EDX Final Project

Dario Taba

31/05/2021

1 Introduction

One of the most widely used banking products is credit cards. It is issued by a bank or financial institution that authorizes the person to use it as a means of payment. It is another form of financing; therefore, the user must assume the obligation to return the amount provided and to pay interest, bank fees and expenses. The problem that we will be addressing is related to inactivity and subsequent abandonment of the product. For this, we will use the dataset provided by the Kaggle platform. (https://www.kaggle.com/sakshigoyal7/credit-card-customers)

The project will consist of creating a predictive model to estimate the situation (Churned or not Churned) of the client, so that the company can then proactively communicate with the client and be able to improve and / or change the offer of products and services and be able to reverse their situation .

All files used for this project are in: (https://github.com/damtaba/EDX-Final-Project)

1.1 Data Loading

The base that will be used is the one found on the Kaggle website. Once downloaded, it will be imported to be able to use it.

```
#Importing de csv downloaded from https://www.kaggle.com/sakshigoyal7/credit-card-customers.
#It will also be provided in my GitHub repo(https://github.com/damtaba/EDX-Final-Project) along with th

data<- read.csv(file="BankChurners.csv")


#There are two variables that were calculated later and are not part of the original base. We'll remove
data$Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Educatio
data$Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Educatio

str(data)
```

```
## 'data.frame':    10127 obs. of  21 variables:
##  $ CLIENTNUM                : int  768805383 818770008 713982108 769911858 709106358 713061558 810347
##  $ Attrition_Flag           : Factor w/ 2 levels "Attrited Customer",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ Customer_Age             : int  45 49 51 40 40 44 51 32 37 48 ...
##  $ Gender                   : Factor w/ 2 levels "F","M": 2 1 2 1 2 2 2 2 2 2 ...
##  $ Dependent_count          : int  3 5 3 4 3 2 4 0 3 2 ...
##  $ Education_Level          : Factor w/ 7 levels "College","Doctorate",..: 4 3 3 4 6 3 7 4 6 3 ...
##  $ Marital_Status           : Factor w/ 4 levels "Divorced","Married",..: 2 3 2 4 2 2 2 4 3 3 ...
```

```
##  $ Income_Category         : Factor w/ 6 levels "$120K +","$40K - $60K",..: 3 5 4 5 3 2 1 3 3 4 ...
##  $ Card_Category           : Factor w/ 4 levels "Blue","Gold",..: 1 1 1 1 1 1 2 4 1 1 ...
##  $ Months_on_book          : int  39 44 36 34 21 36 46 27 36 36 ...
##  $ Total_Relationship_Count: int  5 6 4 3 5 3 6 2 5 6 ...
##  $ Months_Inactive_12_mon  : int  1 1 1 4 1 1 1 2 2 3 ...
##  $ Contacts_Count_12_mon   : int  3 2 0 1 0 2 3 2 0 3 ...
##  $ Credit_Limit            : num  12691 8256 3418 3313 4716 ...
##  $ Total_Revolving_Bal     : int  777 864 0 2517 0 1247 2264 1396 2517 1677 ...
##  $ Avg_Open_To_Buy         : num  11914 7392 3418 796 4716 ...
##  $ Total_Amt_Chng_Q4_Q1    : num  1.33 1.54 2.59 1.4 2.17 ...
##  $ Total_Trans_Amt         : int  1144 1291 1887 1171 816 1088 1330 1538 1350 1441 ...
##  $ Total_Trans_Ct          : int  42 33 20 20 28 24 31 36 24 32 ...
##  $ Total_Ct_Chng_Q4_Q1     : num  1.62 3.71 2.33 2.33 2.5 ...
##  $ Avg_Utilization_Ratio   : num  0.061 0.105 0 0.76 0 0.311 0.066 0.048 0.113 0.144 ...
```

The database has 10,127 records (each one represented by a client) and 21 variables, on which the assembly of the model will be based.

1.2 Packages Loading

The following packages will be used for the rest of the project.

```
# Package names we'll need
packages <- c("tidyverse",
              "rmarkdown",
              "fastDummies",
              "caret",
              "fastDummies",
              "knitr",
              "gmodels",
              "C50",
              "tinytex"
              )


# Install packages not yet installed
installed_packages <- packages %in% rownames(installed.packages())
if (any(installed_packages == FALSE)) {
  install.packages(packages[!installed_packages])
}

# Packages loading
invisible(lapply(packages, library, character.only = TRUE))
```

2 Methods and Analysis

Next, the analysis of the available variables.

2.1 Data Analysis

2.1.1 CLIENTNUM

CLIENTNUM is in itself the identification with which each client has.

2.1.2 Attrition_Flag

Our variable target. It is the one that tells if the client is churned or not.

```
data %>%
  group_by(Attrition_Flag) %>%
  summarize(n=n()) %>%
  mutate(Participation=n/sum(n)) %>%
  kable()
```

Attrition_Flag	n	Participation
Attrited Customer	1627	0.1606596
Existing Customer	8500	0.8393404

Attrited Customer refers to a lost customer, contrary to Existing Customer. We can see a bias in the information in both the proportion of cases with lost clients and existing ones; 83% vs. 4% .

2.1.3 Customer_Age

A summary of the feature

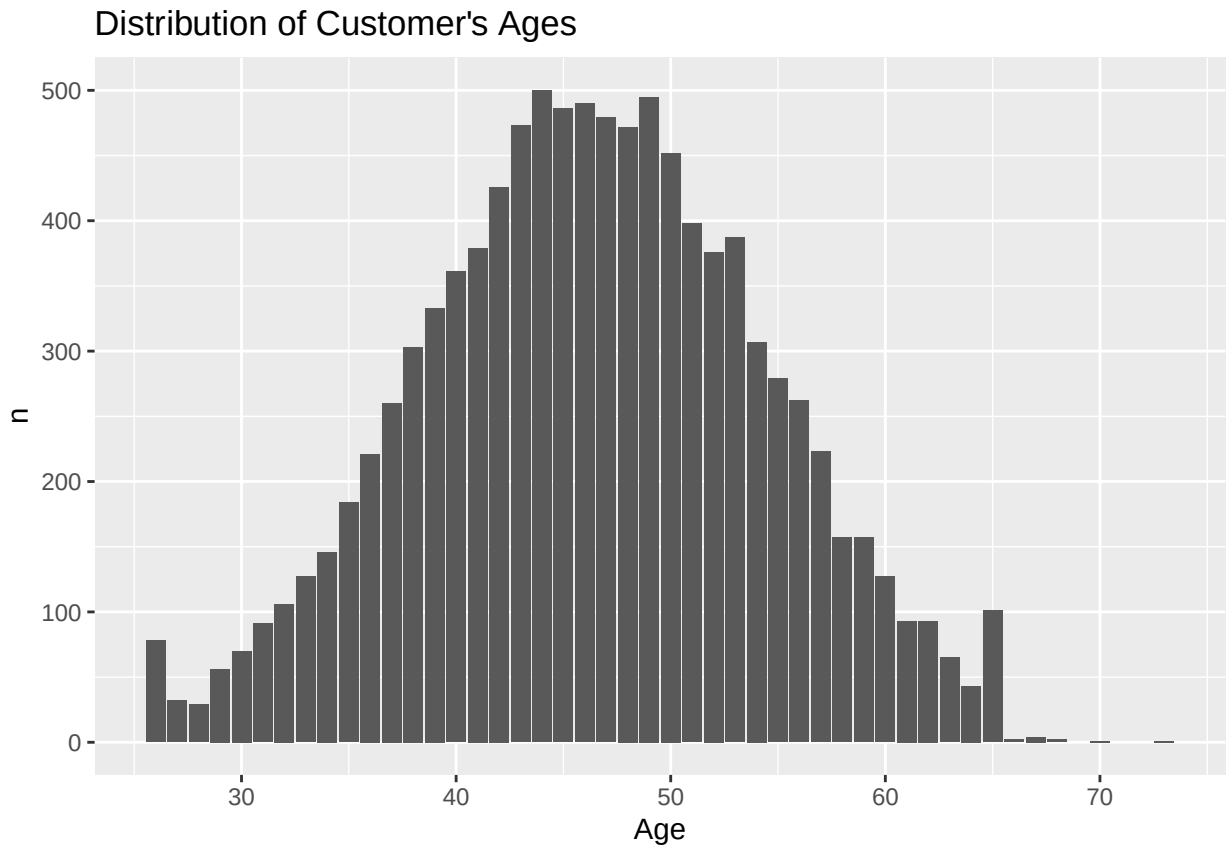
```
summary(data$Customer_Age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   26.00   41.00   46.00   46.33   52.00   73.00
```

It can be seen that there is not a great difference between the median and the mean, so we can assume that the data do not present a great dispersion; half of the records are at age 46, similar to the average age of the clients, 46.33

Graphically, the age distribution can be anticipated to resemble a normal distribution.

```
data %>%
  ggplot(aes(x=Customer_Age)) +
  geom_bar() +
  ggtitle("Distribution of Customer's Ages") +
  xlab("Age") +
  ylab("n")
```



To make sure, thanks to the Shapiro-Wilk test (used to test the normality of a data set) we can ensure that it is most likely a normal distribution (this is said in Royston (1995) to be adequate for p.value < 0.1.)

```
shapiro.test(sample(data$Customer_Age,5000))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  sample(data$Customer_Age, 5000)
## W = 0.99626, p-value = 6.246e-10
```

2.1.4 Gender

Cross tables will be used in the analysis of several variables. At each intersection, in order, provide the following information: 1) Intersection amount, 2) Group share of row total, 3) Column total share, and 4) Grand total share

```
#At first glance it does not seem that gender can be a good predictor; is almost evenly distributed
CrossTable(data$Gender,data$Attrition_Flag,prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
```

```
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  10127
##
##
##              | data$Attrition_Flag
##  data$Gender | Attrited Customer | Existing Customer |         Row Total |
## -------------|-------------------|-------------------|-------------------|
##            F |               930 |              4428 |              5358 |
##              |             0.174 |             0.826 |             0.529 |
##              |             0.572 |             0.521 |                   |
##              |             0.092 |             0.437 |                   |
## -------------|-------------------|-------------------|-------------------|
##            M |               697 |              4072 |              4769 |
##              |             0.146 |             0.854 |             0.471 |
##              |             0.428 |             0.479 |                   |
##              |             0.069 |             0.402 |                   |
## -------------|-------------------|-------------------|-------------------|
## Column Total |              1627 |              8500 |             10127 |
##              |             0.161 |             0.839 |                   |
## -------------|-------------------|-------------------|-------------------|
##
##
```

At first glance it does not seem that gender can be a good predictor; is almost evenly distributed

2.1.5 Dependent_count

Refers to the number of close family members. A summary of the feature

```
summary(data$Customer_Age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   26.00   41.00   46.00   46.33   52.00   73.00
```

Looking at the participation of the lost clients within each group of number of close relatives, no group is observed especially prone to becoming an inactive client.

```
CrossTable(data$Dependent_count,data$Attrition_Flag,prop.chisq =  FALSE,prop.t = FALSE,prop.c = FALSE)
```

