```r
# Removing unnecessary columns ID and zipcode
raw_data$id <- NULL
raw_data$zip <- NULL
```

Next, we change the categorical vairables **perosonal_loan**, **education**, **family**, **securities_account**, **online**, **credit_card** into factors.

```r
# Do necessary type conversions | Categoridcal Data
raw_data$personal_loan <- as_factor(raw_data$personal_loan)
raw_data$education <- as_factor(raw_data$education)
raw_data$family <- as_factor(raw_data$family)
raw_data$securities_account <- as_factor(raw_data$securities_account)
raw_data$cd_account <- as_factor(raw_data$cd_account)
raw_data$online <- as_factor(raw_data$online)
raw_data$credit_card <- as_factor(raw_data$credit_card)
```

We note that the types of variables have been updated as required.

```r
# Structure of the dataset
str(raw_data)
```

```
## tibble [5,000 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ age                : num [1:5000] 25 45 39 35 35 37 53 50 35 34 ...
##  $ experience         : num [1:5000] 1 19 15 9 8 13 27 24 10 9 ...
##  $ income             : num [1:5000] 49 34 11 100 45 29 72 22 81 180 ...
##  $ family             : Factor w/ 4 levels "1","2","3","4": 4 3 1 1 4 4 2 1 3 1 ...
##  $ credit_card_spend  : num [1:5000] 1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
##  $ education          : Factor w/ 3 levels "1","2","3": 1 1 1 2 2 2 2 3 2 3 ...
##  $ mortgage           : num [1:5000] 0 0 0 0 0 155 0 0 104 0 ...
##  $ personal_loan      : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
##  $ securities_account: Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 1 1 1 ...
##  $ cd_account         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ online             : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 2 1 2 1 ...
##  $ credit_card        : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 2 1 1 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   id = col_double(),
##   ..   age = col_double(),
##   ..   experience = col_double(),
##   ..   income = col_double(),
##   ..   zip = col_double(),
##   ..   family = col_double(),
##   ..   credit_card_spend = col_double(),
##   ..   education = col_double(),
##   ..   mortgage = col_double(),
##   ..   personal_loan = col_double(),
##   ..   securities_account = col_double(),
##   ..   cd_account = col_double(),
##   ..   online = col_double(),
##   ..   credit_card = col_double()
##   .. )
```

3 Exploratory Data Analysis

3.1 Odds

We begin by calculating the odds of a customer taking a personal loan based on whether they have a securities account, whether they have a cd account, whether they engage in online banking, whether they use the bank credit card.

```
# Calculating odds of taking a personal loan based on whether securities_account = 1
limited = raw_data[raw_data$securities_account == "1",]
(likely_securities=sum(limited$personal_loan=="1")/sum(limited$personal_loan=="0"))
```

```
## [1] 0.1298701
```

If people opened securities account, it is 0.12 times more likely that people would borrow than not

```
# Calculating odds of taking a personal loan based on whether cd_account = 1
limited = raw_data[raw_data$cd_account == "1",]
(likely_CD=sum(limited$personal_loan=="1")/sum(limited$personal_loan=="0"))
```

```
## [1] 0.8641975
```

If people opened CD account, it is 0.86 times more likely that people would borrow than not

```
# Calculating odds of taking a personal loan based on whether online = 1
limited = raw_data[raw_data$online == "1",]
(likely_Online=sum(limited$personal_loan=="1")/sum(limited$personal_loan=="0"))
```

```
## [1] 0.1080579
```

If people engaged in Online banking, it is 0.108 times more likely that people would borrow than not

```
# Calculating odds of taking a personal loan based on whether credit_card = 1
limited = raw_data[raw_data$credit_card == "1",]
(likely_CC=sum(limited$personal_loan=="1")/sum(limited$personal_loan=="0"))
```

```
## [1] 0.1077619
```

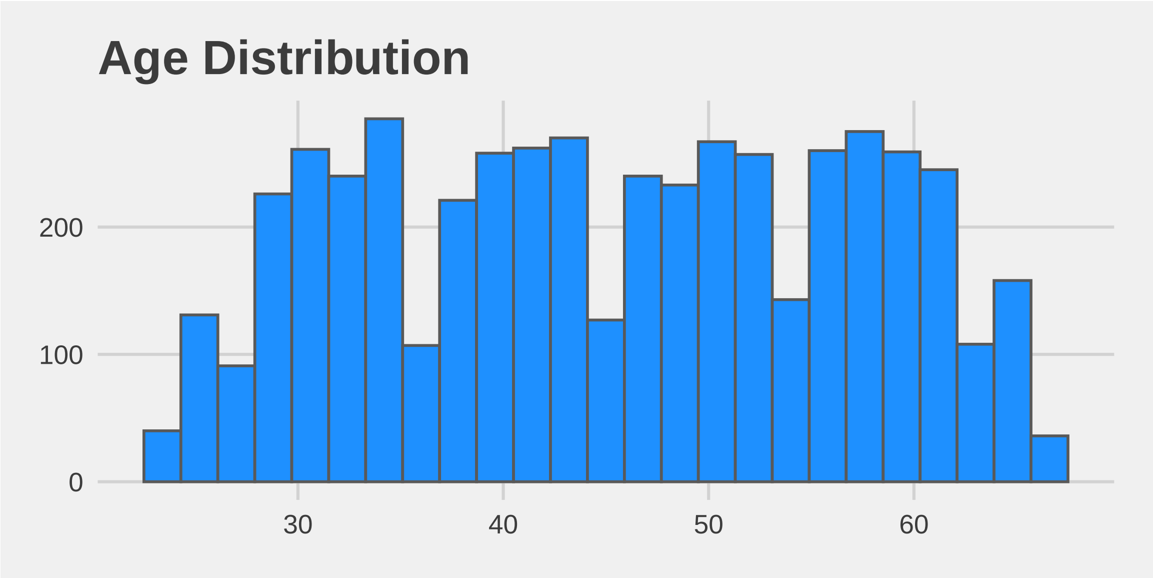
If people used bank credit crads, it is 0.107 times more likely that people would borrow than not

3.2 Visuals

3.2.1 Univariate plots

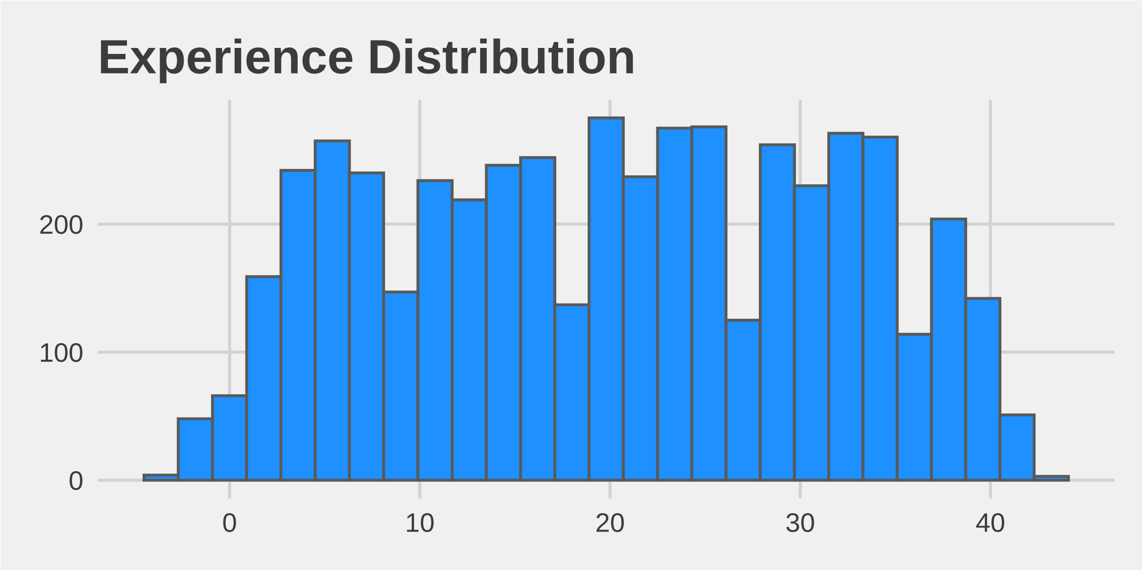
Age Distribution

```
# Age Distribution bar plot
raw_data %>%
  ggplot(aes(x = age)) +
  geom_histogram(stat = 'bin', binwidth = 1.8, color = '#595959', fill = '#1E90FF') +
  labs(title = "Age Distribution") +
  theme_fivethirtyeight()
```



