Credit Card Customer Churn

Esther Waweru - 171444

2024-07-27

Introduction

In the highly competitive financial services industry, customer retention is a critical component for sustaining profitability and growth. Credit card issuers, in particular, face significant challenges in maintaining their customer base due to the diverse range of products and services available to consumers. Customer churn—the phenomenon where customers cease their relationship with a company—represents a substantial cost in terms of lost revenue and increased marketing expenditures to acquire new customers. As such, predicting and mitigating customer churn is of paramount importance for credit card companies aiming to optimize their retention strategies.

This report focuses on the development and implementation of different predictive models for identifying credit card customers who are at risk of churning. Leveraging a comprehensive dataset of customer profiles and transaction histories, the objective is to harness advanced analytical techniques to understand the underlying factors driving churn,to build robust prediction models and compares their effectiveness based on performance metrics. By doing so, credit card companies can proactively address customer dissatisfaction and implement targeted interventions to enhance customer loyalty.

Methods

- **Data Source**

This report depends on the dataset of credit card customer churn for banks; the dataset was collected from Kaggle . Customers can select from four types of credit cards: blue, silver, gold, or platinum. When customers decide to leave the bank, they are recorded as churn customers. Churn customers contribute to a decrease in the bank's profits. As a result, there is growing interest among banking professionals in developing an early-warning system to classify customers as either churn or non-churn. The dataset contains 21 variables: 1 dependentvariable and 20 independent variables. The total number of customers is 10,127, with 1627 churn customers. This shows that our dataset is highly imblanced since 16% of our datset consisting of customers who have exited the bank.

- **Variables of the study**

The dependent variable was Attrition_Flag which consist of the customer status, i.e. whether existing or exited. The rest of the continuous and categorical variables are independent variables and are : CLIENT_NUM (customer account number.), age, gender, number of dependents, education level, marital status, income category, card category (blue, silver, gold, platinum), period of relationship with bank, total number of products held by the customer, number of months inactive of the last 12 months, the number of interactions between the bank and the customer in the last 12 months, credit limit on the credit card, total revolving balance on the credit card, open to buy credit line which is the difference between the credit limit set for the cardholder's account and the current balance(average of last 12 months), change in transaction amount (Q4 over Q1), total transaction amount (last 12 months), total transaction count(last 12 months), change in transaction count (Q4 over Q1), and average card utilization ratio.

- **Data Analysis**

The dataset was complete with no missing values initially. The CLIENT_NUM variable was excluded from the analysis as it did not significantly contribute to the study. For the categorical variables Education_Level, Marital_Status, and Income_Category, the 'Unknown' level was treated as missing values. Since these 'Unknown' entries constituted about 33% of the dataset, they were removed. After this cleaning process, the dataset comprised 7,081 rows and 20 variables.

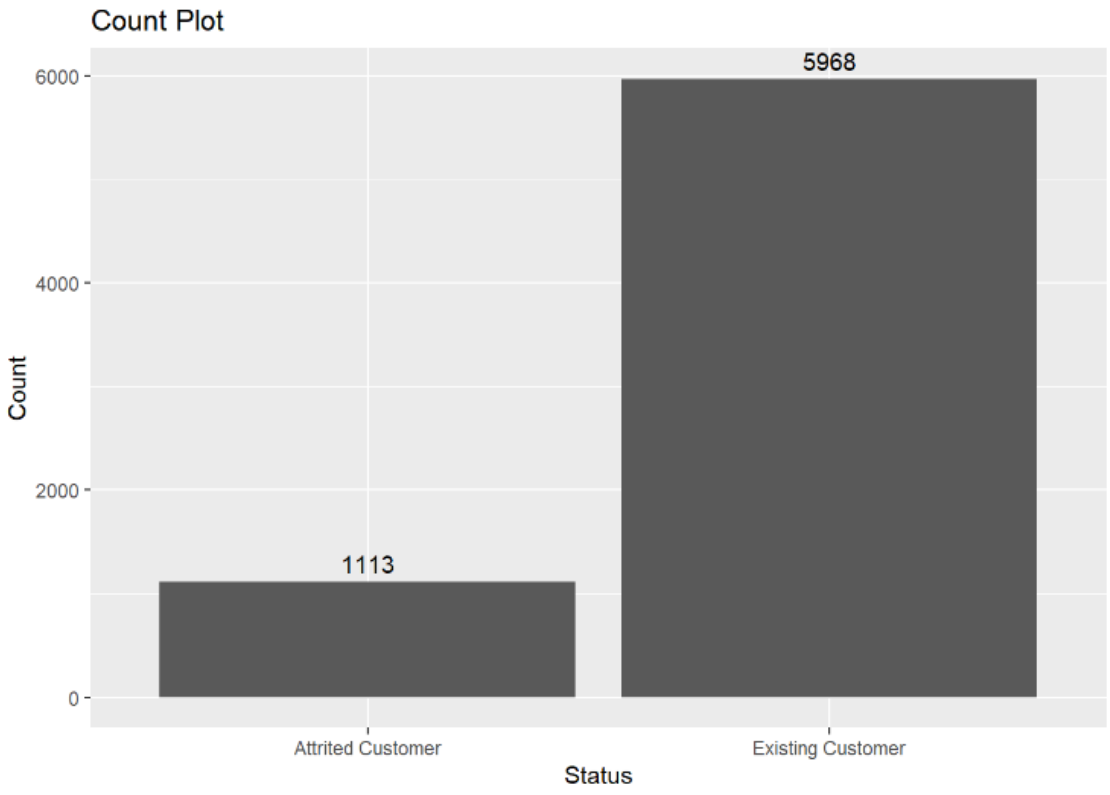


Figure 1: Count Plot

Our dataset is highly imbalanced with 1113 being customers who exited the bank while 5968 being customers who are still with the bank.

