notebook

December 2, 2022

1 Business Problem: Customer Churn Prediction

- Acquiring new customers is important, but retaining them accelerates profitable growth.
- While churn prediction offers the knowledge of which customers are going to stop doing business with your brand at present and the reasons behind them leaving, churn forecasting depicts the total number of customers who are most likely to leave in the near future.

1.1 Project Tasks

A business manager of a consumer credit card portfolio is facing the problem of customer attrition, also known as customer churn. While acquiring new customers is important, simultaneously retaining customers is key to driving profitable growth for the company. In order to prevent the further loss of customers in the future, the manager asks our data analysis team to complete these 3 tasks: - Analyze the data to find out the potential signals for customers leaving - Predict which customers are likely to leave in the near future - Provide data-driven solutions to prevent future customer churn

The manager hopes to leverage this information to retain loyal customers, focus and target marketing campaigns towards customers likely to leave, boost profits, and minimize loss.

Customer churn prediction has several benefits, such as:

- Retain the loyalty of customers: By knowing which customers are most likely to leave, companies can create customized marketing strategies for individual customers, thereby retaining their loyalty.
- Get the business back on track: By identifying the root causes of customer churn, brands can rethink and rebuild their product/service, marketing, and acquisition strategies.
- Boost profits: Acquiring new customers is usually very expensive. However, by selling to existing customers, you can boost profits significantly.
- Avert Loss: Customer churn leads to substantial losses. Hence, churn prediction allows businesses to avoid losses by retaining more existing customers through innovative business strategies.

1.2 Summary of Data and Modeling

- Credit card data: 10127 obs. of 21 variables (Kaggle)
- Review data: 7513 obs. of 9 variables
- Visualization: histogram, scatterplot, barplot …
- Supervised Method - Classification Model: Decision TREE, Logistic Regression, Random Forest,XGBoost to classify if a customer is going to drop off.
- Unsupervised Method - Clustering Model: Kmeans model to segment attrited customers to get to know more about the pattern in each customer segmentation.
- Text Mining for Credit Card customer reviews
- Improvement: Handling imbalanced data. (ROSE), Deep learning models

[723]:
```r
# Read data
bank <- read.csv("data/bank_clean.csv", stringsAsFactors = TRUE)
head(bank)
```

A data.frame: 6 × 21

	CLIENTNUM <int>	Attrition_Flag <int>	Customer_Age <int>	Gender <fct>	Dependent_count <int>	Educ <fct
1	768805383	0	45	M	3	High
2	818770008	0	49	F	5	Grac
3	713982108	0	51	M	3	Grac
4	769911858	0	40	F	4	High
5	709106358	0	40	M	3	Unec
6	713061558	0	44	M	2	Grac

[724]:
```r
summary(bank)
```

```
   CLIENTNUM           Attrition_Flag     Customer_Age     Gender    Dependent_count
 Min.   :708082083   Min.   :0.0000   Min.   :26.00   F:5358   Min.   :0.000
 1st Qu.:713036770   1st Qu.:0.0000   1st Qu.:41.00   M:4769   1st Qu.:1.000
 Median :717926358   Median :0.0000   Median :46.00            Median :2.000
 Mean   :739177606   Mean   :0.1607   Mean   :46.33            Mean   :2.346
 3rd Qu.:773143533   3rd Qu.:0.0000   3rd Qu.:52.00            3rd Qu.:3.000
 Max.   :828343083   Max.   :1.0000   Max.   :73.00            Max.   :5.000
```

```
     Education_Level  Marital_Status       Income_Category  Card_Category
 College       :1013   Divorced: 748   $120K +        : 727   Blue    :9436
 Doctorate     : 451   Married :4687   $40K - $60K    :1790   Gold    : 116
 Graduate      :3128   Single  :3943   $60K - $80K    :1402   Platinum:  20
 High School   :2013   Unknown : 749   $80K - $120K   :1535   Silver  : 555
 Post-Graduate: 516                    Less than $40K:3561
 Uneducated    :1487                   Unknown        :1112
 Unknown       :1519
 Months_on_book  Total_Relationship_Count Months_Inactive_12_mon
 Min.   :13.00   Min.   :1.000            Min.   :0.000
 1st Qu.:31.00   1st Qu.:3.000            1st Qu.:2.000
 Median :36.00   Median :4.000            Median :2.000
 Mean   :35.93   Mean   :3.813            Mean   :2.341
 3rd Qu.:40.00   3rd Qu.:5.000            3rd Qu.:3.000
 Max.   :56.00   Max.   :6.000            Max.   :6.000

 Contacts_Count_12_mon  Credit_Limit   Total_Revolving_Bal Avg_Open_To_Buy
 Min.   :0.000          Min.   : 1438   Min.   :   0        Min.   :    3
 1st Qu.:2.000          1st Qu.: 2555   1st Qu.: 359        1st Qu.: 1324
 Median :2.000          Median : 4549   Median :1276        Median : 3474
 Mean   :2.455          Mean   : 8632   Mean   :1163        Mean   : 7469
 3rd Qu.:3.000          3rd Qu.:11068   3rd Qu.:1784        3rd Qu.: 9859
 Max.   :6.000          Max.   :34516   Max.   :2517        Max.   :34516

 Total_Amt_Chng_Q4_Q1 Total_Trans_Amt Total_Trans_Ct   Total_Ct_Chng_Q4_Q1
 Min.   :0.0000       Min.   :  510   Min.   : 10.00   Min.   :0.0000
 1st Qu.:0.6310       1st Qu.: 2156   1st Qu.: 45.00   1st Qu.:0.5820
 Median :0.7360       Median : 3899   Median : 67.00   Median :0.7020
 Mean   :0.7599       Mean   : 4404   Mean   : 64.86   Mean   :0.7122
 3rd Qu.:0.8590       3rd Qu.: 4741   3rd Qu.: 81.00   3rd Qu.:0.8180
 Max.   :3.3970       Max.   :18484   Max.   :139.00   Max.   :3.7140

 Avg_Utilization_Ratio
 Min.   :0.0000
 1st Qu.:0.0230
 Median :0.1760
 Mean   :0.2749
 3rd Qu.:0.5030
 Max.   :0.9990
```

[725]:
```
str(bank)
```

```
'data.frame':	10127 obs. of  21 variables:
 $ CLIENTNUM               : int  768805383 818770008 713982108 769911858
709106358 713061558 810347208 818906208 710930508 719661558 …
 $ Attrition_Flag          : int  0 0 0 0 0 0 0 0 0 0 …
```

```
 $ 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 1.41 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 …
```

1.3 Data Validation:

- **CLIENTNUM**: Unique identifier for the customer holding the account, int, such as "768805383"
- **Attrition_Flag**: Internal event (customer activity) variable - if the account is closed then 1 else 0: Exisiting customer 84%, Attrited customer 16%.
- **Customer_Age**: Demographic variable - Customer's Age in Years, int, 26–73
- **Gender**: Demographic variable - M=Male, F=Female, F 53%, M =47%
- **Dependent_count**:Demographic variable - Number of dependents, int, 0-5
- **Education_Level**: Demographic variable - Educational Qualification of the account holder, chr, "high school" 20%, "graduate" 31%, … 7 levels,
- **Marital_Status**: Demographic variable, chr, "Married", "Single", "Divorced", "Unknown", 4 levels
- **Income_Category**: Demographic variable - Annual Income Category of the account holder, chr, (< $40K, $40K - 60K, $60K - $80K, $80K-$120K, > $120K, Unknown) 6 levels,
- **Card_Category**: Product Variable - Type of Card (Blue, Silver, Gold, Platinum) 4 levels, Blue 93%
- **Months_on_book**: Period of relationship with bank, int 13 - 56 months.
- **Total_Relationship_Count**: Total no. of products held by the customer, int 1-6
- **Months_Inactive_12_mon**: No. of months inactive in the last 12 months, int, 0-6
- **Contacts_Count_12_mon**: No. of Contacts in the last 12 months, int, 0-6
- **Credit_Limit**: Credit Limit on the Credit Card, num, 1.44K-34.5K

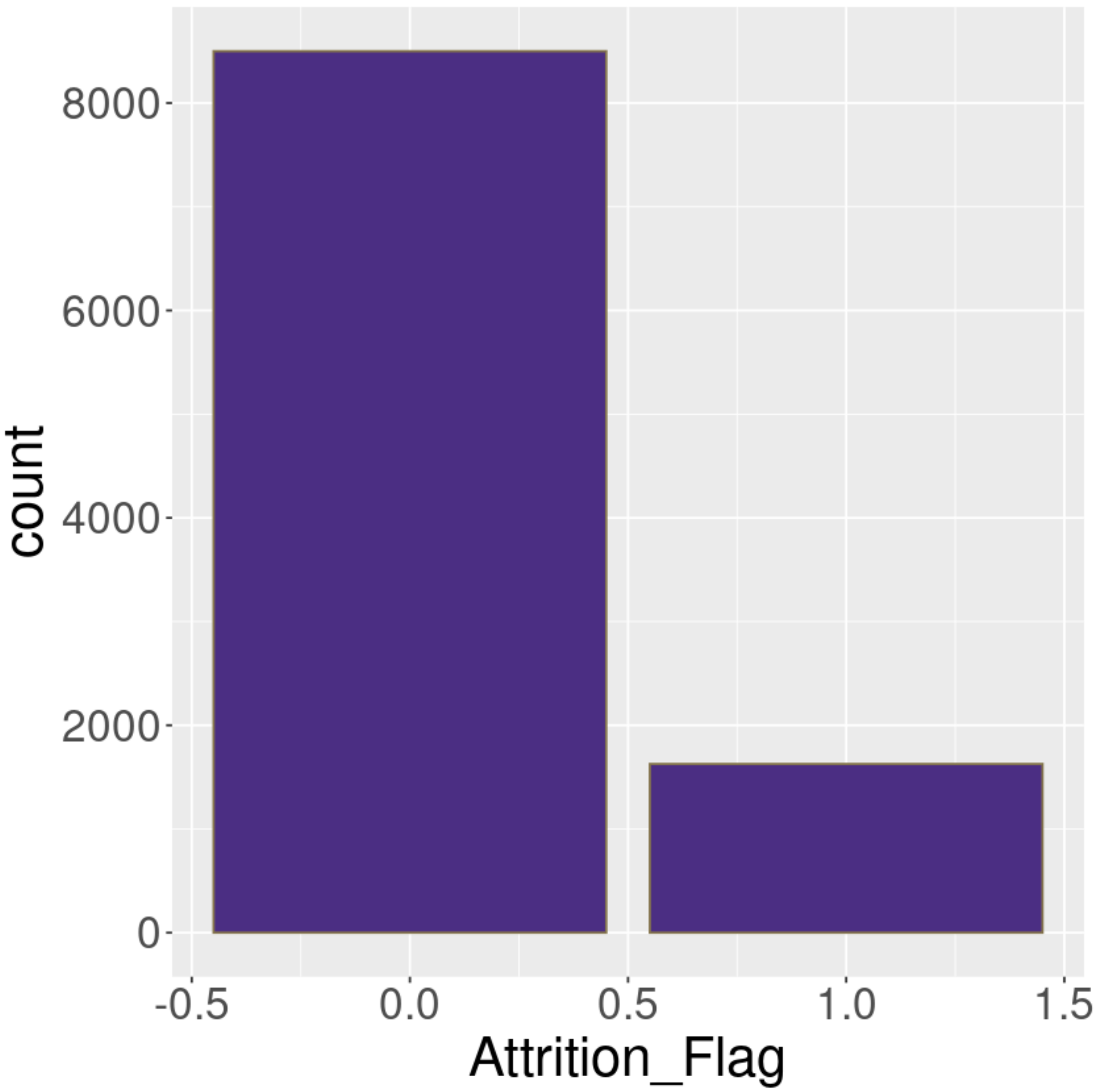
- **Total_Revolving_Bal****:Total Revolving Balance on the Credit Card,int, 0 - 2517 # the meaning
- **Avg_Open_To_Buy**:Open to Buy Credit Line (Average of last 12 months), num, 3 - 34.5k
- **Total_Amt_Chng_Q4_Q1**: Change in Transaction Amount (Q4 over Q1),num, 0-3.4
- **Total_Trans_Amt**: Total Transaction Amount (Last 12 months), int,510-18.5k
- **Total_Trans_Ct**: Total Transaction Count (Last 12 months), int, 10 - 139
- **Total_Ct_Chng_Q4_Q1**: Change in Transaction Count (Q4 over Q1), num, 0 - 3.71
- **Avg_Utilization_Ratio**, Average Card Utilization Ratio, num 0 - 100%

1.4 Data Visualiaztion and Exploratory Analysis

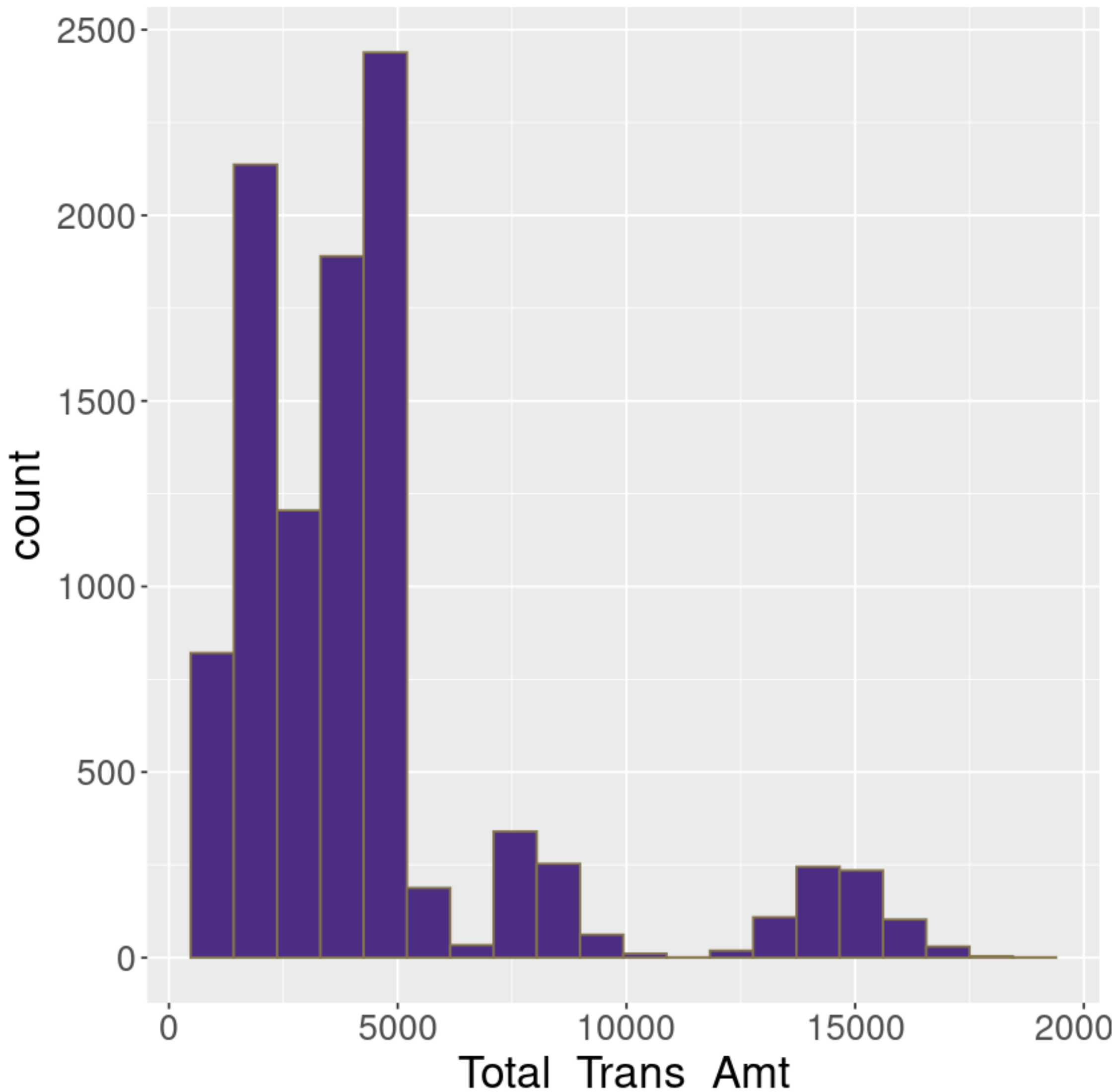
[726]:
```
# Target Variable: Attition_Flag
library(ggplot2)
mean(bank$Attrition_Flag == 1)
```

0.16065962279056

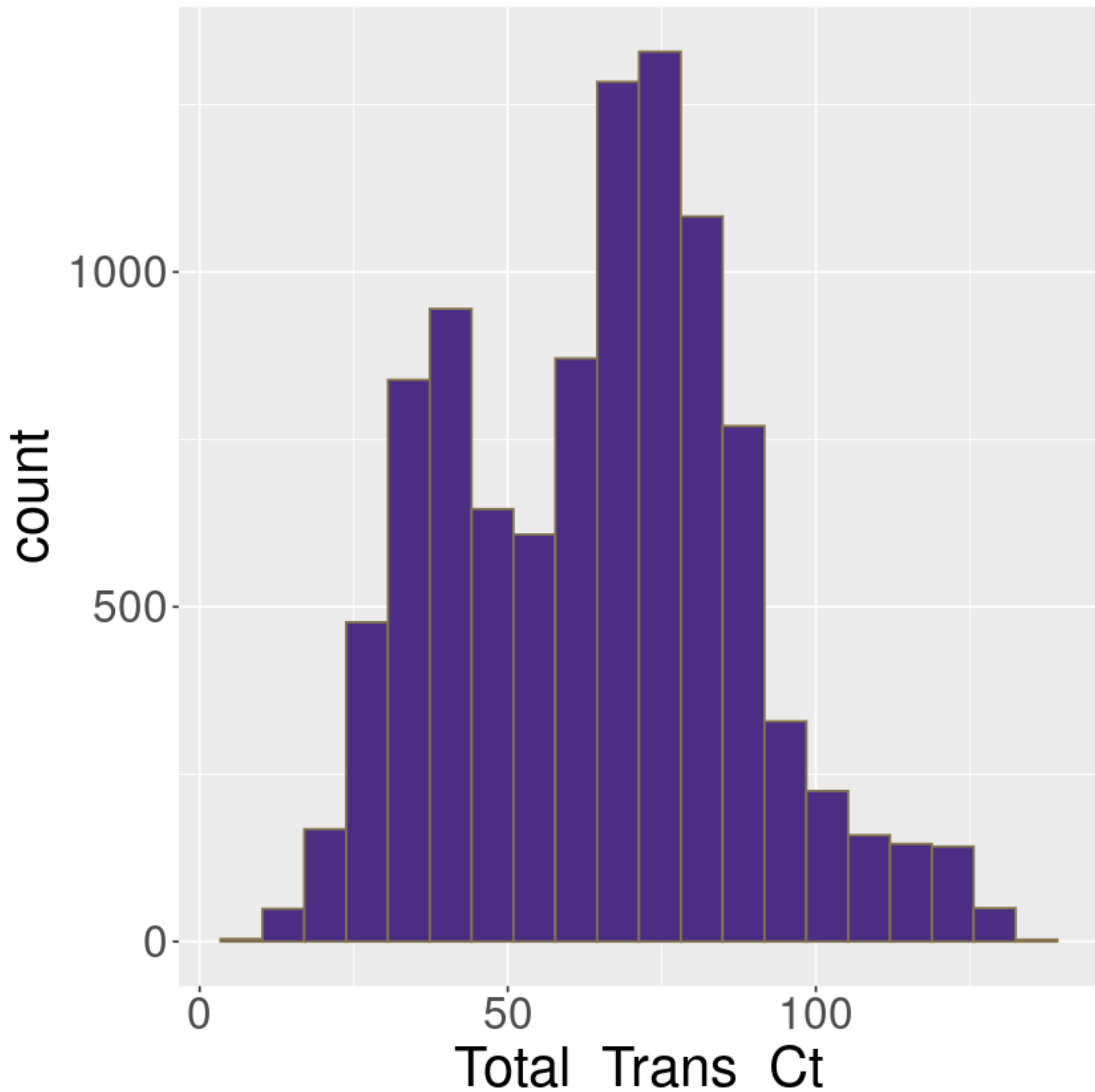
[727]:
```
attrition_gg <- ggplot(data = bank) + geom_bar(aes(x=Attrition_Flag), fill =
↪"#4b2e83", color = "#85754d")
attrition_gg + theme(text = element_text(size = 25))
```



[728]:
```
Trans_Amt_gg <- ggplot(data = bank) + geom_histogram(aes(x=Total_Trans_Amt),␣
 ↪fill = "#4b2e83", color = "#85754d", bins = 20)
Trans_Amt_gg + theme(text = element_text(size = 20))
```



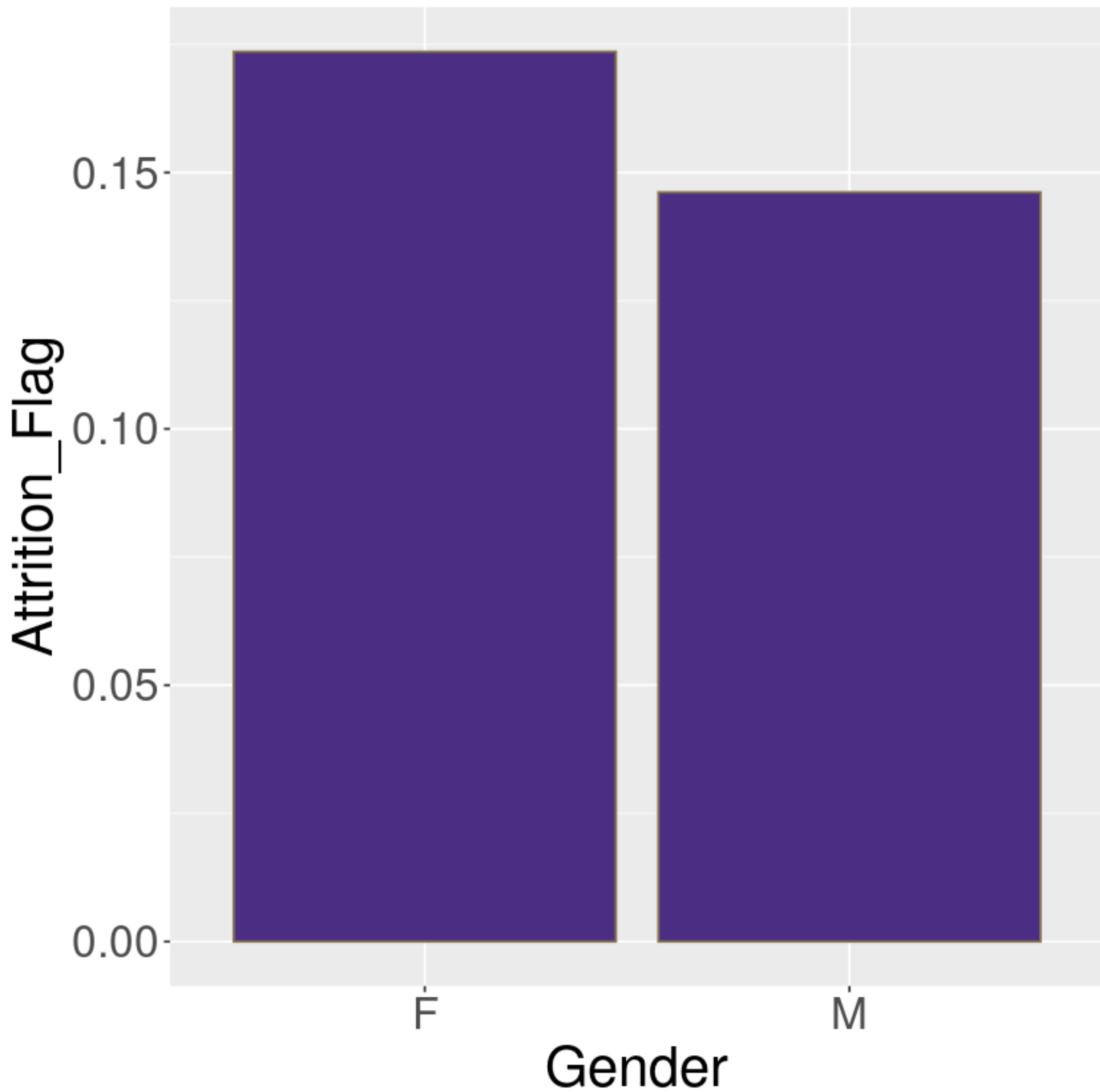
[729]:
```
Trans_Ct_gg <- ggplot(data = bank) + geom_histogram(aes(x=Total_Trans_Ct), fill␣
↪= "#4b2e83", color = "#85754d", bins = 20)
Trans_Ct_gg + theme(text = element_text(size = 25))
```



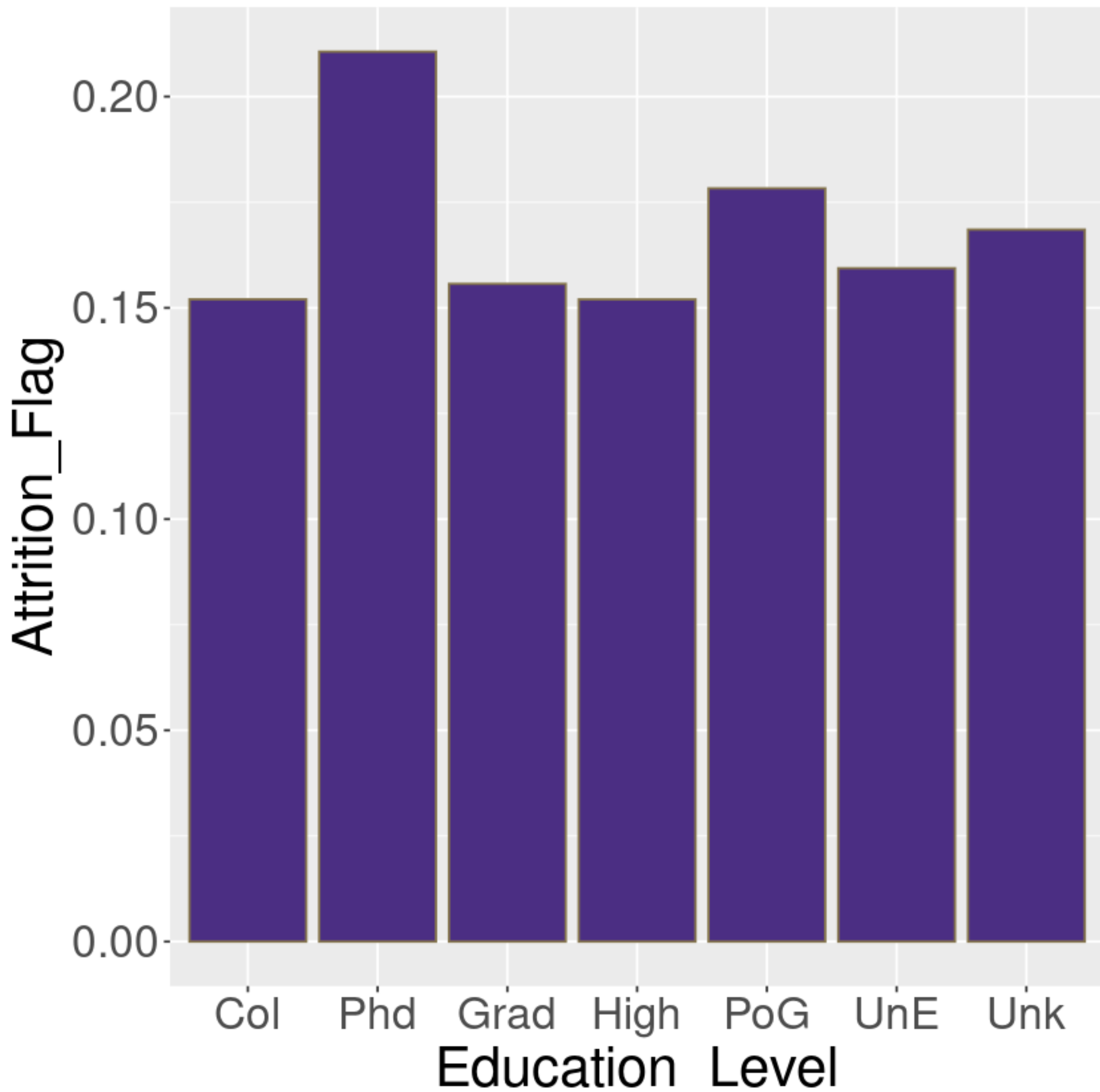
1.4.1 Inspecting the relationships between target variable and categorical variables

- Gender
- Education_Level
- Marital_Status
- Income_Category
- Card_Category

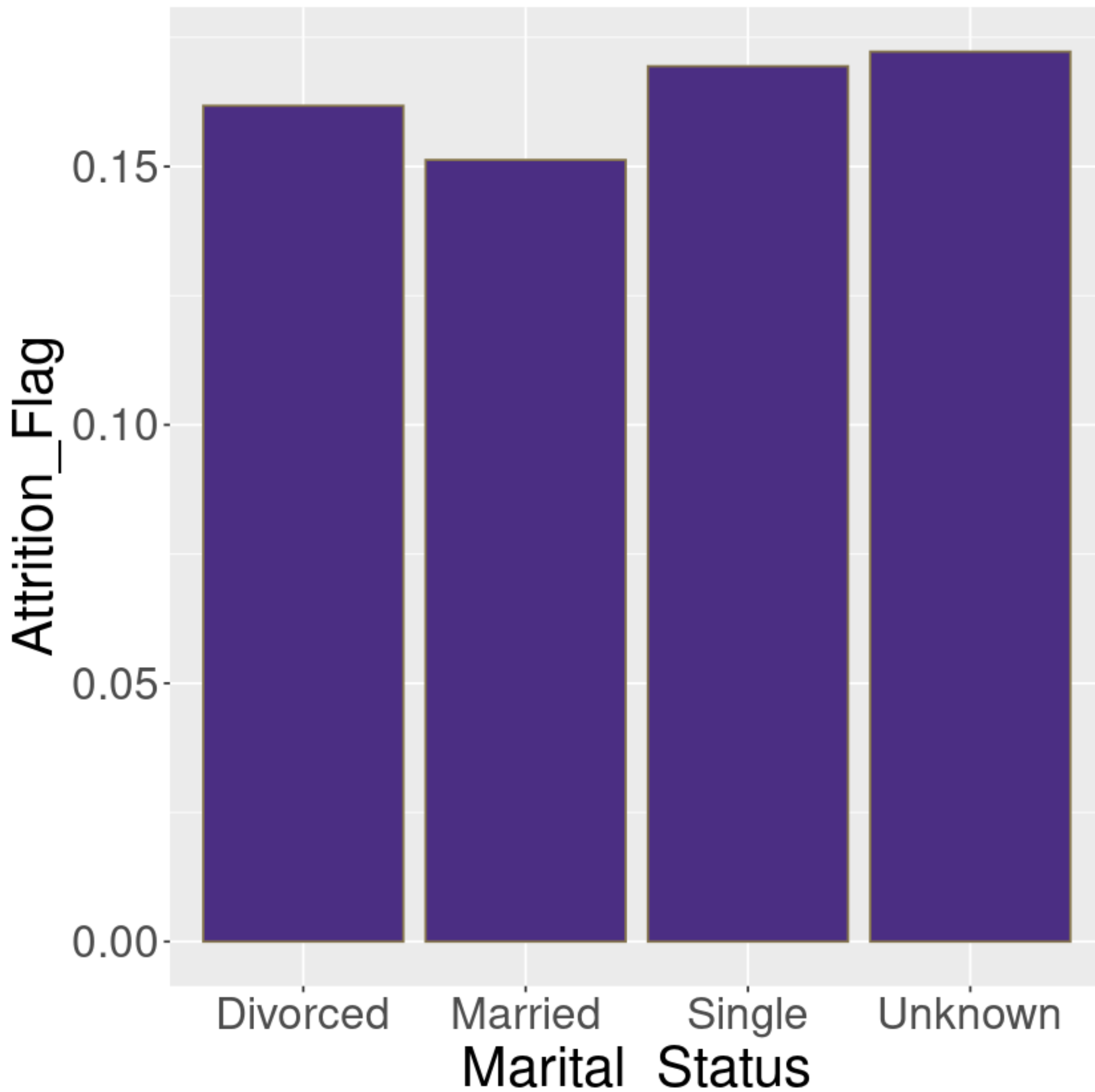
[730]:
```
gender_gg <- ggplot(data = bank) + geom_bar(aes(x = Gender, y =␣
↪Attrition_Flag),  fill = "#4b2e83", color = "#85754d",stat = "summary", fun␣
↪= "mean")
gender_gg + theme(text = element_text(size = 25))
```