Experience Distribution

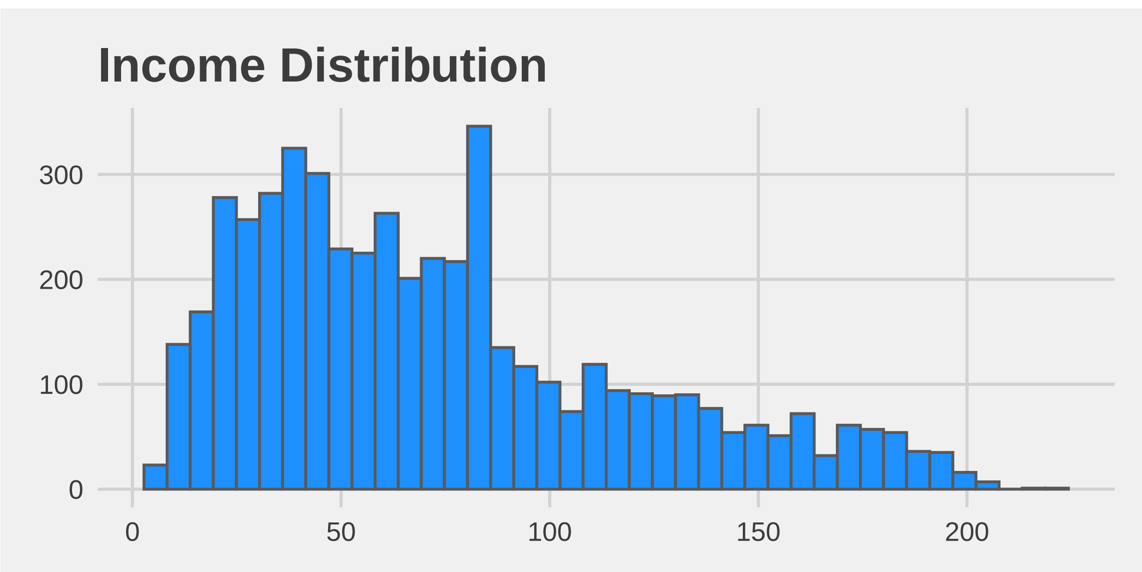
```
# Experience distribution bar plot
raw_data %>%
  ggplot(aes(x = experience)) +
  geom_histogram(stat = 'bin', binwidth = 1.8, color = '#595959', fill = '#1E90FF') +
  labs(title = "Experience Distribution") +
  theme_fivethirtyeight()
```



We note that age and experience distributions look similar, They might be highly correlated. If so, we might have to remove one of the variables so that our models do not fail.

Income Distribution

```
# Income Distribution bar plot
raw_data %>%
  ggplot(aes(x = income)) +
  geom_bar(stat = 'bin', bins = 40, color = '#595959', fill = '#1E90FF') +
  labs(title = "Income Distribution") +
  theme_fivethirtyeight()
```

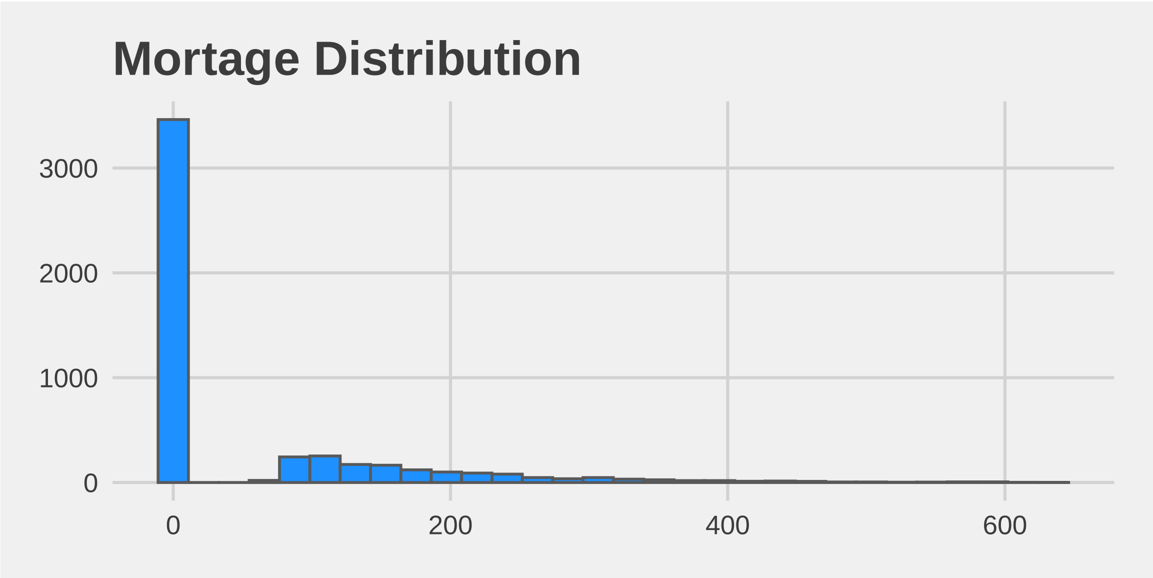


Mortgage Distribution

```
# Mortgage Distribution bar plot
raw_data %>%
  ggplot(aes(x = mortgage)) +
  geom_bar(stat = 'bin', color = '#595959', fill = '#1E90FF') +
```

```
  labs(title = "Mortage Distribution") +
  theme_fivethirtyeight()
```

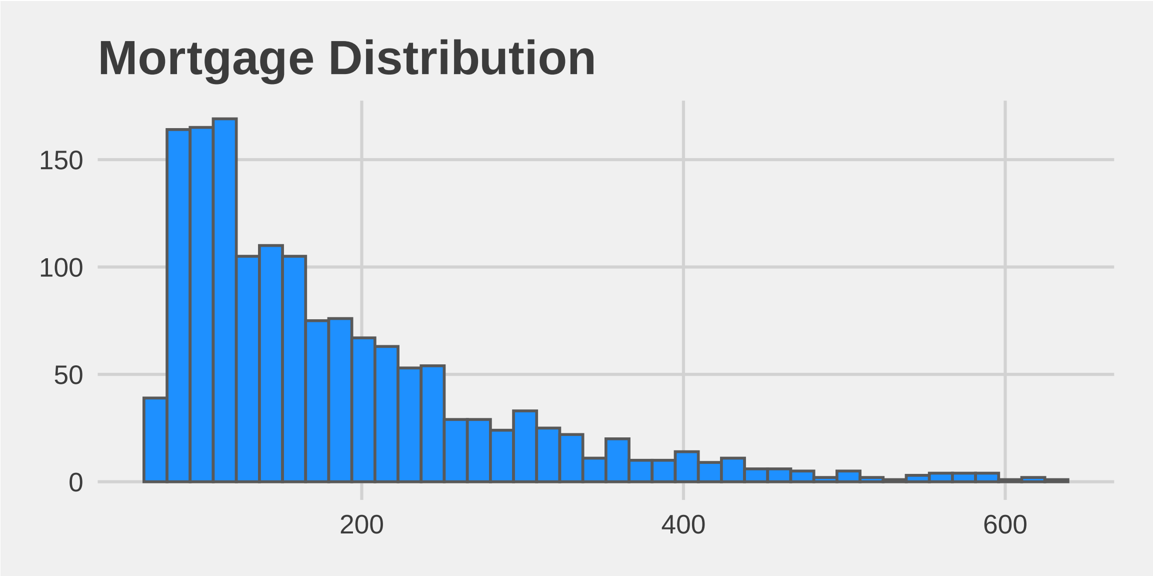
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Seems like most people have no mortage (i.e, mortage is 0); So, we produce another plot removing those without mortgages.