Some of the insights between the target variable and categorical variables are :

- There is no much difference in attrition between the genders but females have a slightly higher attrition rate compared to males.
- Most of the customers that exited are Graduates.
- Most of the customers that attrited are either Married or Single.
- Most of our customers that attrited earn less than $40K.
- Most of our customers fall within the Blue Card Category.
- Most attrited customers are those who interacted the most with the bank which may indicate the Bank was not able to solve problems raised by the customers leading to their exit.
- Customers exit increases between month 2 and 4 of inactivity

Some insights from our numerical variables are :

- Customer's age ranges from 26 years to 73 years with a mean and median of 46 years and is normally distributed.
- The mean of the number of dependents of the customers is 3 with maximum being 5.
- At least the customers have been with the bank for more than a year and the median is 3 years.
- At least 50% of the customers use 4 Bank products or less.
- Out of the 12 months, on average the customers have been inactive for 2 months.
- On average, the customers reached out to the bank twice within the 12 months.
- The range between the credit limit is very big from 1438 to 34516.
- Average revolving balance is 1163.
- Average amount that goes unused by the customer is 7469.
- Some customers made total transactions as high as 18484 while others as low as 510.
- Transactions done by customers range from as low as 10 and as high as 139.
- On average customers used approximately 27% of the available credit amount of their card, with 75% of the customers utilizing around 50% or less of their available credit amount. Its distribution is skewed to the right which is not a positive sign for the bank as most of the customers are not utilizing their credit amount.

On Correlation analysis :

- There's a strong positive correlation between Months_on_book and Customer_Age, Total_Revolving_Bal and Avg_Utilization_Ratio, Total_Trans_Amt and Total_Trans_Ct.
- There's a negative correlation of Total_Relationship_count with Total_Trans_Amt and Total_Trans_Ct, Avg_Utilization_Ratio with Credit_Limit and Avg_Open_To_Buy.

Based on the Line Plots : For credit Limit, the limit keeps increasing for males upto around 50 years when it starts to decrease but for females the credit limit is almost constant even as age increases.Despite females having a lower credit limit than males, the total transaction amount trend is almost similar.For utilization ratio, females utilized their funds more compared to males whose utilization rate was low between 30 and 60 years but peaked at 60 years.

Results

- **Building the model**

Given our dataset's 20 variables, we performed dimensionality reduction using Factor Analysis of Mixed Data (FAMD). FAMD is a principal component method designed to analyze datasets containing both quantitative and qualitative variables. This approach allows for the examination of similarities between individuals, considering the mixed types of variables, and also facilitates the exploration of associations among all variables.

The FAMD results indicated that only the first six dimensions have eigenvalues greater than 1, meaning they account for more variance than each of the original variables. These six dimensions collectively explain 40.22% of the total variance in the dataset.

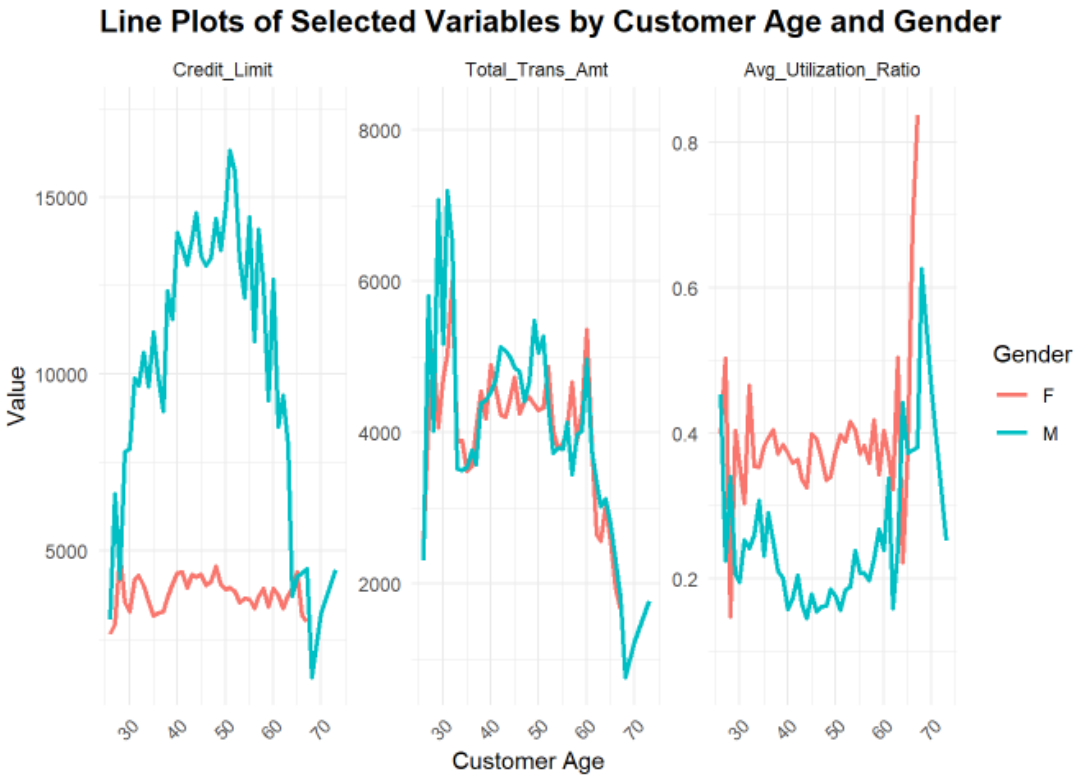


Figure 2: Line Plot

Subsequently, we conducted feature selection using Recursive Feature Elimination (RFE) on the FAMD-transformed data. This process identified nine important dimensions: Dim.4, Dim.2, Dim.22, Dim.5, Dim.1, Dim.25, Dim.23, Dim.3, and Dim.18.

We then used three different datasets to build our prediction models.