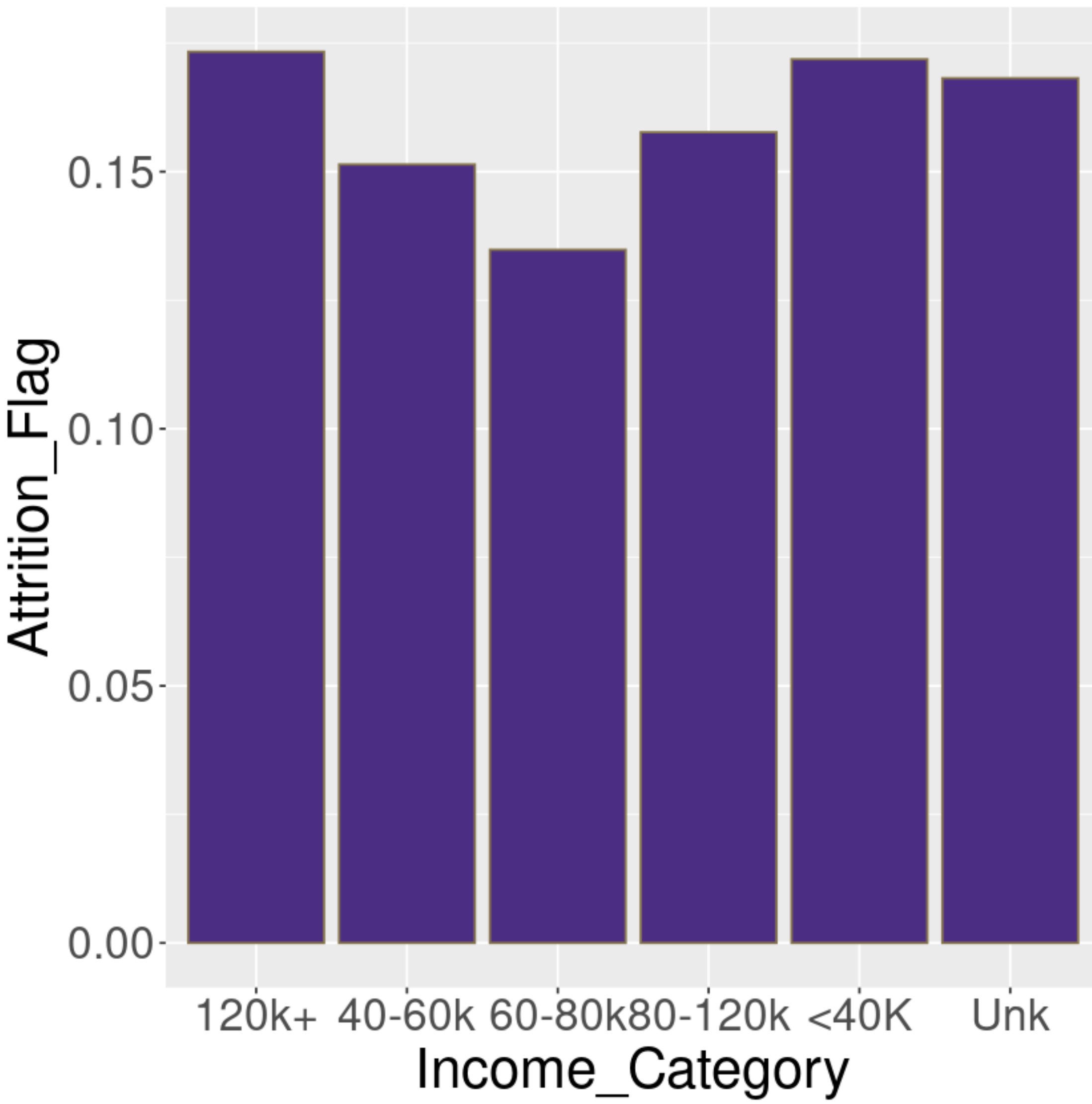
[731]:
```
levels(bank$Education_Level) <- c('Col','Phd','Grad','High','PoG','UnE','Unk')

edu_gg <- ggplot(data = bank) + geom_bar(aes(x = Education_Level, y =␣
 ↪Attrition_Flag), fill = "#4b2e83", color = "#85754d",stat = "summary", fun =␣
 ↪"mean")
edu_gg + theme(text = element_text(size = 25))

levels(bank$Education_Level) <- c('College','Doctorate','Graduate','High␣
 ↪School','Post-Graduate','Uneducated','Unknown')
```



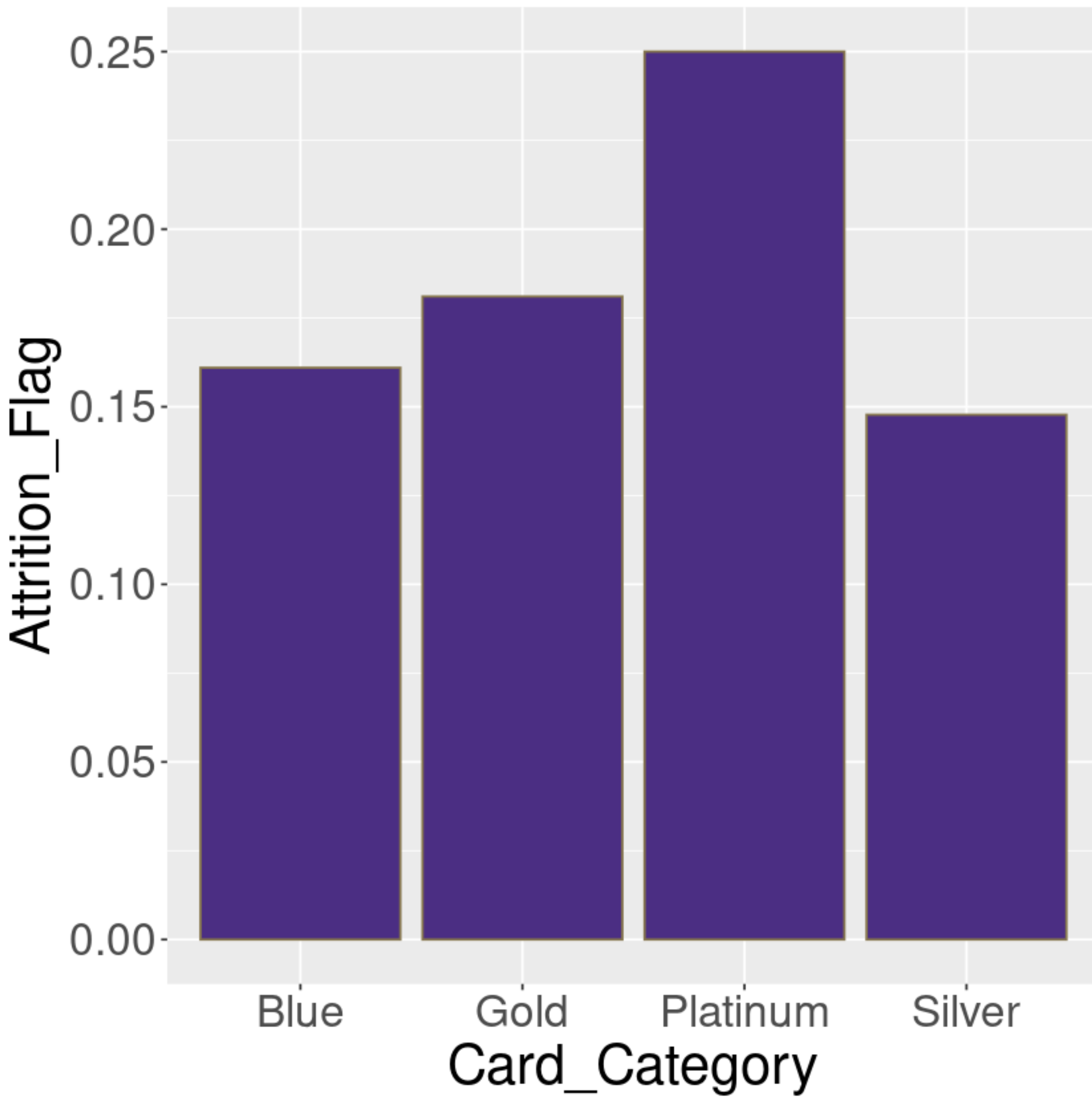
[732]:
```
mar_gg <- ggplot(data = bank) + geom_bar(aes(x = Marital_Status, y =
↪Attrition_Flag), fill = "#4b2e83", color = "#85754d", stat = "summary", fun
↪= "mean")
mar_gg + theme(text = element_text(size = 25))
```



[733]:
```
# rename to show the income more clearly
levels(bank$Income_Category) <- c('120k+','40-60k','60-80k','80-120k',␣
↪'<40K','Unk')

income_gg <- ggplot(data = bank) + geom_bar(aes(x = Income_Category, y =␣
↪Attrition_Flag),fill = "#4b2e83", color = "#85754d", stat = "summary", fun =␣
↪"mean")
income_gg + theme(text = element_text(size = 25))

#levels(bank$Income_Category)

levels(bank$Income_Category) <- c('$120K +','$40K - $60K','$60K - $80K','$80K -␣
↪$120K','Less than $40K','Unknown')
```



[734]:
```
card_gg <- ggplot(data = bank) + geom_bar(aes(x = Card_Category, y =
↪Attrition_Flag), fill = "#4b2e83", color = "#85754d", stat = "summary", fun
↪= "mean")
card_gg + theme(text = element_text(size = 25))
```



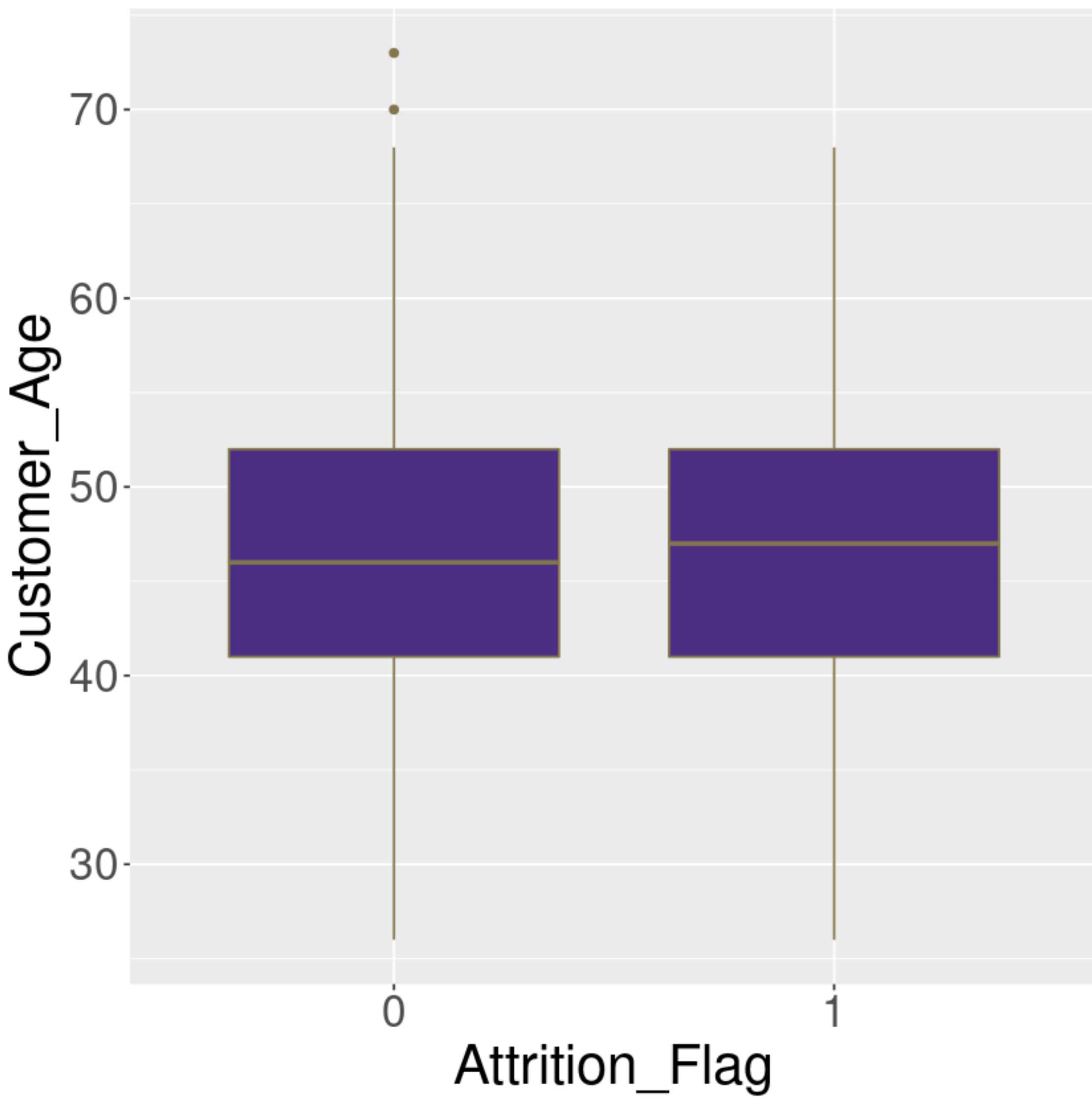
1.4.2 Inspecting the Relationships between Target Variable and Numerical Variables

- Customer_Age
- Dependent_count
- Months_on_book
- Total_Relationship_Count
- Months_Inactive_12_mon
- Contacts_Count_12_mon
- Credit_Limit
- Total_Revolving_Bal
- Avg_Open_To_Buy
- Total_Amt_Chng_Q4_Q1
- Total_Trans_Amt

- Total_Trans_Ct
- Total_Ct_Chng_Q4_Q1
- Avg_Utilization_Ratio

[735]:

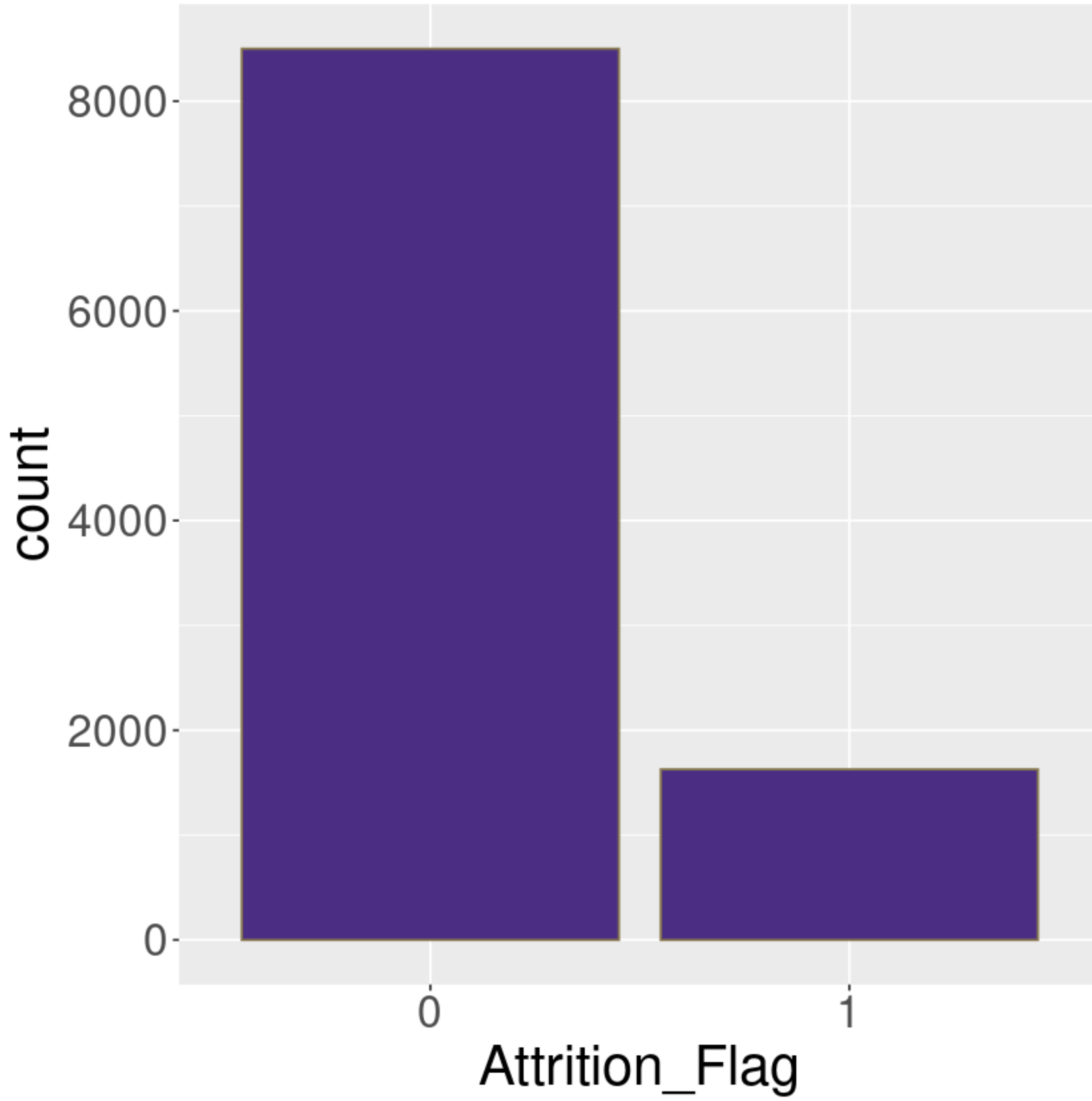
```
bank$Attrition_Flag <- factor(bank$Attrition_Flag)
gender_gg <- ggplot(data = bank) + geom_boxplot(aes(x = Attrition_Flag, y =␣
↪Customer_Age ),fill = "#4b2e83", color = "#85754d")
gender_gg + theme(text = element_text(size = 25))

# higher age, higher attrtion
```



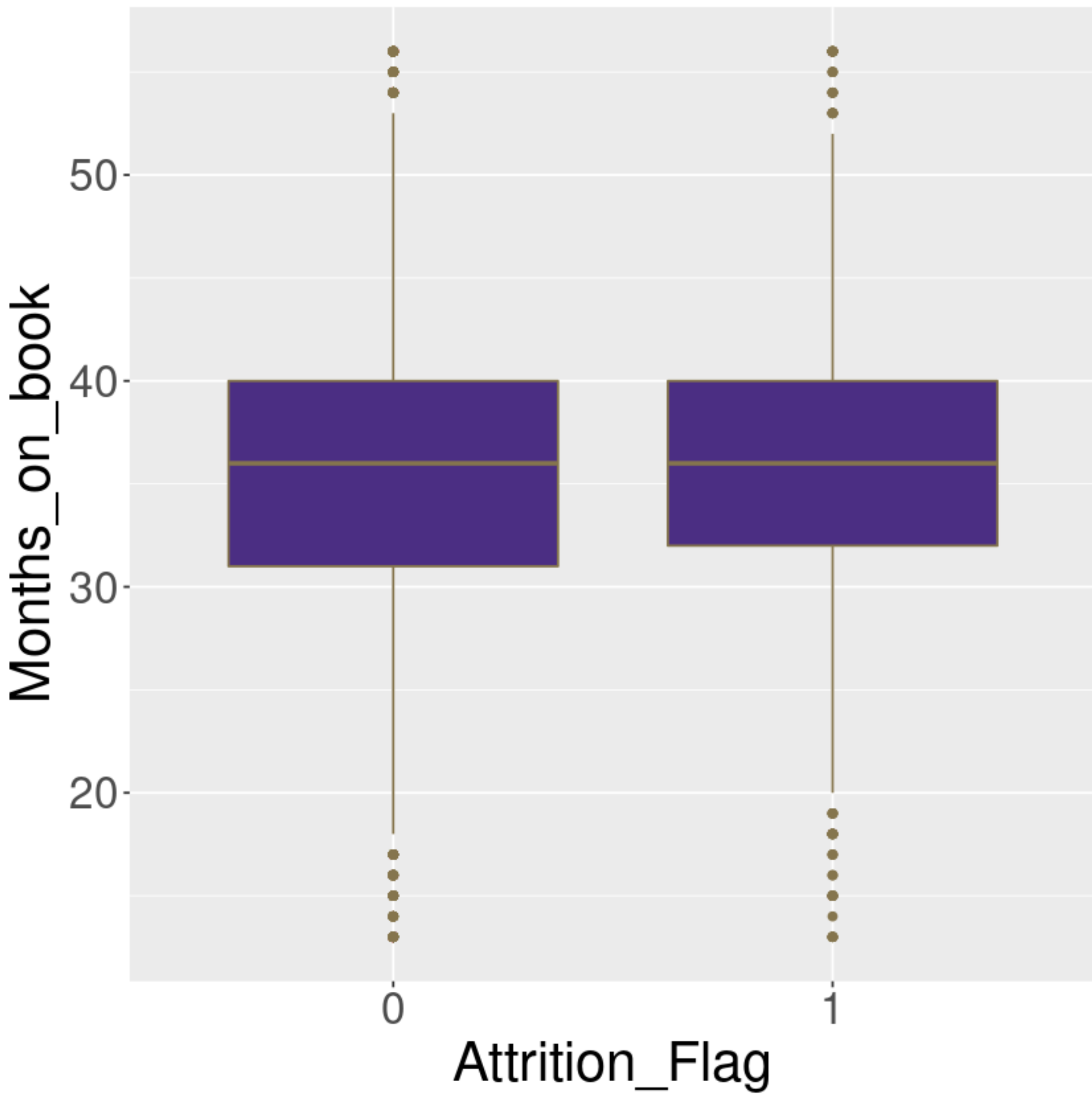
[736]:

```
# Improve the visualization: as.factor for Attrition_Flag
```

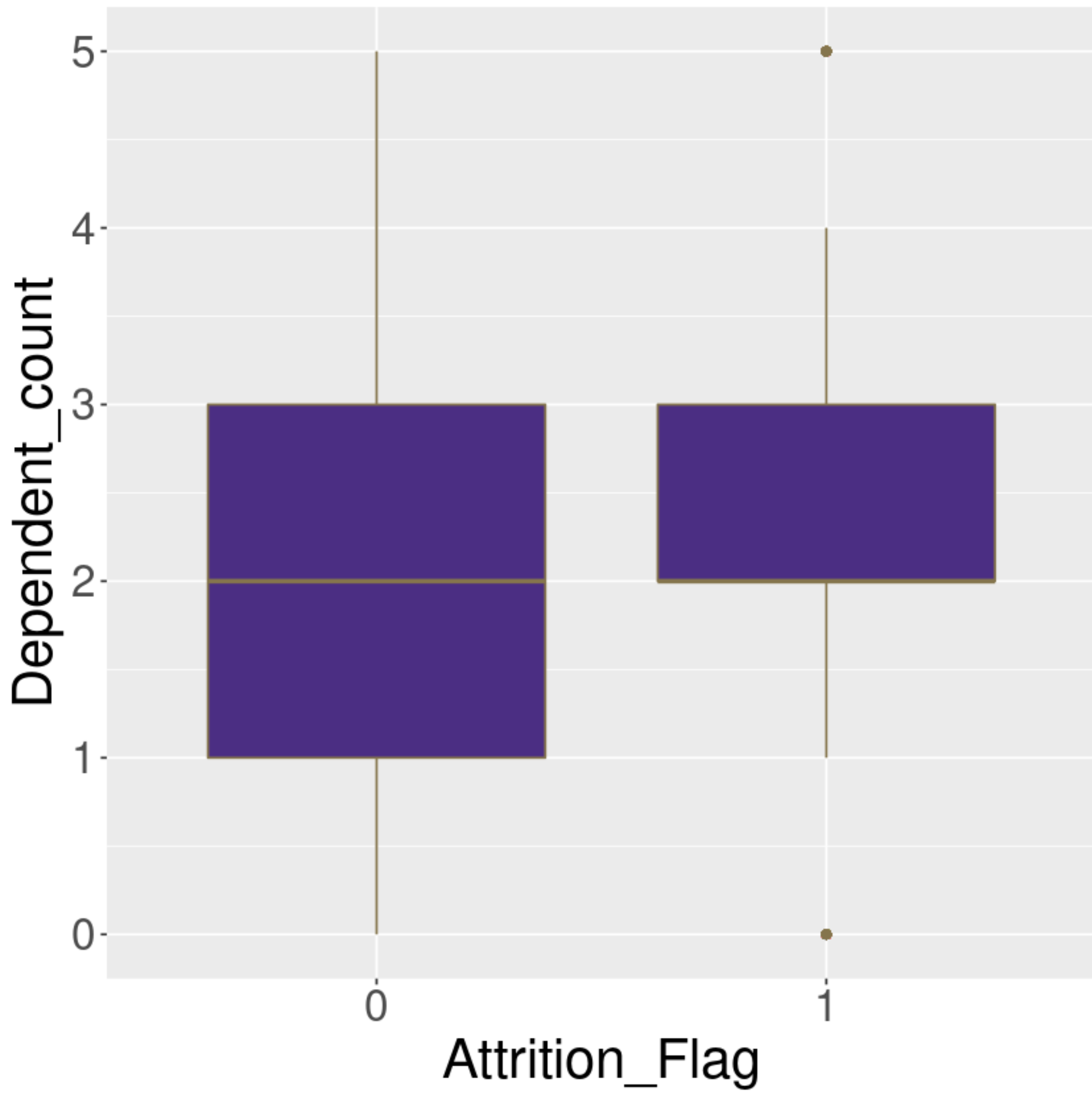
```
attrition_gg <- ggplot(data = bank) + geom_bar(aes(x=Attrition_Flag), fill =␣
↪"#4b2e83", color = "#85754d")
attrition_gg + theme(text = element_text(size = 25))
```



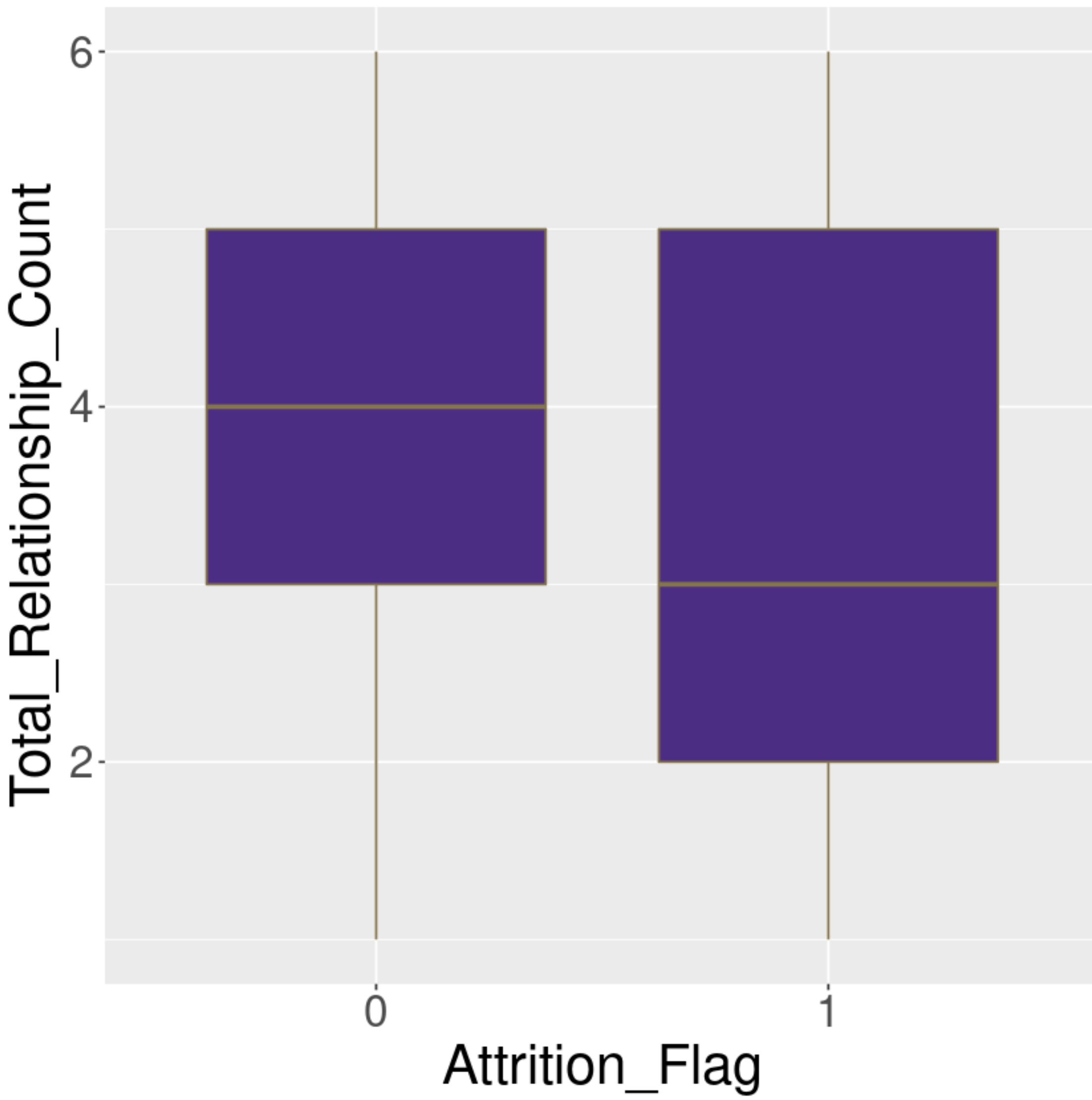
[737]:
```
month_gg <- ggplot(data = bank) + geom_boxplot(aes(x = Attrition_Flag, y=␣
↪Months_on_book), fill = "#4b2e83", color = "#85754d")
month_gg + theme(text = element_text(size = 25))
```