Mortgage Distribution exclding people with no mortgages

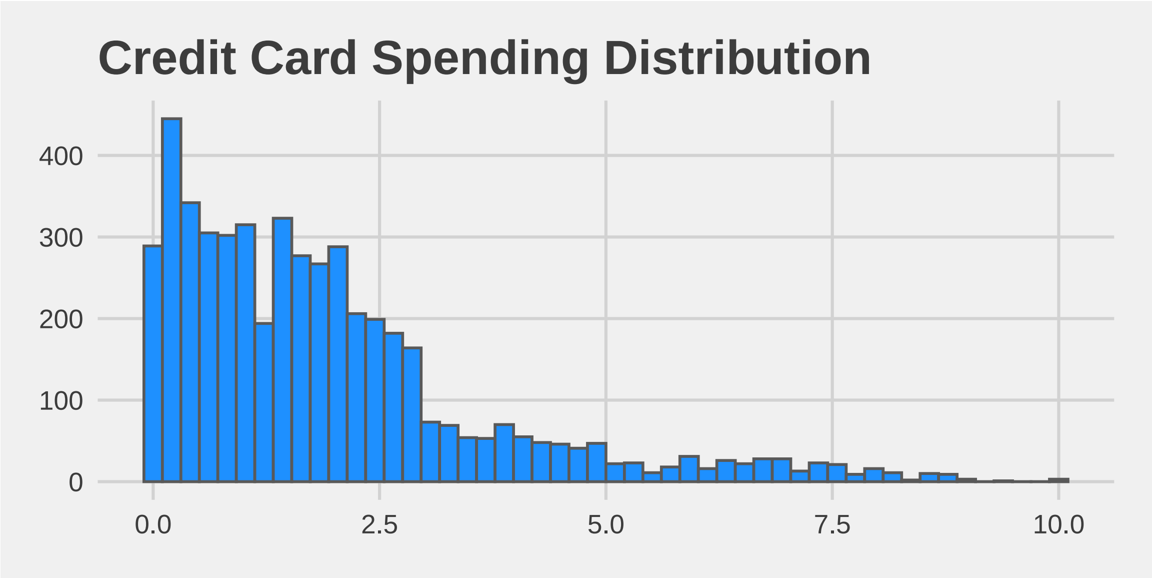
```
# Mortage Distribution bar plot for people with mortgages (exclude 0)
raw_data %>%
  filter(mortgage > 0) %>%
  ggplot(aes(x = mortgage)) +
  geom_bar(stat = 'bin', bins = 40, color = '#595959', fill = '#1E90FF') +
  labs(title = "Mortgage Distribution") +
  theme_fivethirtyeight()
```



```
# It is a right skewed distribution
```

Credit Card Spending Distribution

```
# Credit Card Spending Distribution bar plot for people with mortgages
raw_data %>%
  ggplot(aes(x = credit_card_spend)) +
  geom_bar(stat = 'bin', bins = 50, color = '#595959', fill = '#1E90FF') +
  labs(title = "Credit Card Spending Distribution") +
  theme_fivethirtyeight()
```

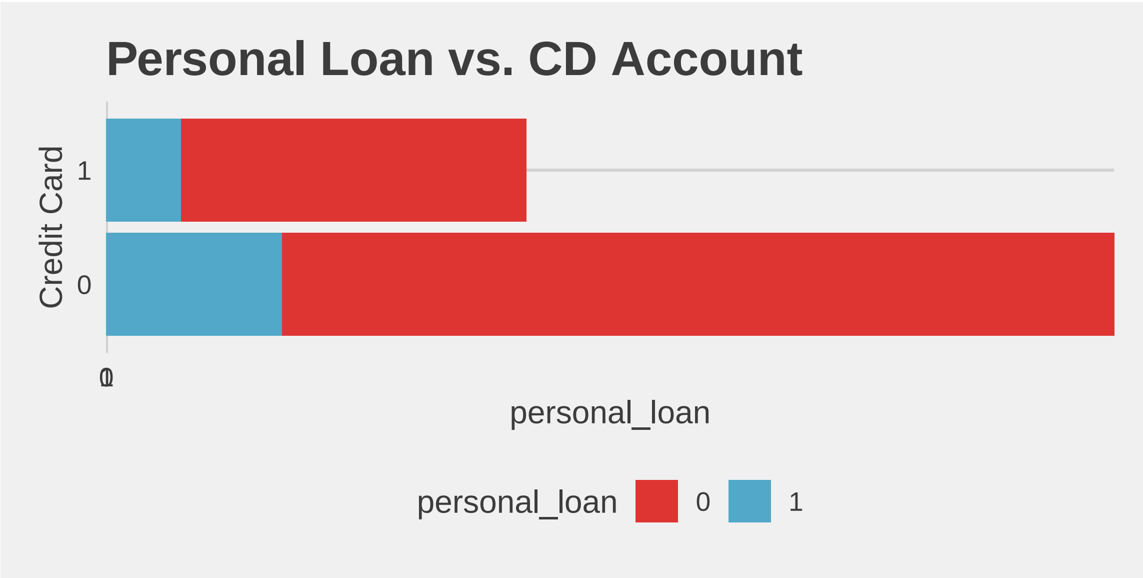



```
# It is a right skewed distribution
```

3.2.2 Bivariate Plots

Personal Loan vs. CD Account

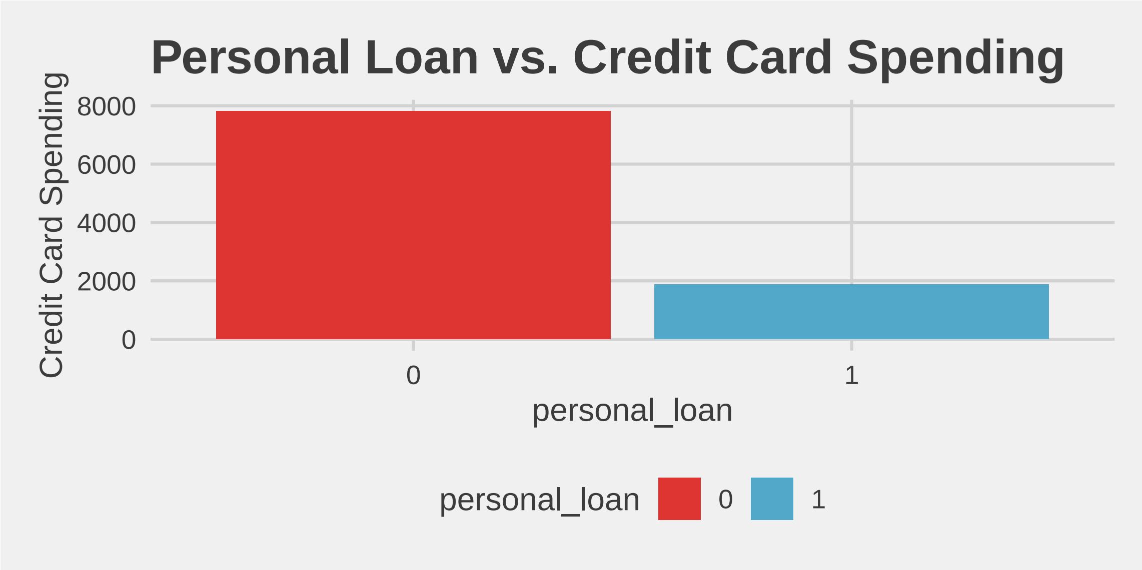
```
# Stacked bar plot to test for Credit Card vs. personal loan
raw_data %>%
  ggplot(aes(x = credit_card, y = personal_loan, fill = personal_loan)) +
  geom_bar(stat = 'identity') +
  coord_flip() +
  scale_fill_manual(values=c('#DE3533','#51A8C9')) +
  labs(title = "Personal Loan vs. CD Account") +
  theme_fivethirtyeight() +
  theme(axis.title = element_text()) +
  xlab('Credit Card')
```



We can observe that people who do not have a credit card are more likey to get personal loan.