Data.Set	Variables
Dataset 1	Contains only the first 6 dimensions from performing FAMD
Dataset 2	Contains the selected features from Recursive Feature Elimination
Dataset 3	Contains all the variables without performing FAMD

- **Machine Learning Models**

The datasets were split into training and testing sets, with 80% allocated for training and 20% for testing. The performance of the customer churn prediction models was evaluated using classification metrics such as precision, accuracy, sensitivity, and specificity.

1. **Random Forest**

Data.Set	Accuracy	Specificity	Sensitivity	Precision	Balanced.Accuracy
Dataset 1	89.61	97.07	49.55	75.86	73.31
Dataset 2	90.95	97.57	55.41	80.92	76.48

Data.Set	Accuracy	Specificity	Sensitivity	Precision	Balanced.Accuracy
Dataset 3	95.48	98.67	78.38	91.58	88.52

Based on all the performance metrics, Dataset 3 is the best model. It consistently has the highest values across all key metrics, indicating strong overall performance in both identifying attrited customers and minimizing false positives.

2. **Logistic Regression**

Data.Set	Accuracy	Specificity	Sensitivity	Precision	Balanced.Accuracy
Dataset 1	88.69	98.07	38.29	78.70	68.18
Dataset 2	89.12	96.65	48.65	72.97	72.65
Dataset 3	90.11	95.81	59.56	72.53	77.63

Dataset 3 is still the best model overall. It has the highest accuracy, sensitivity, and balanced accuracy, indicating strong overall performance in both identifying attrited customers and minimizing false positives. While Dataset 1 has the highest specificity and precision, these are not enough to outweigh the overall better performance metrics of Dataset 3.

3. **Support Vector Machine**

Data.Set	Accuracy	Specificity	Sensitivity	Precision	Balanced.Accuracy
Dataset 1	89.61	97.74	45.95	79.07	71.84
Dataset 2	90.18	97.49	50.90	79.02	74.19
Dataset 3	86.64	99.83	15.77	94.60	57.80

Based on the performance metrics for the SVM models, Dataset 2 is the best model overall. It has the highest accuracy, sensitivity, and balanced accuracy, indicating strong overall performance in both identifying attrited customers and minimizing false positives. While Dataset 3 has the highest specificity and precision, the very low sensitivity and balanced accuracy indicate poor performance in identifying attrited customers. Therefore, Dataset 2 is the most balanced and effective model.

4. **Naives Bayes**

Data.Set	Accuracy	Specificity	Sensitivity	Precision	Balanced.Accuracy
Dataset 1	88.27	98.58	32.88	81.11	65.73
Dataset 2	89.33	98.32	40.99	81.98	69.66
Dataset 3	88.62	92.96	65.32	63.32	79.14

Based on the performance metrics for the Naive Bayes models, Dataset 3 is the best model overall. It has the highest sensitivity and balanced accuracy, indicating strong performance in identifying attrited customers and balanced performance across both classes. While Dataset 2 has the highest accuracy and precision, Dataset 3's better balance in sensitivity and overall balanced accuracy makes it the most effective model for this context.

5. **Decision Tree**

Data.Set	Accuracy	Specificity	Sensitivity	Precision	Balanced.Accuracy
Dataset 1	88.13	96.31	44.14	69.01	70.23
Dataset 2	88.76	94.89	55.86	67.03	75.37
Dataset 3	93.29	96.56	75.68	80.38	86.12

Based on the performance metrics for the Decision Tree models, Dataset 3 is the best model overall. It has the highest values in all key metrics (Accuracy, Specificity, Sensitivity, Precision, and Balanced Accuracy), indicating strong overall performance in identifying both attrited and existing customers effectively.

Discussion

Models	Accuracy	Specificity	Sensitivity	Precision	Balanced.Accuracy
Random Forest	95.48	98.67	78.38	91.58	88.52
Logistic Regression	90.11	95.81	59.56	72.53	77.63
Support Vector Machine	90.18	97.49	50.90	79.02	74.19
Naives Bayes	88.62	92.96	65.32	68.32	79.14
Decision Tree	93.29	96.56	75.68	80.38	86.12

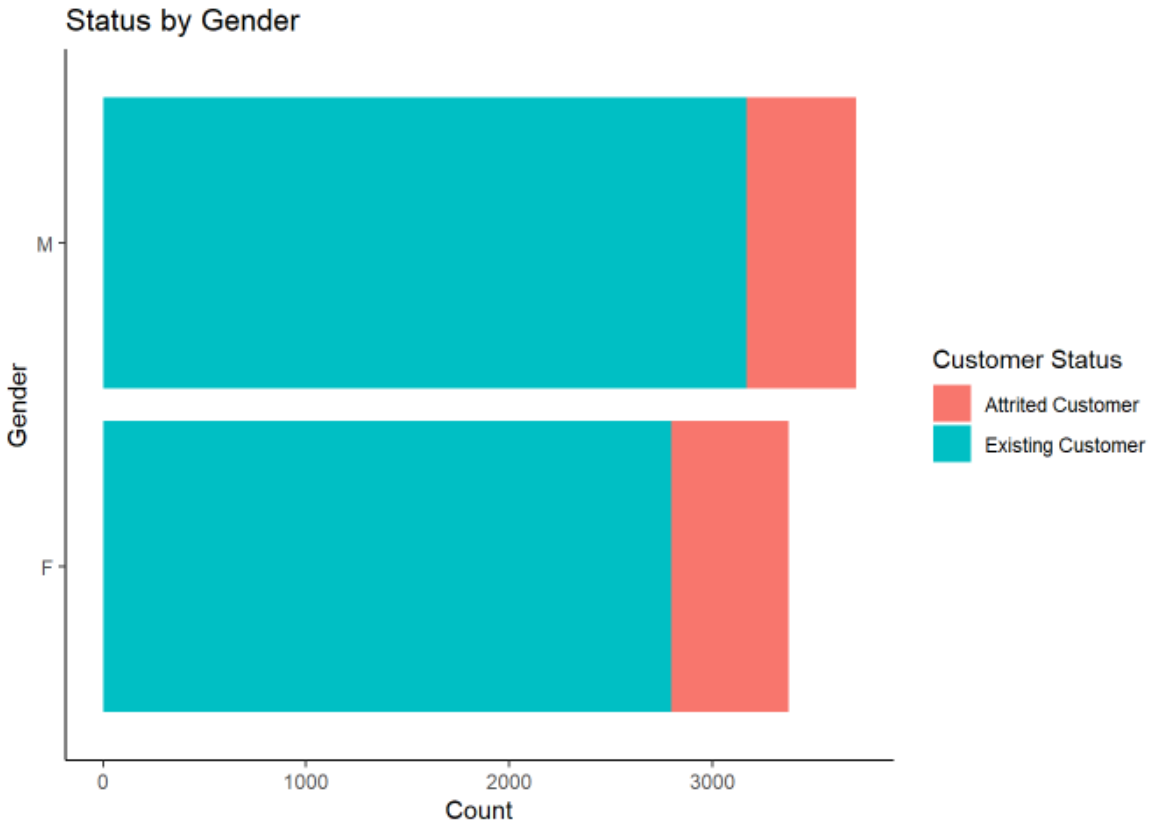
Based on the overall performance metrics, the Random Forest model is the best model. It has the highest values in all key metrics (Accuracy, Specificity, Sensitivity, Precision, and Balanced Accuracy), indicating strong overall performance in identifying both attrited and existing customers effectively. The Random Forest model outperforms all other models in every metric, making it the most reliable and effective model for predicting customer churn.