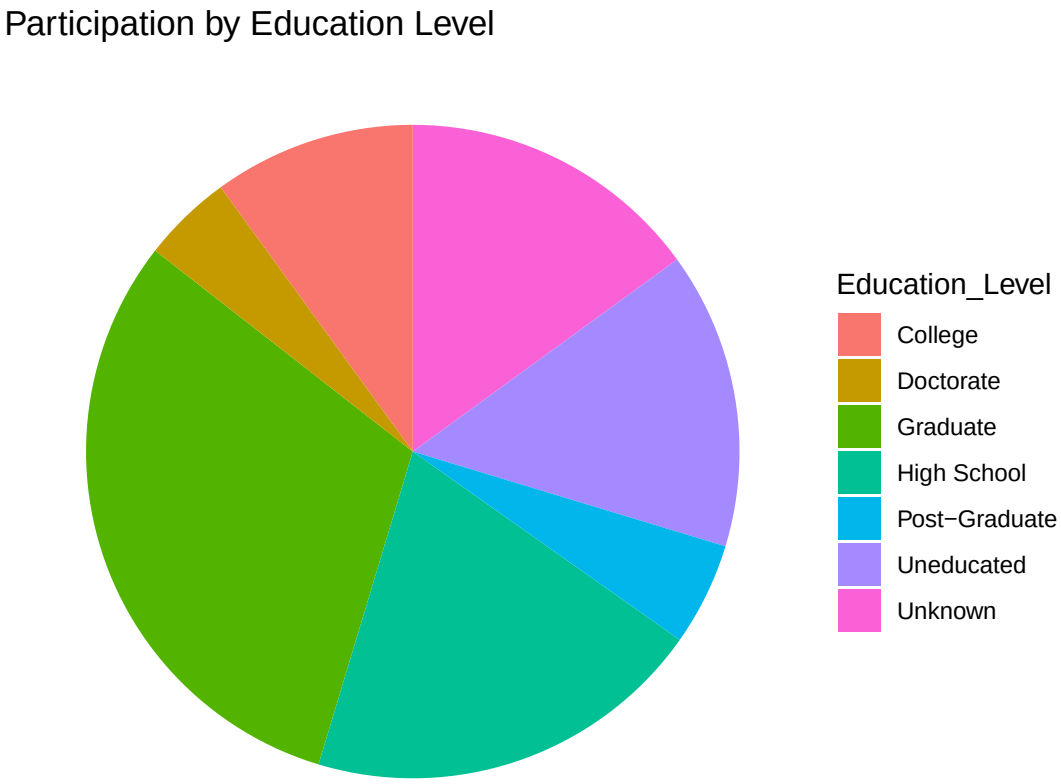
```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
```

```
## |-------------------------|
##
##
## Total Observations in Table:  10127
##
##
##                       | data$Attrition_Flag
## data$Dependent_count  | Attrited Customer | Existing Customer |         Row Total |
## ----------------------|-------------------|-------------------|-------------------|
##                     0 |               135 |               769 |               904 |
##                       |             0.149 |             0.851 |             0.089 |
## ----------------------|-------------------|-------------------|-------------------|
##                     1 |               269 |              1569 |              1838 |
##                       |             0.146 |             0.854 |             0.181 |
## ----------------------|-------------------|-------------------|-------------------|
##                     2 |               417 |              2238 |              2655 |
##                       |             0.157 |             0.843 |             0.262 |
## ----------------------|-------------------|-------------------|-------------------|
##                     3 |               482 |              2250 |              2732 |
##                       |             0.176 |             0.824 |             0.270 |
## ----------------------|-------------------|-------------------|-------------------|
##                     4 |               260 |              1314 |              1574 |
##                       |             0.165 |             0.835 |             0.155 |
## ----------------------|-------------------|-------------------|-------------------|
##                     5 |                64 |               360 |               424 |
##                       |             0.151 |             0.849 |             0.042 |
## ----------------------|-------------------|-------------------|-------------------|
##          Column Total |              1627 |              8500 |             10127 |
## ----------------------|-------------------|-------------------|-------------------|
##
##
```

2.1.6 Education_Level

We can see that most of the clients that we identify in the dataset are graduates and / or finished high school (almost 50% of the dataset).

```
data %>%
  group_by(Education_Level) %>%
  summarize(Q=n()) %>%
  mutate(porc=Q/sum(Q)) %>%
  arrange(desc(porc)) %>%
  ggplot(aes(x="",y=Q,fill=Education_Level)) +
  geom_bar(stat="identity", width=1)+
  coord_polar("y", start=0)+
  theme_void()+
  ggtitle("Participation by Education Level")
```




```
data %>%
  group_by(Education_Level) %>%
  summarize(Q=n()) %>%
  mutate(Participation=Q/sum(Q)) %>%
  arrange(desc(Participation)) %>%
  kable()
```

Education_Level	Q	Participation
Graduate	3128	0.3088773
High School	2013	0.1987756
Unknown	1519	0.1499951
Uneducated	1487	0.1468352
College	1013	0.1000296
Post-Graduate	516	0.0509529
Doctorate	451	0.0445344

2.1.7 Marital_Status

Almost half of the dataset corresponds to married

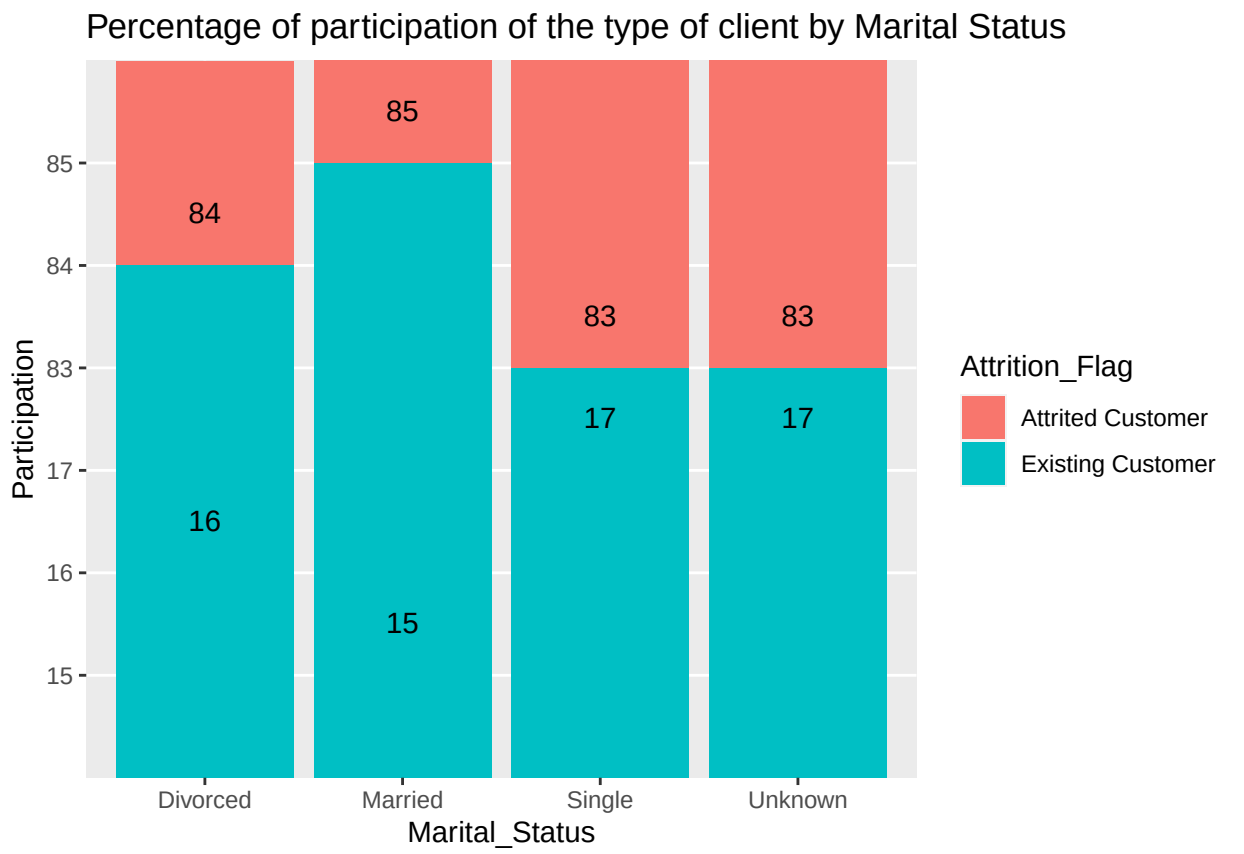
```
data %>%
  group_by(Marital_Status) %>%
  summarize(n=n()) %>%
```

```
  ungroup() %>%
  mutate(Participation=n/sum(n)) %>%
  arrange(desc(Participation)) %>%
  kable()
```

Marital_Status	n	Participation
Married	4687	0.4628222
Single	3943	0.3893552
Unknown	749	0.0739607
Divorced	748	0.0738620

We can see that regardless of the marital_status, the proportion of each type of client is very similar among all

```
data %>%
  group_by(Marital_Status,Attrition_Flag) %>%
  summarize(n=n()) %>%
  mutate(Participation=format(round(n/sum(n)*100), 2), nsmall = 2) %>%
  ggplot(aes(x=Marital_Status,y=Participation,fill=Attrition_Flag))+
  geom_bar(stat = "identity")+
  geom_text(aes(x=Marital_Status,y=Participation,label=Participation,vjust=-2))+
  ggtitle("Percentage of participation of the type of client by Marital Status")
```



2.1.8 Income_Category

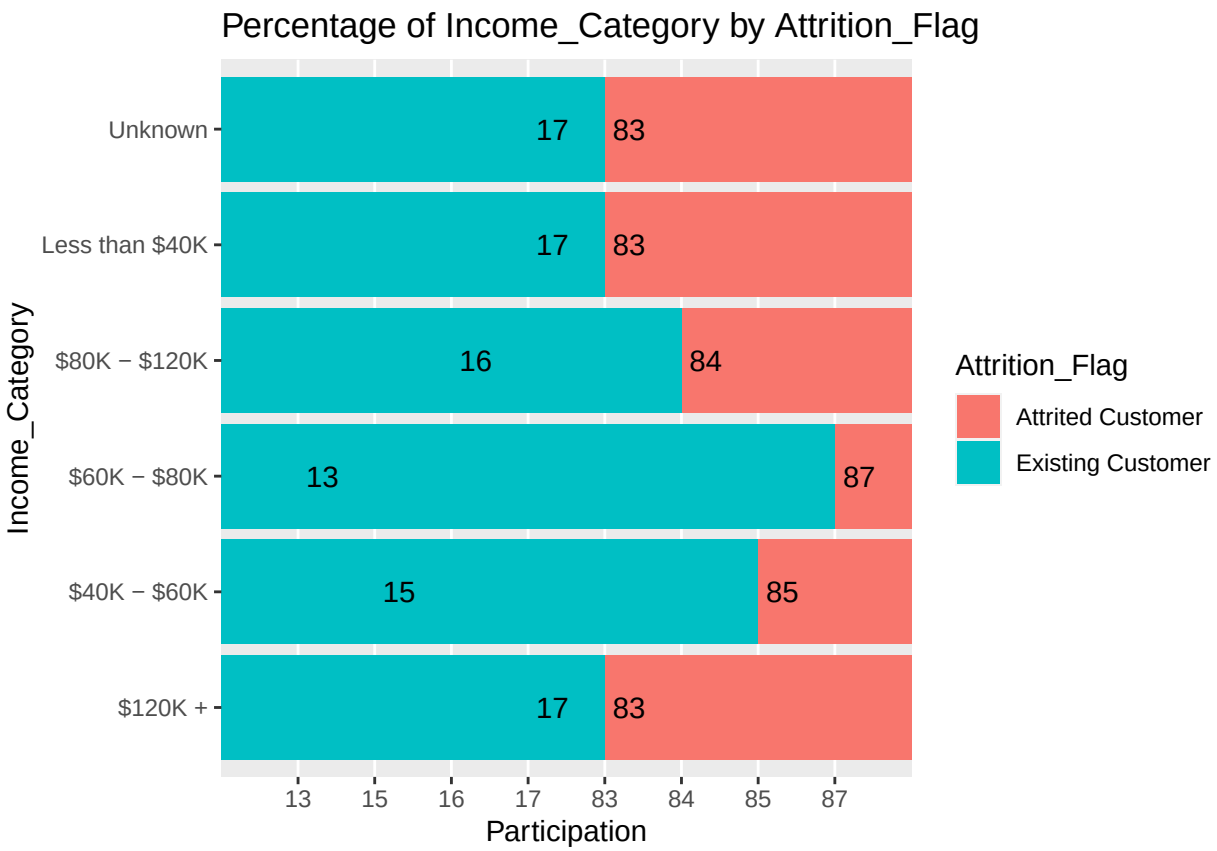
The mode is “Less than $40K”, with 3561 observations

```
data %>%
  group_by(Income_Category) %>%
  summarize(n=n()) %>%
  ungroup() %>%
  mutate(Participation=n/sum(n)) %>%
  arrange(desc(Participation)) %>%
  kable()
```

Income_Category	n	Participation
Less than $40K	3561	0.3516342
$40K - $60K	1790	0.1767552
$80K - $120K	1535	0.1515750
$60K - $80K	1402	0.1384418
Unknown	1112	0.1098055
$120K +	727	0.0717883

The distribution according to the income category.