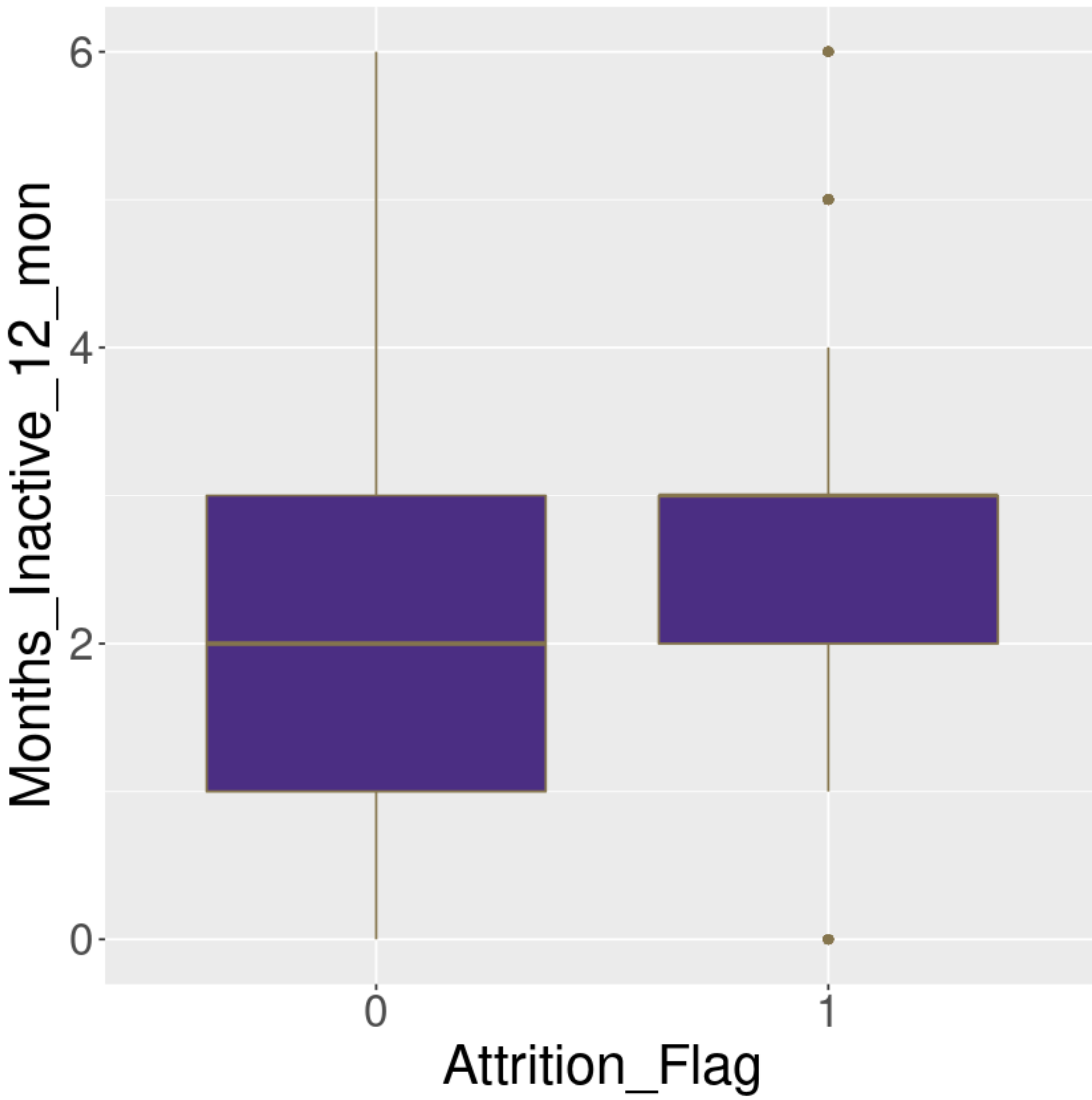
[738]:
```
de_gg <- ggplot(data = bank) + geom_boxplot(aes(x = Attrition_Flag, y =␣
↪Dependent_count), fill = "#4b2e83", color = "#85754d")

de_gg + theme(text = element_text(size = 25))
```



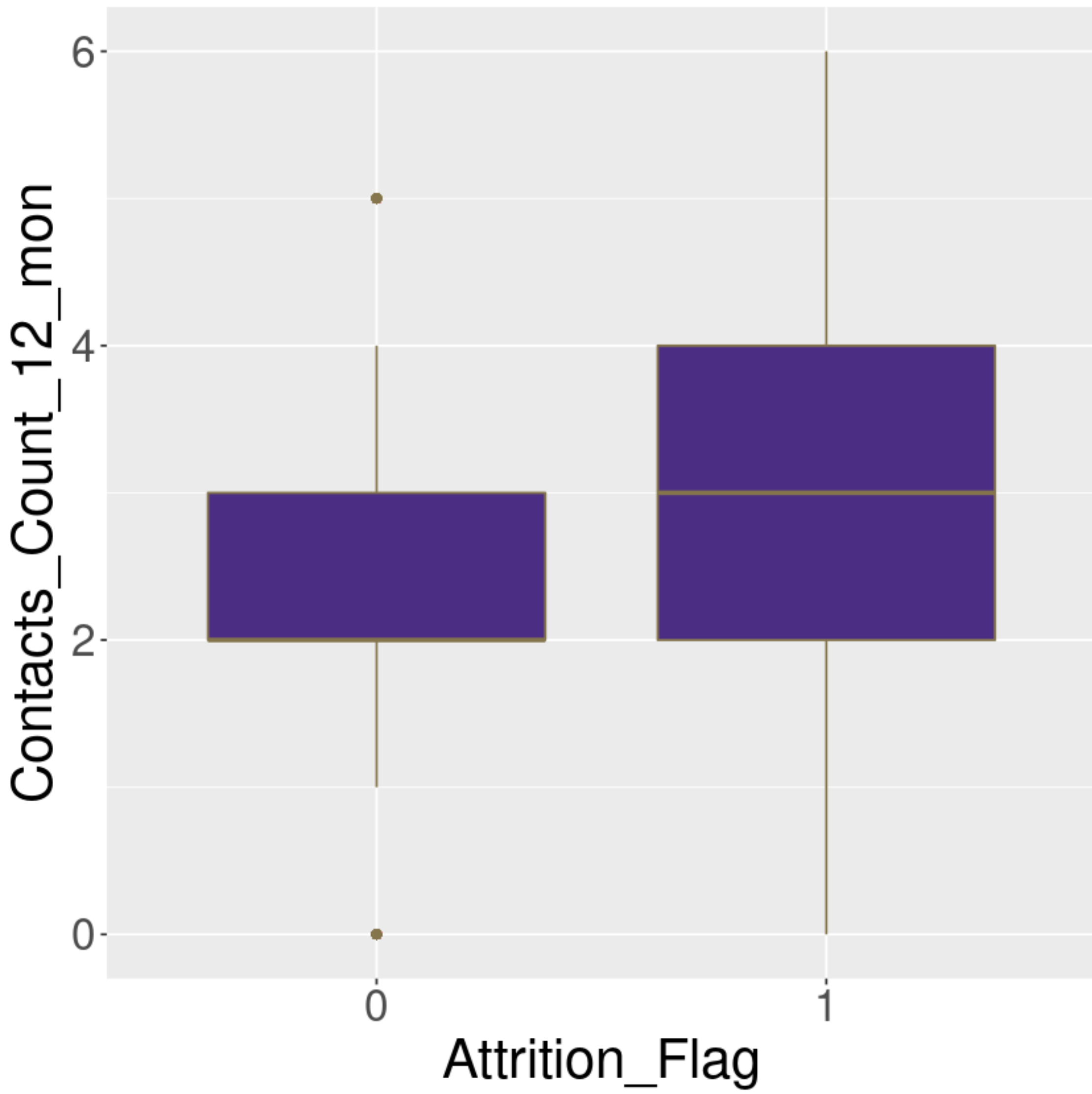
[739]:
```
relation_gg <- ggplot(data = bank) + geom_boxplot(aes(x = Attrition_Flag, y=
    Total_Relationship_Count), fill = "#4b2e83", color = "#85754d")

relation_gg + theme(text = element_text(size = 25))
```



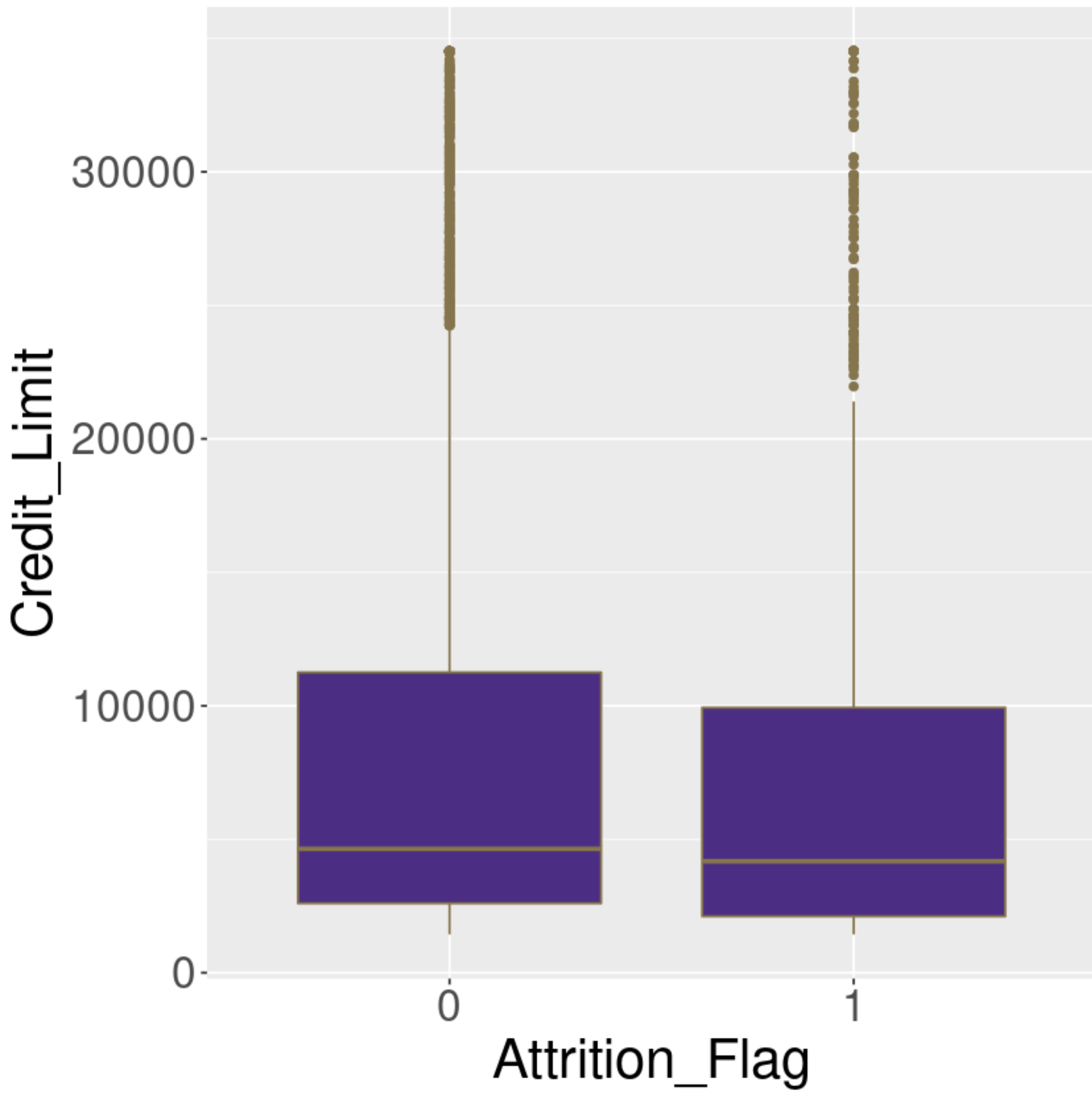
[740]:
```
inative_gg <- ggplot(data = bank) + geom_boxplot(aes(x = Attrition_Flag, y=␣
↪Months_Inactive_12_mon), fill = "#4b2e83", color = "#85754d")

inative_gg + theme(text = element_text(size = 25))
```



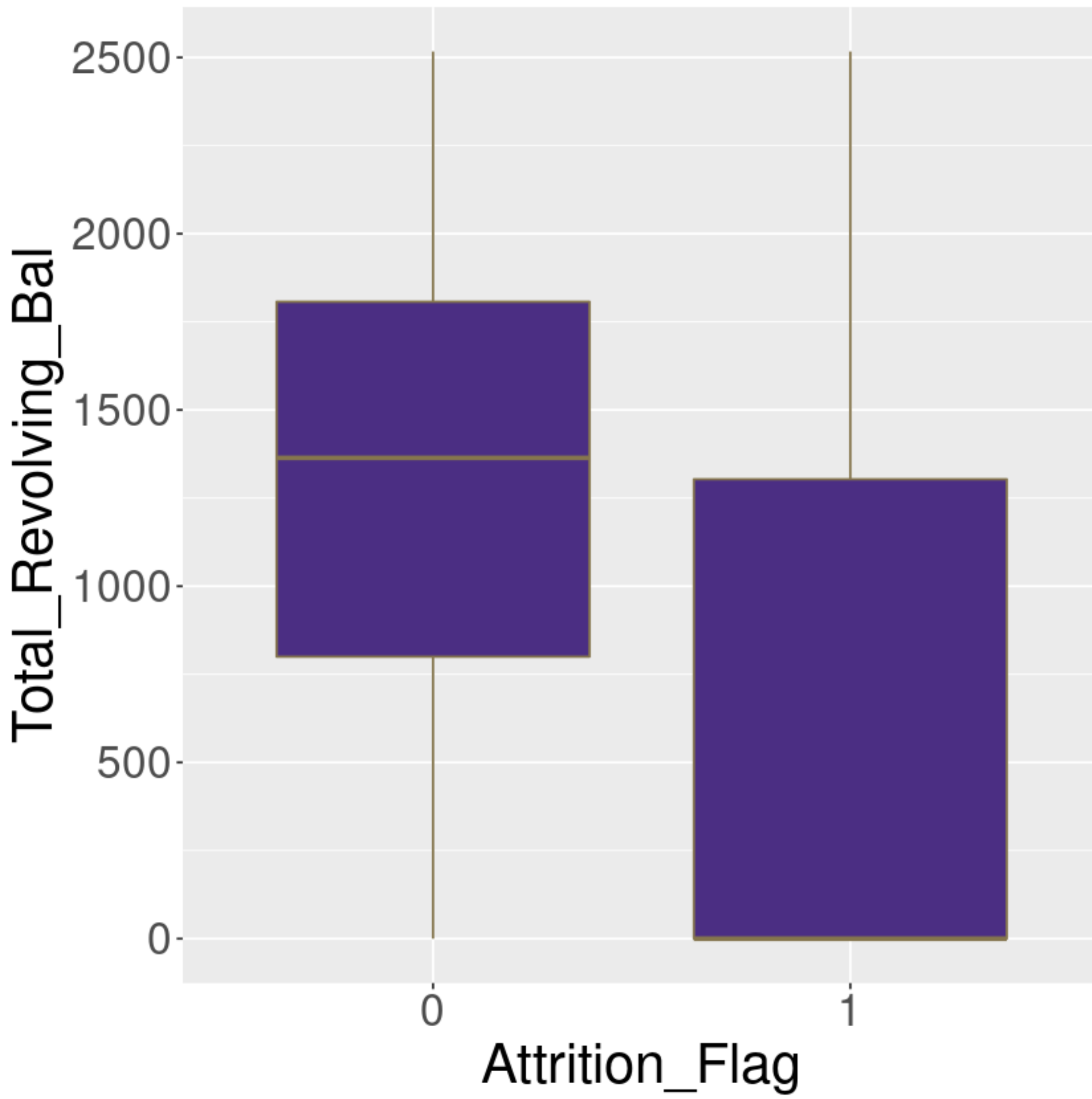
[741]:
```
contact_gg <- ggplot(data = bank) + geom_boxplot(aes(x = Attrition_Flag, y=␣
↪Contacts_Count_12_mon), fill = "#4b2e83", color = "#85754d")

contact_gg + theme(text = element_text(size = 25))
```



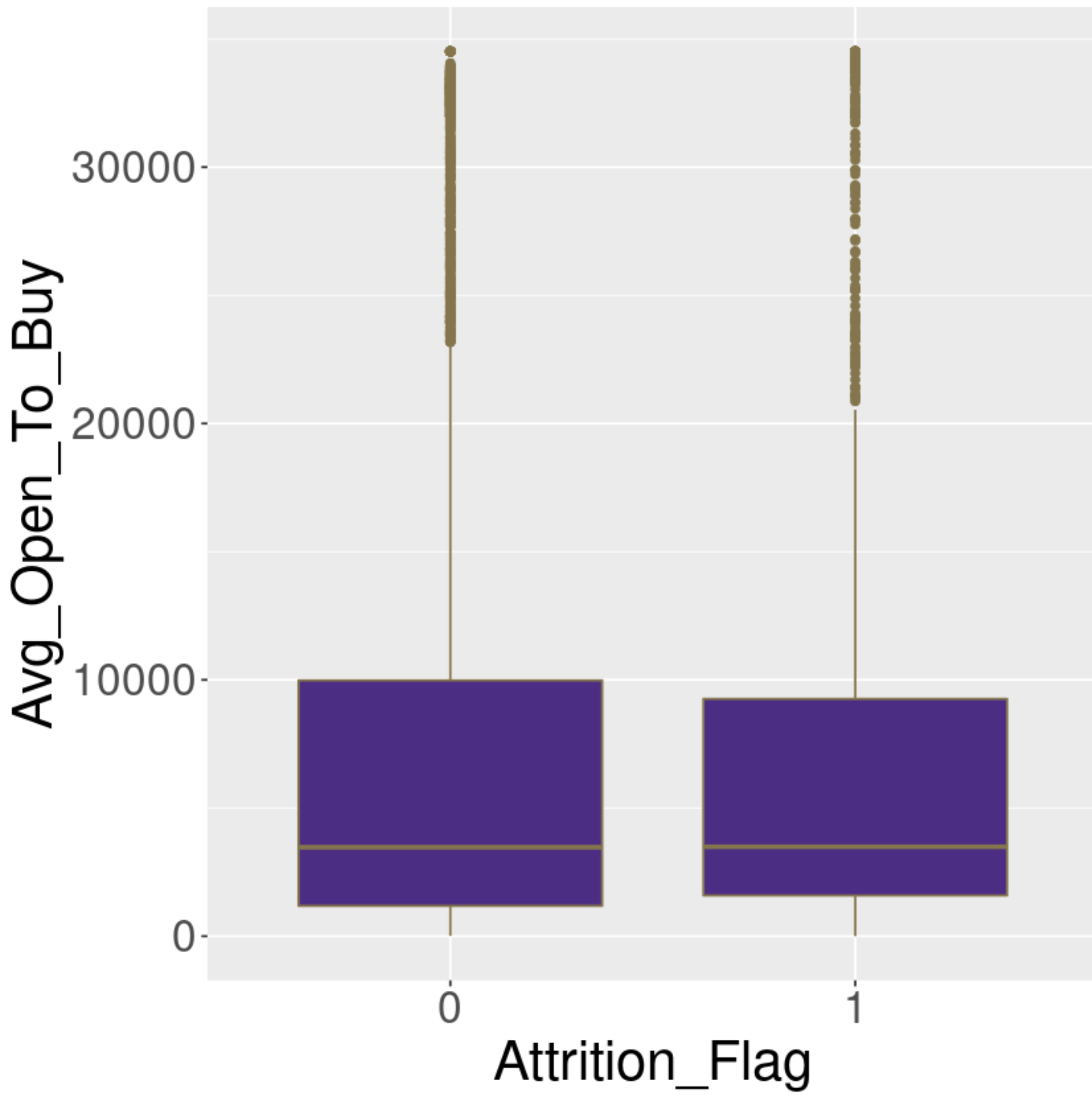
[742]:
```
credit_gg<- ggplot(data = bank) + geom_boxplot(aes(x = Attrition_Flag, y=
 ↪Credit_Limit), fill = "#4b2e83", color = "#85754d")

credit_gg + theme(text = element_text(size = 25))
```



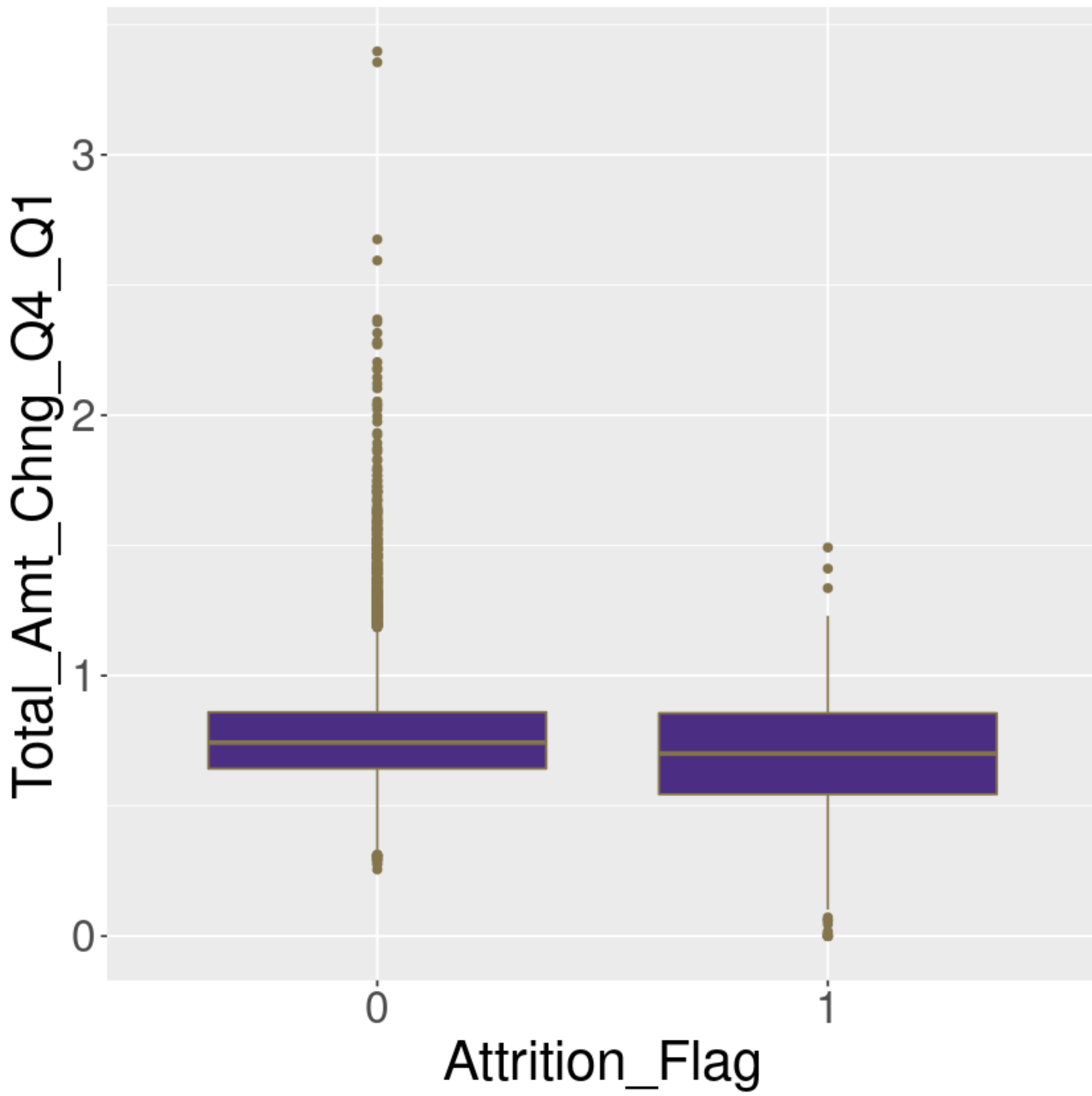
[743]:
```
rev_gg <- ggplot(data = bank) + geom_boxplot(aes(x = Attrition_Flag, y=
    Total_Revolving_Bal), fill = "#4b2e83", color = "#85754d")

rev_gg + theme(text = element_text(size = 25))
```



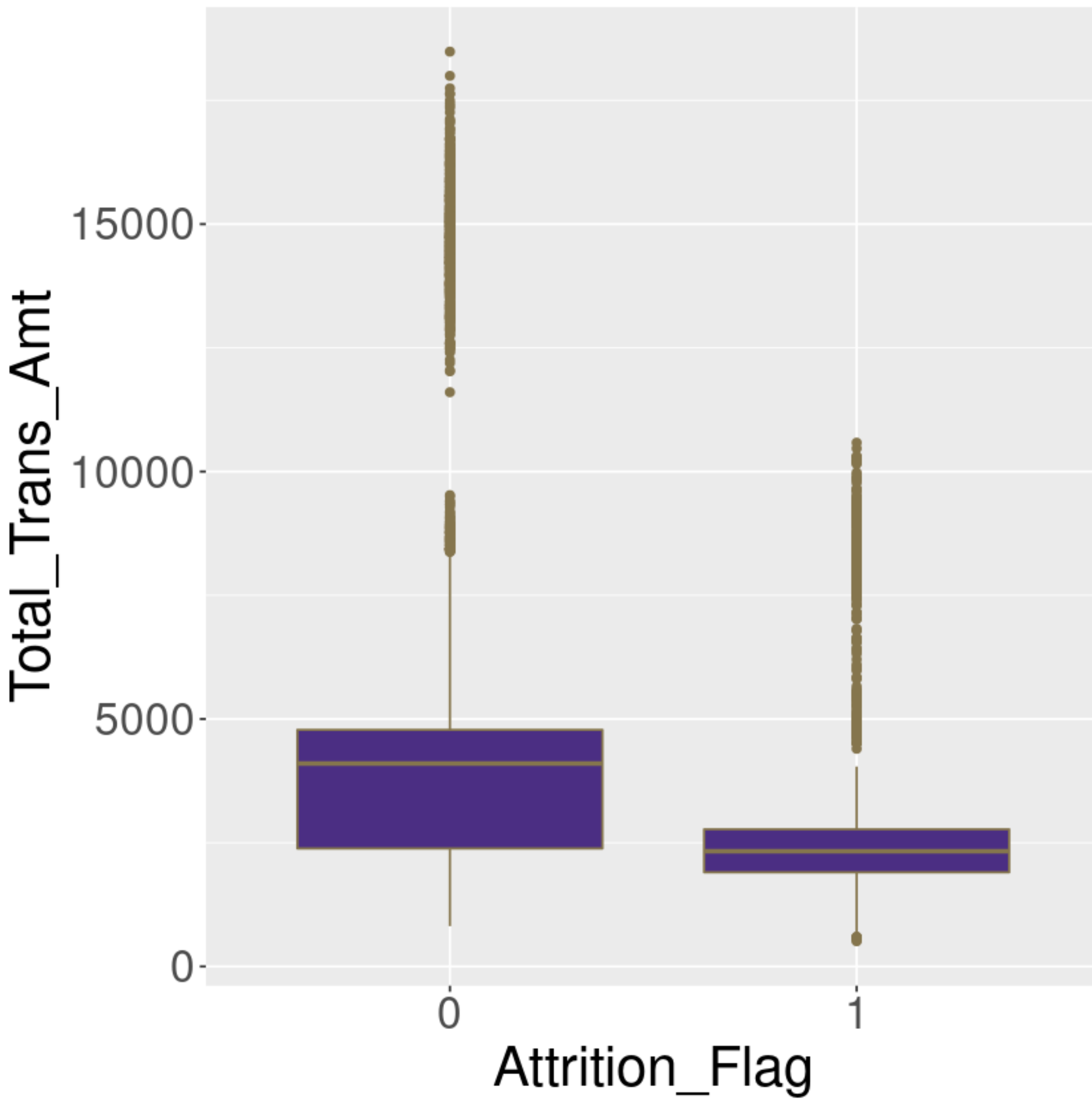
[744]:
```
open_gg <- ggplot(data = bank) + geom_boxplot(aes(x = Attrition_Flag, y=␣
↪Avg_Open_To_Buy), fill = "#4b2e83", color = "#85754d")

open_gg + theme(text = element_text(size = 25))
```