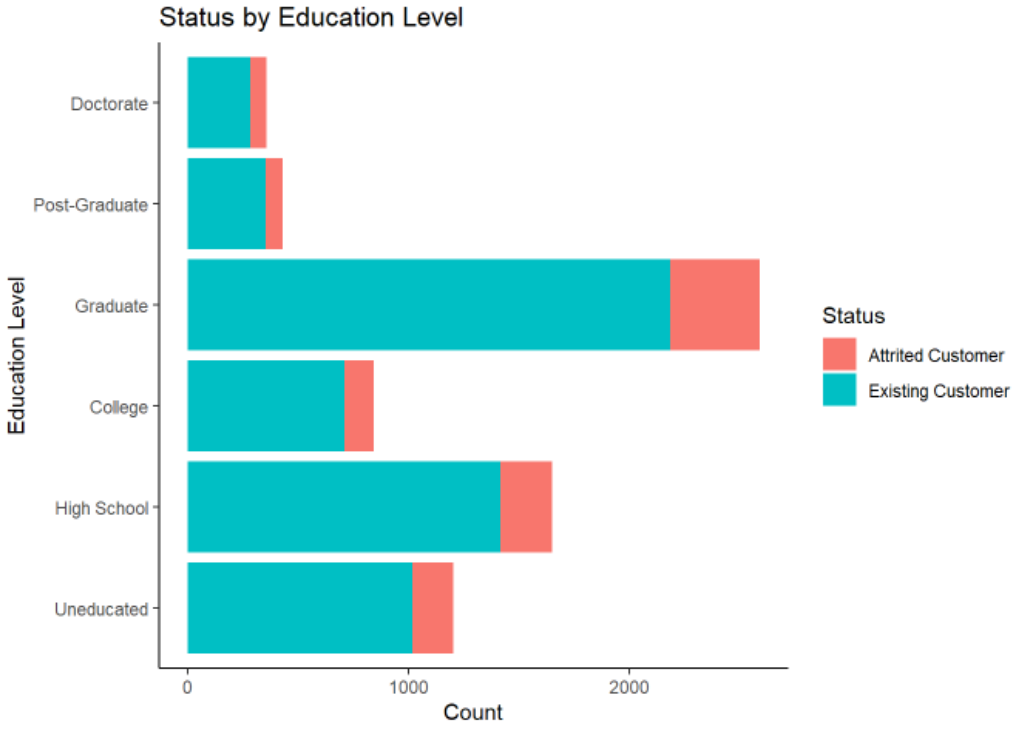
Conclusion

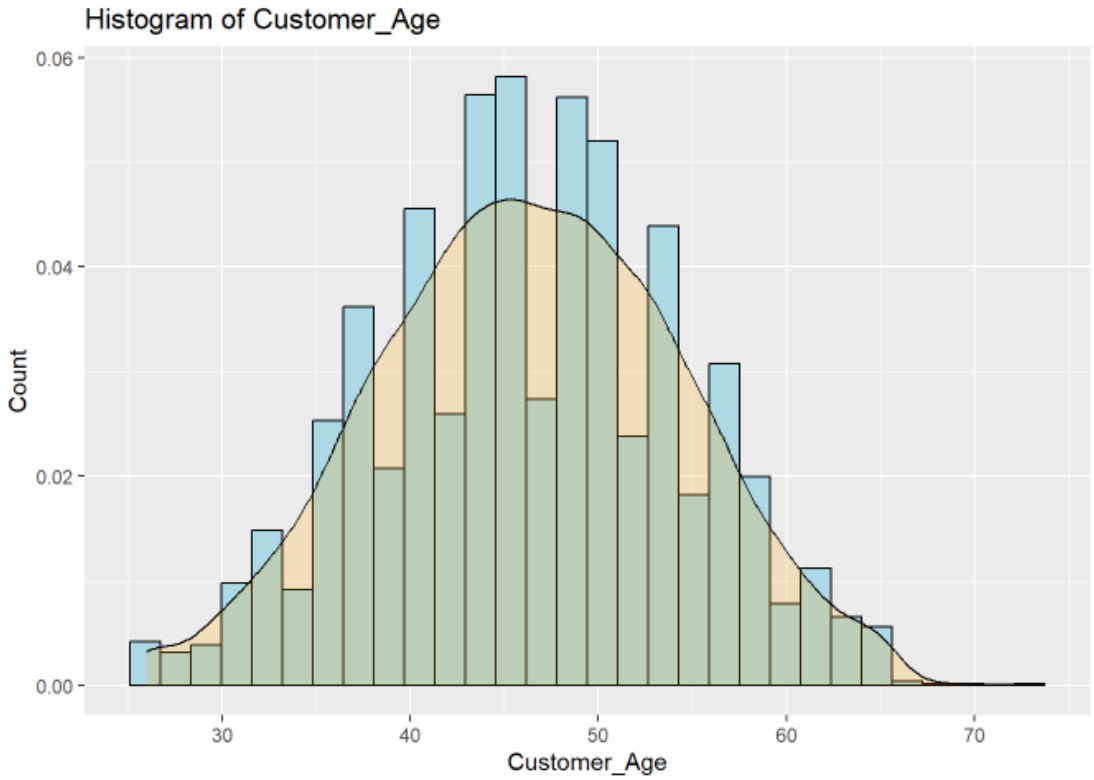
Based on our analysis of multiple machine learning models (Random Forest, Logistic Regression, Support Vector Machine, Naive Bayes, and Decision Tree), we can conclude that the Random Forest model performs the best overall, even considering the imbalance in our data. To address data imbalance, one could perform techniques like SMOTE (Synthetic Minority Over-sampling Technique) or random undersampling to balance the dataset; or Adjust the class weights in the Random Forest model to give more importance to the minority class (churn customers).