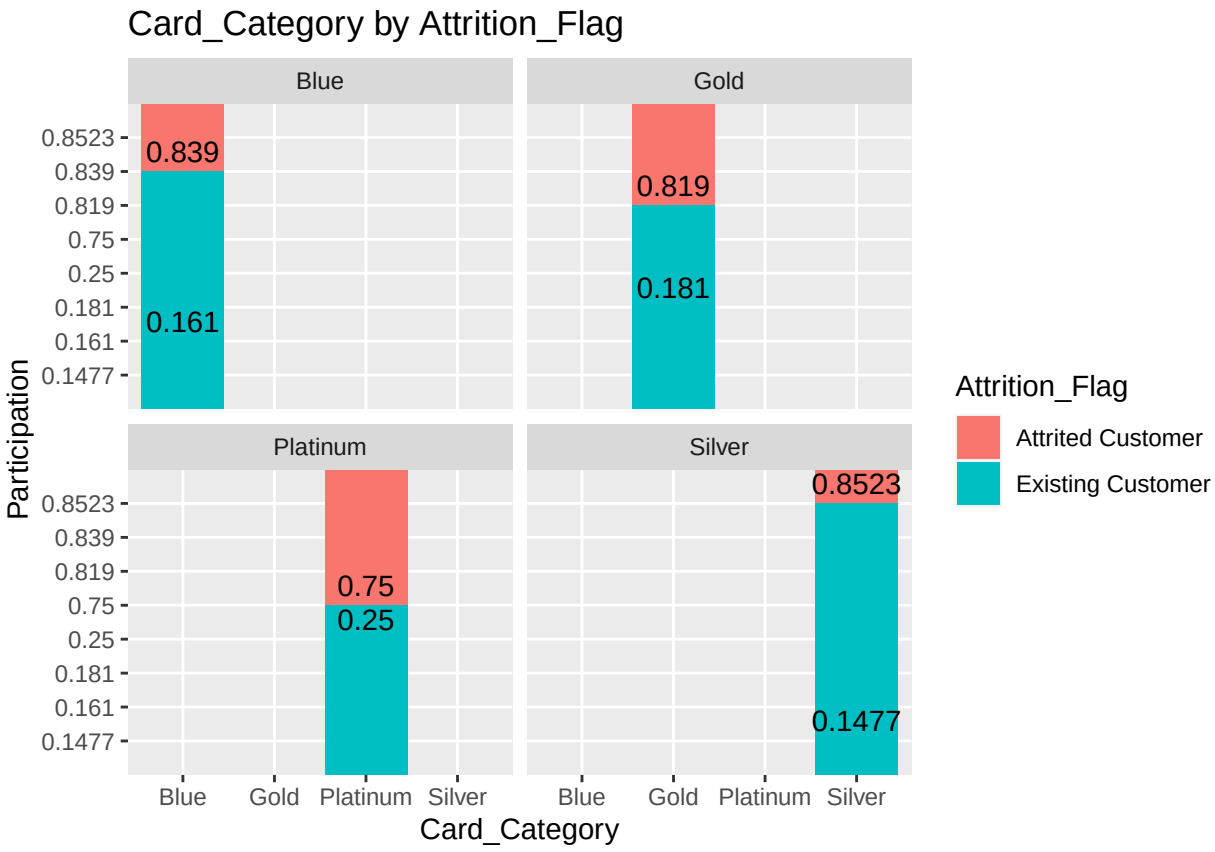
```
data %>%
  group_by(Attrition_Flag,Income_Category) %>%
  summarize(Q=n()) %>%
  ungroup() %>% group_by(Income_Category) %>%
  mutate(Participation=Q/sum(Q)) %>%
  mutate(Participation=format(round(Participation*100,digits = 0),digits = 4)) %>%
  ggplot(aes(x=Income_Category, y = Participation, fill = Attrition_Flag))+
  geom_bar(stat="identity")+
  coord_flip()+
  ggtitle("Percentage of Income_Category by Attrition_Flag")+
  geom_text(aes(x=Income_Category, y = Participation,label=Participation),hjust=-0.25)
```



2.1.9 Card_Category

Graphs of participation of the type of client by the category of the card. You can see in this case some with different composition. The implication of this should be discussed later, because not all types of credit cards carry the same weight; some there are very few cases.

```
data %>%
  group_by(Card_Category,Attrition_Flag) %>%
  summarize(Q=n()) %>%
  mutate(Participation=Q/sum(Q)) %>%
  mutate(Participation=format(round(Participation,digits = 4),digits = 4)) %>%
  ggplot(aes(x=Card_Category,y=Participation,fill=Attrition_Flag))+
  geom_col()+
  ggtitle("Card_Category by Attrition_Flag")+
  geom_text(aes(x=Card_Category, y = Participation,label=Participation,vjust=-0.45))+
  facet_wrap(~ Card_Category)
```




```
data %>%
  group_by(Card_Category) %>%
  summarize(Q=n()) %>%
  mutate(Participation=Q/sum(Q)) %>%
  kable()
```

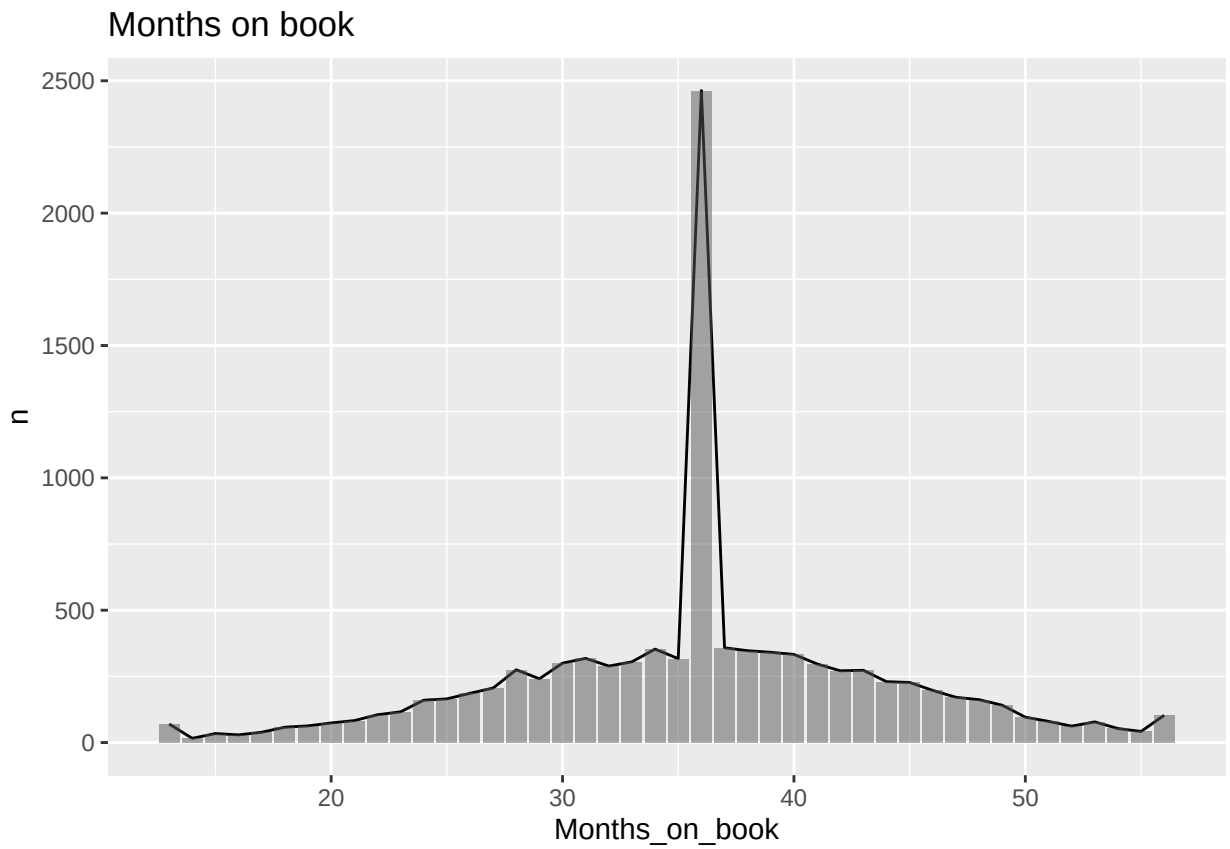
Card_Category	Q	Participation
Blue	9436	0.9317666
Gold	116	0.0114545
Platinum	20	0.0019749
Silver	555	0.0548040

2.1.10 Months_on_book

It refers to the periods that the client has been related to the bank for banking products. A visible characteristic of this variable is its kurtosis. The kurtosis is a statistical measure that determines the degree of concentration that the values of a variable present around the central zone of the frequency distribution. It can be seen that it is a leptokurtic: there is a great concentration of the values around its mean.

```
data %>%
  group_by(Months_on_book) %>%
  summarize(n=n()) %>%
  arrange(desc(Months_on_book)) %>%
```