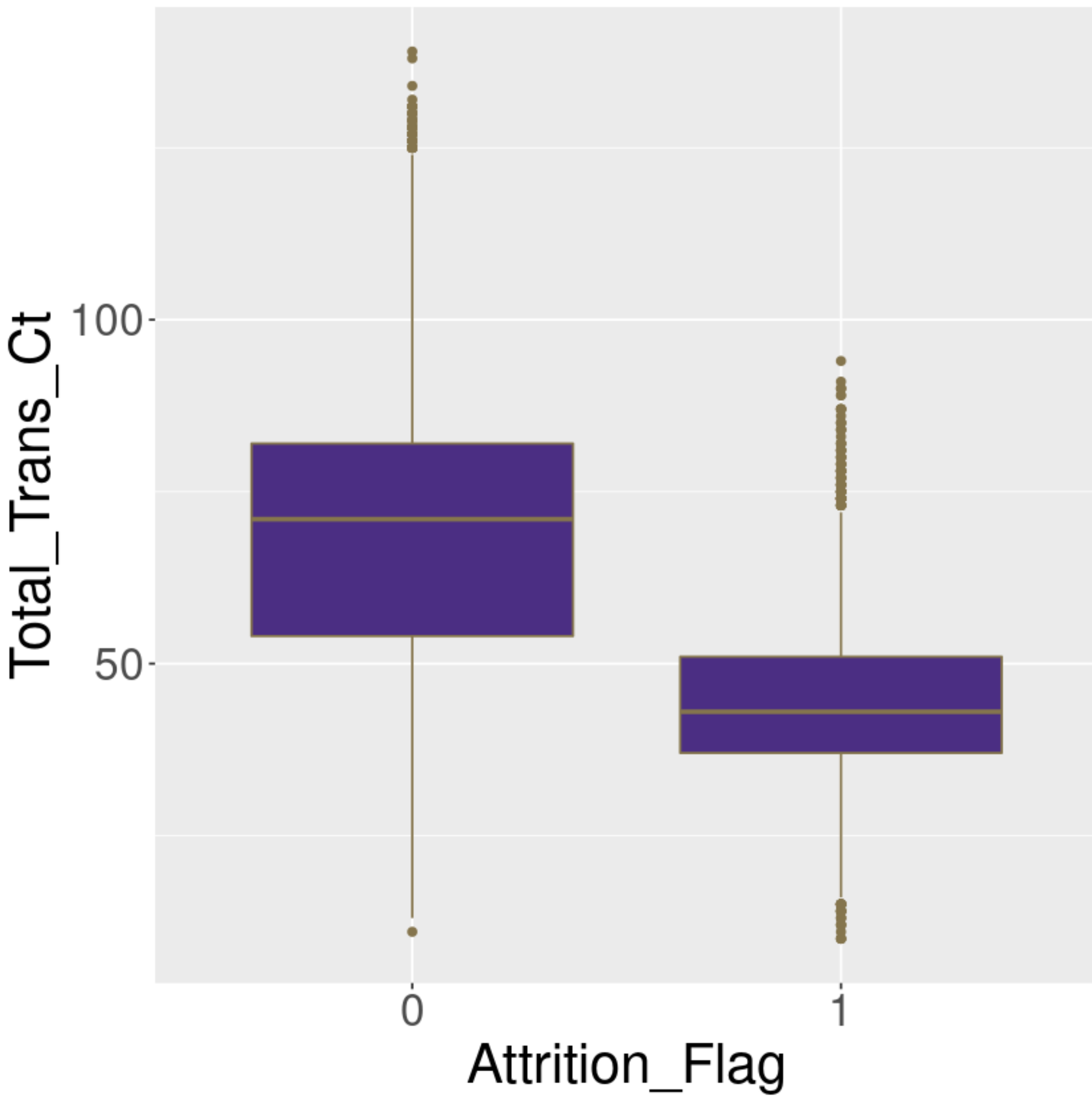
[745]:
```
amtq4_gg <-ggplot(data = bank) + geom_boxplot(aes(x = Attrition_Flag, y=␣
↪Total_Amt_Chng_Q4_Q1), fill = "#4b2e83", color = "#85754d")

amtq4_gg + theme(text = element_text(size = 25))
```



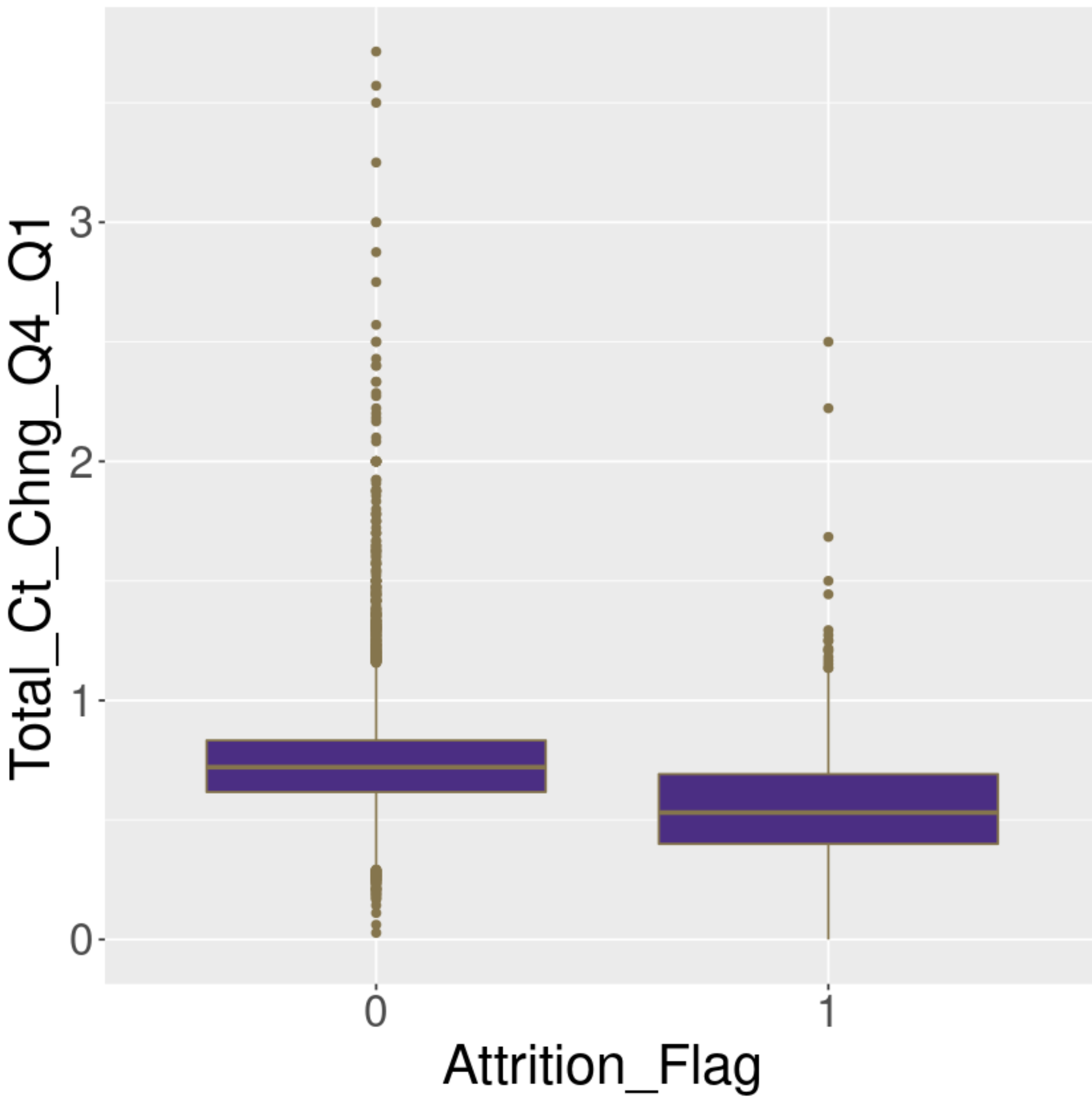
[746]:
```
amt_gg <- ggplot(data = bank) + geom_boxplot(aes(x = Attrition_Flag, y=
 ↪Total_Trans_Amt), fill = "#4b2e83", color = "#85754d")

amt_gg + theme(text = element_text(size = 25))
```



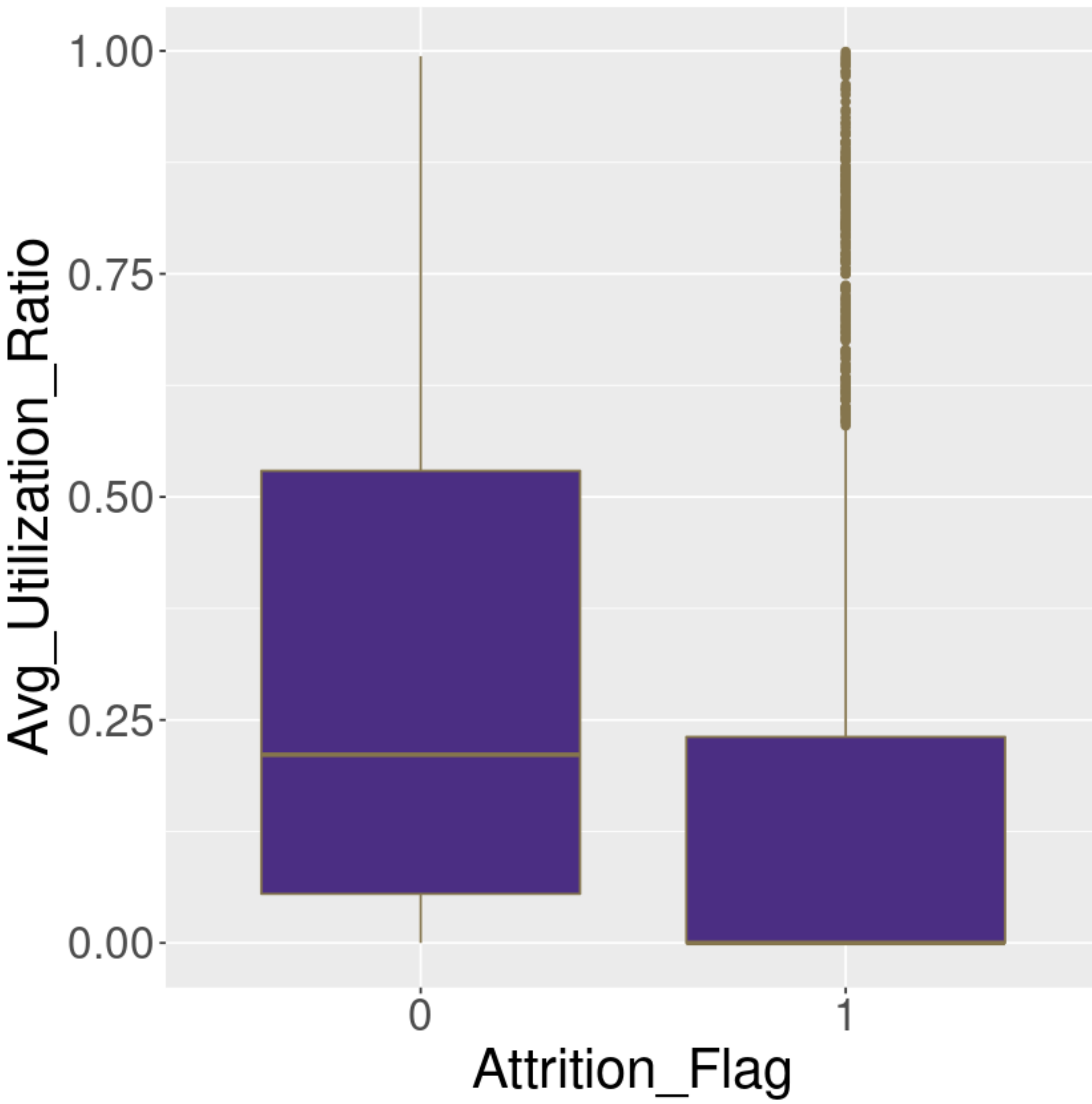
[747]:
```
ct_gg <- ggplot(data = bank) + geom_boxplot(aes(x = Attrition_Flag, y=␣
↪Total_Trans_Ct), fill = "#4b2e83", color = "#85754d")

ct_gg + theme(text = element_text(size = 25))
```



[748]:
```
ctq_gg <- ggplot(data = bank) + geom_boxplot(aes(x = Attrition_Flag, y=
↪Total_Ct_Chng_Q4_Q1),fill = "#4b2e83", color = "#85754d")

ctq_gg + theme(text = element_text(size = 25))
```



[749]:
```
uti_gg <- ggplot(data = bank) + geom_boxplot(aes(x = Attrition_Flag, y=
  ↪Avg_Utilization_Ratio), fill = "#4b2e83", color = "#85754d")

uti_gg + theme(text = element_text(size = 25))
```



1.4.3 Feature Engineering:

- Categorical Variables: Gender, Education_Level, Marital_Status,Income_Category, Card_Category
- Numerical Variables: Months_Inactive_12_mon, Total_Revolving_Bal, Total_Trans_Amt, Total_Trans_Ct, Total_Ct_Chng_Q4_Q1, Avg_Utilization_Ratio

1.5 Supervised Learning

- Classification Model: Decision Tree, Logistic Regression, Random Forest, XGBoost to classify if a customer is going to drop off.
- Compare different models

1.5.1 Decision Tree Model

[750]:
```
str(bank)
```

```
'data.frame':	10127 obs. of  21 variables:
 $ CLIENTNUM               : int  768805383 818770008 713982108 769911858
709106358 713061558 810347208 818906208 710930508 719661558 …
 $ Attrition_Flag          : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 …
 $ 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 1.41 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 …
```

Partition the records into 70% training vs. 30% validation with the seed set to 1234. To create a train and validation set, we are going to split randomly this data set into 70% training set and 30% validation set.

We have nrow(bank) data points in total. We find the observations we want to include in the training set by randomly sample 70% $\times$ nrow(bank) indexes out of nrow(bank) observations.

Random seed is set using set.seed(1234) to control reproducibility so that every time you run the random partitioning, it will give you the same result.

[751]:
```
set.seed(1234)  # set seed for reproducing the partition
# Random sample indexes
train.index <- sample(1:nrow(bank), nrow(bank)*0.7)
train.df <- bank[train.index, ]
valid.df <- bank[-train.index, ]
```

Build classification tree from training data. Overfitting can definitely happen if your tree is too big. There are two ways to limit the tree size. 1. Set criteria to stop tree growth. 2. First grow the tree to full size, and then prune it back. In order to use the CART model in R, we need to install and use the rpart library.

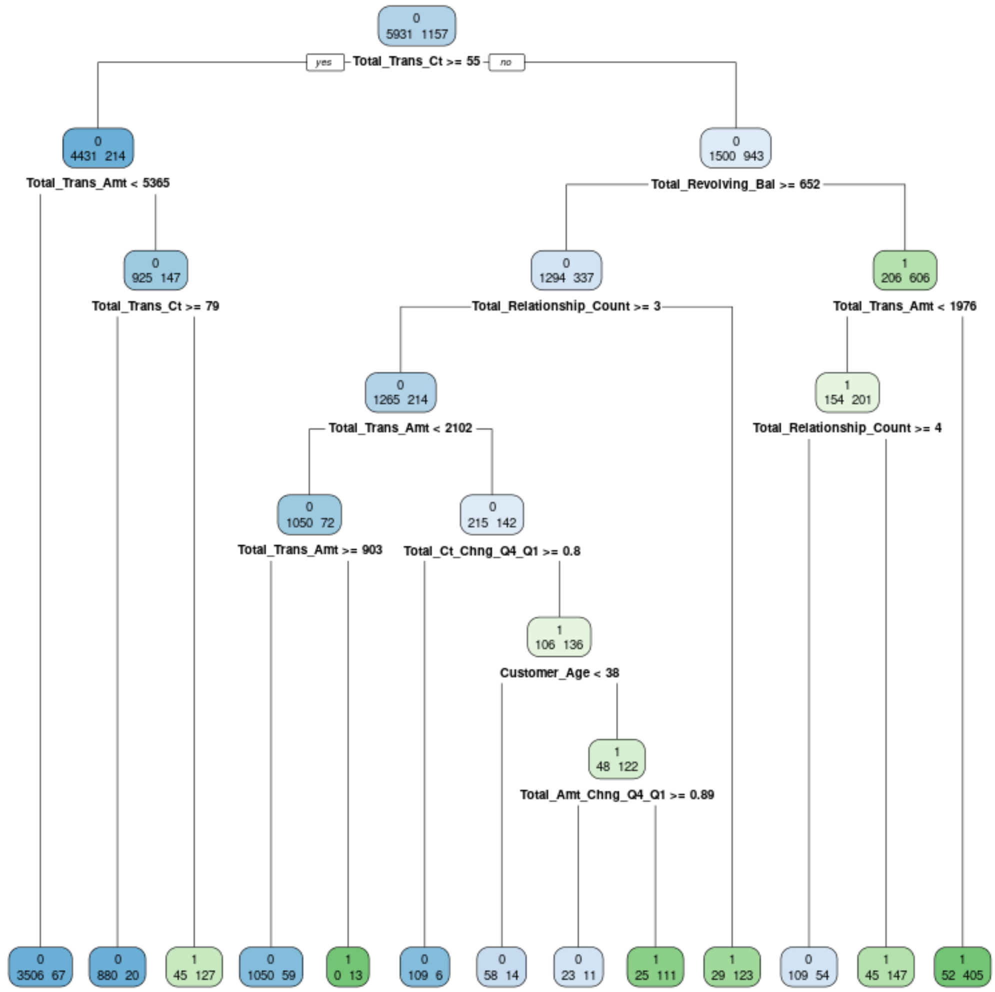
[752]:
```
library(rpart)
```

[753]:
```
install.packages("rpart.plot")
```

```
Installing rpart.plot [3.1.1] …
        OK [linked cache]
```

[754]:
```
library(rpart.plot)
```

[755]:
```
# Default classification tree
default.ct <- rpart(Attrition_Flag ~ ., data = train.df[,-1], method = "class")␣
↪# remove first row (id) for the model
# plot tree
rpart.plot(default.ct, extra = 1)
```



[756]:
```
# Interpret the fitted tree
default.ct
```

```
n= 7088

node), split, n, loss, yval, (yprob)
      * denotes terminal node

  1) root 7088 1157 0 (0.83676637 0.16323363)
    2) Total_Trans_Ct>=54.5 4645  214 0 (0.95392896 0.04607104)
      4) Total_Trans_Amt< 5365 3573   67 0 (0.98124825 0.01875175) *
      5) Total_Trans_Amt>=5365 1072  147 0 (0.86287313 0.13712687)
       10) Total_Trans_Ct>=78.5 900   20 0 (0.97777778 0.02222222) *
```

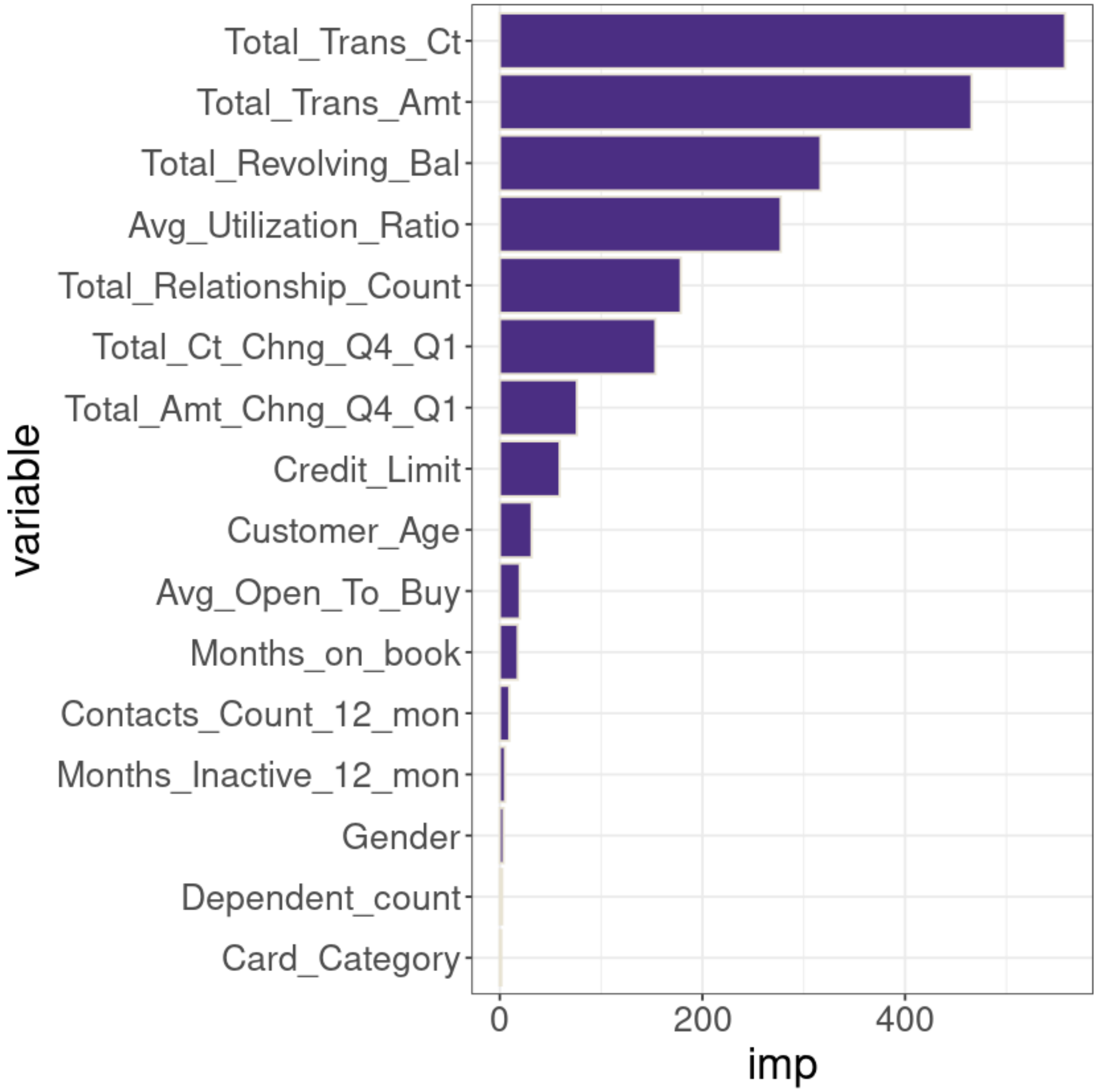
```
      11) Total_Trans_Ct< 78.5 172   45 1 (0.26162791 0.73837209) *
   3) Total_Trans_Ct< 54.5 2443  943 0 (0.61399918 0.38600082)
     6) Total_Revolving_Bal>=651.5 1631  337 0 (0.79337830 0.20662170)
      12) Total_Relationship_Count>=2.5 1479  214 0 (0.85530764 0.14469236)
        24) Total_Trans_Amt< 2101.5 1122   72 0 (0.93582888 0.06417112)
          48) Total_Trans_Amt>=902.5 1109   59 0 (0.94679892 0.05320108) *
          49) Total_Trans_Amt< 902.5 13    0 1 (0.00000000 1.00000000) *
        25) Total_Trans_Amt>=2101.5 357  142 0 (0.60224090 0.39775910)
          50) Total_Ct_Chng_Q4_Q1>=0.7965 115    6 0 (0.94782609 0.05217391) *
          51) Total_Ct_Chng_Q4_Q1< 0.7965 242  106 1 (0.43801653 0.56198347)
           102) Customer_Age< 37.5 72   14 0 (0.80555556 0.19444444) *
           103) Customer_Age>=37.5 170   48 1 (0.28235294 0.71764706)
             206) Total_Amt_Chng_Q4_Q1>=0.8865 34   11 0 (0.67647059 0.
↪32352941) *
             207) Total_Amt_Chng_Q4_Q1< 0.8865 136   25 1 (0.18382353 0.
↪81617647) *
      13) Total_Relationship_Count< 2.5 152   29 1 (0.19078947 0.80921053) *
     7) Total_Revolving_Bal< 651.5 812  206 1 (0.25369458 0.74630542)
      14) Total_Trans_Amt< 1975.5 355  154 1 (0.43380282 0.56619718)
        28) Total_Relationship_Count>=3.5 163   54 0 (0.66871166 0.33128834) *
        29) Total_Relationship_Count< 3.5 192   45 1 (0.23437500 0.76562500) *
      15) Total_Trans_Amt>=1975.5 457   52 1 (0.11378556 0.88621444) *
```

[757]:
```
# important features
default.ct$variable.importance
```

Total_Trans_Ct 557.868804263505 **Total_Trans_Amt** 465.500816720497 **Total_Revolving_Bal** 316.556841819563 **Avg_Utilization_Ratio** 277.286448387531 **Total_Relationship_Count** 178.571606378485 **Total_Ct_Chng_Q4_Q1** 153.563170129513 **Total_Amt_Chng_Q4_Q1** 76.0956202597641 **Credit_Limit** 59.1018114007548 **Customer_Age** 31.4002472662327 **Avg_Open_To_Buy** 19.6918219431393 **Months_on_book** 17.7034505304081 **Contacts_Count_12_mon** 9.36294565841625 **Months_Inactive_12_mon** 5.1014759967271 **Gender** 3.68881560775965 **Dependent_count** 2.27906610282253 **Card_Category** 1.44435162595001

[758]:
```
# plot the importance of variables
library(rpart)
library(tidyverse)
df <- data.frame(imp = default.ct$variable.importance)
df2 <- df %>%
  tibble::rownames_to_column() %>%
  dplyr::rename("variable" = rowname) %>%
  dplyr::arrange(imp) %>%
  dplyr::mutate(variable = forcats::fct_inorder(variable))
dt_imp <-  ggplot(df2) +
  geom_col(aes(x = variable, y = imp),fill = "#4b2e83",
           col = "#e8e3d3", show.legend = F) +
```

```
  coord_flip() +
  scale_fill_grey() +
  theme_bw()

dt_imp + theme(text = element_text(size = 20))
```



The result of important features (top 5) - Total_Trans_Amt - Total_Trans_Ct - Total_Revolving_Bal - Avg_Utilization_Ratio - Total_Relationship_Count

Performance Evaluation

[759]:
```
# classify records in the validation data.
# set argument type = "class" in predict() to generate predicted class␣
↪membership.
```

```
# Otherwise, a probablity of belonging to each class
default.ct.point.pred <- predict(default.ct, valid.df[,-1], type = "class")
```

[760]:
```
install.packages("caret")
```

```
Installing caret [6.0-93] ...
        OK [linked cache]
```

[761]:
```
library(caret)
```

[762]:
```
# generate confusion matrix for validation data
confusionMatrix(default.ct.point.pred, factor(valid.df$Attrition_Flag))
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2456  112
         1  113  358

               Accuracy : 0.926
                 95% CI : (0.9161, 0.935)
    No Information Rate : 0.8453
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7171

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9560
            Specificity : 0.7617
         Pos Pred Value : 0.9564
         Neg Pred Value : 0.7601
             Prevalence : 0.8453
         Detection Rate : 0.8082
   Detection Prevalence : 0.8450
      Balanced Accuracy : 0.8589

       'Positive' Class : 0
```

1.5.2 Logistic Regression

- Logistic regression is a method for fitting a regression curve, y=f(x), when y is a binary variable. The typical use of this model is predicting y given a set of predictors x. The predictors can be continuous, categorical or a mix of both.

[763]:
```
str(bank)
```

```
'data.frame':	10127 obs. of  21 variables:
 $ CLIENTNUM               : int  768805383 818770008 713982108 769911858
709106358 713061558 810347208 818906208 710930508 719661558 …
 $ Attrition_Flag          : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 …
 $ 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 1.41 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 …
```

[764]:
```r
# Preprocess the data
levels(bank$Gender)
```

1. 'F' 2. 'M'

[765]:
```r
levels(bank$Education_Level)
```

1. 'College' 2. 'Doctorate' 3. 'Graduate' 4. 'High School' 5. 'Post-Graduate' 6. 'Uneducated' 7. 'Unknown'

[766]:
```r
levels(bank$Marital_Status)
```

1. 'Divorced' 2. 'Married' 3. 'Single' 4. 'Unknown'

[767]:
```r
levels(bank$Income_Category)
```

1. '$120K +' 2. '$40K - $60K' 3. '$60K - $80K' 4. '$80K - $120K' 5. 'Less than $40K' 6. 'Unknown'

[768]:
```r
levels(bank$Card_Category )
```

1. 'Blue' 2. 'Gold' 3. 'Platinum' 4. 'Silver'

[769]:
```
# create reference category or base level
bank$Gender <- relevel(bank$Gender, ref = "F")
bank$Education_Level <- relevel(bank$Education_Level, ref = "College")
bank$Marital_Status <- relevel(bank$Marital_Status, ref = "Married")
bank$Income_Category <- relevel(bank$Income_Category, ref = "$40K - $60K")
bank$ Card_Category <- relevel(bank$ Card_Category, ref = "Blue")
bank$Attrition_Flag <- as.numeric(bank$Attrition_Flag == "1")
```

[770]:
```
str(bank)
```

```
'data.frame':	10127 obs. of  21 variables:
 $ CLIENTNUM               : int  768805383 818770008 713982108 769911858
709106358 713061558 810347208 818906208 710930508 719661558 …
 $ Attrition_Flag          : num  0 0 0 0 0 0 0 0 0 0 …
 $ 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 "Married","Divorced",..: 1 3 1 4
1 1 1 4 3 3 …
 $ Income_Category         : Factor w/ 6 levels "$40K - $60K",..: 3 5 4 5 3 1 2
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 1.41 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 …
```

[771]:
```
summary(bank)
```

```
   CLIENTNUM         Attrition_Flag    Customer_Age    Gender   Dependent_count
 Min.   :708082083   Min.   :0.0000   Min.   :26.00   F:5358   Min.   :0.000
 1st Qu.:713036770   1st Qu.:0.0000   1st Qu.:41.00   M:4769   1st Qu.:1.000
 Median :717926358   Median :0.0000   Median :46.00            Median :2.000
```

```
 Mean   :739177606   Mean   :0.1607   Mean   :46.33            Mean   :2.346
 3rd Qu.:773143533   3rd Qu.:0.0000   3rd Qu.:52.00            3rd Qu.:3.000
 Max.   :828343083   Max.   :1.0000   Max.   :73.00            Max.   :5.000

      Education_Level  Marital_Status       Income_Category  Card_Category
 College       :1013   Married :4687   $40K - $60K    :1790   Blue    :9436
 Doctorate     : 451   Divorced: 748   $120K +        : 727   Gold    : 116
 Graduate      :3128   Single  :3943   $60K - $80K    :1402   Platinum:  20
 High School   :2013   Unknown : 749   $80K - $120K   :1535   Silver  : 555
 Post-Graduate: 516                    Less than $40K:3561
 Uneducated    :1487                   Unknown        :1112
 Unknown       :1519
 Months_on_book  Total_Relationship_Count Months_Inactive_12_mon
 Min.   :13.00   Min.   :1.000            Min.   :0.000
 1st Qu.:31.00   1st Qu.:3.000            1st Qu.:2.000
 Median :36.00   Median :4.000            Median :2.000
 Mean   :35.93   Mean   :3.813            Mean   :2.341
 3rd Qu.:40.00   3rd Qu.:5.000            3rd Qu.:3.000
 Max.   :56.00   Max.   :6.000            Max.   :6.000

 Contacts_Count_12_mon  Credit_Limit   Total_Revolving_Bal Avg_Open_To_Buy
 Min.   :0.000         Min.   : 1438   Min.   :   0        Min.   :    3
 1st Qu.:2.000         1st Qu.: 2555   1st Qu.: 359        1st Qu.: 1324
 Median :2.000         Median : 4549   Median :1276        Median : 3474
 Mean   :2.455         Mean   : 8632   Mean   :1163        Mean   : 7469
 3rd Qu.:3.000         3rd Qu.:11068   3rd Qu.:1784        3rd Qu.: 9859
 Max.   :6.000         Max.   :34516   Max.   :2517        Max.   :34516

 Total_Amt_Chng_Q4_Q1 Total_Trans_Amt Total_Trans_Ct   Total_Ct_Chng_Q4_Q1
 Min.   :0.0000       Min.   :  510   Min.   : 10.00   Min.   :0.0000
 1st Qu.:0.6310       1st Qu.: 2156   1st Qu.: 45.00   1st Qu.:0.5820
 Median :0.7360       Median : 3899   Median : 67.00   Median :0.7020
 Mean   :0.7599       Mean   : 4404   Mean   : 64.86   Mean   :0.7122
 3rd Qu.:0.8590       3rd Qu.: 4741   3rd Qu.: 81.00   3rd Qu.:0.8180
 Max.   :3.3970       Max.   :18484   Max.   :139.00   Max.   :3.7140

 Avg_Utilization_Ratio
 Min.   :0.0000
 1st Qu.:0.0230
 Median :0.1760
 Mean   :0.2749
 3rd Qu.:0.5030
 Max.   :0.9990
```