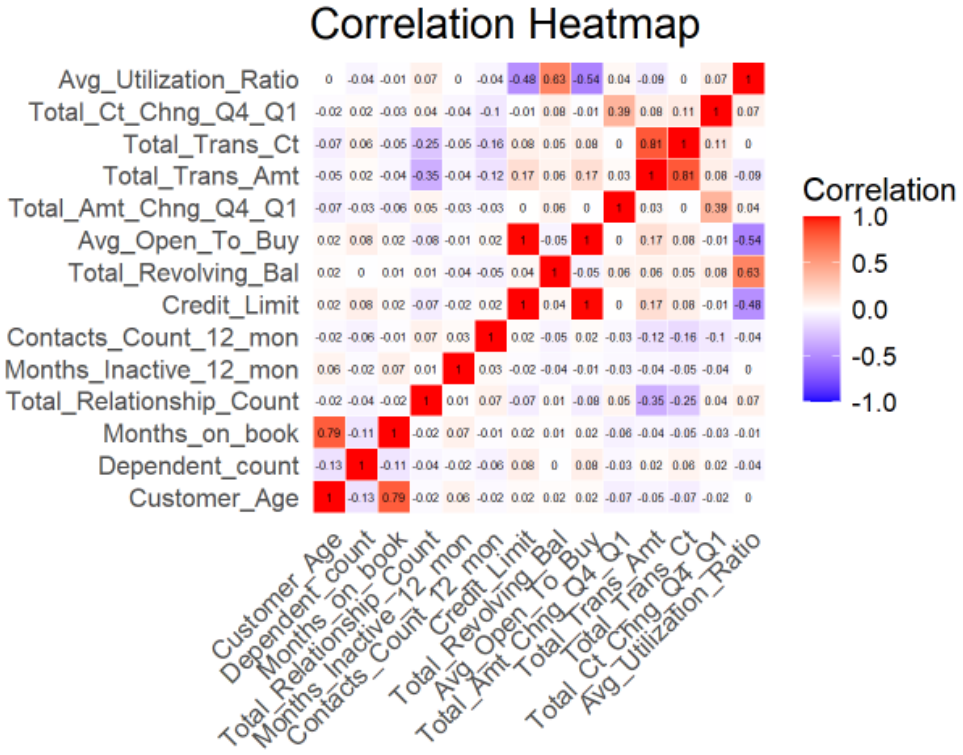
Appendix

Plots


Status by Gender
Gender
M
F
0
1000
2000
3000
Count
Customer Status
Attrited Customer
Existing Customer


Status by Education Level
Education Level
Doctorate
Post-Graduate
Graduate
College
High School
Uneducated
0
1000
2000
Count
Status
Attrited Customer
Existing Customer