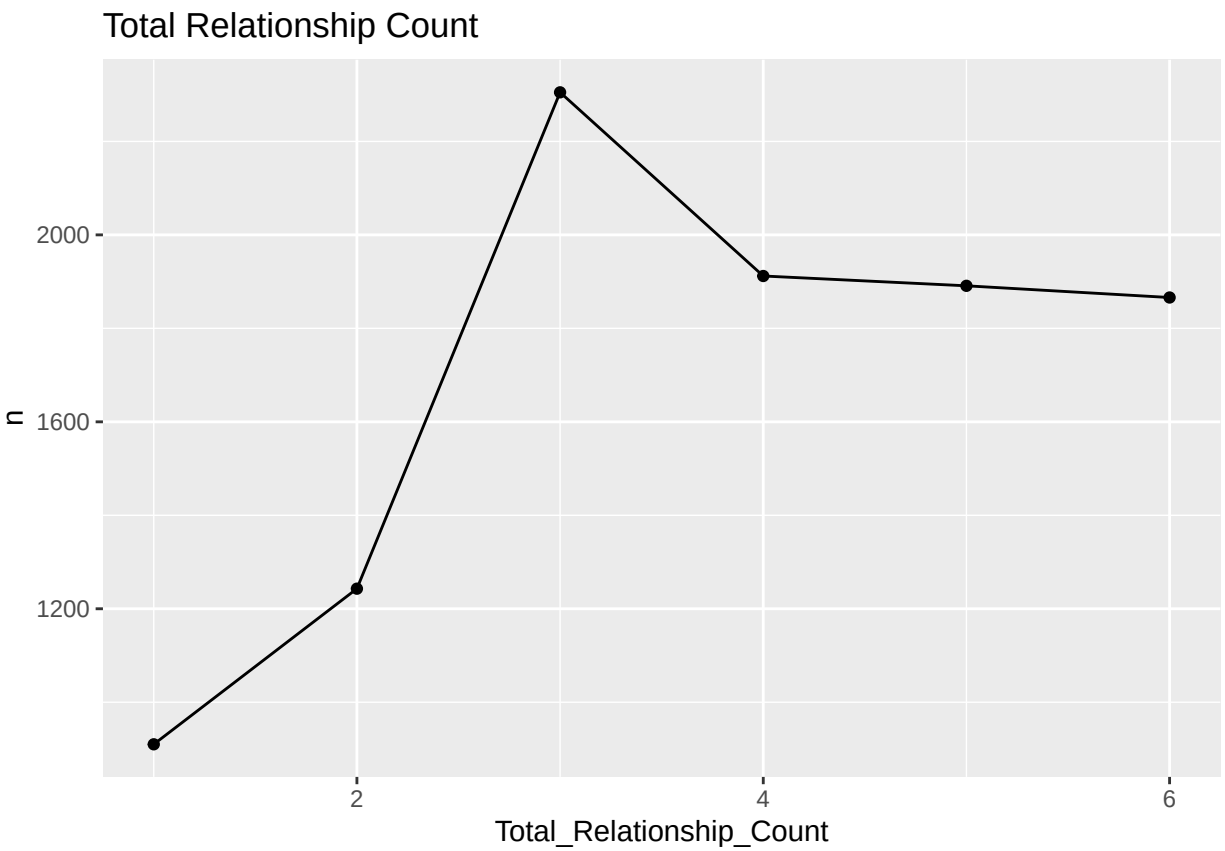
```
  ggplot(aes(x=Months_on_book,y=n)) +
  geom_line()+
  geom_col(alpha=0.5)+
  ggtitle("Months on book")
```



2.1.11 Total_Relationship_Count

Total no. of products held by the customer. It is to be expected that there will be a more common number of products that a customer owns. In the case of this dataset, 3 appears to be the dominant number of products a customer has.

```
data %>%
  group_by(Total_Relationship_Count) %>%
  summarize(n=n()) %>%
  ggplot(aes(Total_Relationship_Count,n))+
  geom_point()+
  geom_line()+
  ggtitle("Total Relationship Count")
```



2.1.12 Months_Inactive_12_m

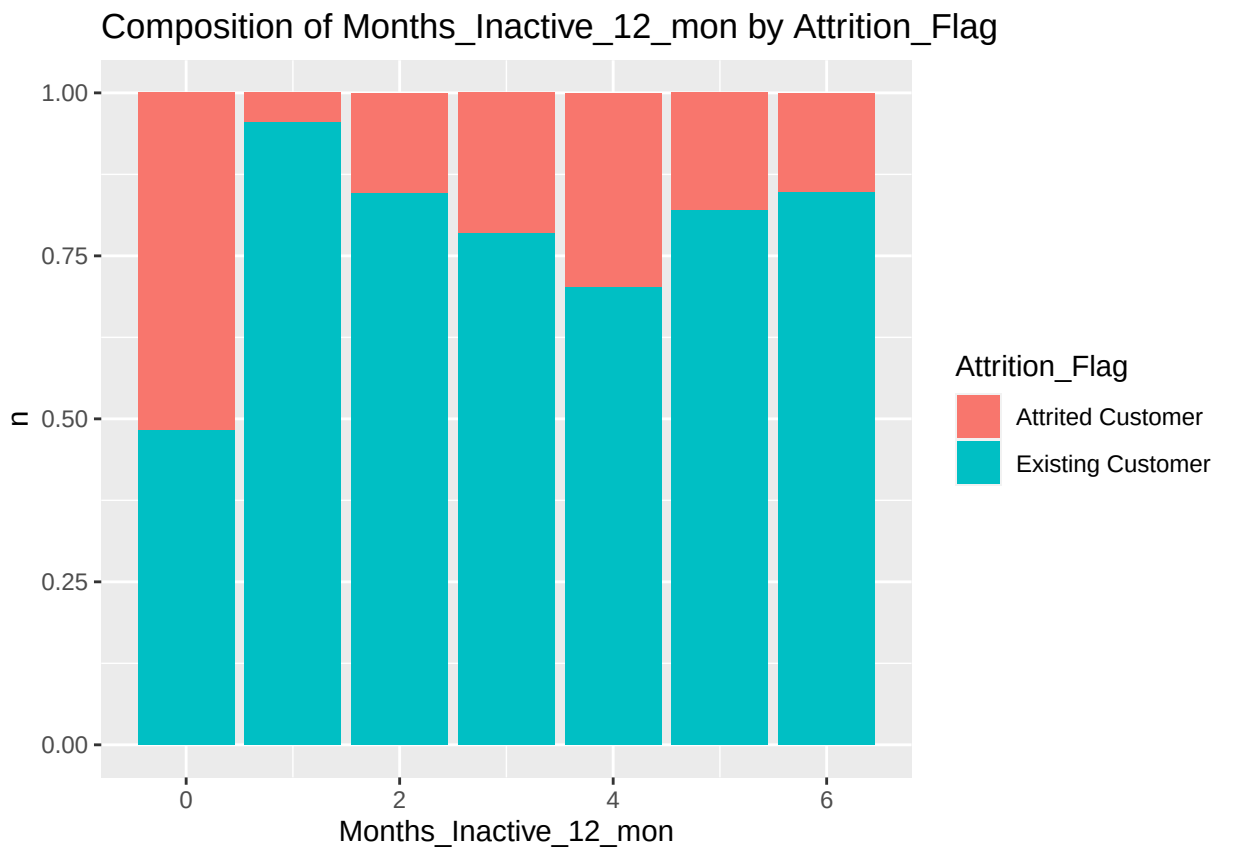
No. of months inactive in the last 12 months. Intuitively it can be predicted that this will be a strong variable. It can be seen how as the inactive months increase, the composition by type of customer also changes. The extremes of the graph should not be taken literally, as they are strata with few records.

```
data %>%
  group_by(Months_Inactive_12_mon) %>%
  summarize(n=n()) %>%
  kable()
```

Months_Inactive_12_mon	n
0	29
1	2233
2	3282
3	3846
4	435
5	178
6	124

```
data %>%
  group_by(Months_Inactive_12_mon,Attrition_Flag) %>%
  summarize(n=n()) %>%
```

```
  ggplot(aes(Months_Inactive_12_mon,n,fill=Attrition_Flag))+
  geom_col(position="fill")+
  ggtitle("Composition of Months_Inactive_12_mon by Attrition_Flag")
```



2.1.13 Contacts_Count_12_mon

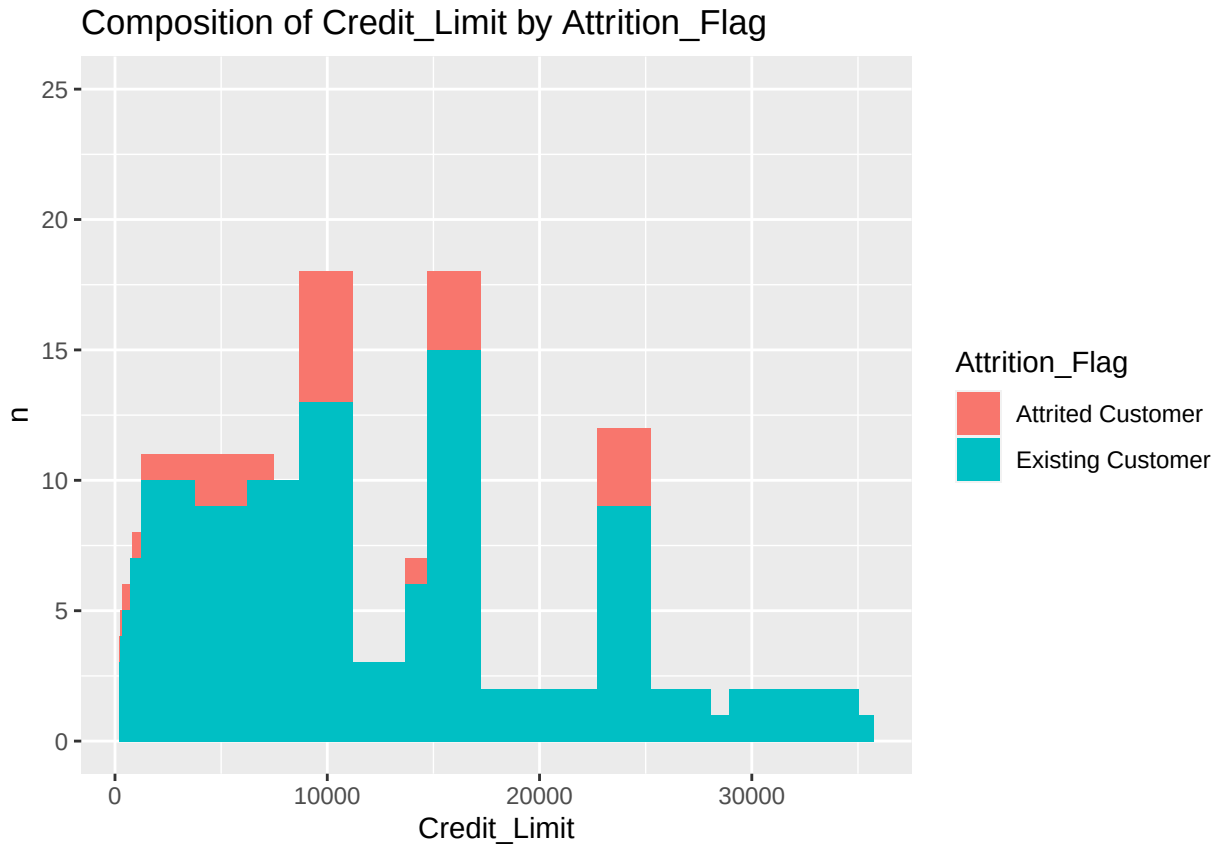
```
data %>%
  group_by(Contacts_Count_12_mon) %>%
  summarize(n=n()) %>%
  mutate(Participation=n/sum(n)) %>%
  arrange(Contacts_Count_12_mon) %>%
  kable()
```

Contacts_Count_12_mon	n	Participation
0	399	0.0393996
1	1499	0.1480201
2	3227	0.3186531
3	3380	0.3337612
4	1392	0.1374543
5	176	0.0173793
6	54	0.0053323

2.1.14 Credit_Limit

It seems that the credit card limit has some relevance in determining whether it is going to be a lost customer. The number of lost customers is more concentrated for the lower limits

```
data %>%
  group_by(Attrition_Flag,Credit_Limit) %>%
  summarize(n=n()) %>%
  ggplot(aes(x=Credit_Limit,y=n,fill=Attrition_Flag)) +
  geom_col(width = 2500)+
  ylim(0,25)+
  ggtitle("Composition of Credit_Limit by Attrition_Flag")
```



2.1.15 Total_Revolving_Bal

Refers to the total unpaid balance.

```
summary(data$Total_Revolving_Bal)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0     359    1276    1163    1784    2517
```