[772]:

```
# Partition data
set.seed(1234)
```

```r
train.index <- sample(1:nrow(bank), nrow(bank)*0.7)
train.df <- bank[train.index, ]
valid.df <- bank[-train.index, ]
```

[773]:
```r
# Fit a logistic regression model
# run logistic model, and show coefficients
train_data <- train.df[,c(-1,-16)]
logit.reg <- glm(Attrition_Flag ~ ., data = train_data, family = "binomial")
summary(logit.reg)
```

```
Call:
glm(formula = Attrition_Flag ~ ., family = "binomial", data = train_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7205  -0.3613  -0.1669  -0.0644   3.4991

Coefficients:
                                Estimate Std. Error z value Pr(>|z|)
(Intercept)                    5.662e+00  4.816e-01  11.756  < 2e-16 ***
Customer_Age                  -1.144e-02  9.331e-03  -1.226 0.220159
GenderM                       -8.353e-01  1.721e-01  -4.855  1.2e-06 ***
Dependent_count                1.373e-01  3.593e-02   3.821 0.000133 ***
Education_LevelDoctorate       2.586e-01  2.460e-01   1.051 0.293177
Education_LevelGraduate       -1.053e-01  1.634e-01  -0.645 0.519247
Education_LevelHigh School    -2.080e-01  1.757e-01  -1.184 0.236469
Education_LevelPost-Graduate   1.778e-01  2.432e-01   0.731 0.464671
Education_LevelUneducated     -1.786e-02  1.839e-01  -0.097 0.922599
Education_LevelUnknown        -3.709e-02  1.842e-01  -0.201 0.840437
Marital_StatusDivorced         5.041e-01  1.851e-01   2.723 0.006464 **
Marital_StatusSingle           6.834e-01  1.014e-01   6.738  1.6e-11 ***
Marital_StatusUnknown          6.170e-01  1.750e-01   3.526 0.000421 ***
Income_Category$120K +         7.817e-01  2.390e-01   3.270 0.001075 **
Income_Category$60K - $80K     2.039e-02  2.016e-01   0.101 0.919455
Income_Category$80K - $120K    4.815e-01  2.010e-01   2.396 0.016568 *
Income_CategoryLess than $40K  1.070e-01  1.446e-01   0.740 0.459445
Income_CategoryUnknown        -1.483e-02  1.883e-01  -0.079 0.937251
Card_CategoryGold              1.025e+00  4.123e-01   2.486 0.012930 *
Card_CategoryPlatinum          1.005e+00  7.494e-01   1.341 0.179827
Card_CategorySilver            1.678e-01  2.454e-01   0.683 0.494306
Months_on_book                -2.087e-03  9.214e-03  -0.226 0.820834
Total_Relationship_Count      -4.443e-01  3.293e-02 -13.493  < 2e-16 ***
Months_Inactive_12_mon         4.977e-01  4.542e-02  10.958  < 2e-16 ***
Contacts_Count_12_mon          5.441e-01  4.402e-02  12.361  < 2e-16 ***
Credit_Limit                  -1.549e-05  8.153e-06  -1.900 0.057419 .
Total_Revolving_Bal           -8.792e-04  8.636e-05 -10.181  < 2e-16 ***
```

```
Total_Amt_Chng_Q4_Q1          -5.091e-01  2.267e-01  -2.246 0.024714 *
Total_Trans_Amt                4.783e-04  2.790e-05  17.141  < 2e-16 ***
Total_Trans_Ct                -1.199e-01  4.516e-03 -26.556  < 2e-16 ***
Total_Ct_Chng_Q4_Q1           -2.837e+00  2.267e-01 -12.517  < 2e-16 ***
Avg_Utilization_Ratio         -2.911e-01  2.960e-01  -0.983 0.325421
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6308.2  on 7087  degrees of freedom
Residual deviance: 3288.1  on 7056  degrees of freedom
AIC: 3352.1

Number of Fisher Scoring iterations: 6
```

[774]:
```
# Generate outcome by comparing predicted probability with the cutoff␣
 ↪probability
# use predict() with type = "response" to compute predicted probabilities
# if type not specified, log-odds will be returned
logit.reg.pred <- predict(logit.reg, valid.df[,c(-1,-16)],  type = "response")
```

[775]:
```
# Choose cutoff value and evaluate classification performance
pred <- ifelse(logit.reg.pred > 0.5, 1, 0)
```

[776]:
```
confusionMatrix(factor(pred), factor(valid.df$Attrition_Flag), positive = "1")
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2473  199
         1   96  271

               Accuracy : 0.9029
                 95% CI : (0.8918, 0.9132)
    No Information Rate : 0.8453
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5922

 Mcnemar's Test P-Value : 2.873e-09

            Sensitivity : 0.57660
            Specificity : 0.96263
         Pos Pred Value : 0.73842
         Neg Pred Value : 0.92552
```

```
             Prevalence : 0.15466
         Detection Rate : 0.08917
   Detection Prevalence : 0.12076
      Balanced Accuracy : 0.76961

       'Positive' Class : 1
```

[777]:
```
#Change the cutoff value to 0.1693
pred <- ifelse(logit.reg.pred > 0.1438, 1, 0)
confusionMatrix(factor(pred), factor(valid.df$Attrition_Flag), positive = "1")
```

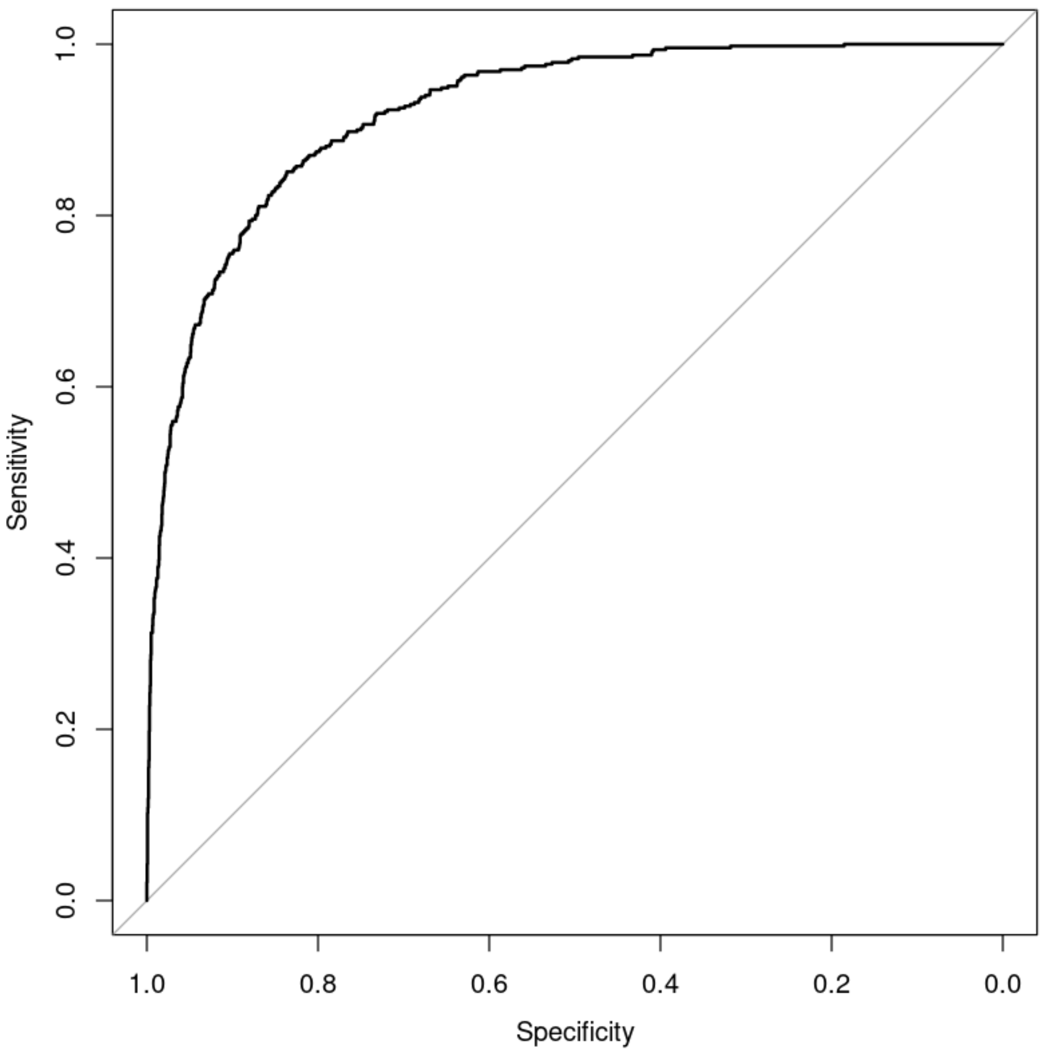
```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2149   70
         1  420  400

               Accuracy : 0.8388
                 95% CI : (0.8252, 0.8517)
    No Information Rate : 0.8453
    P-Value [Acc > NIR] : 0.8481

                  Kappa : 0.5272

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.8511
            Specificity : 0.8365
         Pos Pred Value : 0.4878
         Neg Pred Value : 0.9685
             Prevalence : 0.1547
         Detection Rate : 0.1316
   Detection Prevalence : 0.2698
      Balanced Accuracy : 0.8438

       'Positive' Class : 1
```

[778]:
```
# Generate ROC curve
library(pROC)
```

[779]:
```
r <- roc(valid.df$Attrition_Flag, logit.reg.pred)
```

```
Setting levels: control = 0, case = 1

Setting direction: controls < cases
```

[780]:
```
plot.roc(r)
```



[781]:
```
# find the best threshold, with a default best method "youden"
coords(r, x = "best")
```

A data.frame: 1 × 3

threshold <dbl>	specificity <dbl>	sensitivity <dbl>
0.1437976	0.8365123	0.8510638

1.5.3 Random Forest

[782]:
```
install.packages("randomForest")
```

```
Installing randomForest [4.7-1.1] ...
        OK [linked cache]
```

[783]:
```
library(randomForest)
```

[784]:
```
# model

bank$Attrition_Flag <- factor(bank$Attrition_Flag) # classification model
set.seed(1234)
train.index <- sample(1:nrow(bank), nrow(bank)*0.7)
train.df <- bank[train.index, ]
valid.df <- bank[-train.index, ]
train_data <-train.df[,-1]
random_forest <- randomForest(Attrition_Flag ~., data = train_data)
random_forest
```

```
Call:
 randomForest(formula = Attrition_Flag ~ ., data = train_data)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 3.84%
Confusion matrix:
     0   1 class.error
0 5855  76  0.01281403
1  196 961  0.16940363
```

[785]:
```
confusionMatrix(predict(random_forest,valid.df[,-1]), factor(valid.
 ↪df$Attrition_Flag), positive = "1")
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2539   87
         1   30  383

               Accuracy : 0.9615
                 95% CI : (0.954, 0.9681)
    No Information Rate : 0.8453
    P-Value [Acc > NIR] : < 2.2e-16
```

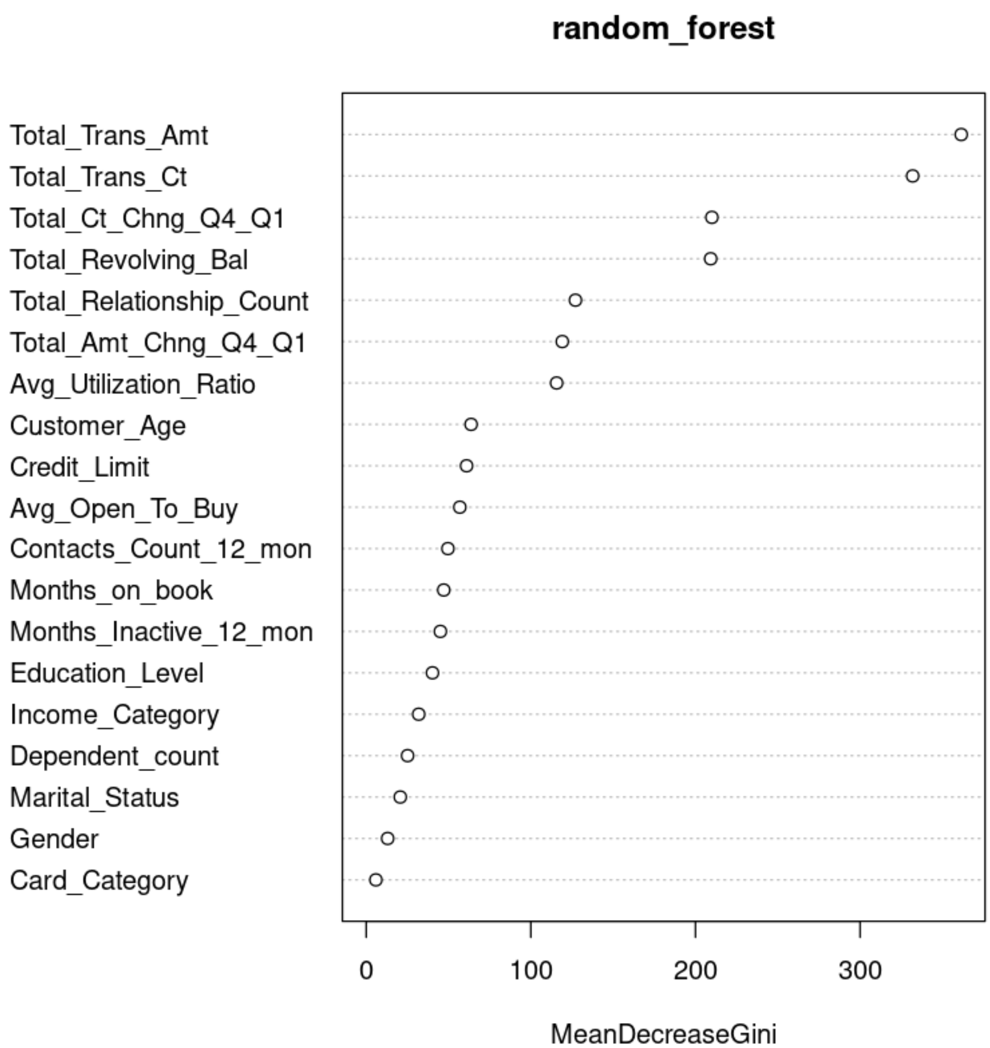
```
                 Kappa : 0.8451

Mcnemar's Test P-Value : 2.252e-07

           Sensitivity : 0.8149
           Specificity : 0.9883
        Pos Pred Value : 0.9274
        Neg Pred Value : 0.9669
            Prevalence : 0.1547
        Detection Rate : 0.1260
  Detection Prevalence : 0.1359
     Balanced Accuracy : 0.9016

      'Positive' Class : 1
```

[786]:
```
## feature importance
#Evaluate variable importance
importance(random_forest)
```

A matrix: 19 × 1 of type dbl

	MeanDecreaseGini
Customer_Age	63.721819
Gender	13.024036
Dependent_count	25.125507
Education_Level	40.262626
Marital_Status	20.706181
Income_Category	31.901363
Card_Category	5.881078
Months_on_book	47.100347
Total_Relationship_Count	127.096229
Months_Inactive_12_mon	45.060071
Contacts_Count_12_mon	49.631937
Credit_Limit	60.971329
Total_Revolving_Bal	209.227133
Avg_Open_To_Buy	56.867859
Total_Amt_Chng_Q4_Q1	119.103705
Total_Trans_Amt	361.421092
Total_Trans_Ct	331.926869
Total_Ct_Chng_Q4_Q1	209.969848
Avg_Utilization_Ratio	115.656658

[787]:
```
varImpPlot(random_forest)
```



1.5.4 XGBoost

[788]:
```
install.packages("xgboost")
```

```
Installing xgboost [1.6.0.1] …
        OK [linked cache]
```

[789]:
```
library(e1071)
library(xgboost)
```

[790]:
```
train <- train.df[,-1]
valid <- valid.df[,-1]
X_train <- data.matrix(train[,-1])
```

```
y_train <- train[,1]
X_valid <-data.matrix(valid[,-1])
y_valid <- valid[,1]

# convert the train and valid data into xgboost matrix type.
xgboost_train = xgb.DMatrix(data=X_train, label=y_train)
xgboost_valid = xgb.DMatrix(data=X_valid, label=y_valid)
```

[791]:
```
summary(y_valid)
```

0	2569	**1**	470

[792]:
```
xgboost1 <- xgboost(data = xgboost_train,                    # the data
                 max.depth=6,                           # max depth
                 nrounds=500)                             # max number of␣
 ↪boosting iterations
```

```
[1]	train-rmse:0.550101
[2]	train-rmse:0.406525
[3]	train-rmse:0.309713
[4]	train-rmse:0.246395
[5]	train-rmse:0.203813
[6]	train-rmse:0.177623
[7]	train-rmse:0.159691
[8]	train-rmse:0.146643
[9]	train-rmse:0.137778
[10]	train-rmse:0.132146
[11]	train-rmse:0.126801
[12]	train-rmse:0.122946
[13]	train-rmse:0.121052
[14]	train-rmse:0.117589
[15]	train-rmse:0.115390
[16]	train-rmse:0.113138
[17]	train-rmse:0.111854
[18]	train-rmse:0.111328
[19]	train-rmse:0.110138
[20]	train-rmse:0.108170
[21]	train-rmse:0.106657
[22]	train-rmse:0.106170
[23]	train-rmse:0.104947
[24]	train-rmse:0.104090
[25]	train-rmse:0.102088
[26]	train-rmse:0.100956
[27]	train-rmse:0.098490
[28]	train-rmse:0.096728
[29]	train-rmse:0.095847
[30]	train-rmse:0.094349
[31]	train-rmse:0.092135
```

```
[32]	train-rmse:0.091240
[33]	train-rmse:0.090667
[34]	train-rmse:0.090214
[35]	train-rmse:0.089815
[36]	train-rmse:0.088509
[37]	train-rmse:0.087260
[38]	train-rmse:0.086975
[39]	train-rmse:0.086319
[40]	train-rmse:0.084663
[41]	train-rmse:0.083406
[42]	train-rmse:0.082615
[43]	train-rmse:0.081537
[44]	train-rmse:0.081033
[45]	train-rmse:0.080654
[46]	train-rmse:0.079378
[47]	train-rmse:0.078103
[48]	train-rmse:0.076555
[49]	train-rmse:0.075500
[50]	train-rmse:0.075073
[51]	train-rmse:0.074243
[52]	train-rmse:0.073926
[53]	train-rmse:0.072844
[54]	train-rmse:0.071841
[55]	train-rmse:0.071439
[56]	train-rmse:0.070959
[57]	train-rmse:0.070577
[58]	train-rmse:0.070477
[59]	train-rmse:0.069493
[60]	train-rmse:0.068459
[61]	train-rmse:0.068033
[62]	train-rmse:0.067205
[63]	train-rmse:0.066583
[64]	train-rmse:0.065297
[65]	train-rmse:0.065062
[66]	train-rmse:0.064433
[67]	train-rmse:0.064269
[68]	train-rmse:0.063920
[69]	train-rmse:0.063489
[70]	train-rmse:0.062622
[71]	train-rmse:0.062003
[72]	train-rmse:0.061599
[73]	train-rmse:0.061194
[74]	train-rmse:0.060648
[75]	train-rmse:0.059340
[76]	train-rmse:0.058915
[77]	train-rmse:0.058226
[78]	train-rmse:0.057141
[79]	train-rmse:0.056326
```

```
[80]	train-rmse:0.055951
[81]	train-rmse:0.055301
[82]	train-rmse:0.054669
[83]	train-rmse:0.054181
[84]	train-rmse:0.053346
[85]	train-rmse:0.052841
[86]	train-rmse:0.052573
[87]	train-rmse:0.051942
[88]	train-rmse:0.050822
[89]	train-rmse:0.050759
[90]	train-rmse:0.050491
[91]	train-rmse:0.049422
[92]	train-rmse:0.048900
[93]	train-rmse:0.048651
[94]	train-rmse:0.047687
[95]	train-rmse:0.047109
[96]	train-rmse:0.046867
[97]	train-rmse:0.046303
[98]	train-rmse:0.045830
[99]	train-rmse:0.045475
[100]	train-rmse:0.045075
[101]	train-rmse:0.044604
[102]	train-rmse:0.043794
[103]	train-rmse:0.043550
[104]	train-rmse:0.043375
[105]	train-rmse:0.043255
[106]	train-rmse:0.042536
[107]	train-rmse:0.042071
[108]	train-rmse:0.041748
[109]	train-rmse:0.041244
[110]	train-rmse:0.040672
[111]	train-rmse:0.039974
[112]	train-rmse:0.039440
[113]	train-rmse:0.039184
[114]	train-rmse:0.038774
[115]	train-rmse:0.038399
[116]	train-rmse:0.037997
[117]	train-rmse:0.037658
[118]	train-rmse:0.037337
[119]	train-rmse:0.037002
[120]	train-rmse:0.036974
[121]	train-rmse:0.036866
[122]	train-rmse:0.036548
[123]	train-rmse:0.036111
[124]	train-rmse:0.035840
[125]	train-rmse:0.035560
[126]	train-rmse:0.035356
[127]	train-rmse:0.035059
```

```
[128]	train-rmse:0.034707
[129]	train-rmse:0.034308
[130]	train-rmse:0.034215
[131]	train-rmse:0.033989
[132]	train-rmse:0.033831
[133]	train-rmse:0.033798
[134]	train-rmse:0.033739
[135]	train-rmse:0.033421
[136]	train-rmse:0.032966
[137]	train-rmse:0.032817
[138]	train-rmse:0.032470
[139]	train-rmse:0.032099
[140]	train-rmse:0.032083
[141]	train-rmse:0.031798
[142]	train-rmse:0.031570
[143]	train-rmse:0.031201
[144]	train-rmse:0.031040
[145]	train-rmse:0.030757
[146]	train-rmse:0.030666
[147]	train-rmse:0.030523
[148]	train-rmse:0.030442
[149]	train-rmse:0.030086
[150]	train-rmse:0.029452
[151]	train-rmse:0.029258
[152]	train-rmse:0.029176
[153]	train-rmse:0.029032
[154]	train-rmse:0.028936
[155]	train-rmse:0.028820
[156]	train-rmse:0.028708
[157]	train-rmse:0.028674
[158]	train-rmse:0.028616
[159]	train-rmse:0.028550
[160]	train-rmse:0.028088
[161]	train-rmse:0.028018
[162]	train-rmse:0.027950
[163]	train-rmse:0.027746
[164]	train-rmse:0.027536
[165]	train-rmse:0.027410
[166]	train-rmse:0.027299
[167]	train-rmse:0.027102
[168]	train-rmse:0.027017
[169]	train-rmse:0.026780
[170]	train-rmse:0.026676
[171]	train-rmse:0.026419
[172]	train-rmse:0.026231
[173]	train-rmse:0.026152
[174]	train-rmse:0.025741
[175]	train-rmse:0.025419
```

```
[176]	train-rmse:0.025385
[177]	train-rmse:0.025173
[178]	train-rmse:0.024887
[179]	train-rmse:0.024807
[180]	train-rmse:0.024572
[181]	train-rmse:0.024124
[182]	train-rmse:0.024089
[183]	train-rmse:0.024008
[184]	train-rmse:0.023963
[185]	train-rmse:0.023695
[186]	train-rmse:0.023666
[187]	train-rmse:0.023489
[188]	train-rmse:0.023229
[189]	train-rmse:0.023065
[190]	train-rmse:0.022898
[191]	train-rmse:0.022785
[192]	train-rmse:0.022696
[193]	train-rmse:0.022445
[194]	train-rmse:0.022319
[195]	train-rmse:0.022143
[196]	train-rmse:0.021890
[197]	train-rmse:0.021784
[198]	train-rmse:0.021673
[199]	train-rmse:0.021521
[200]	train-rmse:0.021435
[201]	train-rmse:0.021394
[202]	train-rmse:0.021257
[203]	train-rmse:0.021054
[204]	train-rmse:0.020727
[205]	train-rmse:0.020412
[206]	train-rmse:0.020106
[207]	train-rmse:0.019960
[208]	train-rmse:0.019804
[209]	train-rmse:0.019598
[210]	train-rmse:0.019369
[211]	train-rmse:0.019265
[212]	train-rmse:0.019083
[213]	train-rmse:0.018961
[214]	train-rmse:0.018838
[215]	train-rmse:0.018721
[216]	train-rmse:0.018674
[217]	train-rmse:0.018530
[218]	train-rmse:0.018368
[219]	train-rmse:0.018265
[220]	train-rmse:0.018144
[221]	train-rmse:0.017943
[222]	train-rmse:0.017929
[223]	train-rmse:0.017876
```

```
[224]	train-rmse:0.017696
[225]	train-rmse:0.017565
[226]	train-rmse:0.017428
[227]	train-rmse:0.017065
[228]	train-rmse:0.016999
[229]	train-rmse:0.016887
[230]	train-rmse:0.016717
[231]	train-rmse:0.016417
[232]	train-rmse:0.016300
[233]	train-rmse:0.016195
[234]	train-rmse:0.016182
[235]	train-rmse:0.016123
[236]	train-rmse:0.015972
[237]	train-rmse:0.015799
[238]	train-rmse:0.015793
[239]	train-rmse:0.015740
[240]	train-rmse:0.015653
[241]	train-rmse:0.015506
[242]	train-rmse:0.015501
[243]	train-rmse:0.015405
[244]	train-rmse:0.015352
[245]	train-rmse:0.015263
[246]	train-rmse:0.015187
[247]	train-rmse:0.015065
[248]	train-rmse:0.015016
[249]	train-rmse:0.014945
[250]	train-rmse:0.014856
[251]	train-rmse:0.014728
[252]	train-rmse:0.014640
[253]	train-rmse:0.014573
[254]	train-rmse:0.014549
[255]	train-rmse:0.014452
[256]	train-rmse:0.014323
[257]	train-rmse:0.014209
[258]	train-rmse:0.014137
[259]	train-rmse:0.014091
[260]	train-rmse:0.014038
[261]	train-rmse:0.013941
[262]	train-rmse:0.013846
[263]	train-rmse:0.013768
[264]	train-rmse:0.013701
[265]	train-rmse:0.013628
[266]	train-rmse:0.013522
[267]	train-rmse:0.013450
[268]	train-rmse:0.013351
[269]	train-rmse:0.013288
[270]	train-rmse:0.013171
[271]	train-rmse:0.013141
```

```
[272]	train-rmse:0.013023
[273]	train-rmse:0.013003
[274]	train-rmse:0.012895
[275]	train-rmse:0.012769
[276]	train-rmse:0.012624
[277]	train-rmse:0.012463
[278]	train-rmse:0.012437
[279]	train-rmse:0.012425
[280]	train-rmse:0.012373
[281]	train-rmse:0.012364
[282]	train-rmse:0.012310
[283]	train-rmse:0.012297
[284]	train-rmse:0.012195
[285]	train-rmse:0.012107
[286]	train-rmse:0.012043
[287]	train-rmse:0.011964
[288]	train-rmse:0.011882
[289]	train-rmse:0.011799
[290]	train-rmse:0.011731
[291]	train-rmse:0.011695
[292]	train-rmse:0.011677
[293]	train-rmse:0.011619
[294]	train-rmse:0.011518
[295]	train-rmse:0.011381
[296]	train-rmse:0.011359
[297]	train-rmse:0.011309
[298]	train-rmse:0.011165
[299]	train-rmse:0.011073
[300]	train-rmse:0.011054
[301]	train-rmse:0.010941
[302]	train-rmse:0.010833
[303]	train-rmse:0.010800
[304]	train-rmse:0.010790
[305]	train-rmse:0.010751
[306]	train-rmse:0.010716
[307]	train-rmse:0.010699
[308]	train-rmse:0.010680
[309]	train-rmse:0.010620
[310]	train-rmse:0.010588
[311]	train-rmse:0.010522
[312]	train-rmse:0.010454
[313]	train-rmse:0.010423
[314]	train-rmse:0.010388
[315]	train-rmse:0.010338
[316]	train-rmse:0.010311
[317]	train-rmse:0.010233
[318]	train-rmse:0.010165
[319]	train-rmse:0.010069
```

```
[320]	train-rmse:0.010013
[321]	train-rmse:0.009920
[322]	train-rmse:0.009892
[323]	train-rmse:0.009859
[324]	train-rmse:0.009775
[325]	train-rmse:0.009695
[326]	train-rmse:0.009588
[327]	train-rmse:0.009521
[328]	train-rmse:0.009492
[329]	train-rmse:0.009459
[330]	train-rmse:0.009438
[331]	train-rmse:0.009404
[332]	train-rmse:0.009394
[333]	train-rmse:0.009386
[334]	train-rmse:0.009319
[335]	train-rmse:0.009274
[336]	train-rmse:0.009238
[337]	train-rmse:0.009192
[338]	train-rmse:0.009123
[339]	train-rmse:0.009085
[340]	train-rmse:0.009018
[341]	train-rmse:0.008971
[342]	train-rmse:0.008923
[343]	train-rmse:0.008896
[344]	train-rmse:0.008819
[345]	train-rmse:0.008773
[346]	train-rmse:0.008738
[347]	train-rmse:0.008722
[348]	train-rmse:0.008693
[349]	train-rmse:0.008589
[350]	train-rmse:0.008529
[351]	train-rmse:0.008515
[352]	train-rmse:0.008493
[353]	train-rmse:0.008454
[354]	train-rmse:0.008416
[355]	train-rmse:0.008355
[356]	train-rmse:0.008306
[357]	train-rmse:0.008289
[358]	train-rmse:0.008259
[359]	train-rmse:0.008238
[360]	train-rmse:0.008220
[361]	train-rmse:0.008187
[362]	train-rmse:0.008138
[363]	train-rmse:0.008095
[364]	train-rmse:0.008068
[365]	train-rmse:0.008050
[366]	train-rmse:0.008031
[367]	train-rmse:0.007977
```

```
[368]	train-rmse:0.007913
[369]	train-rmse:0.007855
[370]	train-rmse:0.007752
[371]	train-rmse:0.007700
[372]	train-rmse:0.007644
[373]	train-rmse:0.007633
[374]	train-rmse:0.007623
[375]	train-rmse:0.007578
[376]	train-rmse:0.007541
[377]	train-rmse:0.007523
[378]	train-rmse:0.007487
[379]	train-rmse:0.007439
[380]	train-rmse:0.007370
[381]	train-rmse:0.007289
[382]	train-rmse:0.007242
[383]	train-rmse:0.007150
[384]	train-rmse:0.007096
[385]	train-rmse:0.007088
[386]	train-rmse:0.007064
[387]	train-rmse:0.006975
[388]	train-rmse:0.006920
[389]	train-rmse:0.006858
[390]	train-rmse:0.006795
[391]	train-rmse:0.006758
[392]	train-rmse:0.006750
[393]	train-rmse:0.006719
[394]	train-rmse:0.006680
[395]	train-rmse:0.006662
[396]	train-rmse:0.006654
[397]	train-rmse:0.006592
[398]	train-rmse:0.006570
[399]	train-rmse:0.006527
[400]	train-rmse:0.006468
[401]	train-rmse:0.006435
[402]	train-rmse:0.006391
[403]	train-rmse:0.006383
[404]	train-rmse:0.006314
[405]	train-rmse:0.006276
[406]	train-rmse:0.006261
[407]	train-rmse:0.006249
[408]	train-rmse:0.006199
[409]	train-rmse:0.006170
[410]	train-rmse:0.006137
[411]	train-rmse:0.006129
[412]	train-rmse:0.006107
[413]	train-rmse:0.006054
[414]	train-rmse:0.006028
[415]	train-rmse:0.006014
```

```
[416]	train-rmse:0.005994
[417]	train-rmse:0.005979
[418]	train-rmse:0.005937
[419]	train-rmse:0.005906
[420]	train-rmse:0.005878
[421]	train-rmse:0.005871
[422]	train-rmse:0.005827
[423]	train-rmse:0.005794
[424]	train-rmse:0.005726
[425]	train-rmse:0.005668
[426]	train-rmse:0.005616
[427]	train-rmse:0.005582
[428]	train-rmse:0.005563
[429]	train-rmse:0.005544
[430]	train-rmse:0.005530
[431]	train-rmse:0.005510
[432]	train-rmse:0.005490
[433]	train-rmse:0.005454
[434]	train-rmse:0.005420
[435]	train-rmse:0.005390
[436]	train-rmse:0.005366
[437]	train-rmse:0.005307
[438]	train-rmse:0.005251
[439]	train-rmse:0.005233
[440]	train-rmse:0.005186
[441]	train-rmse:0.005152
[442]	train-rmse:0.005104
[443]	train-rmse:0.005082
[444]	train-rmse:0.005049
[445]	train-rmse:0.005043
[446]	train-rmse:0.004989
[447]	train-rmse:0.004979
[448]	train-rmse:0.004971
[449]	train-rmse:0.004930
[450]	train-rmse:0.004910
[451]	train-rmse:0.004883
[452]	train-rmse:0.004851
[453]	train-rmse:0.004839
[454]	train-rmse:0.004817
[455]	train-rmse:0.004784
[456]	train-rmse:0.004761
[457]	train-rmse:0.004752
[458]	train-rmse:0.004744
[459]	train-rmse:0.004688
[460]	train-rmse:0.004672
[461]	train-rmse:0.004646
[462]	train-rmse:0.004638
[463]	train-rmse:0.004608
```

```
[464]	train-rmse:0.004572
[465]	train-rmse:0.004532
[466]	train-rmse:0.004507
[467]	train-rmse:0.004502
[468]	train-rmse:0.004466
[469]	train-rmse:0.004448
[470]	train-rmse:0.004408
[471]	train-rmse:0.004392
[472]	train-rmse:0.004390
[473]	train-rmse:0.004360
[474]	train-rmse:0.004341
[475]	train-rmse:0.004304
[476]	train-rmse:0.004299
[477]	train-rmse:0.004277
[478]	train-rmse:0.004251
[479]	train-rmse:0.004236
[480]	train-rmse:0.004215
[481]	train-rmse:0.004200
[482]	train-rmse:0.004199
[483]	train-rmse:0.004188
[484]	train-rmse:0.004157
[485]	train-rmse:0.004130
[486]	train-rmse:0.004127
[487]	train-rmse:0.004082
[488]	train-rmse:0.004054
[489]	train-rmse:0.004037
[490]	train-rmse:0.004027
[491]	train-rmse:0.004014
[492]	train-rmse:0.004009
[493]	train-rmse:0.003974
[494]	train-rmse:0.003947
[495]	train-rmse:0.003925
[496]	train-rmse:0.003911
[497]	train-rmse:0.003899
[498]	train-rmse:0.003875
[499]	train-rmse:0.003845
[500]	train-rmse:0.003830
```