Code

```
## ----setup, include=FALSE-------------------------------------------------------
knitr::opts_chunk$set(echo = TRUE)
```

```r
## ------------------------------------------------------------------------
library(tidyverse)
library(dplyr)
library(plyr)
library(corrplot)
library(ggplot2)
library(gridExtra)
library(reshape2)
library(scales)
library(caret)
library(forcats)
library(lattice)
library(MASS)
library(FactoMineR)
library(factoextra)
library(grid)
library(randomForest)
library(partykit)
library(e1071) #contains sum
library(caTools)



## ------------------------------------------------------------------------
data <- read.csv('C://Users//pc//Documents//Masters_Statistical_Sciences//Module 3//Multivariate data an
attach(data)
head(data)


## ------------------------------------------------------------------------
colnames(data)


## ------------------------------------------------------------------------
dim(data)


## ------------------------------------------------------------------------
str(data)


## ------------------------------------------------------------------------
#chack for missing values

colSums(is.na(data))


## ------------------------------------------------------------------------
duplicated_rows <- duplicated(data)
sum(duplicated_rows)
```

```r
## ------------------------------------------------------------------------
#drop unnecessary columns
data <- data[,-1]
head(data)


## ------------------------------------------------------------------------
#Check for unique values
unique(Education_Level)
unique(Marital_Status)
unique(Income_Category)
unique(Card_Category)


## ------------------------------------------------------------------------
data_A <- data[data$Education_Level == 'Unknown',]
unknown_per <- (nrow(data_A)/nrow(data))*100
unknown_per


## ------------------------------------------------------------------------
data_B <- data[data$Marital_Status == 'Unknown',]
unknown_per_b <- (nrow(data_B)/nrow(data))*100
unknown_per_b


## ------------------------------------------------------------------------
data_C <- data[data$Income_Category == 'Unknown',]
unknown_per_c <- (nrow(data_C)/nrow(data))*100
unknown_per_c


## ------------------------------------------------------------------------
data_D <- data[c(data$Education_Level == 'Unknown',data$Marital_Status == 'Unknown',data$Income_Categor
unknown_per_d <- (nrow(data_D)/nrow(data))*100
unknown_per_d



## ------------------------------------------------------------------------
# Drop rows with 'unknown' in any column
data <- data %>%
  filter_all(all_vars(. != 'Unknown'))

nrow(data)


## ------------------------------------------------------------------------
#Convert to factor
data[sapply(data,is.character)] <- lapply(data[sapply(data,is.character)],as.factor)

str(data)
```

```r
## ------------------------------------------------------------------------
ggplot(data, aes(x = Attrition_Flag)) +
  geom_bar() +
  geom_text(stat = "count", aes(label = ..count..), vjust = -0.5) +
  labs(title = "Count Plot", x = "Status", y = "Count")


## ------------------------------------------------------------------------
#Attribution_flag by Gender
ggplot(data , aes(y = Gender)) +
 geom_bar(aes(fill =  Attrition_Flag), position = position_stack(reverse = FALSE)) +
 theme(legend.position = "top") + theme_classic() + xlab("Count") + ylab("Gender") + ggtitle("Status by


## ------------------------------------------------------------------------
# Attrition by Education Level

#Reorder factor levels
data$Education_Level <- factor(data$Education_Level, levels = c('Uneducated','High School','College','G

#Plot
ggplot(data , aes(y = Education_Level)) +
 geom_bar(aes(fill = Attrition_Flag), position = position_stack(reverse = FALSE)) +
 theme(legend.position = "top") + theme_classic() + xlab("Count") + ylab("Education Level") + ggtitle("


## ------------------------------------------------------------------------
#Attrition Flag by Marital Status

ggplot(data , aes(y = Marital_Status)) +
 geom_bar(aes(fill = Attrition_Flag), position = position_stack(reverse = FALSE)) +
 theme(legend.position = "top") + theme_classic() + xlab("Count") + ylab("Marital Status") + ggtitle("S



## ------------------------------------------------------------------------
#Attrition Flag by Income
ggplot(data , aes(y = Income_Category)) +
 geom_bar(aes(fill = Attrition_Flag), position = position_stack(reverse = FALSE)) +
 theme(legend.position = "top") + theme_classic() + xlab("Count") + ylab("Income Category") + ggtitle("


## ------------------------------------------------------------------------
#Attrition by Card Category
ggplot(data , aes(y = Card_Category)) +
 geom_bar(aes(fill = Attrition_Flag), position = position_stack(reverse = FALSE)) +
 theme(legend.position = "top") + theme_classic() + xlab("Count") + ylab("Card Category") + ggtitle("Sta


## ------------------------------------------------------------------------
#Attrition by Contacts_Count_12_mon
ggplot(data , aes(y = Contacts_Count_12_mon)) +
 geom_bar(aes(fill = Attrition_Flag), position = position_stack(reverse = FALSE)) +
```

```
 theme(legend.position = "top") + theme_classic() + xlab("Count") + ylab("Contacts_Count_12_mon") + ggti


## ------------------------------------------------------------------------
#Attrition by Months_Inactive_12_mon
ggplot(data , aes(y = Months_Inactive_12_mon)) +
 geom_bar(aes(fill = Attrition_Flag), position = position_stack(reverse = FALSE)) +
 theme(legend.position = "top") + theme_classic() + xlab("Count") + ylab("Months_Inactive_12_mon") + ggt


## ------------------------------------------------------------------------
summary(data[-c(1,3,5,6,7,8)])


## ------------------------------------------------------------------------

numeric_cols <- sapply(data, is.numeric)
numeric_data <- data[,numeric_cols]

par(mfrow = c(5,3))

for(col in names(numeric_data)){
    p <- ggplot(data, aes_string(x = col)) +
    geom_histogram(aes(y = ..density..),fill = "lightblue", color = "black") +
      geom_density(alpha = .2 ,fill = 'orange') +
    labs(title = paste("Histogram of", col), x = col, y = "Count")
    print(p)
  }


## ------------------------------------------------------------------------

# Calculate the correlation matrix
correlation_matrix <- cor(numeric_data, use = "complete.obs")

# Convert the correlation matrix to a long format
correlation_long <- melt(correlation_matrix)

# Create the heatmap

ggplot(data = correlation_long, aes(Var1, Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1, 1), space = "Lab",
                       name="Correlation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1,