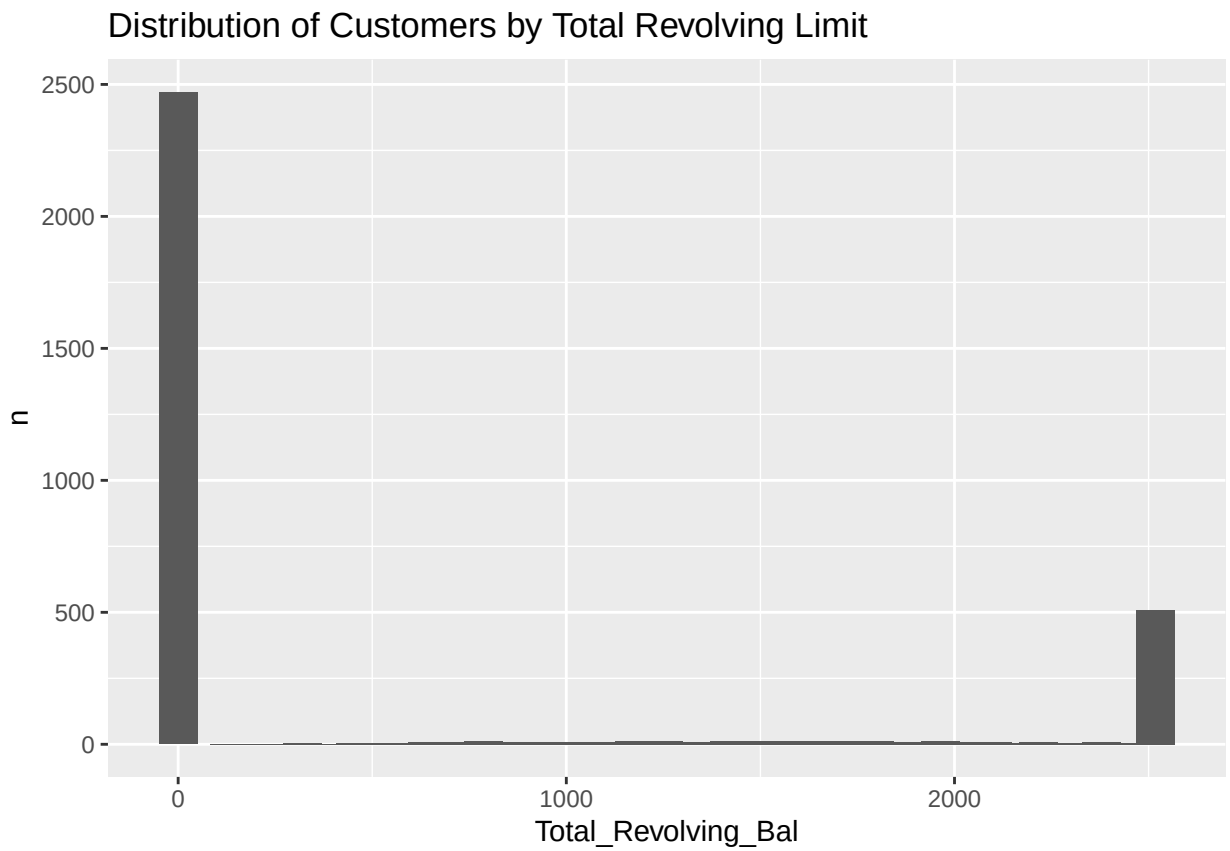
The most common is that it does not have a balance as debt, being 25% of the total dataset.

```
data %>%
  group_by(Total_Revolving_Bal) %>%
  summarize(n=n()) %>%
  mutate(porc=n/sum(n)) %>%
  arrange(desc(porc))
```

```
## # A tibble: 1,974 x 3
##    Total_Revolving_Bal     n     porc
##                  <int> <int>    <dbl>
##  1                   0  2470 0.244
##  2                2517   508 0.0502
##  3                1480    12 0.00118
##  4                1965    12 0.00118
##  5                1434    11 0.00109
##  6                1664    11 0.00109
##  7                1720    11 0.00109
##  8                 787    10 0.000987
##  9                1175    10 0.000987
## 10                1176    10 0.000987
## # ... with 1,964 more rows
```

It can be seen that the distribution concentrates at the extremes

```
data %>%
  group_by(Total_Revolving_Bal) %>%
  summarize(n=n()) %>%
  ggplot(aes(x=Total_Revolving_Bal,y=n))+
  geom_col(width = 100)+
  ggtitle("Distribution of Customers by Total Revolving Limit")
```

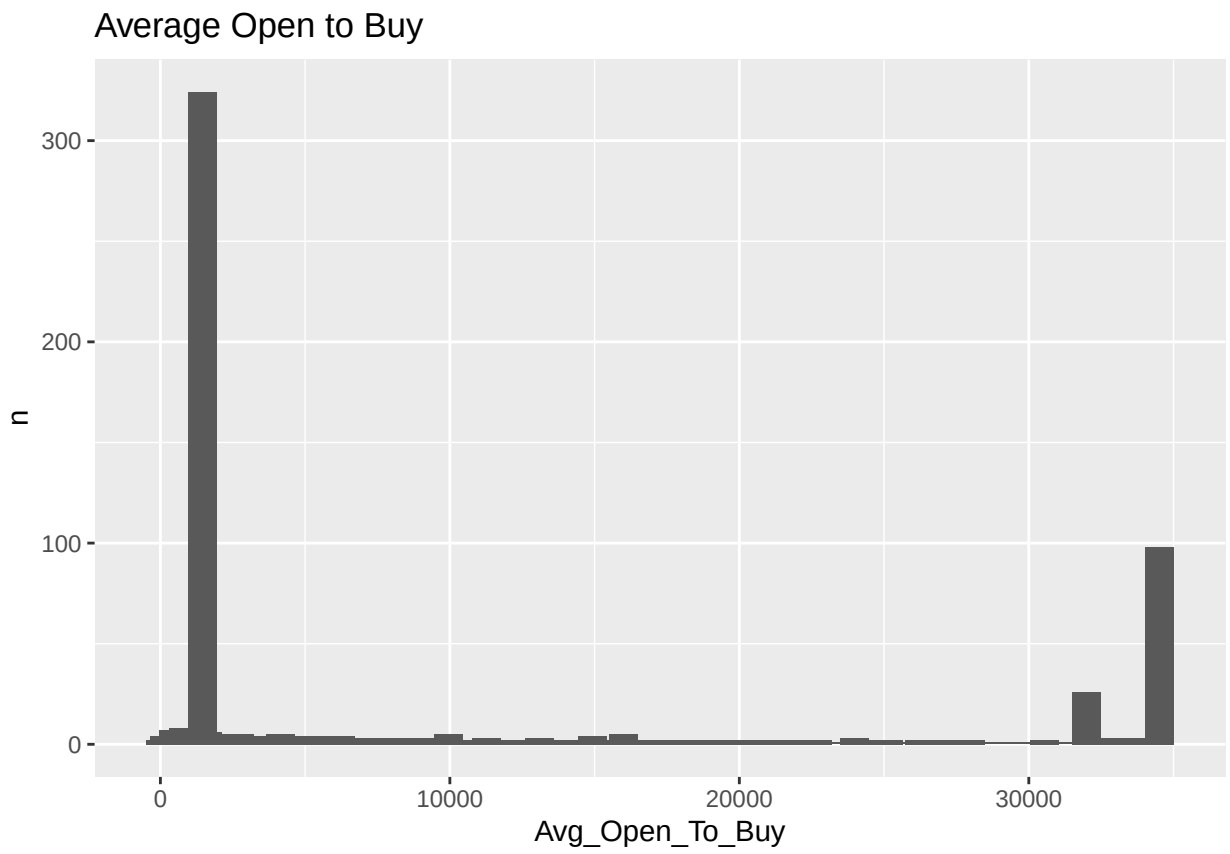



```
#  kable()
```

2.1.16 Avg_Open_To_Buy

A notable characteristic of this variable is its positive asymmetry. Skewness is the measure that indicates the symmetry of the distribution of a variable with respect to the arithmetic mean. In this case, the distribution is stretched (to the right) for values above the mean.

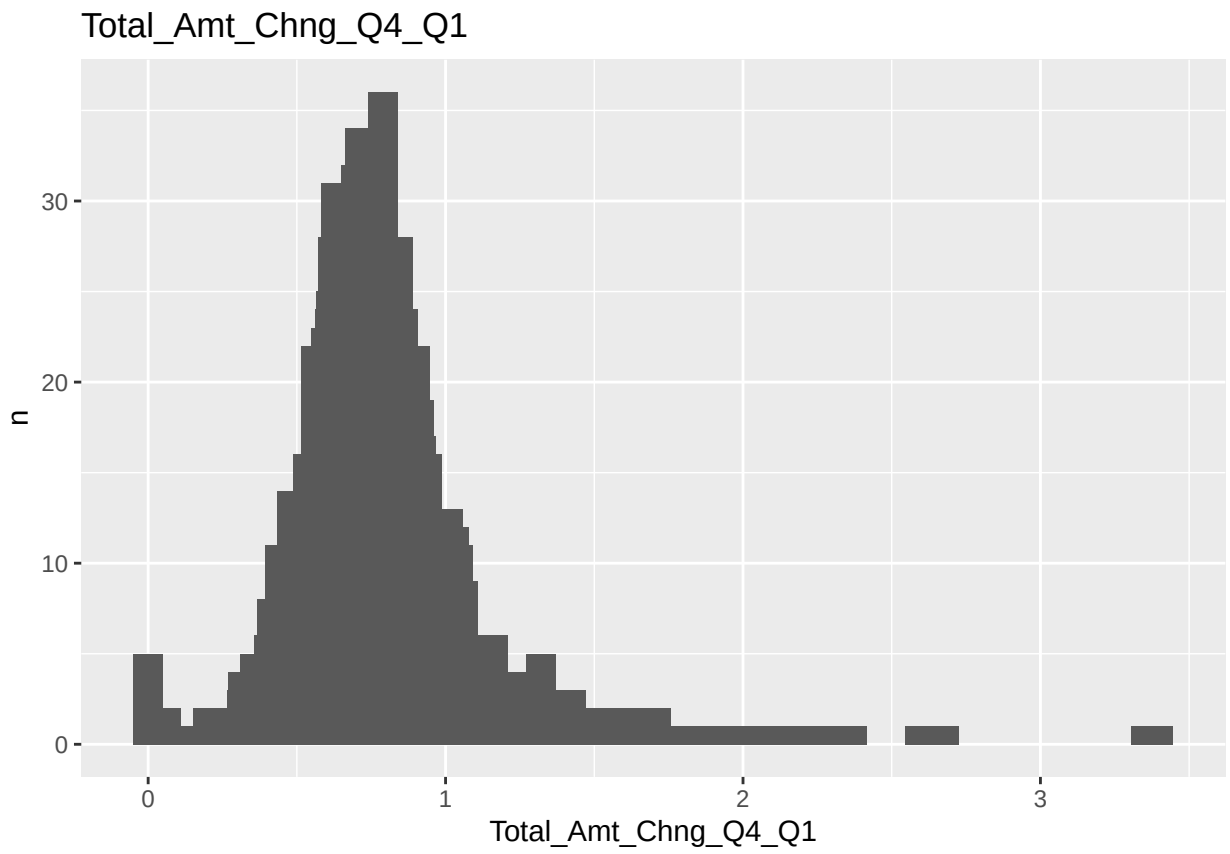
```
data %>%
  group_by(Avg_Open_To_Buy) %>%
  summarize(n=n()) %>%
  mutate(porc=n/sum(n)) %>%
  ggplot(aes(Avg_Open_To_Buy,n)) +
  geom_col(width = 1000)+
  ggtitle("Average Open to Buy")
```



2.1.17 Total_Amt_Chng_Q4_Q1

Refers to change in Transaction Amount (Q4 over Q1).