Personal Loan vs. Credit Card Spending

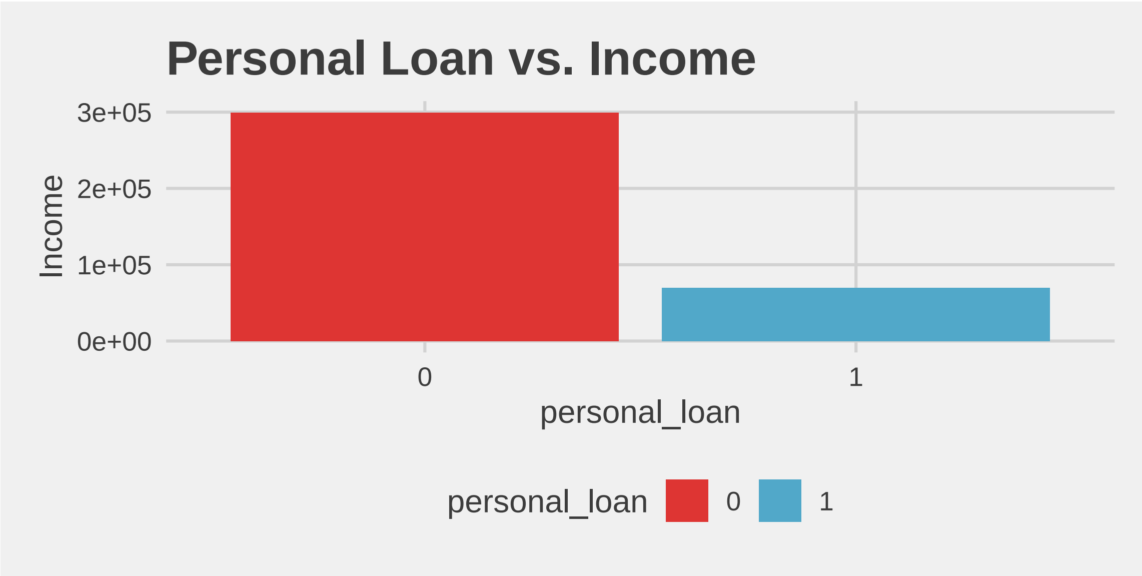
```
# Bar plot to test for Credit Card Spending vs. personal loan
raw_data %>%
  ggplot(aes(x = credit_card_spend, y = personal_loan, fill = personal_loan)) +
  geom_bar(stat = 'identity') +
  coord_flip() +
  scale_fill_manual(values=c('#DE3533','#51A8C9')) +
  labs(title = "Personal Loan vs. Credit Card Spending") +
  theme_fivethirtyeight() +
  theme(axis.title = element_text()) +
  xlab('Credit Card Spending')
```



We can observe that people whose credit card spending is high do not generally borrow loans

Personal Loan vs. Income

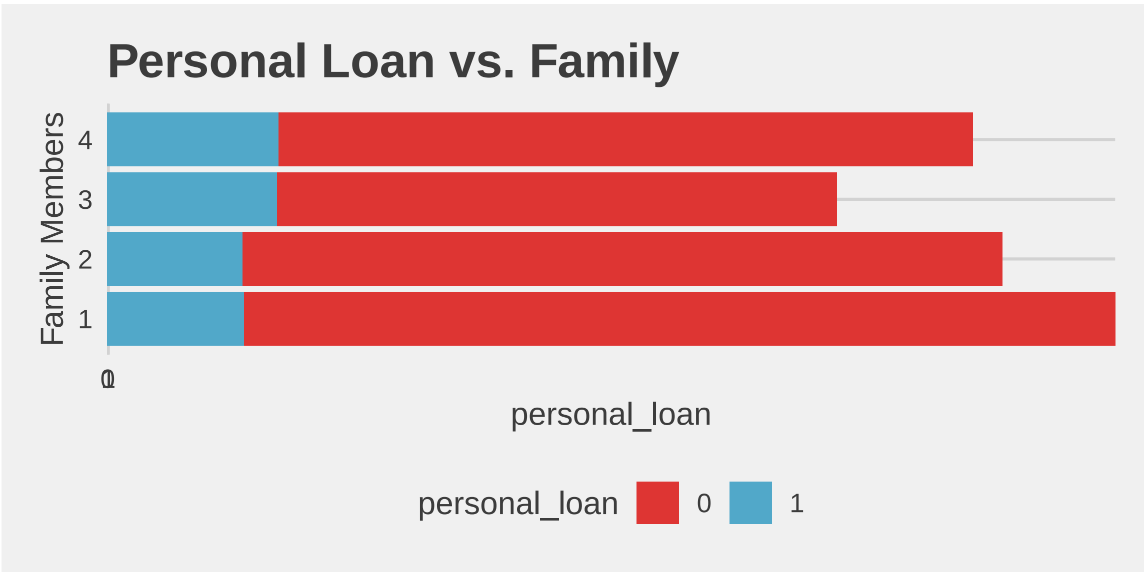
```
# Bar plot to test for Income vs. personal loan
raw_data %>%
  ggplot(aes(x = income, y = personal_loan, fill = personal_loan)) +
  geom_bar(stat = 'identity') +
  coord_flip() +
  scale_fill_manual(values=c('#DE3533','#51A8C9')) +
  labs(title = "Personal Loan vs. Income") +
  theme_fivethirtyeight() +
  theme(axis.title = element_text()) +
  xlab('Income')
```



We can observe that people with higher incomes do not generally borrow loans

Personal Loan vs. Family

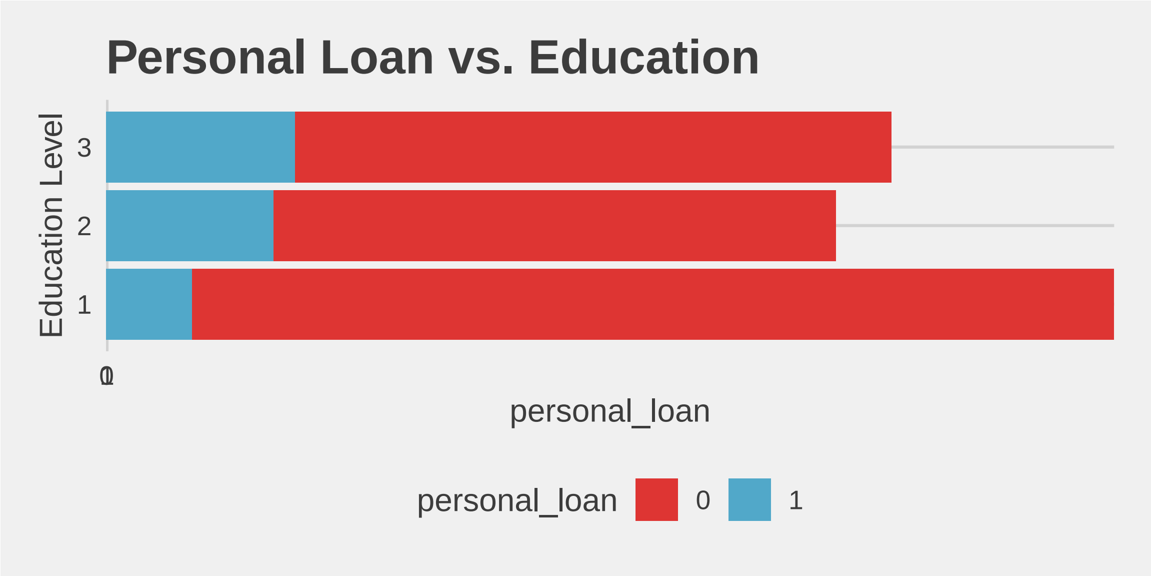
```r
# Stacked bar plot to test for family vs. personal loan
raw_data %>%
  ggplot(aes(x = family, y = personal_loan, fill = personal_loan)) +
  geom_bar(stat = 'identity') +
  coord_flip() +
  scale_fill_manual(values=c('#DE3533','#51A8C9')) +
  labs(title = "Personal Loan vs. Family") +
  theme_fivethirtyeight() +
  theme(axis.title = element_text()) +
  xlab('Family Members')
```



We observe the differnce in whether or not people take loans based on the number of family members.