[793]:
```
xgboost1
```

```
##### xgb.Booster
raw: 1.8 Mb
call:
  xgb.train(params = params, data = dtrain, nrounds = nrounds,
    watchlist = watchlist, verbose = verbose, print_every_n = print_every_n,
    early_stopping_rounds = early_stopping_rounds, maximize = maximize,
    save_period = save_period, save_name = save_name, xgb_model = xgb_model,
    callbacks = callbacks, max.depth = 6)
```

```
params (as set within xgb.train):
  max_depth = "6", validate_parameters = "1"
xgb.attributes:
  niter
callbacks:
  cb.print.evaluation(period = print_every_n)
  cb.evaluation.log()
# of features: 19
niter: 500
nfeatures : 19
evaluation_log:
    iter  train_rmse
       1 0.550101327
       2 0.406525233
---
     499 0.003844771
     500 0.003829790
```

[794]:
```
summary(xgboost1)
```

```
               Length  Class              Mode
handle               1 xgb.Booster.handle externalptr
raw            1882430 -none-             raw
niter                1 -none-             numeric
evaluation_log       2 data.table         list
call                14 -none-             call
params               2 -none-             list
callbacks            2 -none-             list
feature_names       19 -none-             character
nfeatures            1 -none-             numeric
```

[795]:
```
#use model to make predictions on valid data
pred = predict(xgboost1, xgboost_valid)

pred
```

1. 0.984222590923309 2. 0.959598779678345 3. 1.02995204925537 4. 1.29087901115417
5. 0.98189640045166 6. 1.26966643333435 7. 0.974927067756653 8. 0.992498278617859
9. 1.05641782283783 10. 1.00185930728912 11. 0.998172402381897 12. 1.0516711473465
13. 1.30043864250183 14. 1.1025755405426 15. 1.10412418842316 16. 1.0516471862793
17. 1.25880932807922 18. 1.90219390392303 19. 1.58192479610443 20. 0.973842144012451
21. 0.950345277786255 22. 1.18904101848602 23. 2.15802025794983 24. 1.06553781032562
25. 1.17191159725189 26. 1.01347553730011 27. 0.972853183746338 28. 1.00586867332458
29. 1.41475963592529 30. 0.98992246389389 31. 0.92238849401474 32. 2.06302285194397
33. 1.16370534896851 34. 0.98002302646637 35. 1.36190390586853 36. 1.00928699970245
37. 0.946904003620148 38. 1.00435853004456 39. 1.10054183006287 40. 0.996864438056946
41. 1.40883278846741 42. 1.14175128936768 43. 2.0360906124115 44. 1.03078615665436
45. 1.38465178012848 46. 0.964209377765656 47. 1.21187782287598 48. 1.00853931903839

49. 1.01823842525482 50. 1.00234067440033 51. 1.33672297000885 52. 1.59729790687561
53. 0.987927079200745 54. 1.90348744392395 55. 0.976078510284424 56. 1.41064250469208
57. 1.04388499259949 58. 1.18615198135376 59. 1.68860530853271 60. 1.01284873485565
61. 0.973057925701141 62. 1.43729937076569 63. 1.05019974708557 64. 0.979315578937531
65. 0.943922340869904 66. 0.941778242588043 67. 0.896911680698395 68. 0.753844320774078
69. 0.977749407291412 70. 1.81678068637848 71. 1.07555997371674 72. 1.18939089775085
73. 1.12472653388977 74. 1.00801527500153 75. 0.902799546718597 76. 1.44709455966949
77. 1.1320241689682 78. 0.991944015026093 79. 1.02032995223999 80. 1.11494970321655
81. 1.4304164648056 82. 0.956844091415405 83. 1.04660427570343 84. 1.6430789232254
85. 1.12521028518677 86. 0.897537887096405 87. 1.1265606880188 88. 0.979135513305664
89. 0.961659252643585 90. 1.72743010520935 91. 1.00591719150543 92. 0.972215950489044
93. 0.940824925899506 94. 0.965178489685059 95. 1.04196047782898 96. 0.999716699123383
97. 0.990419745445251 98. 1.02498173713684 99. 1.40276157855988 100. 1.07437252998352
101. 0.963455677032471 102. 1.20620739459991 103. 0.985227763652802 104. 1.01782155036926
105. 1.04055655002594 106. 1.00000691413879 107. 0.991972804069519 108. 0.960342288017273
109. 1.05062544345856 110. 1.97442889213562 111. 1.0526567697525 112. 0.976860523223877
113. 0.964484930038452 114. 0.962137997150421 115. 1.01040041446686 116. 1.0003879070282
117. 0.971704721450806 118. 1.00279307365417 119. 1.00958693027496 120. 1.16383039951324
121. 2.07981538772583 122. 0.89018702507019 123. 1.39951491355896 124. 0.951729118824005
125. 0.952286720275879 126. 0.92825311422348 127. 2.03304934501648 128. 0.913326561450958
129. 0.98595529794693 130. 0.909296691417694 131. 1.02807593345642 132. 1.00118732452393
133. 1.18884289264679 134. 1.33783948421478 135. 0.986512243747711 136. 1.06768941879272
137. 0.976172268390656 138. 1.000816822052 139. 1.00254857540131 140. 1.05112838745117
141. 1.8913140296936 142. 0.88639622926712 143. 1.07953023910522 144. 1.07467317581177
145. 1.00780069828033 146. 0.961557805538177 147. 1.09766387939453 148. 0.990422785282135
149. 0.886929214000702 150. 0.940750062465668 151. 1.15084874629974 152. 1.03335201740265
153. 0.891560077667236 154. 0.99143522977829 155. 0.936356365680695 156. 1.00655722618103
157. 0.980536341667175 158. 0.97863245010376 159. 1.03347933292389 160. 1.96390533447266
161. 1.07348573207855 162. 1.11929416656494 163. 1.17220258712769 164. 0.970976173877716
165. 0.94805371761322 166. 0.974750280380249 167. 0.986356914043427 168. 1.19931077957153
169. 0.928806364536285 170. 2.22209596633911 171. 2.01533913612366 172. 0.804018318653107
173. 1.01822125911713 174. 0.971790790557861 175. 1.95649135112762 176. 0.949447870254517
177. 0.944019913673401 178. 1.87347650527954 179. 1.04375088214874 180. 2.05826020240784
181. 1.00284731388092 182. 2.20491862297058 183. 1.1635148525238 184. 1.07507491111755
185. 1.00814616680145 186. 0.946508824825287 187. 0.94988352060318 188. 1.0011819601059
189. 1.02641904354095 190. 0.972756862640381 191. 0.906652271747589 192. 1.7224907875061
193. 1.01755785942078 194. 1.00518560409546 195. 1.02754342556 196. 1.05318176746368
197. 1.09658789634705 198. 1.06773138046265 199. 1.11376917362213 200. 1.14176070690155 201.
202. 0.995185792446136 203. 1.12533020973206 204. 1.00895714759827 205. 1.02639591693878
206. 1.10104310512543 207. 2.1021523475647 208. 1.49554431438446 209. 1.00360810756683
210. 1.82847249507904 211. 1.0035172700882 212. 1.07361268997192 213. 1.90805983543396
214. 0.981694996356964 215. 0.997653305530548 216. 1.01675522327423 217. 0.988426148891449
218. 1.02153813838959 219. 1.01783204078674 220. 1.01666367053986 221. 1.39463949203491
222. 0.993237376213074 223. 0.982315063476562 224. 1.03456008434296 225. 1.854567527771
226. 1.03139305114746 227. 1.23013758659363 228. 1.90668988227844 229. 0.995100677013397
230. 1.01565861701965 231. 1.01006472110748 232. 1.00611793994904 233. 1.00664722919464
234. 0.997082889080048 235. 1.73617219924927 236. 1.04565227031708 237. 1.56085669994354
238. 0.972264170646667 239. 2.11213803291321 240. 1.68948006629944 241. 1.18821060657501

242. 0.991358697414398 243. 0.987740635871887 244. 1.00274455547333 245. 0.98257714509964 246. 1.00805711746216 247. 0.99950635433197 248. 0.994574844837189 249. 1.01082611083984 250. 2.1429545879364 251. 1.0200001001358 252. 1.51870286464691 253. 0.981020271778107 254. 2.13199710845947 255. 1.02167975902557 256. 0.996750235557556 257. 0.986021876335144 258. 1.02728259563446 259. 1.00726282596588 260. 0.999320030212402 261. 0.994095206260681 262. 1.00021553039551 263. 2.19690942764282 264. 0.997773587703705 265. 0.985890746116638 266. 1.00434839725494 267. 1.0754919052124 268. 0.975574851036072 269. 1.93244886398315 270. 1.01901650428772 271. 2.11483478546143 272. 1.41537797451019 273. 1.01965034008026 274. 0.983173072338104 275. 1.00082385540009 276. 1.59829556941986 277. 0.991594672203064 278. 0.934461951255798 279. 1.01386761665344 280. 0.997194349765778 281. 0.99889200925827 282. 1.40694427490234 283. 1.08140957355499 284. 0.97942054271698 285. 0.999252617359161 286. 1.21109461784363 287. 1.03175902366638 288. 1.90753519535065 289. 1.04584491252899 290. 0.995562374591827 291. 0.973886489868164 292. 1.20334565639496 293. 1.80133211612701 294. 2.06192946434021 295. 0.975162148475647 296. 0.989398658275604 297. 1.92375087738037 298. 2.03374671936035 299. 0.991609752178192 300. 1.91925859451294 301. 1.01464366912842 302. 1.71973598003387 303. 1.0060293674469 304. 0.953539431095123 305. 0.996875464916229 306. 1.85510873794556 307. 1.0110114812851 308. 0.977624297142029 309. 2.2570424079895 310. 1.71495091915131 311. 1.04819560050964 312. 1.74864792823792 313. 1.89881551265717 314. 2.04479002952576 315. 1.08683514595032 316. 2.04578709602356 317. 1.00063526630402 318. 1.06937992572784 319. 1.11286818981171 320. 1.00915110111237 321. 1.05348598957062 322. 1.84640920162201 323. 1.1828510761261 324. 1.92677080631256 325. 1.77364814281464 326. 0.992467939853668 327. 0.99974513053894 328. 1.58847498893738 329. 0.966410100460052 330. 1.01343929767609 331. 2.02444267272949 332. 0.982317686080933 333. 1.27573478221893 334. 1.97277057170868 335. 1.00935125350952 336. 0.994649589061737 337. 0.982609748840332 338. 1.02272653579712 339. 0.993005692958832 340. 1.00711750984192 341. 1.0676441192627 342. 2.09345436096191 343. 0.988238155841827 344. 1.40429651737213 345. 0.99621593952179 346. 0.99098014831543 347. 1.00091576576233 348. 1.1723325252533 349. 1.9487612247467 350. 1.03753221035004 351. 1.97848677635193 352. 1.00519859790802 353. 1.00358021259308 354. 2.04189586639404 355. 0.955494701862335 356. 1.02724456787109 357. 1.03441071510315 358. 2.15286135673523 359. 1.00889444351196 360. 1.00439369678497 361. 1.00784862041473 362. 1.00410151481628 363. 0.96275782585144 364. 2.16348099708557 365. 1.00376296043396 366. 1.05028915405273 367. 0.995083689689636 368. 1.00454473495483 369. 2.02556347846985 370. 0.987571954727173 371. 1.83739244937897 372. 1.00963544845581 373. 1.92121374607086 374. 0.997591137886047 375. 0.953283309936523 376. 1.0010062456131 377. 1.00300431251526 378. 0.999464094638824 379. 1.0084593296051 380. 1.00226891040802 381. 1.9736829996109 382. 1.04344141483307 383. 1.00807559490204 384. 0.980873286724091 385. 1.93388414382935 386. 1.00902962684631 387. 1.01058769226074 388. 0.994260966777802 389. 0.995881795883179 390. 0.961664140224457 391. 1.00688409805298 392. 0.99584686756134 393. 1.03545093536377 394. 1.06798112392426 395. 0.964677453041077 396. 1.32541286945343 397. 1.00604772567749 398. 1.04345333576202 399. 1.00848686695099 400. 1.99644804000854 401. 1.9351681470871

[796]:
```
# Convert prediction to factor type

pred[(pred>3)] = 3
pred_y = as.factor((levels(y_valid))[round(pred)])
print(pred_y)
```

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

```
  [38] 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
  [75] 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
 [112] 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
 [149] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 0 1 0 1 0 0 0
 [186] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 1 1 0
 [223] 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
 [260] 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [297] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
 [334] 1 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
 [371] 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [408] 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 1 1 0 0 0 0 0
 [445] 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0
 [482] 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [519] 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 0
 [556] 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0
 [593] 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [630] 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [667] 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
 [704] 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0
 [741] 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 [778] 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0
 [815] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0
 [852] 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
 [889] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [926] 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
 [963] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
[1000] 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
[1037] 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0
[1074] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0
[1111] 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0
[1148] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1
[1185] 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0
[1222] 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0
[1259] 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[1296] 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0
[1333] 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0
[1370] 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 1 0 1 0
[1407] 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 1 0 1
[1444] 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0 1 0 0 0 1 0 0 1
[1481] 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 1 0 0 0
[1518] 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1
[1555] 1 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0
[1592] 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
[1629] 0 1 0 0 1 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0
[1666] 0 0 0 0 0 0 0 1 1 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
[1703] 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1 0 1 0 1 0 0
[1740] 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1
[1777] 0 1 0 0 1 0 1 0 0 1 1 1 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
```

```
[1814] 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
[1851] 0 1 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0
[1888] 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
[1925] 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 1 0 0 1 0 0 0 0 0 1 1 0 0 1 0
[1962] 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0
[1999] 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0
[2036] 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0
[2073] 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0
[2110] 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[2147] 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0
[2184] 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0
[2221] 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 1 1 0 0
[2258] 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[2295] 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1
[2332] 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
[2369] 0 0 1 1 0 1 0 0 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
[2406] 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0
[2443] 1 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0
[2480] 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
[2517] 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
[2554] 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 1 0 0 1 0 0 1 1 0 1 0 0 1 0 0 1 0 0 0 0 0 0
[2591] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[2628] 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
[2665] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
[2702] 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0
[2739] 1 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 1 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0
[2776] 0 1 1 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0
[2813] 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0
[2850] 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0
[2887] 0 1 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0
[2924] 0 0 1 0 0 0 0 1 1 0 0 1 1 0 1 0 1 0 0 0 1 0 0 1 1 0 1 1 1 0 1 0 0 0 0 0 1
[2961] 0 1 1 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 1 0
[2998] 0 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
[3035] 0 0 0 1 1
Levels: 0 1
```

[797]:
```
# There is NA in pred_y[2947]

summary(pred_y)
```

0	2589	**1**	450

[798]:
```
summary(y_valid)
```

0	2569	**1**	470

[799]:
```
confusionMatrix(pred_y,y_valid, positive = "1")
```

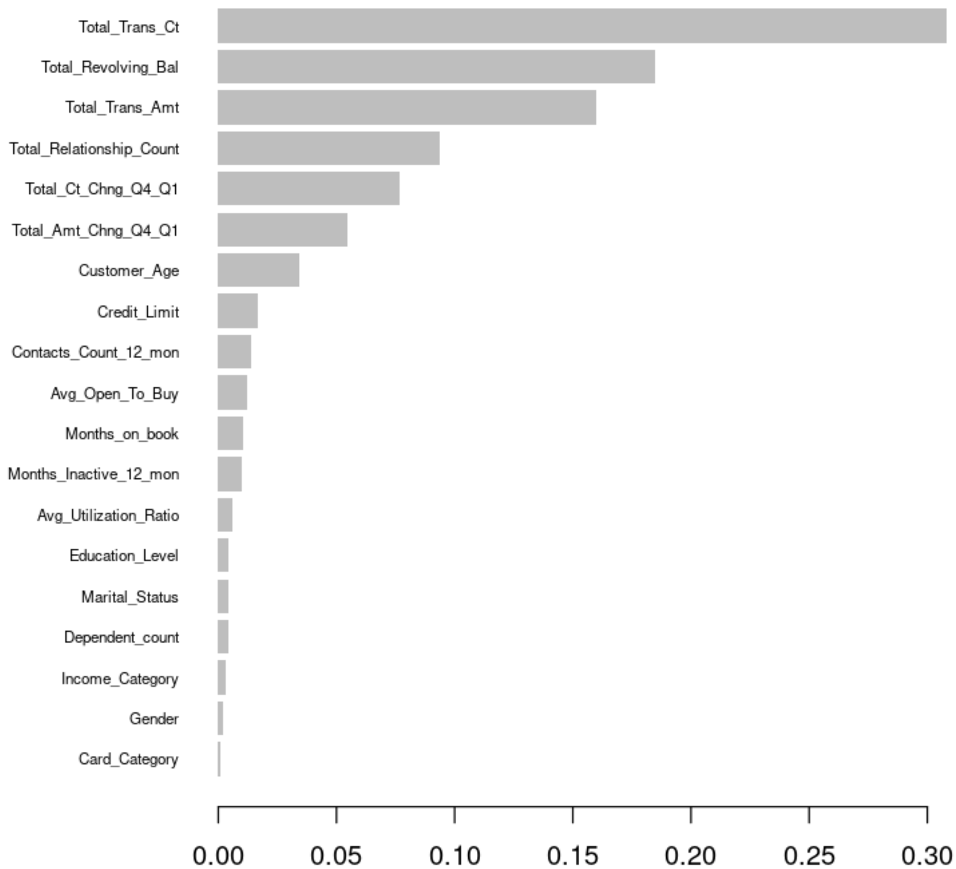
```
Confusion Matrix and Statistics
```

```
          Reference
Prediction    0    1
         0 2524   65
         1   45  405

               Accuracy : 0.9638
                 95% CI : (0.9565, 0.9702)
    No Information Rate : 0.8453
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.8591

 Mcnemar's Test P-Value : 0.07005

            Sensitivity : 0.8617
            Specificity : 0.9825
         Pos Pred Value : 0.9000
         Neg Pred Value : 0.9749
             Prevalence : 0.1547
         Detection Rate : 0.1333
   Detection Prevalence : 0.1481
      Balanced Accuracy : 0.9221

       'Positive' Class : 1
```

[800]:
```
# Compute feature importance matrix
importance_matrix <-  xgb.importance(colnames(xgboost_train), model = xgboost1)
importance_matrix
xgb_gg <- xgb.plot.importance(importance_matrix[1:19,])
```

A data.table: 19 × 4

Feature	Gain	Cover	Frequency
<chr>	<dbl>	<dbl>	<dbl>
Total_Trans_Ct	0.3079314980	0.068669327	0.074264673
Total_Revolving_Bal	0.1850734308	0.062500903	0.057450867
Total_Trans_Amt	0.1598863533	0.190410963	0.130029724
Total_Relationship_Count	0.0936545614	0.011462799	0.031231977
Total_Ct_Chng_Q4_Q1	0.0767662587	0.105933591	0.087839936
Total_Amt_Chng_Q4_Q1	0.0549043735	0.137838096	0.095869748
Customer_Age	0.0340213771	0.037173791	0.111574464
Credit_Limit	0.0167120037	0.106755295	0.082738122
Contacts_Count_12_mon	0.0137869512	0.018345788	0.025863981
Avg_Open_To_Buy	0.0124648973	0.115987454	0.058781775
Months_on_book	0.0102968952	0.033694826	0.046448693
Months_Inactive_12_mon	0.0096621118	0.012370599	0.021738166
Avg_Utilization_Ratio	0.0060217064	0.057756527	0.034115612
Education_Level	0.0045522073	0.009751655	0.039882880
Marital_Status	0.0041735500	0.007463162	0.020762167
Dependent_count	0.0040210290	0.006852109	0.037487245
Income_Category	0.0030935000	0.011500740	0.027283617
Gender	0.0022487288	0.002254082	0.013175990
Card_Category	0.0007285664	0.003278293	0.003460361



1.5.5 Model Comparison

Metric/Model	Decision Tree	Random Forest	Logistic Regression	XGBoost
Accuracy	0.9260	0.9615	0.8388	0.9638
Sensitivity	0.7617	0.8149	0.8511	0.8617
Specificity	0.9560	0.9883	0.8365	0.9825

We will choose the XGboost Model to predict attrition.

1.6 Unsupervised Model : Customer Segmentation

[801]:
```
# only attrition customers
# select part of part of variables
bank1 <- read.csv("data/bank_clean.csv")
attrition <- bank1[bank1$Attrition_Flag == "1",c(6,8,15,18,20,21)]
str(attrition)
```

```
'data.frame':   1627 obs. of  6 variables:
 $ Education_Level      : chr  "Graduate" "Doctorate" "Graduate" "Graduate" …
 $ Income_Category      : chr  "Less than $40K" "Unknown" "Less than $40K"
"$120K +" …
 $ Total_Revolving_Bal  : int  0 605 808 0 0 0 1628 2227 0 0 …
 $ Total_Trans_Amt      : int  692 704 705 602 691 615 836 720 673 530 …
 $ Total_Ct_Chng_Q4_Q1  : num  0.6 0.143 0.9 0.364 0.5 0.714 0.385 0.353 0.8 1
…
 $ Avg_Utilization_Ratio: num  0 0.077 0.562 0 0 0 0.299 0.191 0 0 …
```

[802]:
```
head(attrition)
```

A data.frame: 6 × 6

	Education_Level <chr>	Income_Category <chr>	Total_Revolving_Bal <int>	Total_Trans_Amt <int>	T <
22	Graduate	Less than $40K	0	692	0
40	Doctorate	Unknown	605	704	0
52	Graduate	Less than $40K	808	705	0
55	Graduate	$120K +	0	602	0
62	Graduate	$60K - $80K	0	691	0
83	Unknown	$40K - $60K	0	615	0

[803]:
```
attrition$Education_Level[attrition$Education_Level == 'College'] <- 4
attrition$Education_Level[attrition$Education_Level == 'Doctorate'] <- 7
attrition$Education_Level[attrition$Education_Level == 'Graduate'] <- 5
attrition$Education_Level[attrition$Education_Level == 'High School'] <- 3
attrition$Education_Level[attrition$Education_Level == 'Post-Graduate'] <- 6
attrition$Education_Level[attrition$Education_Level == 'Uneducated'] <- 2
attrition$Education_Level[attrition$Education_Level == 'Unknown'] <- 1
```

[804]:
```
attrition$Education_Level
```

1. '5' 2. '7' 3. '5' 4. '5' 5. '5' 6. '1' 7. '1' 8. '5' 9. '3' 10. '4' 11. '2' 12. '5' 13. '1' 14. '3' 15. '5' 16. '6' 17. '2' 18. '2' 19. '1' 20. '4' 21. '4' 22. '3' 23. '3' 24. '6' 25. '3' 26. '5' 27. '3' 28. '3' 29. '5' 30. '5' 31. '1' 32. '2' 33. '6' 34. '2' 35. '4' 36. '5' 37. '5' 38. '2' 39. '1' 40. '5' 41. '2' 42. '5' 43. '2' 44. '6' 45. '2' 46. '4' 47. '6' 48. '2' 49. '1' 50. '4' 51. '7' 52. '5' 53. '6' 54. '3' 55. '3' 56. '4' 57. '1' 58. '4' 59. '3' 60. '1' 61. '2' 62. '7' 63. '2' 64. '1' 65. '5' 66. '4' 67. '7' 68. '5' 69. '5' 70. '3' 71. '1' 72. '5' 73. '3' 74. '3' 75. '5' 76. '1' 77. '1' 78. '1' 79. '4' 80. '5' 81. '2' 82. '2' 83. '3' 84. '7' 85. '5' 86. '3' 87. '3' 88. '5' 89. '3' 90. '2' 91. '3' 92. '4' 93. '3' 94. '3' 95. '2' 96. '3' 97. '5' 98. '2' 99. '3' 100. '3' 101. '1' 102. '7' 103. '5' 104. '5' 105. '3' 106. '5' 107. '1' 108. '5' 109. '6' 110. '3' 111. '3' 112. '4' 113. '5' 114. '5' 115. '3' 116. '4' 117. '1' 118. '3' 119. '2' 120. '5' 121. '4' 122. '5' 123. '2' 124. '3'

125. '3' 126. '5' 127. '5' 128. '3' 129. '1' 130. '2' 131. '5' 132. '4' 133. '5' 134. '5' 135. '3' 136. '5' 137. '5' 138. '6' 139. '6' 140. '6' 141. '5' 142. '1' 143. '4' 144. '5' 145. '5' 146. '3' 147. '3' 148. '3' 149. '5' 150. '4' 151. '4' 152. '5' 153. '4' 154. '2' 155. '4' 156. '6' 157. '2' 158. '5' 159. '3' 160. '2' 161. '5' 162. '3' 163. '4' 164. '5' 165. '5' 166. '3' 167. '2' 168. '5' 169. '3' 170. '3' 171. '6' 172. '2' 173. '4' 174. '5' 175. '7' 176. '2' 177. '6' 178. '5' 179. '1' 180. '3' 181. '3' 182. '2' 183. '5' 184. '4' 185. '4' 186. '5' 187. '7' 188. '5' 189. '5' 190. '7' 191. '3' 192. '2' 193. '5' 194. '5' 195. '3' 196. '4' 197. '2' 198. '5' 199. '2' 200. '5' 201. 202. '5' 203. '3' 204. '5' 205. '5' 206. '1' 207. '1' 208. '6' 209. '2' 210. '4' 211. '3' 212. '5' 213. '6' 214. '3' 215. '5' 216. '1' 217. '4' 218. '3' 219. '2' 220. '1' 221. '3' 222. '5' 223. '2' 224. '3' 225. '7' 226. '2' 227. '1' 228. '5' 229. '5' 230. '5' 231. '2' 232. '5' 233. '5' 234. '5' 235. '5' 236. '5' 237. '5' 238. '1' 239. '2' 240. '5' 241. '6' 242. '4' 243. '1' 244. '1' 245. '4' 246. '6' 247. '5' 248. '2' 249. '1' 250. '4' 251. '2' 252. '1' 253. '1' 254. '5' 255. '6' 256. '6' 257. '3' 258. '2' 259. '3' 260. '3' 261. '2' 262. '6' 263. '2' 264. '4' 265. '3' 266. '1' 267. '2' 268. '5' 269. '6' 270. '5' 271. '1' 272. '4' 273. '1' 274. '3' 275. '7' 276. '5' 277. '1' 278. '5' 279. '4' 280. '3' 281. '1' 282. '5' 283. '5' 284. '1' 285. '2' 286. '2' 287. '4' 288. '7' 289. '2' 290. '2' 291. '5' 292. '6' 293. '5' 294. '1' 295. '5' 296. '5' 297. '3' 298. '3' 299. '5' 300. '3' 301. '5' 302. '7' 303. '5' 304. '2' 305. '4' 306. '6' 307. '5' 308. '3' 309. '5' 310. '3' 311. '1' 312. '2' 313. '5' 314. '1' 315. '5' 316. '5' 317. '3' 318. '3' 319. '6' 320. '3' 321. '1' 322. '7' 323. '5' 324. '3' 325. '6' 326. '2' 327. '5' 328. '5' 329. '7' 330. '5' 331. '3' 332. '2' 333. '3' 334. '1' 335. '5' 336. '5' 337. '2' 338. '1' 339. '2' 340. '1' 341. '1' 342. '5' 343. '2' 344. '1' 345. '4' 346. '2' 347. '5' 348. '5' 349. '2' 350. '1' 351. '5' 352. '5' 353. '3' 354. '5' 355. '1' 356. '2' 357. '2' 358. '7' 359. '3' 360. '1' 361. '5' 362. '2' 363. '4' 364. '2' 365. '1' 366. '7' 367. '3' 368. '5' 369. '3' 370. '5' 371. '5' 372. '1' 373. '3' 374. '5' 375. '7' 376. '5' 377. '4' 378. '2' 379. '3' 380. '6' 381. '7' 382. '5' 383. '3' 384. '3' 385. '2' 386. '1' 387. '1' 388. '5' 389. '5' 390. '3' 391. '3' 392. '5' 393. '5' 394. '4' 395. '3' 396. '1' 397. '2' 398. '1' 399. '3' 400. '5' 401. '5'