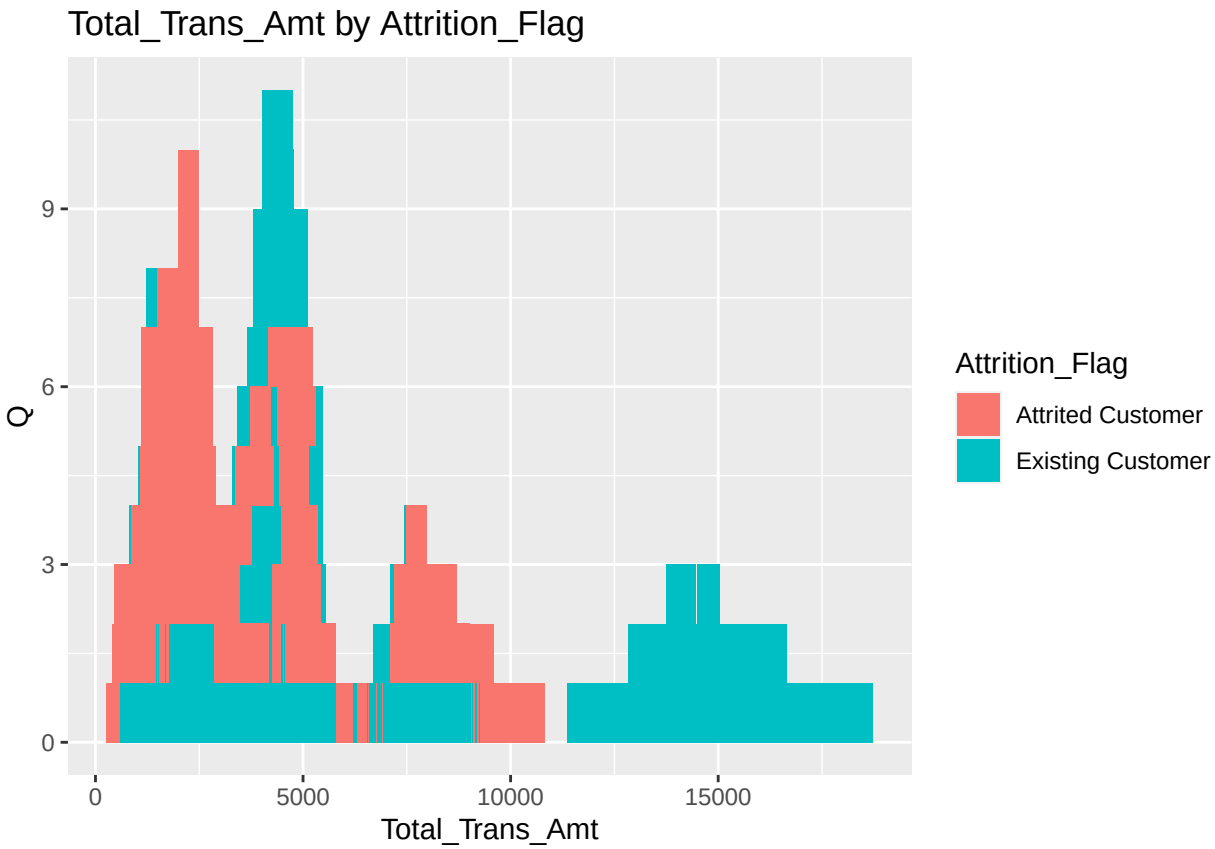
```
data %>%
  group_by(Total_Amt_Chng_Q4_Q1) %>%
  summarize(n=n()) %>%
  mutate(porc=n/sum(n)) %>%
  ggplot(aes(Total_Amt_Chng_Q4_Q1,n))+
  geom_col(width = 0.1)+
  ggtitle("Total_Amt_Chng_Q4_Q1")
```



2.1.18 Total_Trans_Amt

This variable will also be seen as a promising variable to predict what we need.It is the total transaction amount (Last 12 months).We can see how it separates the types of clients in the intervals with the lowest amounts per transaction

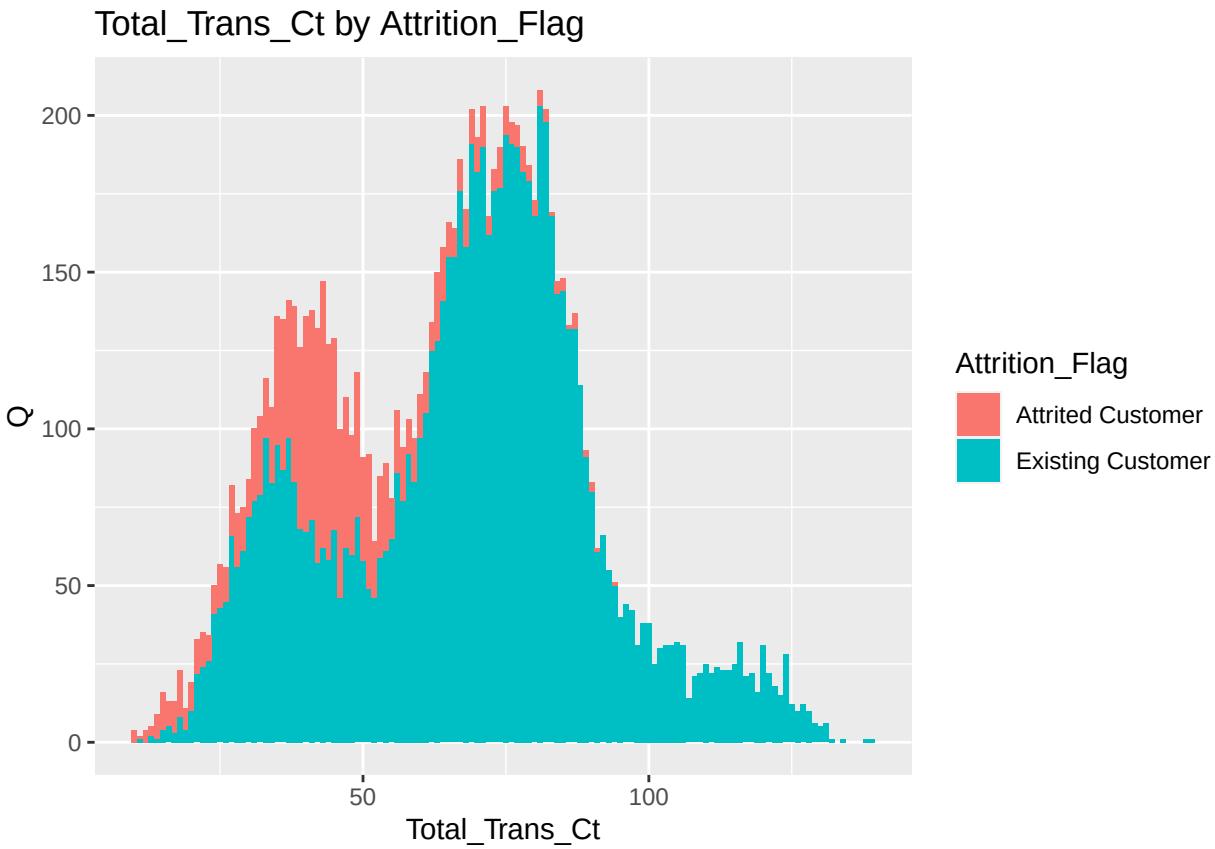
```
data %>%
  group_by(Total_Trans_Amt,Attrition_Flag) %>%
  summarize(Q=n()) %>%
  mutate(porc=Q/sum(Q)) %>%
  arrange(desc(porc)) %>%
  ggplot(aes(Total_Trans_Amt,Q,fill=Attrition_Flag))+
  geom_col(width = 500)+
  ggtitle("Total_Trans_Amt by Attrition_Flag")
```



2.1.19 Total_Trans_Ct

Similar to the previous one, but instead of amounts, it counts the amount. It is probably highly correlated with the previous one, not providing new information to the model. You can see the same effect; separation of lost customers in the band with the least amount of transactions.

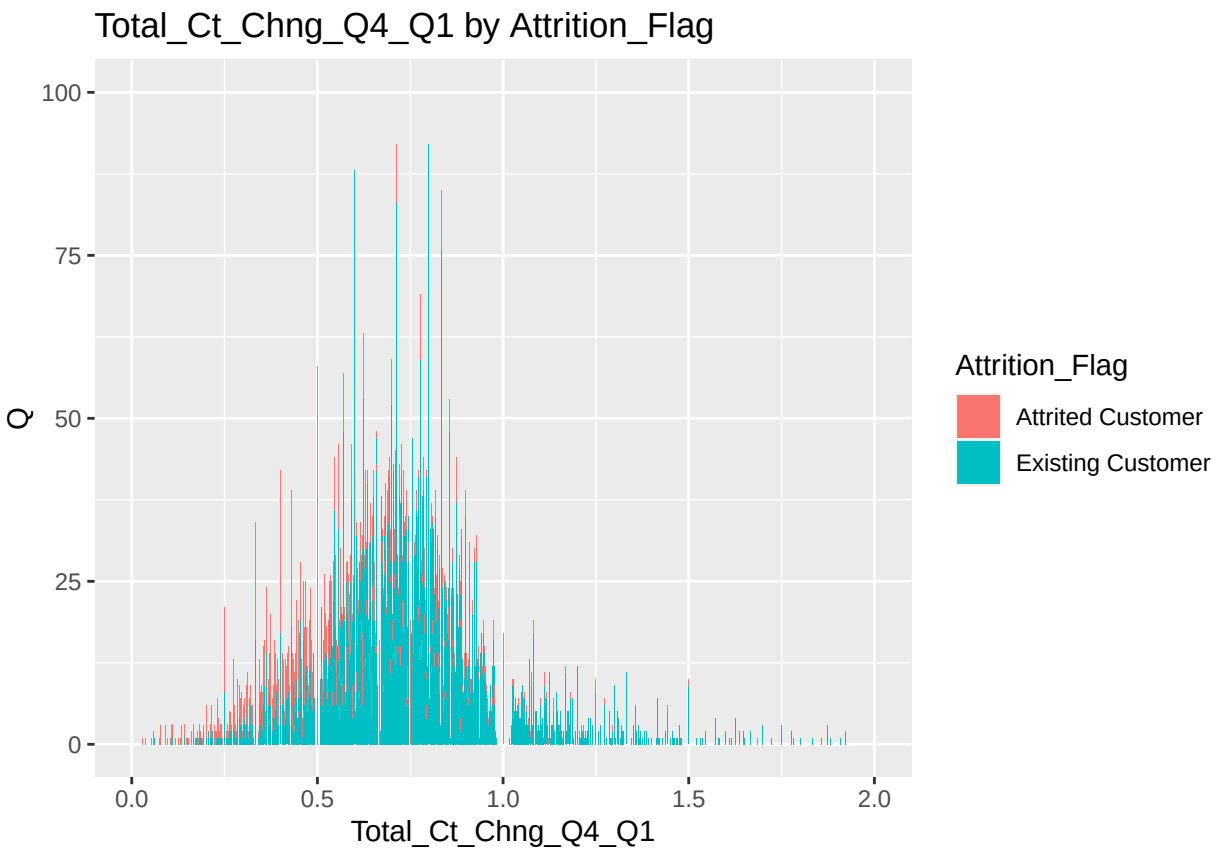
```
data %>%
  group_by(Total_Trans_Ct,Attrition_Flag) %>%
  summarize(Q=n()) %>%
  mutate(porc=Q/sum(Q)) %>%
  arrange(desc(porc)) %>%
  ggplot(aes(Total_Trans_Ct,Q,fill=Attrition_Flag))+
  geom_col()+
  ggtitle("Total_Trans_Ct by Attrition_Flag")
```



2.1.20 Total_Ct_Chng_Q4_Q1

Change in Transaction Count (Q4 over Q1)).