Personal Loan vs. Education

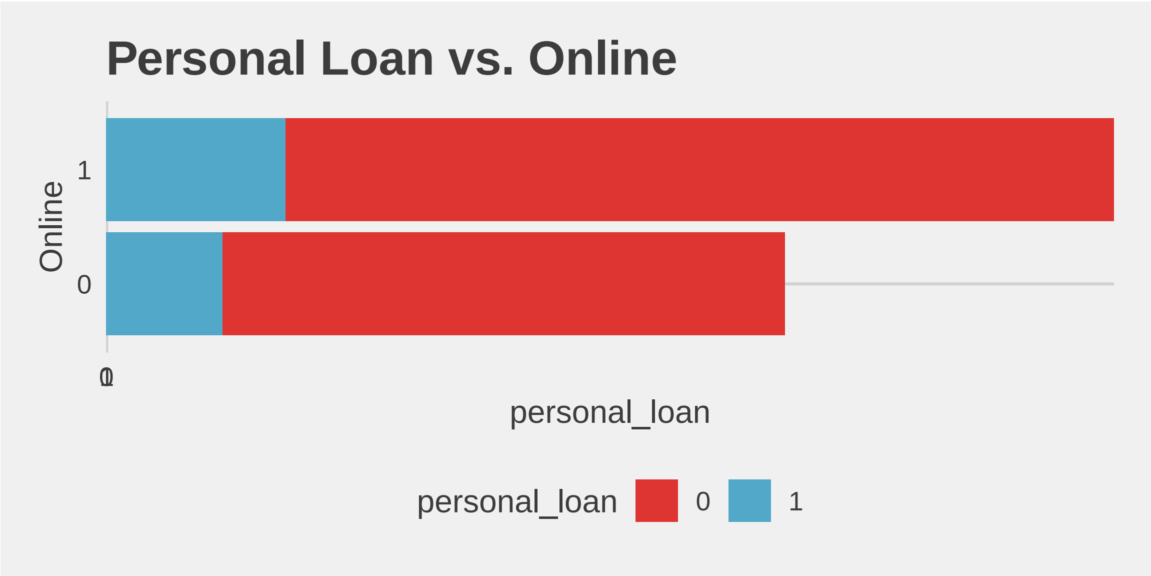
```r
# Stacked bar plot to test for education vs. personal loan
raw_data %>%
  ggplot(aes(x = education, y = personal_loan, fill = personal_loan)) +
  geom_bar(stat = 'identity') +
  coord_flip() +
  scale_fill_manual(values=c('#DE3533','#51A8C9')) +
  labs(title = "Personal Loan vs. Education") +
  theme_fivethirtyeight() +
  theme(axis.title = element_text()) +
  xlab('Education Level')
```



We can observe some differneces.

Personal Loan vs. Online

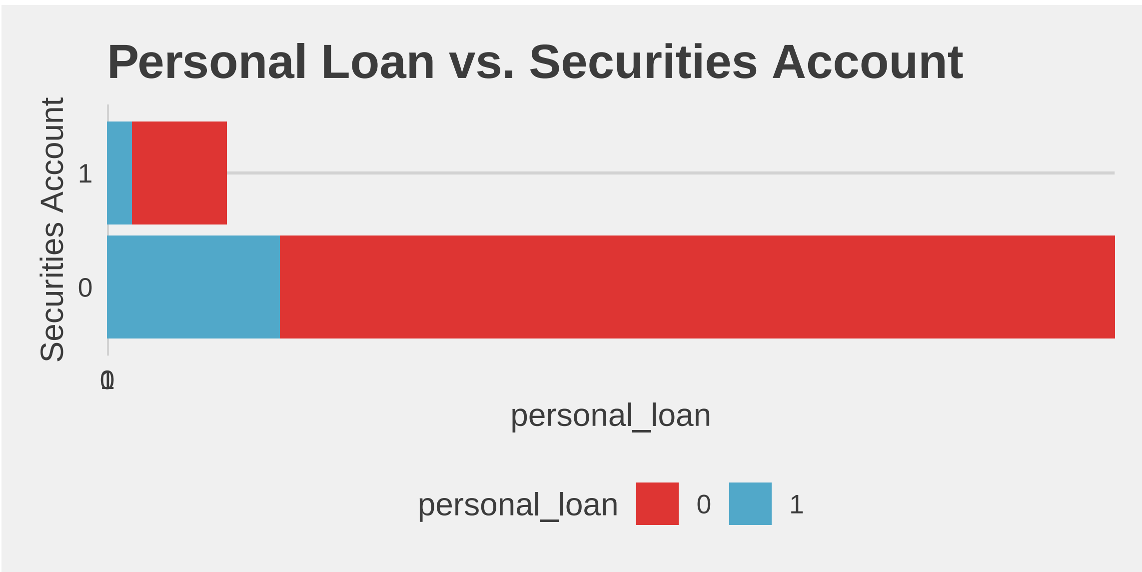
```
# Stacked bar plot to test for online vs. personal loan
raw_data %>%
  ggplot(aes(x = online, y = personal_loan, fill = personal_loan)) +
  geom_bar(stat = 'identity') +
  coord_flip() +
  scale_fill_manual(values=c('#DE3533','#51A8C9')) +
  labs(title = "Personal Loan vs. Online") +
  theme_fivethirtyeight() +
  theme(axis.title = element_text()) +
  xlab('Online')
```



We can observe the differnce in whether or not people borrow loans based on whether they engage in online banking.

Personal Loan vs. Securities Account

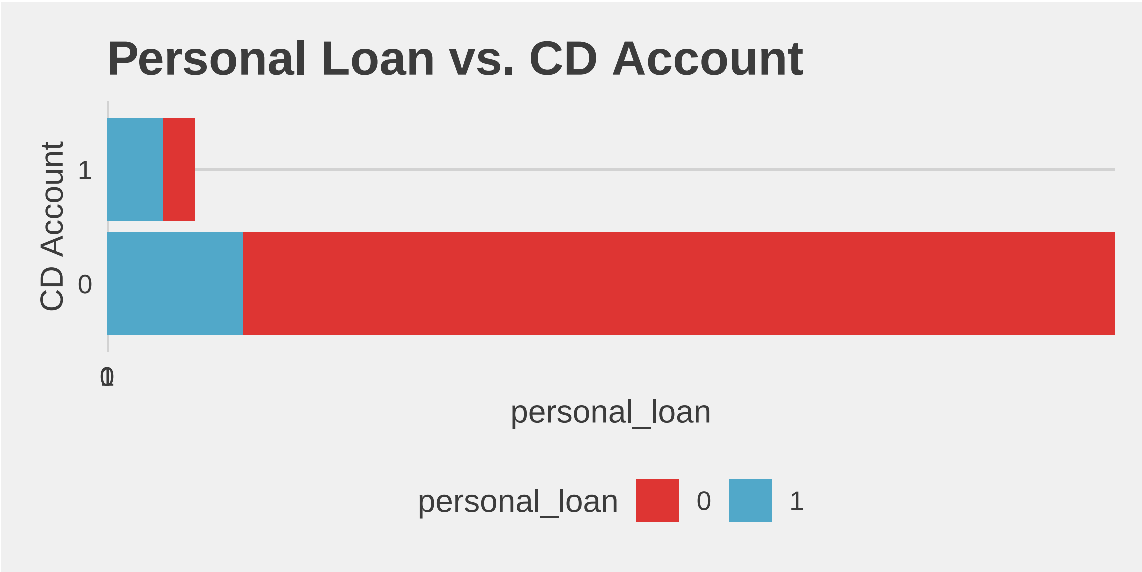
```
# Stacked bar plot to test for Securites Account vs. personal loan
raw_data %>%
  ggplot(aes(x = securities_account, y = personal_loan, fill = personal_loan)) +
  geom_bar(stat = 'identity') +
  coord_flip() +
  scale_fill_manual(values=c('#DE3533','#51A8C9')) +
  labs(title = "Personal Loan vs. Securities Account") +
  theme_fivethirtyeight() +
  theme(axis.title = element_text()) +
  xlab('Securities Account')
```