                                   size = 12, hjust = 1),  axis.text.y = element_text(size = 12),
        plot.title = element_text(size = 20),
        legend.title = element_text(size = 15),
        legend.text = element_text(size = 12),
        plot.margin = unit(c(0.01,0.01,0.01,0.01), "cm"),
        panel.grid.major = element_blank(),
```

```
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.ticks = element_blank()) +
  coord_fixed() +
  geom_text(aes(Var1, Var2, label = round(value, 2)), color = "black", size = 2) +
  labs(title = "Correlation Heatmap", x = "", y = "")


## ------------------------------------------------------------------------
# Filter the necessary columns
selected_cols <- c("Customer_Age", "Gender", "Credit_Limit", "Total_Trans_Amt", "Avg_Utilization_Ratio"
data_selected <- data[, selected_cols]

# Melt the data into a long format
data_long <- reshape2::melt(data_selected, id.vars = c("Customer_Age", "Gender"),
                            measure.vars = c("Credit_Limit", "Total_Trans_Amt", "Avg_Utilization_Ratio"))

# Create the faceted line plot
ggplot(data_long, aes(x = Customer_Age, y = value, color = Gender)) +
  geom_line(stat = "summary", fun.y = mean, size = 1) +
  facet_wrap(~ variable, scales = "free_y") +
  theme_minimal() +
  labs(title = "Line Plots of Selected Variables by Customer Age and Gender",
       x = "Customer Age",
       y = "Value") +
  theme(plot.title = element_text(size = 14, face = "bold"),
        axis.text.x = element_text(angle = 45, hjust = 1))


## ------------------------------------------------------------------------
#numerical_vars <- data %>% dplyr :: select(where(is.numeric)) %>% colnames()

# Scale numerical variables
#data[numerical_vars] <- scale(data[numerical_vars])


## ------------------------------------------------------------------------
# Replace values in the "Attrition_Flag" column
#data <- data %>% mutate(Attrition_Flag = ifelse(Attrition_Flag == "Existing Customer", 0,ifelse(Attrit

#head(data)


## ------------------------------------------------------------------------
#One Hot encoding
# Define the categorical columns to be one-hot encoded
#catcols <- c("Gender", "Marital_Status", "Card_Category")

# Create the dummyVars model
#dum_model <- dummyVars(~ ., data = data[catcols])

# Apply the model to the data to create the one-hot encoded variables
#one_hot_encoded <- predict(dum_model, newdata = data[catcols])
```

```r
# Convert the result to a data frame
#one_hot_encoded_df <- as.data.frame(one_hot_encoded)

# Combine the one-hot encoded variables with the original data
#df_encoded <- cbind(data, one_hot_encoded_df)

# Optionally, remove the original categorical columns
#df_encoded <- df_encoded[, !(names(df_encoded) %in% catcols)]


#Label Encoding

# Label encode the "Education_Level" column
#df_encoded$Education_Level <- as.numeric(df_encoded$Education_Level)

# Label encode the "Income_Category" column
#df_encoded$Income_Category <- as.numeric(df_encoded$Income_Category)

# Print the first few rows of the encoded data frame
#head(df_encoded)


## -----------------------------------------------------------------------------
#str(df_encoded)


## -----------------------------------------------------------------------------

data.famd <- FAMD(data,sup.var = 1,graph = FALSE, ncp = 25)


## -----------------------------------------------------------------------------
#inspect dimensions
eig.val <- get_eigenvalue(data.famd)
eig.val


## -----------------------------------------------------------------------------
#Visualize the eigenvalues/variances.

fviz_screeplot(data.famd)


## -----------------------------------------------------------------------------
var <- get_famd_var(data.famd)
var



## -----------------------------------------------------------------------------
#plot both quantitative and qualitative variables

fviz_famd_var(data.famd, repel = TRUE)
```

```r
## ------------------------------------------------------------------------
# Contribution to the first dimension
fviz_contrib(data.famd, "var", axes = 1)


## ------------------------------------------------------------------------
# Contribution to the second dimension
fviz_contrib(data.famd, "var", axes = 2)


## ------------------------------------------------------------------------
# Contribution to the third dimension
fviz_contrib(data.famd, "var", axes = 3)


## ------------------------------------------------------------------------
#Visualise quantitative
fviz_famd_var(data.famd, "quanti.var", repel = TRUE,
              col.var = "contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"))


## ------------------------------------------------------------------------
#Visualise qualitative
fviz_famd_var(data.famd, "quali.var", col.var = "contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07")
             )


## ------------------------------------------------------------------------
# Extract the FAMD components
famd_components <- data.famd$ind$coord

# Convert to a data frame
famd_components_df <- data.frame(famd_components,Attrition_Flag = data$Attrition_Flag)

# Display the first few rows of the FAMD components
head(famd_components_df)


## ------------------------------------------------------------------------
# Define the target variable
target_variable <- famd_components_df$Attrition_Flag  # Replace 'Attrition_Flag' with the actual target

# Feature selection using Recursive Feature Elimination
control <- rfeControl(functions = rfFuncs, method = "cv", number = 10)

# Perform RFE
set.seed(123)  # For reproducibility
#rfe_result <- rfe(famd_components_df, target_variable, sizes = c(1:10), rfeControl = control)
rfe_result <- rfe(famd_components_df[, -ncol(famd_components_df)],target_variable, sizes = c(1:10), rfe
# Display the results
```

```r
#print(rfe_result)

# Get the selected features
selected_features <- predictors(rfe_result)

# Display the selected features
print(selected_features)


## ----------------------------------------------------------------------------
# Ensure selected_features are in famd_data
missing_features <- setdiff(selected_features, colnames(famd_components_df))
if (length(missing_features) > 0) {
  stop(paste("The following selected features are missing in famd_components_df:", paste(missing_feature
}

selected_features


## ----------------------------------------------------------------------------
# Subset the original data with the selected features
final_data <- famd_components_df[,c(selected_features,"Attrition_Flag")]
head(final_data)
#head(famd_components_df)
#sel