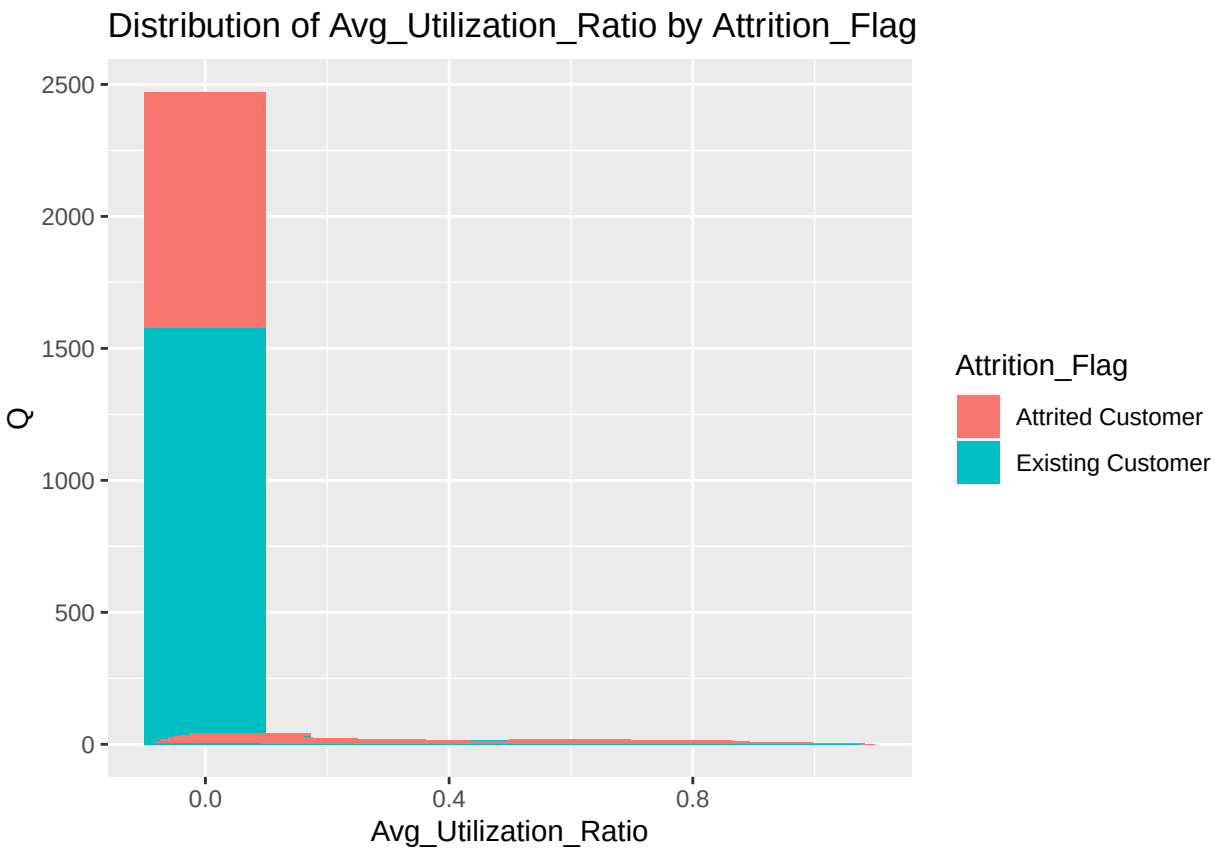
```
data %>%
  group_by(Total_Ct_Chng_Q4_Q1,Attrition_Flag) %>%
  summarize(Q=n()) %>%
  mutate(porc=Q/sum(Q)) %>%
  arrange(desc(porc)) %>%
  ggplot(aes(Total_Ct_Chng_Q4_Q1,Q,fill=Attrition_Flag))+
  geom_col()+
  ylim(0,100)+
  xlim(0,2)+
  ggtitle("Total_Ct_Chng_Q4_Q1 by Attrition_Flag")
```



2.1.21 Avg_Utilization_Ratio

Average Card Utilization Ratio. Initially, it might seem like a good source of information, but when you see how it is distributed, almost all the information is concentrated in the same point

```
data %>%
  group_by(Avg_Utilization_Ratio,Attrition_Flag) %>%
  summarize(Q=n()) %>%
  mutate(porc=Q/sum(Q)) %>%
  arrange(desc(porc)) %>%
  ggplot(aes(Avg_Utilization_Ratio,Q,fill=Attrition_Flag))+
  geom_col(width = 0.2)+
  ggtitle("Distribution of Avg_Utilization_Ratio by Attrition_Flag")
```



2.2 Pre-processing the dataset

First to simplify the variables, dummy variables will be created, with two states for each column: 1 or 0

```
#Making the dummy's features
data_flags <- dummy_cols(data,remove_selected_columns = TRUE,select_columns = c("Gender","Education_Leve

#Factor variables that are mutually exclusive can be separated into n-1 dummy variables.
#This means, if it is not any of the above, it is by default the one that remains, being one of the dis
data_flags <- data_flags %>%
  mutate(Churn_Customers= ifelse(Attrition_Flag =="Attrited Customer",1,0)) %>%
  select(-Attrition_Flag,
         -Gender_M,
         -Education_Level_Doctorate,
         -Marital_Status_Unknown,
         -`Income_Category_$120K +`,
         -Card_Category_Platinum,
         -CLIENTNUM #Does not have any information apart fromthe ID
         )

str(data_flags)
```

```
## 'data.frame':    10127 obs. of  33 variables:
##  $ Customer_Age                  : int  45 49 51 40 40 44 51 32 37 48 ...
##  $ Dependent_count               : int  3 5 3 4 3 2 4 0 3 2 ...
##  $ Months_on_book                : int  39 44 36 34 21 36 46 27 36 36 ...
##  $ Total_Relationship_Count      : int  5 6 4 3 5 3 6 2 5 6 ...
```

```
##  $ Months_Inactive_12_mon        : int  1 1 1 4 1 1 1 2 2 3 ...
##  $ Contacts_Count_12_mon         : int  3 2 0 1 0 2 3 2 0 3 ...
##  $ Credit_Limit                  : num  12691 8256 3418 3313 4716 ...
##  $ Total_Revolving_Bal           : int  777 864 0 2517 0 1247 2264 1396 2517 1677 ...
##  $ Avg_Open_To_Buy               : num  11914 7392 3418 796 4716 ...
##  $ Total_Amt_Chng_Q4_Q1          : num  1.33 1.54 2.59 1.4 2.17 ...
##  $ Total_Trans_Amt               : int  1144 1291 1887 1171 816 1088 1330 1538 1350 1441 ...
##  $ Total_Trans_Ct                : int  42 33 20 20 28 24 31 36 24 32 ...
##  $ Total_Ct_Chng_Q4_Q1           : num  1.62 3.71 2.33 2.33 2.5 ...
##  $ Avg_Utilization_Ratio         : num  0.061 0.105 0 0.76 0 0.311 0.066 0.048 0.113 0.144 ...
##  $ Gender_F                      : int  0 1 0 1 0 0 0 0 0 0 ...
##  $ Education_Level_College       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Education_Level_Graduate      : int  0 1 1 0 0 1 0 0 0 1 ...
##  $ Education_Level_High School   : int  1 0 0 1 0 0 0 1 0 0 ...
##  $ Education_Level_Post-Graduate : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Education_Level_Uneducated    : int  0 0 0 0 1 0 0 0 1 0 ...
##  $ Education_Level_Unknown       : int  0 0 0 0 0 0 1 0 0 0 ...
##  $ Marital_Status_Divorced       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Marital_Status_Married        : int  1 0 1 0 1 1 1 0 0 0 ...
##  $ Marital_Status_Single         : int  0 1 0 0 0 0 0 0 1 1 ...
##  $ Income_Category_$40K - $60K   : int  0 0 0 0 0 1 0 0 0 0 ...
##  $ Income_Category_$60K - $80K   : int  1 0 0 0 1 0 0 1 1 0 ...
##  $ Income_Category_$80K - $120K  : int  0 0 1 0 0 0 0 0 0 1 ...
##  $ Income_Category_Less than $40K: int  0 1 0 1 0 0 0 0 0 0 ...
##  $ Income_Category_Unknown       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Card_Category_Blue            : int  1 1 1 1 1 1 0 0 1 1 ...
##  $ Card_Category_Gold            : int  0 0 0 0 0 0 1 0 0 0 ...
##  $ Card_Category_Silver          : int  0 0 0 0 0 0 0 1 0 0 ...
##  $ Churn_Customers               : num  0 0 0 0 0 0 0 0 0 0 ...
```

By simplifying the situation of the factor variables, they increased the size of the database. To simplify this, the least explanatory variables are eliminated. There are different ways to do it. In this project the correlation matrix will be used. Remembering that the correlation or dependence is any statistical relationship, causal or not, between two random variables or bivariate data, what will be done is to identify the variables that have a lot of correlation with each other (we refer to the relationship between the predictors) and maintain the which has a lower correlation with the rest of the variables (trying). You start by looking at the relationship in pairs, then you compare it to the rest and are left with just one of the two. To do this, we must first calculate the correlation matrix

```
correlationMatrix <- cor(data_flags[,1:32],method="pearson")
```

The level of correlation between the variables can be seen graphically through the size and color of the intersections. Once with this, we will look for the correlations and we will locate the dispensable variables

```
highlyCorrelatedNames <- findCorrelation(correlationMatrix, cutoff=0.5,names=TRUE)
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.5,names=FALSE)

data_model <- data_flags[,-highlyCorrelated]
kable(highlyCorrelatedNames,col.names = "This are the variables highly correlated")
```

This are the variables highly correlated
Avg_Open_To_Buy
Credit_Limit
Avg_Utilization_Ratio
Card_Category_Blue
Gender_F
Total_Trans_Amt
Marital_Status_Married
Customer_Age

2.3 Models

First we will divide the dataset into trainset and testset. The split 50/50 is because of the small dataset.

```
#Using a 50/50 division of the data
index_test <- createDataPartition(data_model$Churn_Customers,p=0.5,list=FALSE)

test_set <- data_flags[index_test,]
train_set <- data_flags[-index_test,]
```

Different models will be tested in order to then be able to choose the model that best suits our needs. The proportion of correct predictions over the total number of observations to be predicted will be used to measure its efficiency.

2.3.1 k-nearest neighbors algorithm (knn)

The first approximation will be through the knn algorithm. Nearest neighbor classifiers are defined by their characteristic of classifying unlabeled examples by assigning them the class of the most similar labeled examples. The kNN algorithm treats entities as coordinates in a multidimensional entity space. Then, having created this "map", by locating the points here, it gives the classification based on the majority of "k" neighbors that it has closest. Traditionally Eucledian distances are used for the location and assembly of this map, following the following formula:

$$dist(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2}$$

As it is based on numerical distances, it is necessary that all variables be of this type. The way categorical variables were converted to dummys earlier is useful in this case because it reduces them to a numerical scale, from 0 to 1

2.3.1.1 Changing the scales of fetures The location of each point within the map (which then classifies the observations to be predicted) is based on distances calculated based on the values of each feature. This means that if there is a feature that is naturally measured in larger absolute values than others, it may have more impact on the measured distances, probably turning the model into a predictor based on only that feature. For example, we have the dummy variables created previously, whose scale would be from 0 to 1 (with the only two possible values being). This variable could present values less incidence, just because of its scale, than the limit for the purchase of credit cards, which is absurd to think that it even has a value of 1 dollar.

For this reason it is necessary, when using the knn algorithm, to work on the database beforehand. For the application of this algorithm, it will be tested with the adjustment of two types of scales: normalized and z-scores.