We can observe that people are more likely to a loan if they do not have a securities account

Personal Loan vs. CD Account

```
# Stacked bar plot to test for CD Account vs. personal loan
raw_data %>%
  ggplot(aes(x = cd_account, y = personal_loan, fill = personal_loan)) +
  geom_bar(stat = 'identity') +
  coord_flip() +
  scale_fill_manual(values=c('#DE3533','#51A8C9')) +
  labs(title = "Personal Loan vs. CD Account") +
  theme_fivethirtyeight() +
  theme(axis.title = element_text()) +
  xlab('CD Account')
```



We can observe that people are more likely to a loan if they do not have a securities account

3.2.3 Correlation Heatmap

We generate a heatmap of **correlations** among the numeric variables in the dataset.

```
# Correlation plot for the numeric columns
numeric_vaiables <- c('age', 'experience', 'income', 'mortgage', 'credit_card_spend')

# Using ggcor from GGally package to produce correlation heatmap
ggcorr(raw_data[, numeric_vaiables], nbreaks = 7,
       low = "#1E90FF", mid = "#AAAAAA", high = "#F21A00",
       label = TRUE, label_size = 7, label_color='black',
       legend.position = 'left') +
  theme_gray()
```

We can observe that age and experience are **highly correlated**, Hence, we have to remove experience.

4 Data Preprocessing

Remove the experience variable

```
# Removing Experience variable as it is highly correlated with age and will mess with our models if lef
raw_data$experience <- NULL

# Data Splitting - Training Data = 60%, Validation Data = 20% & Testing Data = 20%
```

4.1 Split the data

Splitting the dataset into train, validation and test We split the dataset into three sets:

```
1. Training    60%  | To train the models on
2. Validation  20%  | To tune and test our models to select best models
3. Testing     20%  | To evaluate the final model
```

```
# Setting seed for reproducibility of results
# Remove sample.kind = "Rounding" if R version < 3.5
set.seed(7, sample.kind = "Rounding")
```

```
## Warning in set.seed(7, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
# sample into three sets to create indices for train, validation, test sets
idx <- sample(seq(1, 3), size = nrow(raw_data), replace = TRUE, prob = c(.6, .2, .2))

# Split the data into three sets
raw_train <- raw_data[idx == 1,]
raw_val <- raw_data[idx == 2,]
raw_test <- raw_data[idx == 3,]
```

Standardizing the data

We have seen that the numeric data **ranges** are very different. So to bring them into a similar range, we standardize the data. i.e., we scale and centre the dataset.

```
# Standardizing the numeric varaibles as we've seen that the ranges of numerical variables vary quite a
# Using preProcess from caret package
(norm.values <- preProcess(raw_train, method=c("center", "scale")))
```

```
## Created from 3012 samples and 11 variables
##
## Pre-processing:
##   - centered (4)
##   - ignored (7)
##   - scaled (4)
```

```
# Apply the preProcess params to the dat using preProcess.predict
train <- predict(norm.values, raw_train)
val <- predict(norm.values, raw_val)
test <- predict(norm.values, raw_test)
```

Summary of normalized data

```
# Check the new ranges
summary(train)
```

```
##       age               income         family  credit_card_spend education
##  Min.   :-1.96297   Min.   :-1.4310   1:877   Min.   :-1.1181   1:1295
##  1st Qu.:-0.91063   1st Qu.:-0.7584   2:750   1st Qu.:-0.7149   2: 828
##  Median :-0.03369   Median :-0.2160   3:628   Median :-0.1965   3: 889
##  Mean   : 0.00000   Mean   : 0.0000   4:757   Mean   : 0.0000
##  3rd Qu.: 0.84326   3rd Qu.: 0.5217           3rd Qu.: 0.3363
##  Max.   : 1.89559   Max.   : 3.1254           Max.   : 4.2387
##     mortgage       personal_loan securities_account cd_account online
##  Min.   :-0.5555   0:2724        0:2704             0:2834     0:1205
##  1st Qu.:-0.5555   1: 288        1: 308             1: 178     1:1807
##  Median :-0.5555
##  Mean   : 0.0000
##  3rd Qu.: 0.4367
##  Max.   : 5.5668
##  credit_card
##  0:2157
##  1: 855
##
##
##
##
```

5 Model Building

First, we define a function to produce neat confusion matrix plots using **ggcorr from GGally** library. This library is built on top of the ggplot2 library.

```
# Helper function to draw the confusion matrix using ggplot
prettyConfusion <- function(results){
  # Convert the results from confusionMatrix toa  data frame
  table <- data.frame(results$table)

  # Calcualte Proportions and Predicted columns
  plotTable <- table %>%
    mutate(Predicted = ifelse(table$Prediction == table$Reference, "Correct", "Wrong")) %>%
    group_by(Reference) %>%
    mutate(Proportion = Freq/sum(Freq))

  # Fill alpha relative to sensitivity/specificity by
  # proportional outcomes within reference groups
  ggplot(plotTable, aes(Reference, Prediction, fill = Predicted, alpha = Proportion)) +
    geom_tile() +
    geom_text(aes(label = Freq), vjust = .5, fontface  = "bold", size=10) +
    scale_fill_manual(values = c(Correct = "springgreen2", Wrong = "orangered2")) +
    theme_bw() +
    xlim(rev(levels(table$Reference))) +
    theme_map() +
    theme(legend.position = "none")
}
```

Logistic Regression Classifier

```
##--------Logistic Regression Classifier----------##
# train
lr <- train(personal_loan ~ ., data = train, method = "glm", family = binomial)

# predictions
pred_lr <- predict(lr, val)

# Accuracy of the model
(acc_lr <- mean(pred_lr == val$personal_loan))
```

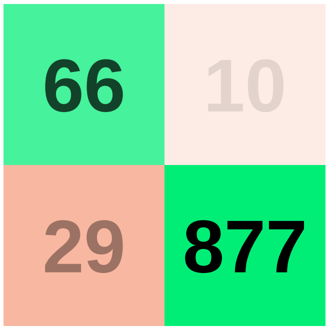
```
## [1] 0.9602851
```

```
# Generate Confusion Matrix of the model
results_lr <- confusionMatrix(pred_lr, val$personal_loan, positive="1")

# F1 Score
(f1_lr <- results_lr$byClass['F1'])
```

```
##        F1
## 0.7719298
```

```
# Draw a pretty confusion matrix using the custom helper function
prettyConfusion(results_lr)
```



Accuracy: 96.0285132
F1-Score: 0.7719298

Naīve Bayes Classifier

```
##------------Naive Bayes Classifier--------------##
# train
nb <- train(personal_loan ~ ., data = train, method = "nb")

# predictions
pred_nb <- predict(nb, val)

# Accuracy of the model
(acc_nb <- mean(pred_nb == val$personal_loan))
```

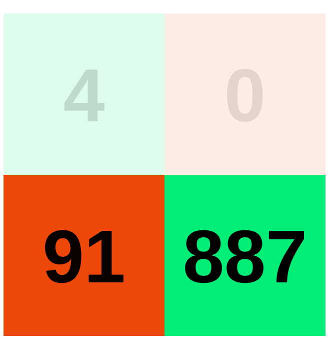
```
## [1] 0.907332
```

```
# Generate Confusion Matrix of the model
results_nb <- confusionMatrix(pred_nb, val$personal_loan, positive="1")

# F1 Score
(f1_nb <- results_nb$byClass['F1'])
```

```
##         F1
## 0.08080808
```

```
# Draw a pretty confusion matrix using the custom helper function
prettyConfusion(results_nb)
```