[805]:
```
attrition$Education_Level <- as.numeric(attrition$Education_Level)
attrition$Education_Level
```

1. 5 2. 7 3. 5 4. 5 5. 5 6. 1 7. 1 8. 5 9. 3 10. 4 11. 2 12. 5 13. 1 14. 3 15. 5 16. 6 17. 2 18. 2 19. 1 20. 4 21. 4 22. 3 23. 3 24. 6 25. 3 26. 5 27. 3 28. 3 29. 5 30. 5 31. 1 32. 2 33. 6 34. 2 35. 4 36. 5 37. 5 38. 2 39. 1 40. 5 41. 2 42. 5 43. 2 44. 6 45. 2 46. 4 47. 6 48. 2 49. 1 50. 4 51. 7 52. 5 53. 6 54. 3 55. 3 56. 4 57. 1 58. 4 59. 3 60. 1 61. 2 62. 7 63. 2 64. 1 65. 5 66. 4 67. 7 68. 5 69. 5 70. 3 71. 1 72. 5 73. 3 74. 3 75. 5 76. 1 77. 1 78. 1 79. 4 80. 5 81. 2 82. 2 83. 3 84. 7 85. 5 86. 3 87. 3 88. 5 89. 3 90. 2 91. 3 92. 4 93. 3 94. 3 95. 2 96. 3 97. 5 98. 2 99. 3 100. 3 101. 1 102. 7 103. 5 104. 5 105. 3 106. 5 107. 1 108. 5 109. 6 110. 3 111. 3 112. 4 113. 5 114. 5 115. 3 116. 4 117. 1 118. 3 119. 2 120. 5 121. 4 122. 5 123. 2 124. 3 125. 3 126. 5 127. 5 128. 3 129. 1 130. 2 131. 5 132. 4 133. 5 134. 5 135. 3 136. 5 137. 5 138. 6 139. 6 140. 6 141. 5 142. 1 143. 4 144. 5 145. 5 146. 3 147. 3 148. 3 149. 5 150. 4 151. 4 152. 5 153. 4 154. 2 155. 4 156. 6 157. 2 158. 5 159. 3 160. 2 161. 5 162. 3 163. 4 164. 5 165. 5 166. 3 167. 2 168. 5 169. 3 170. 3 171. 6 172. 2 173. 4 174. 5 175. 7 176. 2 177. 6 178. 5 179. 1 180. 3 181. 3 182. 2 183. 5 184. 4 185. 4 186. 5 187. 7 188. 5 189. 5 190. 7 191. 3 192. 2 193. 5 194. 5 195. 3 196. 4 197. 2 198. 5 199. 2 200. 5 201. 202. 5 203. 3 204. 5 205. 5 206. 1 207. 1 208. 6 209. 2 210. 4 211. 3 212. 5 213. 6 214. 3 215. 5 216. 1 217. 4 218. 3 219. 2 220. 1 221. 3 222. 5 223. 2 224. 3 225. 7 226. 2 227. 1 228. 5 229. 5 230. 5 231. 2 232. 5 233. 5 234. 5 235. 5 236. 5 237. 5 238. 1 239. 2 240. 5 241. 6 242. 4 243. 1 244. 1 245. 4 246. 6 247. 5 248. 2 249. 1 250. 4 251. 2 252. 1 253. 1 254. 5 255. 6 256. 6 257. 3 258. 2 259. 3 260. 3 261. 2 262. 6 263. 2 264. 4 265. 3 266. 1 267. 2 268. 5 269. 6 270. 5 271. 1 272. 4 273. 1 274. 3 275. 7 276. 5 277. 1 278. 5 279. 4 280. 3 281. 1 282. 5 283. 5 284. 1 285. 2 286. 2 287. 4 288. 7 289. 2 290. 2 291. 5 292. 6 293. 5 294. 1 295. 5 296. 5 297. 3 298. 3 299. 5 300. 3 301. 5 302. 7 303. 5 304. 2 305. 4 306. 6 307. 5 308. 3 309. 5 310. 3 311. 1 312. 2 313. 5 314. 1 315. 5 316. 5 317. 3 318. 3 319. 6 320. 3 321. 1 322. 7 323. 5 324. 3 325. 6 326. 2 327. 5 328. 5 329. 7 330. 5 331. 3 332. 2

333. 3 334. 1 335. 5 336. 5 337. 2 338. 1 339. 2 340. 1 341. 1 342. 5 343. 2 344. 1 345. 4 346. 2 347. 5 348. 5 349. 2 350. 1 351. 5 352. 5 353. 3 354. 5 355. 1 356. 2 357. 2 358. 7 359. 3 360. 1 361. 5 362. 2 363. 4 364. 2 365. 1 366. 7 367. 3 368. 5 369. 3 370. 5 371. 5 372. 1 373. 3 374. 5 375. 7 376. 5 377. 4 378. 2 379. 3 380. 6 381. 7 382. 5 383. 3 384. 3 385. 2 386. 1 387. 1 388. 5 389. 5 390. 3 391. 3 392. 5 393. 5 394. 4 395. 3 396. 1 397. 2 398. 1 399. 3 400. 5 401. 5

[806]:
```
attrition$Income_Category[attrition$Income_Category == '$40K - $60K'] <- 3
attrition$Income_Category[attrition$Income_Category == '$120K +'] <- 6
attrition$Income_Category[attrition$Income_Category == '$60K - $80K'] <- 4
attrition$Income_Category[attrition$Income_Category == '$80K - $120K'] <- 5
attrition$Income_Category[attrition$Income_Category == 'Less than $40K'] <- 2
attrition$Income_Category[attrition$Income_Category == 'Unknown'] <- 1
```

[807]:
```
attrition$Income_Category
```

1. ’2’ 2. ’1’ 3. ’2’ 4. ’6’ 5. ’4’ 6. ’3’ 7. ’5’ 8. ’5’ 9. ’2’ 10. ’4’ 11. ’2’ 12. ’5’ 13. ’5’ 14. ’2’ 15. ’2’ 16. ’4’ 17. ’3’ 18. ’5’ 19. ’2’ 20. ’3’ 21. ’2’ 22. ’6’ 23. ’3’ 24. ’3’ 25. ’3’ 26. ’4’ 27. ’1’ 28. ’6’ 29. ’1’ 30. ’6’ 31. ’3’ 32. ’3’ 33. ’5’ 34. ’6’ 35. ’4’ 36. ’2’ 37. ’2’ 38. ’4’ 39. ’2’ 40. ’2’ 41. ’2’ 42. ’4’ 43. ’5’ 44. ’5’ 45. ’2’ 46. ’6’ 47. ’2’ 48. ’1’ 49. ’6’ 50. ’5’ 51. ’1’ 52. ’4’ 53. ’2’ 54. ’4’ 55. ’4’ 56. ’3’ 57. ’5’ 58. ’4’ 59. ’4’ 60. ’3’ 61. ’2’ 62. ’2’ 63. ’4’ 64. ’2’ 65. ’2’ 66. ’4’ 67. ’5’ 68. ’5’ 69. ’2’ 70. ’5’ 71. ’2’ 72. ’2’ 73. ’5’ 74. ’4’ 75. ’2’ 76. ’3’ 77. ’4’ 78. ’1’ 79. ’2’ 80. ’3’ 81. ’4’ 82. ’2’ 83. ’1’ 84. ’4’ 85. ’5’ 86. ’6’ 87. ’5’ 88. ’2’ 89. ’4’ 90. ’5’ 91. ’3’ 92. ’5’ 93. ’4’ 94. ’4’ 95. ’6’ 96. ’2’ 97. ’5’ 98. ’4’ 99. ’2’ 100. ’4’ 101. ’2’ 102. ’2’ 103. ’5’ 104. ’5’ 105. ’3’ 106. ’2’ 107. ’1’ 108. ’5’ 109. ’6’ 110. ’1’ 111. ’3’ 112. ’5’ 113. ’3’ 114. ’2’ 115. ’3’ 116. ’5’ 117. ’4’ 118. ’2’ 119. ’1’ 120. ’5’ 121. ’6’ 122. ’2’ 123. ’5’ 124. ’1’ 125. ’2’ 126. ’5’ 127. ’4’ 128. ’3’ 129. ’1’ 130. ’5’ 131. ’6’ 132. ’3’ 133. ’4’ 134. ’1’ 135. ’1’ 136. ’6’ 137. ’2’ 138. ’4’ 139. ’2’ 140. ’3’ 141. ’2’ 142. ’3’ 143. ’1’ 144. ’2’ 145. ’2’ 146. ’3’ 147. ’4’ 148. ’3’ 149. ’6’ 150. ’5’ 151. ’5’ 152. ’6’ 153. ’5’ 154. ’2’ 155. ’6’ 156. ’2’ 157. ’5’ 158. ’2’ 159. ’4’ 160. ’3’ 161. ’2’ 162. ’6’ 163. ’5’ 164. ’3’ 165. ’6’ 166. ’2’ 167. ’4’ 168. ’2’ 169. ’5’ 170. ’3’ 171. ’2’ 172. ’6’ 173. ’4’ 174. ’5’ 175. ’6’ 176. ’6’ 177. ’2’ 178. ’6’ 179. ’2’ 180. ’2’ 181. ’4’ 182. ’2’ 183. ’3’ 184. ’2’ 185. ’1’ 186. ’3’ 187. ’2’ 188. ’3’ 189. ’2’ 190. ’5’ 191. ’1’ 192. ’2’ 193. ’2’ 194. ’4’ 195. ’1’ 196. ’5’ 197. ’6’ 198. ’5’ 199. ’5’ 200. ’5’ 201. 202. ’2’ 203. ’5’ 204. ’2’ 205. ’2’ 206. ’4’ 207. ’4’ 208. ’5’ 209. ’5’ 210. ’2’ 211. ’6’ 212. ’5’ 213. ’3’ 214. ’4’ 215. ’5’ 216. ’4’ 217. ’5’ 218. ’5’ 219. ’1’ 220. ’6’ 221. ’2’ 222. ’2’ 223. ’5’ 224. ’5’ 225. ’2’ 226. ’5’ 227. ’1’ 228. ’5’ 229. ’3’ 230. ’3’ 231. ’4’ 232. ’5’ 233. ’2’ 234. ’4’ 235. ’5’ 236. ’4’ 237. ’5’ 238. ’6’ 239. ’5’ 240. ’5’ 241. ’2’ 242. ’5’ 243. ’6’ 244. ’1’ 245. ’5’ 246. ’5’ 247. ’4’ 248. ’6’ 249. ’2’ 250. ’5’ 251. ’2’ 252. ’5’ 253. ’5’ 254. ’3’ 255. ’5’ 256. ’3’ 257. ’5’ 258. ’3’ 259. ’5’ 260. ’1’ 261. ’2’ 262. ’4’ 263. ’2’ 264. ’5’ 265. ’2’ 266. ’2’ 267. ’2’ 268. ’5’ 269. ’2’ 270. ’2’ 271. ’5’ 272. ’4’ 273. ’6’ 274. ’2’ 275. ’2’ 276. ’3’ 277. ’3’ 278. ’4’ 279. ’4’ 280. ’2’ 281. ’1’ 282. ’2’ 283. ’5’ 284. ’2’ 285. ’6’ 286. ’3’ 287. ’5’ 288. ’5’ 289. ’2’ 290. ’5’ 291. ’2’ 292. ’4’ 293. ’3’ 294. ’3’ 295. ’2’ 296. ’5’ 297. ’6’ 298. ’5’ 299. ’6’ 300. ’2’ 301. ’5’ 302. ’2’ 303. ’5’ 304. ’3’ 305. ’3’ 306. ’2’ 307. ’3’ 308. ’4’ 309. ’6’ 310. ’2’ 311. ’3’ 312. ’2’ 313. ’5’ 314. ’5’ 315. ’2’ 316. ’5’ 317. ’5’ 318. ’3’ 319. ’2’ 320. ’2’ 321. ’4’ 322. ’6’ 323. ’3’ 324. ’4’ 325. ’3’ 326. ’4’ 327. ’6’ 328. ’3’ 329. ’4’ 330. ’3’ 331. ’4’ 332. ’5’ 333. ’5’ 334. ’3’ 335. ’2’ 336. ’5’ 337. ’2’ 338. ’2’ 339. ’1’ 340. ’4’ 341. ’5’ 342. ’4’ 343. ’5’ 344. ’6’ 345. ’6’ 346. ’3’ 347. ’5’ 348. ’6’ 349. ’6’ 350. ’1’ 351. ’5’ 352. ’2’ 353. ’4’ 354. ’1’ 355. ’5’ 356. ’5’ 357. ’2’ 358. ’2’ 359. ’1’ 360. ’3’ 361. ’5’ 362. ’2’ 363. ’3’ 364. ’5’ 365. ’6’ 366. ’2’ 367. ’3’ 368. ’2’ 369. ’1’ 370. ’1’ 371. ’4’ 372. ’2’ 373. ’6’ 374. ’4’ 375. ’1’ 376. ’3’ 377. ’5’ 378. ’6’ 379. ’4’ 380. ’3’ 381. ’2’ 382. ’2’ 383. ’2’ 384. ’4’ 385. ’3’ 386. ’3’ 387. ’2’ 388. ’1’ 389. ’6’ 390. ’5’ 391. ’2’ 392. ’4’ 393. ’5’ 394. ’6’ 395. ’4’ 396. ’5’ 397. ’1’ 398. ’3’ 399. ’2’ 400. ’3’ 401. ’2’

[808]:
```
attrition$Income_Category <- as.numeric(attrition$Income_Category)
attrition$Income_Category
```

1. 2 2. 1 3. 2 4. 6 5. 4 6. 3 7. 5 8. 5 9. 2 10. 4 11. 2 12. 5 13. 5 14. 2 15. 2 16. 4 17. 3 18. 5 19. 2 20. 3 21. 2 22. 6 23. 3 24. 3 25. 3 26. 4 27. 1 28. 6 29. 1 30. 6 31. 3 32. 3 33. 5 34. 6 35. 4 36. 2 37. 2 38. 4 39. 2 40. 2 41. 2 42. 4 43. 5 44. 5 45. 2 46. 6 47. 2 48. 1 49. 6 50. 5 51. 1 52. 4 53. 2 54. 4 55. 4 56. 3 57. 5 58. 4 59. 4 60. 3 61. 2 62. 2 63. 4 64. 2 65. 2 66. 4 67. 5 68. 5 69. 2 70. 5 71. 2 72. 2 73. 5 74. 4 75. 2 76. 3 77. 4 78. 1 79. 2 80. 3 81. 4 82. 2 83. 1 84. 4 85. 5 86. 6 87. 5 88. 2 89. 4 90. 5 91. 3 92. 5 93. 4 94. 4 95. 6 96. 2 97. 5 98. 4 99. 2 100. 4 101. 2 102. 2 103. 5 104. 5 105. 3 106. 2 107. 1 108. 5 109. 6 110. 1 111. 3 112. 5 113. 3 114. 2 115. 3 116. 5 117. 4 118. 2 119. 1 120. 5 121. 6 122. 2 123. 5 124. 1 125. 2 126. 5 127. 4 128. 3 129. 1 130. 5 131. 6 132. 3 133. 4 134. 1 135. 1 136. 6 137. 2 138. 4 139. 2 140. 3 141. 2 142. 3 143. 1 144. 2 145. 2 146. 3 147. 4 148. 3 149. 6 150. 5 151. 5 152. 6 153. 5 154. 2 155. 6 156. 2 157. 5 158. 2 159. 4 160. 3 161. 2 162. 6 163. 5 164. 3 165. 6 166. 2 167. 4 168. 2 169. 5 170. 3 171. 2 172. 6 173. 4 174. 5 175. 6 176. 6 177. 2 178. 6 179. 2 180. 2 181. 4 182. 2 183. 3 184. 2 185. 1 186. 3 187. 2 188. 3 189. 2 190. 5 191. 1 192. 2 193. 2 194. 4 195. 1 196. 5 197. 6 198. 5 199. 5 200. 5 201. 202. 2 203. 5 204. 2 205. 2 206. 4 207. 4 208. 5 209. 5 210. 2 211. 6 212. 5 213. 3 214. 4 215. 5 216. 4 217. 5 218. 5 219. 1 220. 6 221. 2 222. 2 223. 5 224. 5 225. 2 226. 5 227. 1 228. 5 229. 3 230. 3 231. 4 232. 5 233. 2 234. 4 235. 5 236. 4 237. 5 238. 6 239. 5 240. 5 241. 2 242. 5 243. 6 244. 1 245. 5 246. 5 247. 4 248. 6 249. 2 250. 5 251. 2 252. 5 253. 5 254. 3 255. 5 256. 3 257. 5 258. 3 259. 5 260. 1 261. 2 262. 4 263. 2 264. 5 265. 2 266. 2 267. 2 268. 5 269. 2 270. 2 271. 5 272. 4 273. 6 274. 2 275. 2 276. 3 277. 3 278. 4 279. 4 280. 2 281. 1 282. 2 283. 5 284. 2 285. 6 286. 3 287. 5 288. 5 289. 2 290. 5 291. 2 292. 4 293. 3 294. 3 295. 2 296. 5 297. 6 298. 5 299. 6 300. 2 301. 5 302. 2 303. 5 304. 3 305. 3 306. 2 307. 3 308. 4 309. 6 310. 2 311. 3 312. 2 313. 5 314. 5 315. 2 316. 5 317. 5 318. 3 319. 2 320. 2 321. 4 322. 6 323. 3 324. 4 325. 3 326. 4 327. 6 328. 3 329. 4 330. 3 331. 4 332. 5 333. 5 334. 3 335. 2 336. 5 337. 2 338. 2 339. 1 340. 4 341. 5 342. 4 343. 5 344. 6 345. 6 346. 3 347. 5 348. 6 349. 6 350. 1 351. 5 352. 2 353. 4 354. 1 355. 5 356. 5 357. 2 358. 2 359. 1 360. 3 361. 5 362. 2 363. 3 364. 5 365. 6 366. 2 367. 3 368. 2 369. 1 370. 1 371. 4 372. 2 373. 6 374. 4 375. 1 376. 3 377. 5 378. 6 379. 4 380. 3 381. 2 382. 2 383. 2 384. 4 385. 3 386. 3 387. 2 388. 1 389. 6 390. 5 391. 2 392. 4 393. 5 394. 6 395. 4 396. 5 397. 1 398. 3 399. 2 400. 3 401. 2

[809]:
```
#factor(attrition$Education_Level)
#levels(attrition$Education_Level) <- c("4","7","5","3","6","2","1")
#levels(attrition$Education_Level)
# 'College''Doctorate''Graduate''High␣
↪School''Post-Graduate''Uneducated''Unknown'
# c("4","7","5","3","6","2","1")

#factor(attrition$Income_Category)
#levels(attrition$Income_Category) <- c("3","6","4","5","2","1")
# '$40K - $60K''$120K +''$60K - $80K''$80K - $120K''Less than $40K''Unknown'
#c("3","6","4","5","2","1")

#levels(attrition$Income_Category)
```

[810]:
```
# attrition$Education_Level <- as.numeric(attrition$Education_Level)
```

[811]:
```
# convert categorical variables to numerical variables
#attrition$Education_Level <- as.numeric(attrition$Education_Level)


#attrition$Income_Category <- as.numeric(attrition$Income_Category)
# select part of part of variables
str(attrition)
```

```
'data.frame':	1627 obs. of  6 variables:
 $ Education_Level      : num  5 7 5 5 5 1 1 5 3 4 …
 $ Income_Category      : num  2 1 2 6 4 3 5 5 2 4 …
 $ Total_Revolving_Bal  : int  0 605 808 0 0 0 1628 2227 0 0 …
 $ Total_Trans_Amt      : int  692 704 705 602 691 615 836 720 673 530 …
 $ Total_Ct_Chng_Q4_Q1  : num  0.6 0.143 0.9 0.364 0.5 0.714 0.385 0.353 0.8 1
…
 $ Avg_Utilization_Ratio: num  0 0.077 0.562 0 0 0 0.299 0.191 0 0 …
```

1.6.1 Use K-means to Cluster the Attrited Customers

Available components of the results of kmeans include:

- cluster: a vector of integers (from 1:K) indicating the cluster to which each point is allocated.
- centers: cluster centers, each cluster has a vector of variable means
- withinss: within-cluster sum of squares, one component per cluster.
- tot.withinss: total within-cluster sum of squares, i.e. sum(withinss).
- size: the number of points in each cluster.

[812]:
```
# Cluster - K-means Model:
# normalize data
library(caret)
# compute mean and standard deviation of each column
norm.values <- preProcess(attrition, method=c("center", "scale"))
# we perform the transformation/normalization
attrition.norm <-predict(norm.values, attrition)
# set seed for reproducibility
set.seed(1234)
km <- kmeans(attrition.norm, 3)
# centroids
km$centers
```

A matrix: 3×6 of type dbl

	Education_Level	Income_Category	Total_Revolving_Bal	Total_Trans_A
1	-0.079536610	0.40210565	0.0279693	2.1288967
2	0.022243655	-0.02667072	-0.5549751	-0.3875653
3	-0.006875095	-0.18877445	1.4281584	-0.3574894

[813]:
```
# within-cluster sum of squared distances
km$withinss
```

1. 1164.63463344399 2. 3025.50568099584 3. 1595.9012361028

[814]:
```
# total within-cluster sum of square
avg3 <- km$tot.withinss/3
avg3
```

1928.68051684754

[815]:
```
km1 <- kmeans(attrition.norm, 1)
avg1 <- km1$tot.withinss
avg1
```

9755.99999999986

[816]:
```
km2 <- kmeans(attrition.norm, 2)
avg2 <- km2$tot.withinss/2
avg2
```

3971.12003176821

[817]:
```
km4<- kmeans(attrition.norm, 4)
avg4 <- km4$tot.withinss/4
avg4
```

1250.18227502076

[818]:
```
km5<- kmeans(attrition.norm, 5)
avg5 <- km5$tot.withinss/5
avg5
```

873.778031994046

[819]:
```
km6 <- kmeans(attrition.norm, 6)
avg6 <- km6$tot.withinss/6
avg6
```

677.274905868985

[820]:
```
km7<- kmeans(attrition.norm, 7)
avg7 <- km7$tot.withinss/7
avg7
```

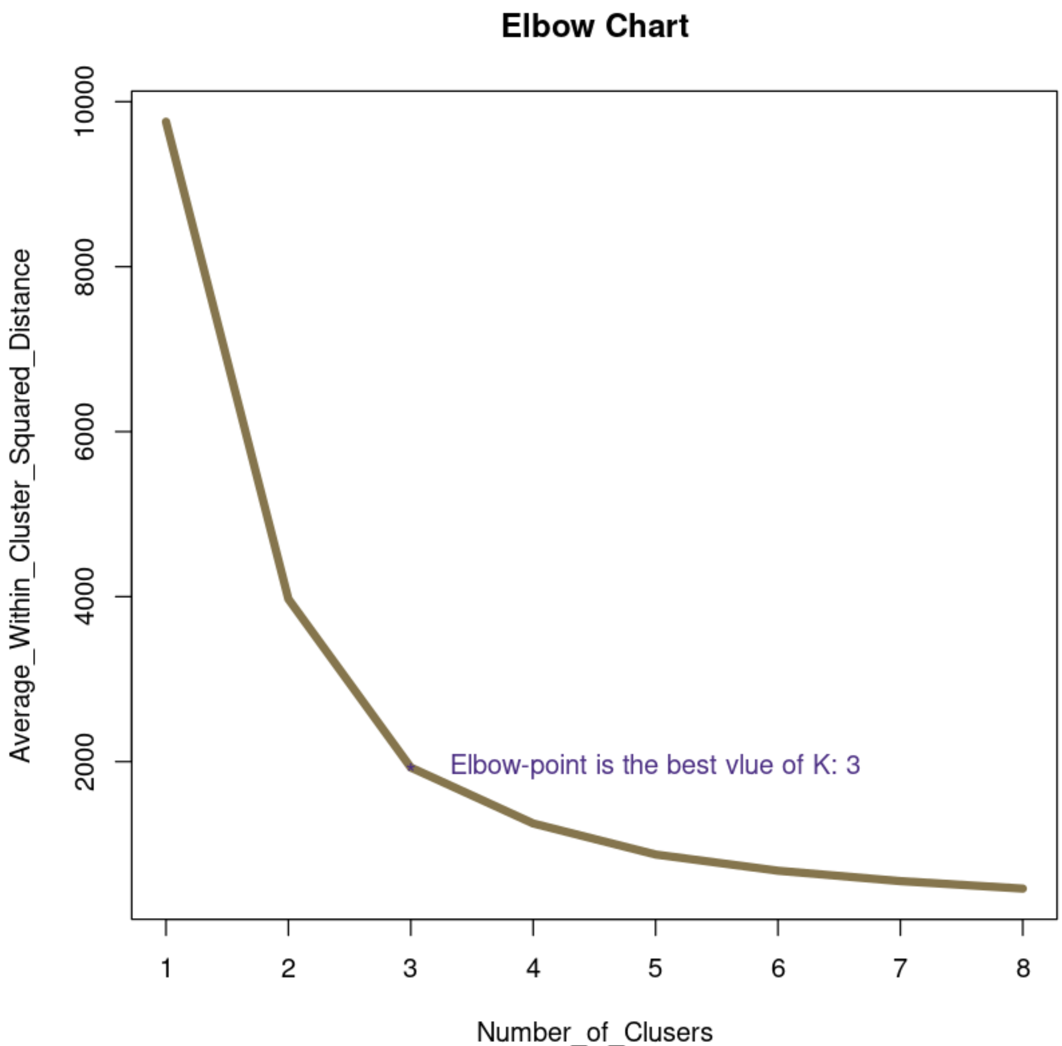
551.211305448226

[821]:
```
km8<- kmeans(attrition.norm, 8)
avg8 <- km8$tot.withinss/8
avg8
```

460.764734983544

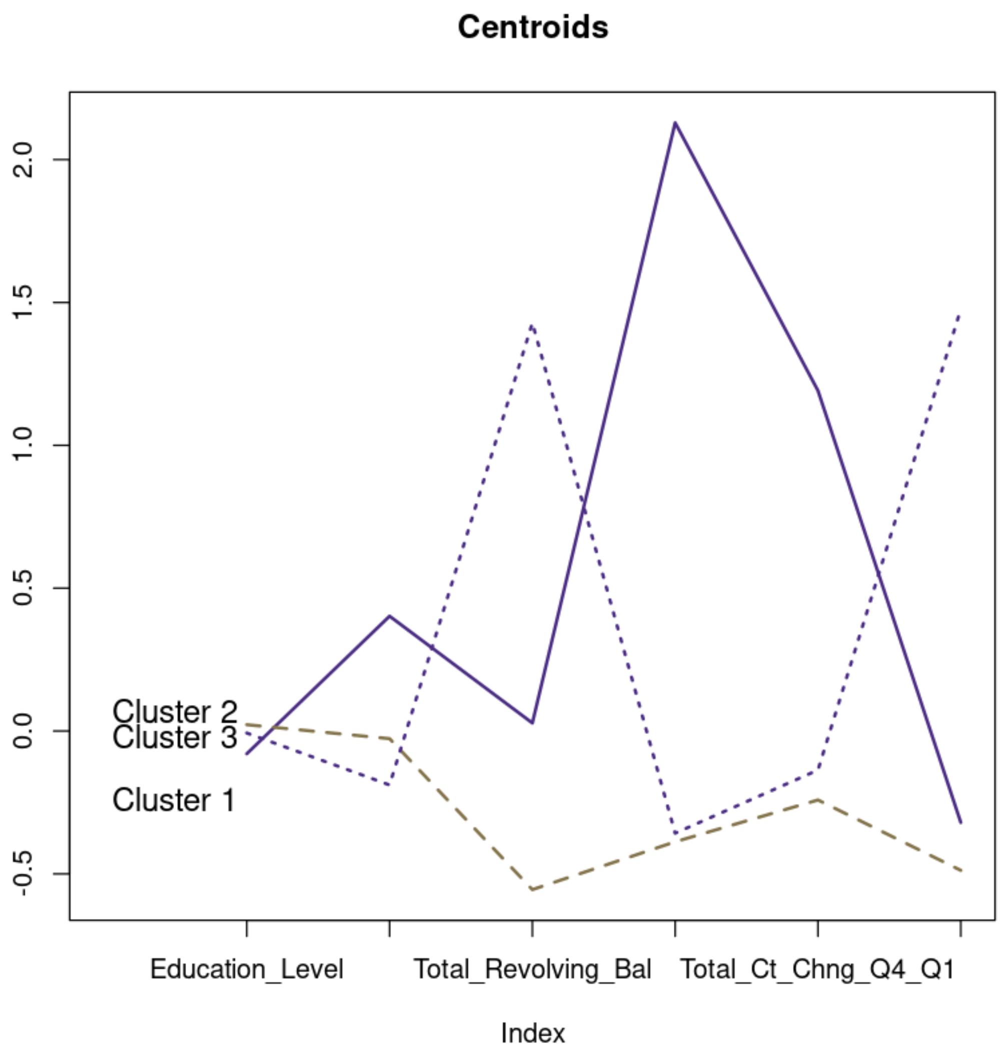
[822]:
```
# Plot the elbow chart to select best value for k
Number_of_Clusers <- c(1,2,3,4,5,6,7,8)
```

```
Average_Within_Cluster_Squared_Distance <- c(avg1, avg2,␣
↪avg3,avg4,avg5,avg6,avg7, avg8)
plot(Number_of_Clusers,Average_Within_Cluster_Squared_Distance,type="l", col =␣
↪"#85754d", lwd=5, main="Elbow Chart")
# Elbow-point is the best vlue of K: 2
text(5, 1937,
     "Elbow-point is the best vlue of K: 3",col = "#4b2e83")
text(3,1937, "*",col= "#4b2e83", font = 80)
```



[823]:
```
# Interpret the resulting clusters
# Profiling centroids
pp <- plot(c(0), xaxt = 'n', ylab = "", type = "l",
```

```
     ylim = c(min(km$centers), max(km$centers)), xlim = c(0, 6),␣
 ↪main="Centroids")

# label x-axes
axis(1, at = c(1:6), labels = names(attrition))

# plot centroids
for (i in c(1:3))
  lines(km$centers[i,], lty = i, lwd = 2, col = ifelse(i %in% c(1,3),
                                                        "#4b2e83" , "#85754d"))

# name clusters
text(x = 0.5, y = km$centers[, 1]*3, cex= 1.2,labels = paste("Cluster", c(1:3)))
```



[824]:
```
#PVisualizing clusters in two dimensions
install.packages("factoextra")
library(factoextra)
```

```
Installing factoextra [1.0.7] ...
        OK [linked cache]
```

[825]:
```
fv <- fviz_cluster(km, geom = "point", data = attrition) + ggtitle("Clustering
 ↪Results")
fv + theme(text = element_text(size = 18))
```

1.6.2 Insights from Clustering:

After we selected the XGBoost model to predict which customers would churn in the future, we wanted to dive deeper into the attrited customer group (16%). We ran the unsupervised learning model K-means to segment customers. Using an elbow chart, we selected 3 as the best value of K (number of clusters). The resulting clusters allowed us to identify some characteristics of each. - Cluster 1 as "High Value", containing customers with high income and recently decreasing transaction activities. - Cluster 2, as "Potential Value", had customers of medium income level and low engagement. - Cluster 3, as "Risky", contained individuals with low income and high revolving balances.