2.3.1.1.1 Normalized Feature When normalizing the variables, what is done is to apply the following formula, to adjust its scale between 0 and 1:

$$Xnormalized = \frac{x - min(x)}{max(x) - min(x)}$$

```
#Creating the function that normalizes features
normalize <- function(x){
  return((x-min(x))/(max(x)-min(x)))}

#Creating both datasets with the new scale
train_features_n <- as.data.frame(lapply(train_set[1:32],normalize))
test_features_n <- as.data.frame(lapply(test_set[1:32],normalize))
```

After the dataset is prepared, the model is created

```
#Making the model
model_knn_n <- train(train_features_n,as.factor(train_set[,33]),method = "knn")

#Using this model to predict
y_hat_knn_n_model <- predict(model_knn_n,test_features_n)

#Checking the proportion of right cases
Mean_knn_n <- mean(test_set$Churn_Customers==y_hat_knn_n_model)

#Creating a table to fill it with the results
Results_table <- data.frame(Model=character(),Mean=numeric(),stringsAsFactors = FALSE)

#Creating the list for this model
Results_knn_n <- list("Knn normalized",Mean_knn_n)

# Adding this model's results to the table
Results_table[1,] <- Results_knn_n
Results_knn_n
```

```
## [[1]]
## [1] "Knn normalized"
##
## [[2]]
## [1] 0.8558452
```

Using this model with normalized features, a percentage of 85.58 correct predictions has been achieved

2.3.1.1.2 z-score Another way to unify or standardize the scale of the variables is through the z-score. This involves applying a calculation:

$$Z = \frac{x - \mu}{\sigma}$$

Being mu the mean and sigma the standard deviation. This scale is easier to apply, since it can be done through the scale () function:

```r
# Scaling the data
train_features_s <- as.data.frame(scale(train_set[,1:32]))
test_features_s <- as.data.frame(scale(test_set[,1:32]))
```

Proceeding to the model

```r
#Making the model
model_knn_s <- train(train_features_s[,1:32],as.factor(train_set[,33]),method = "knn")

#Using the model to predict
y_hat_knn_s_model <- predict(model_knn_s,test_features_s)

#Checking the proportion of right cases
Mean_kss_s <- mean(test_set$Churn_Customers==y_hat_knn_s_model)

#Creating the list for this model
Results_knn_s <- list("Knn with scale z-score",Mean_kss_s)

# Adding this model's results to the table
Results_table[2,] <-Results_knn_s

Results_knn_s
```

```
## [[1]]
## [1] "Knn with scale z-score"
##
## [[2]]
## [1] 0.8720379
```

Using this model with this type of scaling, a percentage of 87.2 correct predictions has been achieved

2.3.2 C5.0 Decision Tree

A decision tree in Machine Learning is a tree structure similar to a flowchart where an internal node represents a characteristic (or attribute), the branch represents a decision rule, and each leaf node represents the result. It is a fairly simple model to understand, you do not need solid knowledge of statistics and probability to understand it, while still being one of the most efficient models. Decision trees are built using a heuristic called recursive partitioning. This approach it is generally known as "divide and conquer" because it uses the values of the characteristics to divide the data in smaller and smaller subsets of similar classes. Starting at the root node, which represents the entire data set, the algorithm choose a characteristic that is the most predictive of the target class. Examples are then divided into groups of values other than this characteristic; this decision forms the first set of tree branches. The algorithm continues to "divide and conquer" the nodes, choose the best candidate feature each time until a stopping criterion is reached.

As a first step, the features now do not need to be numeric, so we will change the format of some that better adapt to the factorial type.

```r
#Transforming into factorial some features in the train set
train_set_C5.0 <- train_set %>% mutate(
  Gender_F = as.factor(Gender_F),
  Education_Level_College= as.factor(Education_Level_College),
  Education_Level_Graduate=as.factor(Education_Level_Graduate),
```

```
  Education_Level_High_school = as.factor(`Education_Level_High School`),
  Education_Level_Post_Graduate = as.factor(`Education_Level_Post-Graduate`),
  Education_Level_Uneducated = as.factor(Education_Level_Uneducated),
  Education_Level_Unknown=as.factor(Education_Level_Unknown),
  Marital_Status_Divorced=as.factor(Marital_Status_Divorced),
  Marital_Status_Married = as.factor(Marital_Status_Married),
  Marital_Status_Single=as.factor(Marital_Status_Single),
  Income_Category_40K_60K= as.factor(`Income_Category_$40K - $60K`),
  Income_Category_60K_80K = as.factor(`Income_Category_$60K - $80K`),
  Income_Category_80K_120K=as.factor(`Income_Category_$80K - $120K`),
  Income_Category_Less_than_40K = as.factor(`Income_Category_Less than $40K`),
  Income_Category_Unknown = as.factor(Income_Category_Unknown),
  Card_Category_Blue = as.factor(Card_Category_Blue),
  Card_Category_Gold = as.factor(Card_Category_Gold),
  Card_Category_Silver = as.factor(Card_Category_Silver),
  Churn_Customers = as.factor(Churn_Customers)
) %>%
  select(
    -`Education_Level_High School`,
    -`Education_Level_Post-Graduate`,
    -`Income_Category_$40K - $60K`,
    -`Income_Category_$60K - $80K`,
    -`Income_Category_$80K - $120K`,
    -`Income_Category_Less than $40K`
  )

#Transforming into factorial some features in the test set
test_set_C5.0 <- test_set %>% mutate(
  Gender_F = as.factor(Gender_F),
  Education_Level_College= as.factor(Education_Level_College),
  Education_Level_Graduate=as.factor(Education_Level_Graduate),
  Education_Level_High_school = as.factor(`Education_Level_High School`),
  Education_Level_Post_Graduate = as.factor(`Education_Level_Post-Graduate`),
  Education_Level_Uneducated = as.factor(Education_Level_Uneducated),
  Education_Level_Unknown=as.factor(Education_Level_Unknown),
  Marital_Status_Divorced=as.factor(Marital_Status_Divorced),
  Marital_Status_Married = as.factor(Marital_Status_Married),
  Marital_Status_Single=as.factor(Marital_Status_Single),
  Income_Category_40K_60K= as.factor(`Income_Category_$40K - $60K`),
  Income_Category_60K_80K = as.factor(`Income_Category_$60K - $80K`),
  Income_Category_80K_120K=as.factor(`Income_Category_$80K - $120K`),
  Income_Category_Less_than_40K = as.factor(`Income_Category_Less than $40K`),
  Income_Category_Unknown = as.factor(Income_Category_Unknown),
  Card_Category_Blue = as.factor(Card_Category_Blue),
  Card_Category_Gold = as.factor(Card_Category_Gold),
  Card_Category_Silver = as.factor(Card_Category_Silver),
  Churn_Customers = as.factor(Churn_Customers)
) %>%
  select(
    -`Education_Level_High School`,
    -`Education_Level_Post-Graduate`,
    -`Income_Category_$40K - $60K`,
    -`Income_Category_$60K - $80K`,
```

```
    -`Income_Category_$80K - $120K`,
    -`Income_Category_Less than $40K`
  )
```

Once the initial base has been processed, continues with the assembly of the model

```
#Making the model
model_c5o <- C5.0(Churn_Customers ~.,data=train_set_C5.0)

#Using the model to predict
y_hat_c50 <- predict(model_c5o,test_set_C5.0)

# Summary of the model
summ_c50 <- summary(model_c5o)
```

```
#Checking the proportion of right cases
mean_c50 <- mean(y_hat_c50==test_set_C5.0$Churn_Customers)

#Creating the list for this model
Results_c50 <- list("C5.0 Decision Tree",mean_c50)

# Adding this model's results to the table
Results_table[3,] <- Results_c50

Results_c50
```

```
## [[1]]
## [1] "C5.0 Decision Tree"
##
## [[2]]
## [1] 0.9397709
```

Using C5.0, a percentage of 93.98 correct predictions has been achieved

2.3.3 C5.0 Decision Tree Boosted

Boosting is rooted in the notion that by combining a number of weak performing learners, you can create a team that is much stronger than any one of the learners alone. Each of the models has a unique set of strengths and weaknesses, and may be better or worse at certain problems. Using a combination of several learners with complementary strengths and weaknesses can therefore dramatically improve the accuracy of a classifier.

It is only necessary to add an additional test parameter that indicates the number of separate decision trees to use. The "trials" parameter sets a upper limit; the algorithm will stop adding trees if it recognizes that additional trials doesn't seem to be improving accuracy.

```
#Making the model
model_c5o_boosted <- C5.0(Churn_Customers ~.,data=train_set_C5.0,trials = 20)

#Using the model to predict
y_hat_c50_boosted <- predict(model_c5o_boosted,test_set_C5.0)
```

```
# Summary of the model
summ_c50_boosted <- summary(model_c5o_boosted)
```

Checking the results

```
#Checking the proportion of right cases
mean_c50_boosted <- mean(y_hat_c50_boosted==test_set_C5.0$Churn_Customers)

#Creating the list for this model
Results_c50_boosted <- list("C5.0 Decision Tree Boosted",mean_c50_boosted)

# Adding this model's results to the table
Results_table[4,] <- Results_c50_boosted

Results_c50_boosted
```

```
## [[1]]
## [1] "C5.0 Decision Tree Boosted"
##
## [[2]]
## [1] 0.9668246
```

Using C5.0 boosted, a percentage of 96.68 correct predictions has been achieved

3 Results

The results of all models are as follows

```
Results_table %>%
  mutate("Correct predictions" = round(Results_table$Mean,digits = 4)) %>%
  select(-Mean) %>%
  kable()
```

Model	Correct predictions
Knn normalized	0.8558
Knn with scale z-score	0.8720
C5.0 Decision Tree	0.9398
C5.0 Decision Tree Boosted	0.9668

It has been possible to improve from the first approximation, using improvements in the model and using another that turns out to be much more efficient.Starting from an efficiency of 85.58, changing the scale that was used, it could be increased to 85.58. This may be due to the fact that in the case of normalized features, all the variation and amplitude of the feature is forced to be compacted between the values 0 to 1. In the case of z-score, it is not restricted to this range, but it is reduce so that all the feeatures vary based on their standard deviation. Once these two options have been explored, the decision tree method is used. From the beginning, a notable improvement in efficiency is achieved, reaching 93.98 values. Once here, we improve the model using boosting, which produces a satisfactory level of 96.68 efficiency.

4 Conclusion

It has been started with a database provided by Kaggle (https://www.kaggle.com/sakshigoyal7/credit-card-customers), so that the information can be used to create a model that can predict whether it will be a lost customer or not. With an initial descriptive analysis, it has been possible to see how some features are more interesting than others in pursuit of our objective. As the first processing of the base, some features that were categorized were converted to dummy's features, increasing the dimension of the base. In order to counteract some of this effect, through the correlation matrix, it has been possible to distinguish variables that could explain the same as others, with the aim of simplifying the size of the model through the amount of features it handles. Once this was done, the base was ready to be handled by the different needs of the models. Once here, different models and variants were used to be able to improve the efficiency of the predictions, achieving an efficiency of 96.68.

The impact that this model could have is interesting, since it could retain customers of the bank or banking entity that provides credit cards as a service. Being able to predict that a client is about to be lost helps to proactively seek solutions to their problems, as well as improve the offer to them so as not to lose interest.

On the other hand, we believe that the base is not large enough to create a model on which a company can base its operations, so we are limited to the base provided. Like future projects, doing this project has provided me with practice, both technical and solving and facing a challenge like this. Being that I work within the banking sector, this project is an initial step for my next project that is related to the behavioral prediction of clients with personal loans