Accuracy: 90.7331976
F1-Score: 0.0808081

Linear Discriminant Analysis

```
##---------Linear Discriminant Analysis-----------##
# train
ld <- train(personal_loan ~ ., data = train, method = "lda", family = binomial)

# predictions
pred_ld <- predict(ld, val)

# Accuracy of the model
(acc_ld <- mean(pred_ld == val$personal_loan))
```

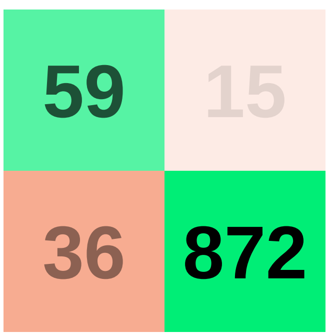
```
## [1] 0.9480652
```

```
# Generate Confusion Matrix of the model
results_ld <- confusionMatrix(pred_ld, val$personal_loan, positive="1")

# F1 Score
(f1_ld <- results_ld$byClass['F1'])
```

```
##        F1
## 0.6982249
```

```
# Draw a pretty confusion matrix using the custom helper function
prettyConfusion(results_ld)
```



Accuracy: 94.8065173
F1-Score: 0.6982249

Loess

```
##---------------------Loess-------------------##

# train
loess <- train(personal_loan ~ ., data = train, method = "gamLoess")
```

```
## Loading required package: gam

## Loading required package: splines

## Loading required package: foreach

##
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##
```

```
##     accumulate, when
```

```
## Loaded gam 1.16.1
```

```r
# predictions
pred_loess <- predict(loess, val)

# Accuracy of the model
(acc_loess <- mean(pred_loess == val$personal_loan))
```

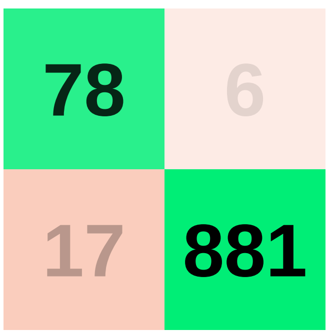
```
## [1] 0.9765784
```

```r
# Generate Confusion Matrix of the model
results_loess <- confusionMatrix(pred_loess, val$personal_loan, positive="1")

# F1 Score
(f1_loess <- results_loess$byClass['F1'])
```

```
##        F1
## 0.8715084
```

```r
# Draw a pretty confusion matrix using the custom helper function
prettyConfusion(results_loess)
```



Accuracy: 97.6578411
F1-Score: 0.8715084

Quadratic Discriminant Analysis

```r
##--------Quadratic Discriminant Analysis----------##

# train
qd <- train(personal_loan ~ ., data = train, method = "qda", family = binomial)

# predictions
pred_qd <- predict(qd, val)

# Accuracy of the model
(acc_qd <- mean(pred_qd == val$personal_loan))
```

```
## [1] 0.9429735
```

```r
# Generate Confusion Matrix of the model
results_qd <- confusionMatrix(pred_qd, val$personal_loan, positive="1")

# F1 Score
```

```r
(f1_qd <- results_qd$byClass['F1'])
```

```
##        F1
## 0.6818182
```

```r
# Draw a pretty confusion matrix using the custom helper function
prettyConfusion(results_qd)
```

Accuracy: 94.2973523
F1-Score: 0.6818182

Support Vector Machine

```r
##----------Support Vector Machine----------------##

# train
svm <- train(personal_loan ~ ., data = train, method = "svmLinear")

# predictions
pred_svm <- predict(loess, val)

# Accuracy of the model
(acc_svm <- mean(pred_svm == val$personal_loan))
```

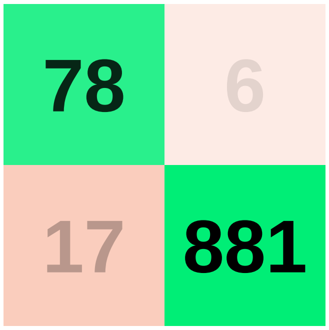
```
## [1] 0.9765784
```

```r
# Generate Confusion Matrix of the model
results_svm <- confusionMatrix(pred_svm, val$personal_loan, positive="1")

# F1 Score
(f1_svm <- results_svm$byClass['F1'])
```

```
##        F1
## 0.8715084
```

```r
# Draw a pretty confusion matrix using the custom helper function
prettyConfusion(results_svm)
```



Accuracy: 97.6578411
F1-Score: 0.8715084

K Nearest Neighbours Classification

```r
##--------------K Nearest Neighbours--------------##

# Setting seed for reproducibility of results
set.seed(7, sample.kind = "Rounding")

# k values to test best k
k_values <- data.frame(k = seq(2, 12, 1))

# train
knn <- train(personal_loan ~ ., data = train, method = "knn",
             tuneGrid = k_values)
# best k value
knn$bestTune
```

```
##   k
## 1 2
```

```r
# predictions
pred_knn <- predict(knn, val)

# Accuracy of the model
(acc_knn <- mean(pred_knn == val$personal_loan))
```

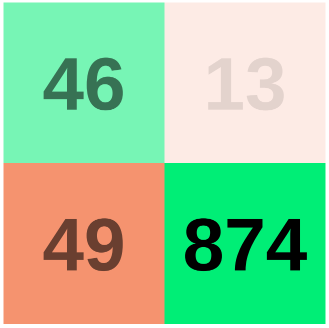
```
## [1] 0.9368635
```

```r
# Generate Confusion Matrix of the model
results_knn <- confusionMatrix(pred_knn, val$personal_loan, positive="1")

# F1 Score
(f1_knn <- results_knn$byClass['F1'])
```

```
##        F1
## 0.5974026
```

```r
# Draw a pretty confusion matrix using the custom helper function
prettyConfusion(results_knn)
```



The best k value is: 2
Accuracy: 93.6863544
F1-Score: 0.5974026

Random Forest Classification

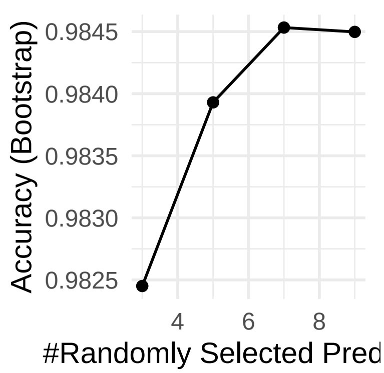
```
##----------------Random Forest-----------------##

# Setting seed for reproducibility of results
set.seed(7, sample.kind = "Rounding")

# Values of mtry
mtryGrid <- data.frame(mtry = c(3,5,7,9))

# train
rf <-  train(personal_loan ~ ., data = train, method = "rf",
             tuneGrid = mtryGrid, importance = T)

# Plot the values of accuracy for values of mtry
ggplot(rf) + theme_minimal()
```




```
# best model param
rf$bestTune
```

```
##   mtry
## 3    7
```

```
# predictions
pred_rf <- predict(rf, val)

# Accuracy of the model
```

```r
(acc_rf <- mean(pred_rf == val$personal_loan))
```

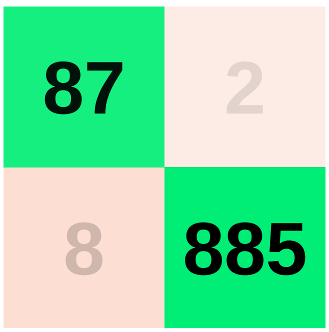
```
## [1] 0.9898167
```

```r
# Generate Confusion Matrix of the model
results_rf <- confusionMatrix(pred_rf, val$personal_loan, positive="1")

# F1 Score
(f1_rf <- results_rf$byClass['F1'])
```

```
##        F1
## 0.9456522
```

```r
# Draw a pretty confusion matrix using the custom helper function
prettyConfusion(results_rf)
```



The best value of mtry is: 7

Accuracy: 98.9816701
F1-Score: 0.9456522

```r
#Variable Importance
varImp(rf)
```

```
## rf variable importance
##
##                     Importance
## income                 100.000
## education3              69.072
## education2              63.508
## family4                 39.166
## family3                 38.807
## credit_card_spend       19.727
## family2                  6.986
## age                      5.080
## cd_account1              3.538
## credit_card1             1.847
## online1                  1.454
## mortgage                 1.002
## securities_account1      0.000
```

We note that income, family and credit card are most important variables. This was also confirmed in our exploratory data analysis section.