## ----------------------------------------------------------------------------
# Subset the original data with the first six dimensions

dim_6 <- colnames(famd_components_df[1:6])

dim_data <- famd_components_df[,c(dim_6,"Attrition_Flag")]
head(dim_data)


## ----------------------------------------------------------------------------
#Split dim data into training and testing
set.seed(123)
intrain <- createDataPartition(dim_data$Attrition_Flag,p=0.80, list = FALSE)
train_dim<- dim_data[intrain,]
test_dim<- dim_data[-intrain,]
dim(train_dim); dim(test_dim)



## ----------------------------------------------------------------------------
#Split final data into training and testing
set.seed(123)
intrain_dim<- createDataPartition(final_data$Attrition_Flag,p=0.80, list = FALSE)
training_dim<- final_data[intrain_dim,]
testing_dim<- final_data[-intrain_dim,]
```

```r
dim(training_dim); dim(testing_dim)



## ------------------------------------------------------------------------------
#Splitting the regular dataset
set.seed(123)
intrain_reg<- createDataPartition(data$Attrition_Flag, p=0.80, list = FALSE)
training_reg<- data[intrain_reg,]
testing_reg<- data[-intrain_reg,]
dim(training_reg); dim(testing_reg)


## ------------------------------------------------------------------------------
#Randomforest for dim data
rand_forest <- randomForest(Attrition_Flag ~ ., ntree= 500, family="binomial", data=train_dim)
print(summary(rand_forest))
#random_forest
rf_pred <- predict(rand_forest, test_dim)
caret::confusionMatrix(rf_pred, test_dim$Attrition_Flag)


## ------------------------------------------------------------------------------
#Randomforest for final data
random_forest <- randomForest(Attrition_Flag ~ ., ntree= 500, family="binomial", data=training_dim)
print(summary(random_forest))
#random_forest
rf_pred <- predict(random_forest, testing_dim)
caret::confusionMatrix(rf_pred, testing_dim$Attrition_Flag)


## ------------------------------------------------------------------------------
#Randomforest for reg data
random_forest_2 <- randomForest(Attrition_Flag ~ ., ntree= 500, family="binomial", data=training_reg)
print(summary(random_forest_2))
#random_forest_2
rf_pred_2 <- predict(random_forest_2, testing_reg)
caret::confusionMatrix(rf_pred_2, testing_reg$Attrition_Flag)


## ------------------------------------------------------------------------------
#Logistic Regression using dim data
LogModel <- glm(Attrition_Flag ~ ., family= "binomial", data = train_dim)
print(summary(LogModel))

#predict on test data
anova(LogModel, test="Chisq")
predictions <- predict(LogModel, test_dim,  type = "response")

#Convert probabilities to binary predictions
y_pred <- ifelse(predictions > 0.5, 2, 1)
y_pred <- as.numeric(y_pred)
target <- as.numeric(test_dim$Attrition_Flag)
```

```r
#prop.table(table(training_pca$Attrition_Flag))
caret::confusionMatrix(table(y_pred, target))


## ---------------------------------------------------------------------------
#Logistic Regression using famd data
LogModel_2 <- glm(Attrition_Flag ~ ., family= "binomial", data = training_dim)
print(summary(LogModel_2))

#predict on test data
anova(LogModel_2, test="Chisq")
predicts <- predict(LogModel_2, testing_dim,  type = "response")

#Convert probabilities to binary predictions
y_pred_1 <- ifelse(predicts > 0.5, 2, 1)
y_pred_1 <- as.numeric(y_pred_1)
target_1 <- as.numeric(testing_dim$Attrition_Flag)
#prop.table(table(training_pca$Attrition_Flag))
caret::confusionMatrix(table(y_pred_1, target_1))


## ---------------------------------------------------------------------------
#Logistic Regression for Regular Data
LogModel_3 <- glm(Attrition_Flag ~ ., family= "binomial", data = training_reg)
print(summary(LogModel_3))

anova(LogModel_3, test="Chisq")
log_reg_3 <- predict(LogModel_3, testing_reg,  type = "response")

y_pred_3 <- ifelse(log_reg_3 > 0.5, 2, 1)
y_pred_3 <- as.numeric(y_pred_3)
target_3 <- as.numeric(testing_reg$Attrition_Flag)
#prop.table(table(training_pca$Attrition_Flag))
caret::confusionMatrix(table(y_pred_3, target_3))


## ---------------------------------------------------------------------------
#SVM for dim Data
svmfit = svm(Attrition_Flag ~ ., data = train_dim, cross = 10, gamma = 0.5, cost = 1)
svm_pred <- predict(svmfit, test_dim)
summary(svmfit)
caret::confusionMatrix(svm_pred, test_dim$Attrition_Flag)


## ---------------------------------------------------------------------------
#SVM for famd Data
svmfit2 = svm(Attrition_Flag ~ ., data = training_dim, cross = 10, gamma = 0.5, cost = 1)
svm_pred2 <- predict(svmfit2, testing_dim)
summary(svmfit2)
caret::confusionMatrix(svm_pred2, testing_dim$Attrition_Flag)


## ---------------------------------------------------------------------------
```

```r
#SVM for Regular Data
svmfit3 = svm(Attrition_Flag ~ ., data = training_reg, cross = 10, gamma = 0.5, cost = 1)
svm_pred3 <- predict(svmfit3, testing_reg)
summary(svmfit3)
caret::confusionMatrix(svm_pred3, testing_reg$Attrition_Flag)


## ------------------------------------------------------------------------
#Naive Bayes for dim Data
naive_bayes<- naiveBayes(Attrition_Flag ~ ., data= train_dim)
naive_bayes
nb_pred<- predict(naive_bayes, test_dim)
caret::confusionMatrix(nb_pred, test_dim$Attrition_Flag)


## ------------------------------------------------------------------------
#Naive Bayes for PCA Data
naive_bayes2<- naiveBayes(Attrition_Flag ~ ., data= training_dim)
naive_bayes2
nb_pred2<- predict(naive_bayes2, testing_dim)
caret::confusionMatrix(nb_pred2, testing_dim$Attrition_Flag)


## ------------------------------------------------------------------------
#Naive Bayes for Regular Data
naive_bayes3<- naiveBayes(Attrition_Flag ~ ., data= training_reg)
naive_bayes3
nb_pred3<- predict(naive_bayes3, testing_reg)
caret::confusionMatrix(nb_pred3, testing_reg$Attrition_Flag)


## ------------------------------------------------------------------------
#Decision tree for dim data
decision_tree <- ctree(Attrition_Flag ~ ., data= train_dim)
decision_tree
dt_pred<- predict(decision_tree, test_dim)
caret::confusionMatrix(dt_pred, test_dim$Attrition_Flag)


## ------------------------------------------------------------------------
#Decision tree for dim data
decision_tree2 <- ctree(Attrition_Flag ~ ., data= training_dim)
decision_tree2
dt_pred2<- predict(decision_tree2, testing_dim)
caret::confusionMatrix(dt_pred2, testing_dim$Attrition_Flag)


## ------------------------------------------------------------------------
#Decision tree for Regular data
decision_tree3 <- ctree(Attrition_Flag ~ ., data= training_reg)
decision_tree3
dt_pred3<- predict(decision_tree3, testing_reg)
caret::confusionMatrix(dt_pred3, testing_reg$Attrition_Flag)
```