1.7 Text Mining for Customer Reviews

1.7.1 Import Data and Build a Corpus

- Build a corpus containing all the docs. The main structure for managing documents in tm is a so-called Corpus, representing a collection of text documents. VectorSource(x) interprets each element of the vector x as a document.
- Another useful way to build a corpus is using ZipSource(x). A ZIP file source extracts a compressed ZIP file via unzip and interprets each file as a document. An example usage can be corp <- Corpus(ZipSource("myfiles.zip", recursive = T)).

[826]:
```
# Text Ming Part
# read reveiw data:
review <- read.csv("data/card_review.csv")
```

[827]:
```
summary(review)
```

```
  Card.Name           Category          Date.Reviewed         Profiles
 Length:7513        Length:7513        Length:7513        Length:7513
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character
   Ratings            Reviews             Source           Bank.Name
 Length:7513        Length:7513        Length:7513        Length:7513
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character
   Old.New
 Length:7513
 Class :character
 Mode  :character
```

[828]:
```
str(review)
```

```
'data.frame':   7513 obs. of  9 variables:
 $ Card.Name    : chr  "RBL Platinum Delight" "AXIS VISTARA" "HDFC JET
PRIVILEGE" "SBI AIR INDIA SIGNATURE" …
 $ Category     : chr  "Rewards" "Travel" "Travel" "Travel" …
 $ Date.Reviewed: chr  " Apr 02, 2019" " Apr 03, 2019" " Apr 03, 2019" " Apr 03,
2019" …
```

```
 $ Profiles     : chr  "User" "User" "User" "User" …
 $ Ratings      : chr  "5" "4" "5" "4" …
 $ Reviews      : chr  "I have been using RBL PLATINUM card since 2017. I am
getting buy one get one movie tickets free from BOOK MY SH"| __truncated__ "I
have just started using AXIS BANK VISTARA CREDIT CARD and the features are good.
The credit limit was average"| __truncated__ "I have taken SBI JET PRIVILEGE
CARD this is life time free card .Credit limit is sufficient  and good.Customer
"| __truncated__ "I have Air India Signature card and its been 2 months. The
credit limit is more than 50K which was okay. They a"| __truncated__ …
 $ Source       : chr  "BANK BAAZAR" "BANK BAAZAR" "BANK BAAZAR" "BANK BAAZAR"
…
 $ Bank.Name    : chr  "RBL" "Axis Bank" "HDFC Bank" "SBI" …
 $ Old.New      : chr  "New" "Old" "Old" "Old" …
```

[829]:
```r
head(review)
```

A data.frame: 6 × 9

	Card.Name <chr>	Category <chr>	Date.Reviewed <chr>	Profiles <chr>	Ratings <chr>	Revie <chr
1	RBL Platinum Delight	Rewards	Apr 02, 2019	User	5	I have
2	AXIS VISTARA	Travel	Apr 03, 2019	User	4	I have
3	HDFC JET PRIVILEGE	Travel	Apr 03, 2019	User	5	I have
4	SBI AIR INDIA SIGNATURE	Travel	Apr 03, 2019	User	4	I have
5	CITIBANK PREMIERMILES	Travel	Apr 03, 2019	User	4	I hold
6	CITIBANK PREMIERMILES	Travel	Apr 03, 2019	User	4	The c

[830]:
```r
review$Ratings[review$Ratings == "0"] <- "Bad"
review$Ratings[review$Ratings == "0.5" ] <- "Bad"
review$Ratings[review$Ratings == 1] <- "Bad"
review$Ratings[review$Ratings == 2] <- "Bad"
review$Ratings[review$Ratings == "2.5"] <- "Bad"
review$Ratings[review$Ratings == "3"] <- "Bad"
review$Ratings[review$Ratings == "3.5"] <- "Bad"
review$Ratings[review$Ratings == "4"] <- "Good"
review$Ratings[review$Ratings == "4.5"] <- "Good"
review$Ratings[review$Ratings == "5"] <- "Good"
review$Ratings[review$Ratings == "None"] <- "Unknown"
```

[831]:
```r
str(review)
```

```
'data.frame':   7513 obs. of  9 variables:
 $ Card.Name    : chr  "RBL Platinum Delight" "AXIS VISTARA" "HDFC JET
PRIVILEGE" "SBI AIR INDIA SIGNATURE" …
 $ Category     : chr  "Rewards" "Travel" "Travel" "Travel" …
 $ Date.Reviewed: chr  " Apr 02, 2019" " Apr 03, 2019" " Apr 03, 2019" " Apr 03,
2019" …
 $ Profiles     : chr  "User" "User" "User" "User" …
 $ Ratings      : chr  "Good" "Good" "Good" "Good" …
 $ Reviews      : chr  "I have been using RBL PLATINUM card since 2017. I am
```

```
getting buy one get one movie tickets free from BOOK MY SH"| __truncated__ "I
have just started using AXIS BANK VISTARA CREDIT CARD and the features are good.
The credit limit was average"| __truncated__ "I have taken SBI JET PRIVILEGE
CARD this is life time free card .Credit limit is sufficient  and good.Customer
"| __truncated__ "I have Air India Signature card and its been 2 months. The
credit limit is more than 50K which was okay. They a"| __truncated__ …
 $ Source       : chr  "BANK BAAZAR" "BANK BAAZAR" "BANK BAAZAR" "BANK BAAZAR"
…
 $ Bank.Name    : chr  "RBL" "Axis Bank" "HDFC Bank" "SBI" …
 $ Old.New      : chr  "New" "Old" "Old" "Old" …
```

[832]:
```r
head(review)
```

A data.frame: 6 × 9

	Card.Name <chr>	Category <chr>	Date.Reviewed <chr>	Profiles <chr>	Ratings <chr>	Revie <chr
1	RBL Platinum Delight	Rewards	Apr 02, 2019	User	Good	I have
2	AXIS VISTARA	Travel	Apr 03, 2019	User	Good	I have
3	HDFC JET PRIVILEGE	Travel	Apr 03, 2019	User	Good	I have
4	SBI AIR INDIA SIGNATURE	Travel	Apr 03, 2019	User	Good	I have
5	CITIBANK PREMIERMILES	Travel	Apr 03, 2019	User	Good	I hold
6	CITIBANK PREMIERMILES	Travel	Apr 03, 2019	User	Good	The c

[833]:
```r
goodreview <- review[review$Ratings == "Good",]
badreview <- review[review$Ratings == "Bad",]

str(goodreview)
```

```
'data.frame':   880 obs. of  9 variables:
 $ Card.Name    : chr  "RBL Platinum Delight" "AXIS VISTARA" "HDFC JET
PRIVILEGE" "SBI AIR INDIA SIGNATURE" …
 $ Category     : chr  "Rewards" "Travel" "Travel" "Travel" …
 $ Date.Reviewed: chr  " Apr 02, 2019" " Apr 03, 2019" " Apr 03, 2019" " Apr 03,
2019" …
 $ Profiles     : chr  "User" "User" "User" "User" …
 $ Ratings      : chr  "Good" "Good" "Good" "Good" …
 $ Reviews      : chr  "I have been using RBL PLATINUM card since 2017. I am
getting buy one get one movie tickets free from BOOK MY SH"| __truncated__ "I
have just started using AXIS BANK VISTARA CREDIT CARD and the features are good.
The credit limit was average"| __truncated__ "I have taken SBI JET PRIVILEGE
CARD this is life time free card .Credit limit is sufficient  and good.Customer
"| __truncated__ "I have Air India Signature card and its been 2 months. The
credit limit is more than 50K which was okay. They a"| __truncated__ …
 $ Source       : chr  "BANK BAAZAR" "BANK BAAZAR" "BANK BAAZAR" "BANK BAAZAR"
…
 $ Bank.Name    : chr  "RBL" "Axis Bank" "HDFC Bank" "SBI" …
 $ Old.New      : chr  "New" "Old" "Old" "Old" …
```

[834]:
```r
str(badreview)
```

```
'data.frame':	736 obs. of  9 variables:
 $ Card.Name    : chr  "HDFC JET PRIVILEGE" "AXIS VISTARA" "AXIS VISTARA
SIGNATURE" "HDFC DINERS CLUB BLACK" …
 $ Category     : chr  "Travel" "Travel" "Travel" "Lifestyle" …
 $ Date.Reviewed: chr  " Apr 05, 2017" " Apr 09, 2019" " Apr 13, 2018" " Apr 30,
2019" …
 $ Profiles     : chr  "User" "User" "User" "User" …
 $ Ratings      : chr  "Bad" "Bad" "Bad" "Bad" …
 $ Reviews      : chr  "Hdfc bank need to improve on their response. Well, I get
good credit limit with this card. I also get the rewar"| __truncated__ "I have
taken a AXIS BANK VISTARA SIGNATURE CREDIT CARD. Annual charges for this card is
Rs. 3000,i am not sure "| __truncated__ "I hold a credit card with Axis Bank.
The process was fine and the team was on time to reach out and I got the c"|
__truncated__ "I have Hdfc  Bank Credit card ,The credit liit  was good ,The
credit card received  on time .I am ,using this c"| __truncated__ …
 $ Source       : chr  "BANK BAAZAR" "BANK BAAZAR" "BANK BAAZAR" "BANK BAAZAR"
…
 $ Bank.Name    : chr  "HDFC Bank" "Axis Bank" "Axis Bank" "HDFC Bank" …
 $ Old.New      : chr  "Old" "Old" "Old" "Old" …
```

[835]:
```
# install.packages("tm")
install.packages("tm")
```

```
Installing tm [0.7-9] …
	OK [linked cache]
```

[836]:
```
# load library(tm) for NLP
library(tm)
```

[837]:
```
# convert "Reviews"column into a corpus
corp <- Corpus(VectorSource(review$Reviews))
corpgood <- Corpus(VectorSource(goodreview$Reviews))
corpbad <- Corpus(VectorSource(badreview$Reviews))
```

1.7.2 Clean and Pre-process the Text Data

Once we have a corpus we typically want to modify the documents in it, e.g., stemming, stopword removal. Pre-processing are done via the tm_map() function which applies (maps) a function to all documents of the corpus.

[838]:
```
#Clean and pre-process the text data
# 1. Switch to lower case
corp <- tm_map(corp, tolower)
corpgood <- tm_map(corpgood, tolower)
corpbad <- tm_map(corpbad, tolower)
# 2. Remove numbers
corp <- tm_map(corp, removeNumbers)
corpgood <- tm_map(corpgood, removeNumbers)
```

```
corpbad <- tm_map(corpbad, removeNumbers)

# 3. Remove punctuation marks
corp <- tm_map(corp, removePunctuation)
corpgood <- tm_map(corpgood, removePunctuation)
corpbad <- tm_map(corpbad, removePunctuation)


# 4. Remove stopwords
# These  include words such as articles (a, an, the), conjunctions (and, or but etc.), common verbs (is), qualifiers (yet, however etc) . The tm package includes  a standard list of such stop words
# examine the list of stopwords by typing in:
# stopwords("english")
corp <- tm_map(corp, removeWords, stopwords("english"))
corpgood <- tm_map(corpgood, removeWords, stopwords("english"))
corpbad <- tm_map(corpbad, removeWords, stopwords("english"))

# 5. remove "Mojibake" words
corp <- tm_map(corp, removeWords, c('â€ ','â€œ','ðŸ ','â€','sbi')) # need to be improved.
corpgood <- tm_map(corpgood, removeWords, c('â€ ','â€œ','ðŸ ','â€','sbi'))
corpbad <- tm_map(corpbad, removeWords, c('â€ ','â€œ','ðŸ ','â€','sbi'))

# 6. Remove extra whitespaces
corp <- tm_map(corp, stripWhitespace)
corpgood <- tm_map(corpgood, stripWhitespace)
corpbad <- tm_map(corpbad, stripWhitespace)
```

```
Warning message in tm_map.SimpleCorpus(corp, tolower):
"transformation drops documents"
Warning message in tm_map.SimpleCorpus(corpgood, tolower):
"transformation drops documents"
Warning message in tm_map.SimpleCorpus(corpbad, tolower):
"transformation drops documents"
Warning message in tm_map.SimpleCorpus(corp, removeNumbers):
"transformation drops documents"
Warning message in tm_map.SimpleCorpus(corpgood, removeNumbers):
"transformation drops documents"
Warning message in tm_map.SimpleCorpus(corpbad, removeNumbers):
"transformation drops documents"
Warning message in tm_map.SimpleCorpus(corp, removePunctuation):
"transformation drops documents"
Warning message in tm_map.SimpleCorpus(corpgood, removePunctuation):
"transformation drops documents"
Warning message in tm_map.SimpleCorpus(corpbad, removePunctuation):
"transformation drops documents"
```

```
Warning message in tm_map.SimpleCorpus(corp, removeWords, stopwords("english")):
"transformation drops documents"
Warning message in tm_map.SimpleCorpus(corpgood, removeWords,
stopwords("english")):
"transformation drops documents"
Warning message in tm_map.SimpleCorpus(corpbad, removeWords,
stopwords("english")):
"transformation drops documents"
Warning message in tm_map.SimpleCorpus(corp, removeWords, c("â€ ", "â€œ", :
"transformation drops documents"
Warning message in tm_map.SimpleCorpus(corpgood, removeWords, c("â€ ", "â€œ", :
"transformation drops documents"
Warning message in tm_map.SimpleCorpus(corpbad, removeWords, c("â€ ", "â€œ", :
"transformation drops documents"
Warning message in tm_map.SimpleCorpus(corp, stripWhitespace):
"transformation drops documents"
Warning message in tm_map.SimpleCorpus(corpgood, stripWhitespace):
"transformation drops documents"
Warning message in tm_map.SimpleCorpus(corpbad, stripWhitespace):
"transformation drops documents"
```

1.7.3 Stemming

Another important preprocessing step is to make a text stemming which reduces words to their root form. In other words, this process removes suffixes from words to make it simple and to get the common origin. Examples:

cats -> cat travel, traveling, traveled -> travel We will next use an R package SnowballC (Snowball stemmers based on the C libstemmer UTF-8 library) to collapse words to a common root to aid comparison of vocabulary.

[839]:
```
install.packages("SnowballC")
```

```
Installing SnowballC [0.7.0] …
        OK [linked cache]
```

[840]:
```
library(SnowballC)
corp <- tm_map(corp, stemDocument)
corpgood <- tm_map(corpgood, stemDocument)
corpbad <- tm_map(corpbad, stemDocument)
```

```
Warning message in tm_map.SimpleCorpus(corp, stemDocument):
"transformation drops documents"
Warning message in tm_map.SimpleCorpus(corpgood, stemDocument):
"transformation drops documents"
Warning message in tm_map.SimpleCorpus(corpbad, stemDocument):
"transformation drops documents"
```

1.7.4 Generate the Spreadsheet Representation of the Documents

Columns are terms (words) Rows are documents Each cell can represent presence, term frequency, or, IDF weighted term frequency (TF-IDF) Basic definition:

Term Frequency TF(t,d): The number of times a term (word) t appears in each document d. Inverse Document Frequency (IDF): IDF(t)=log(total number of documents# documents containing term t) IDF accounts for terms that appear frequently in all the documents. TF-IDF matrix: TF-IDF(t,d)=TF(t,d)×IDF(t)

[841]:
```
# find out Term-Document matirx based on Term Frequency
dtm <- DocumentTermMatrix(corp)
dtm_good <- DocumentTermMatrix(corpgood)
dtm_bad <- DocumentTermMatrix(corpbad)
inspect(dtm)
```

```
<<DocumentTermMatrix (documents: 7513, terms: 9704)>>
Non-/sparse entries: 179028/72727124
Sparsity           : 100%
Maximal term length: 70
Weighting          : term frequency (tf)
Sample             :
      Terms
Docs   bank can card credit get limit offer point use will
  1122   31  20   58     50   8     6     3     1   3   13
  1124   31  20   58     50   8     6     3     1   3   13
  553     1   1   20      7   7     4     2     2   5    0
  566    14   6    6      0   3     0     0     0   1    5
  570    13   6   47     10   7     5     3     4   7    3
  574     0   1   12      7   2     4     1     0   5    5
  599    20   8    9      5   1     0     0     0   1    4
  619     2   1   18      7   8     0     5     8   0    1
  669     5   3    0      0   5     2    13     1   0    2
  676     1   1    3      0   3     0     0     6   0    0
```

[842]:
```
# find out tf-idf
tfidf <- weightTfIdf(dtm)
tfidf_good <- weightTfIdf(dtm_good)
tfidf_bad <- weightTfIdf(dtm_bad)

inspect(tfidf)
```

```
Warning message in weightTfIdf(dtm):
"empty document(s): 241 1503 3073 3160 3916 4113 4244 4762 4786 5290 7104"

<<DocumentTermMatrix (documents: 7513, terms: 9704)>>
Non-/sparse entries: 179028/72727124
Sparsity           : 100%
Maximal term length: 70
```

```
Weighting           : term frequency - inverse document frequency (normalized)
(tf-idf)
Sample              :
      Terms
Docs   bank can card credit get limit point use will yes
  201     0   0    0      0   0     0     0   0    0   0
  2365    0   0    0      0   0     0     0   0    0   0
  2379    0   0    0      0   0     0     0   0    0   0
  3019    0   0    0      0   0     0     0   0    0   0
  4590    0   0    0      0   0     0     0   0    0   0
  4727    0   0    0      0   0     0     0   0    0   0
  5448    0   0    0      0   0     0     0   0    0   0
  61      0   0    0      0   0     0     0   0    0   0
  6904    0   0    0      0   0     0     0   0    0   0
  986     0   0    0      0   0     0     0   0    0   0
```

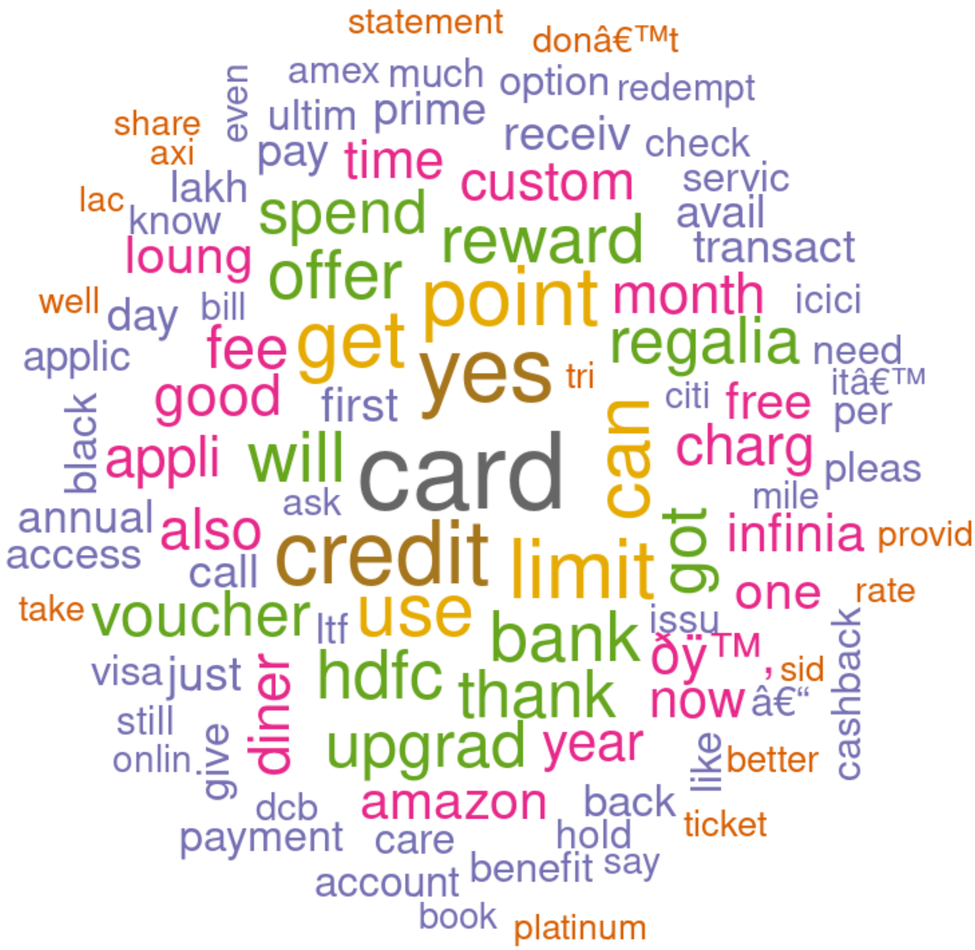
1.7.5 Generate the Word Cloud

The importance of words can be illustrated as a word cloud as follow:

[843]:
```
install.packages("wordcloud")
```

```
Installing wordcloud [2.6] …
        OK [linked cache]
```

[844]:
```
library(wordcloud)
# convert tfidf to a matrix
# for all reviews with rating of 0-5 or Unknown.
m <- as.matrix(tfidf)

# find out the importance of each term by summing up the tf-idf scores over the␣
↪corpus
v <- sort(colSums(m),decreasing=TRUE)
wordcloud(names(v), v, random.order=FALSE, max.words=100, colors=brewer.pal(8,␣
↪"Dark2"))

# brewer.pal(n, name) is used for setting the color palettes
# n: Number of different colors in the palette, minimum 3, maximum depending on␣
↪palette
# name: A palette name
```



[845]:

```r
# Only for the reviews with rating >= 4
m_good <- as.matrix(tfidf_good)

# find out the importance of each term by summing up the tf-idf scores over the corpus
v <- sort(colSums(m_good),decreasing=TRUE)
wordcloud(names(v), v, random.order=FALSE, max.words=100, colors=brewer.pal(8, "Dark2"))

# brewer.pal(n, name) is used for setting the color palettes
# n: Number of different colors in the palette, minimum 3, maximum depending on palette
# name: A palette name
```

```
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"payment could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"increas could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"respons could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"transact could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"now could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"money could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"mani could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"care could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"day could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"happi could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"far could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"account could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"everi could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"give could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"loung could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"book could not be fit on page. It will not be plotted."
```

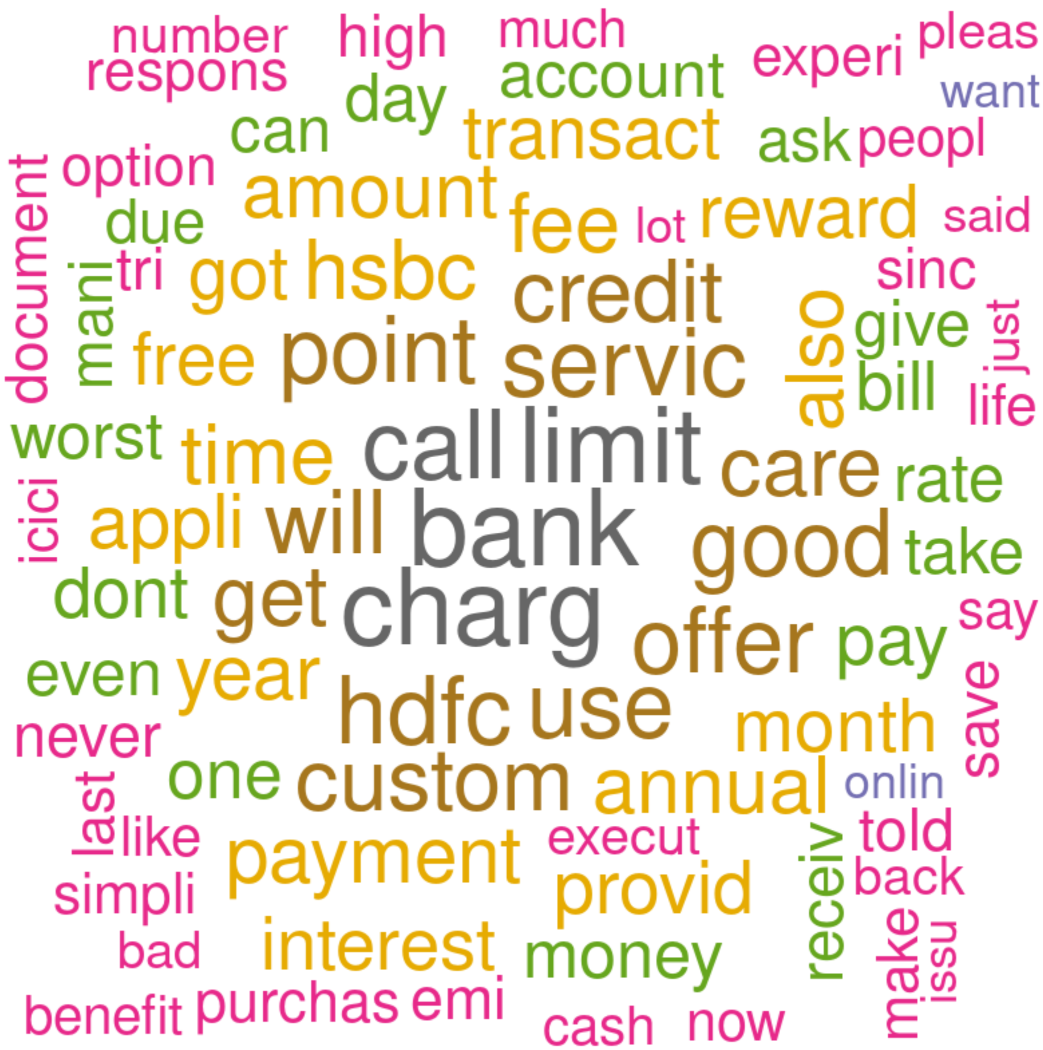
```
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"simpli could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"document could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"henc could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"credit could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"waiv could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"realli could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"airport could not be fit on page. It will not be plotted."
```

[846]:
```
# Only for the reviews with rating < 4

m_bad <- as.matrix(tfidf_bad)

# find out the importance of each term by summing up the tf-idf scores over the
  ↪corpus
v <- sort(colSums(m_bad),decreasing=TRUE)
wordcloud(names(v), v, random.order=FALSE, max.words=100, colors=brewer.pal(8,
  ↪"Dark2"))

# brewer.pal(n, name) is used for setting the color palettes
# n: Number of different colors in the palette, minimum 3, maximum depending on
  ↪palette
```

```
# name: A palette name
```

```
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"inform could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"close could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"accept could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"need could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"cancel could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"know could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"request could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"india could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"satisfi could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"statement could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"proper could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"without could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"problem could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"contact could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"extra could not be fit on page. It will not be plotted."
```

```
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"given could not be fit on page. It will not be plotted."
Warning message in wordcloud(names(v), v, random.order = FALSE, max.words = 100,
:
"though could not be fit on page. It will not be plotted."
```



1.7.6 Insights from Text Mining

We were interested in some qualitative aspects of credit card satisfaction and what customers care about, Using TF (Term Frequency) and IDF (Inverse Document Frequency), we were able to associate words more commonly used in positive (ratings 4.0-5.0) versus negative ratings (0.0-3.5). Word clouds were generated for each. - We can see that satisfied credit card reviews frequently used the words "offer" and "time", suggesting that promotional offers and timely customer service contributed to positive customer experience. - On the other hand, negative reviews often contained the words "charge", "limit" and "call", potentially suggesting that they were unsatisfied with annual charges, the available credit limit on their card, and perhaps had to call customer service in order to resolve issues.

1.8 Conclusion

- The XGBoost classification model gave great predictive accuracy, as well as good detection of attrition and ruling out non-attrition cases. For customers that are predicted to churn, an alert could be generated to flag them as potential attrition and additional attention and service can be targeted toward them.
- From our identified clusters, we can also create different segmentation (High, Medium, and Low attention needed) and tailor customer service to their specific needs. Some potential solutions are:
  - Cluster 1: Contact customers individually to see if they have complaints, better service
  - Cluster 2: Offer incentives
  - Cluster 3: Avoid default risk, help customer manage their debts e.g. negotiate new terms
- Some service improvement suggestions are to offer special promotions, save customer time through better call service, revising the annual fee structure, increasing credit limits, and establishing a customer loyalty campaign.

Further improvements can be made to our analyses moving forward, including tuning hyperparameters, handling imbalanced data, and trying deep-learning models.

1.9 References

- Goyal, Sakshi. "Credit Card Customer." Kaggle, https://www.kaggle.com/datasets/sakshigoyal7/credit-card-customers
- Chidarala, Nagarjuna. "Credit Card Reviews." Kaggle, https://www.kaggle.com/datasets/arjunanc/credit-card-reviews