Ensemble Model (Voting Classifier)

```r
# votes are added up as total predictions from 6models that predict 1
votes <- (pred_rf == 1) + (pred_svm == 1) + (pred_ld == 1) + (pred_lr == 1) + (pred_loess == 1)

# Ensemble prediction is 1 if atleast three of the five models predict 1
pred_ensemble <- ifelse(votes >= 3, 1, 0)

# Accuradcy of the model
(acc_ensemble <- mean(pred_ensemble == val$personal_loan))
```

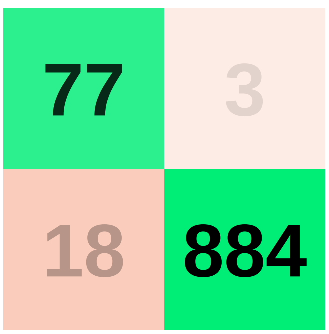
```
## [1] 0.9786151
```

```r
# Generate Confusion Matrix of the model
results_ensemble <- confusionMatrix(factor(pred_ensemble), val$personal_loan, positive="1")

# F1 Score
(f1_ensemble <- results_ensemble$byClass['F1'])
```

```
##   F1
## 0.88
```

```r
# Draw a pretty confusion matrix using the custom helper function
prettyConfusion(results_ensemble)
```



Accuracy: 97.8615071
F1-Score: 0.88

5.1 Validation Metrics

5.1.1 Validation Accuracies

```
#All models Validation Accuracies
models <- c("Logistic regression", "Naive Bayes Classifier",
            "LDA", "QDA", "Loess","K nearest neighbors",
            "Random forest", "Ensemble" , "Support Vector Machine")
# Accuracy list
accuracy <- c(acc_lr, acc_nb, acc_ld, acc_qd, acc_loess, acc_knn, acc_rf, acc_ensemble, acc_svm)

# Output the accuracies table
data.frame(Model = models, Validation_Accuracy = accuracy) %>%
  arrange(desc(Validation_Accuracy)) %>%
  knitr::kable()
```

Model	Validation_Accuracy
Random forest	0.9898167
Ensemble	0.9786151
Loess	0.9765784
Support Vector Machine	0.9765784
Logistic regression	0.9602851
LDA	0.9480652
QDA	0.9429735
K nearest neighbors	0.9368635
Naive Bayes Classifier	0.9073320

5.1.2 Validation F1-Scores

```
# All models Validation F1 Scores
f1 <- c(f1_lr, f1_nb, f1_ld, f1_qd, f1_loess, f1_knn, f1_rf, f1_ensemble, f1_svm)

# Output the f1 table
data.frame(Model = models, Validation_F1 = f1) %>%
  arrange(desc(Validation_F1)) %>%
  knitr::kable()
```

Model	Validation_F1
Random forest	0.9456522
Ensemble	0.8800000
Loess	0.8715084
Support Vector Machine	0.8715084
Logistic regression	0.7719298
LDA	0.6982249
QDA	0.6818182
K nearest neighbors	0.5974026
Naive Bayes Classifier	0.0808081

Model	Validation_F1

The best performing models on validation set are:

```
1. Random Forest
2. Ensemble Voting Classifier
3. Loess
4. Support Vector Machine
5. Logistic Regression Classifier
6. Linear Discriminant Analysis
```

6 Model Evaluation

We generate the predictions for the test set using our best models selected using validation F1-Scores:

```r
# Logistic Regression test set predictions
test_lr <- predict(lr, test)

# Confusion matrix for the test predictions of lr model
test_results_lr <- confusionMatrix(test_lr, test$personal_loan, positive="1")

# F1 Score
(test_f1_lr <- test_results_lr$byClass['F1'])
```

```
##        F1
## 0.7816092
```

```r
# LDA test set predictions
test_ld <- predict(ld, test)

# Confusion matrix for the test predictions of lda model
test_results_ld <- confusionMatrix(test_ld, test$personal_loan, positive="1")

# F1 Score
(test_f1_ld <- test_results_ld$byClass['F1'])
```

```
##        F1
## 0.6936416
```

```r
# Loess test set predictions
test_loess <- predict(loess, test)

# Confusion matrix for the test predictions of loess model
test_results_loess <- confusionMatrix(test_loess, test$personal_loan, positive="1")

# F1 Score
(test_f1_loess <- test_results_loess$byClass['F1'])
```

```
##        F1
## 0.8342246
```

```r
# SVM test set predictions
test_svm <- predict(svm, test)
```

```r
# Confusion matrix for the test predictions of svm model
test_results_svm <- confusionMatrix(test_svm, test$personal_loan, positive="1")

# F1 Score
(test_f1_svm <- test_results_svm$byClass['F1'])
```

```
##        F1
## 0.7664671
```

```r
# Random Forest test set predictions
test_rf <- predict(rf, test)

# Confusion matrix for the test predictions of random forest model
test_results_rf <- confusionMatrix(test_rf, test$personal_loan, positive="1")

# F1 Score
(test_f1_rf <- test_results_rf$byClass['F1'])
```

```
##        F1
## 0.9109948
```

```r
# votes are added up as total predictions from 6models that predict 1
test_votes <- (test_rf == 1) + (test_loess == 1) + (test_svm == 1) + (test_lr == 1) + (test_ld == 1)

# Ensemble test set predictions
test_ensemble <- ifelse(test_votes >= 3, 1, 0)

# Confusion matrix for the test predictions of ensemble model
test_results_ensemble <- confusionMatrix(factor(test_ensemble), test$personal_loan, positive="1")

# F1 Score
(test_f1_ensemble <- test_results_ensemble$byClass['F1'])
```

```
##        F1
## 0.8023256
```

7 Results

Here we obatin the results for the best models on the test set as follows:

7.1 Testing Accuracies

```r
# Selected models
test_models <- c("Random Forest", "Loess" , "Support Vector Machine",
                 "Logistic Regression", "Linear Discriminatn Analyis", "Ensemble")

# Calculate Accuracies for the test set
test_accuracy <- c(mean(test_rf == test$personal_loan),
                   mean(test_loess == test$personal_loan),
                   mean(test_svm == test$personal_loan),
                   mean(test_lr == test$personal_loan),
                   mean(test_ld == test$personal_loan),
                   mean(test_ensemble == test$personal_loan))

# Output the testing accuracies table
data.frame(Model = test_models, Testing_Accuracy = test_accuracy) %>%
  arrange(desc(Testing_Accuracy)) %>%
  knitr::kable()
```

Model	Testing_Accuracy
Random Forest	0.9831014
Loess	0.9691849
Ensemble	0.9662028
Logistic Regression	0.9622266
Support Vector Machine	0.9612326
Linear Discriminatn Analyis	0.9473161

7.2 Testing F1-Scores

```r
# All models Validation F1 Scores
test_f1 <- c(test_f1_rf, test_f1_loess, test_f1_svm,
             test_f1_lr, test_f1_ld, test_f1_ensemble)

# Output the f1 table
data.frame(Model = test_models, Testing_F1 = test_f1) %>%
  arrange(desc(Testing_F1)) %>%
  knitr::kable()
```

Model	Testing_F1
Random Forest	0.9109948
Loess	0.8342246
Ensemble	0.8023256
Logistic Regression	0.7816092
Support Vector Machine	0.7664671
Linear Discriminatn Analyis	0.6936416

8 Conclusion

Finally, we conclude that the best model for this project was the random forest model(7). It had a testing F1-Score of 0.911 and Testing Accuracy 98.31%.
Further scope for this project would be to incorporate neural networks and gradient boosting models which might improve on our models, however, given the time taken to train and tune neural networks. We have left it to the future. Also, we could do some feature engineering to find out the main features and if possible, extract new features from the domain.

This analysis could've been better if we get more data from Bank of India so that the we can be sure that the models are not over-fitting to this specific dataset and generalize better to new or unseeen data.