FACULTY OF ENGINEERING, ENVIRONMENT & COMPUTING

(7082CEM)

Coursework

Big Data Analytics and Visualization Using PySpark

MODULE LEADER: Dr. Marwan Fuad

Student Name: Bharti Tayal

SID:10895366

Customer Churn Prediction in Banking Industry

I can confirm that all work submitted is my own: Yes

Introduction

In this present daunting scenario, there is prominent competition in almost every industry. In this paper, I am going to discuss one major problem of every industry which is Customer Churn. Here, I am going to predict Customer Attrition in Banking Industry using Machine Learning Classification techniques.

Customers can be a fickle bunch—happy one day, gone the next. That's why it's important for companies of all shapes and sizes to monitor when customers leave or "churn." While customer churn is often considered to be a measure of failure rather than success, it is one of the most important metrics to track (churn?, 2020). Customer attrition is basically, when any company loses its clients or customer. It can also be referred to as 'Customer Churn', 'Customer Turnover', or 'Customer Defection'. Banks, telephone service companies, Internet service providers, pay TV companies, insurance firms, and alarm monitoring services, often use customer attrition analysis and customer attrition rates as one of their key business metrics (along with cash flow, EBITDA, etc.) because the cost of retaining an existing customer is far less than acquiring a new one. Companies from these sectors often have customer service branches which attempt to win back defecting clients, because recovered long-term customers can be worth much more to a company than newly recruited clients (Customer attrition - Wikipedia, 2021). Companies care about churning because it has been proved that cost of acquiring new customer is 5 times more than to keep the current ones. If customer defection reduces by 5%, companies can increase their profits by 25-125%. That is the reason why industries want to know more about customer churn so that they can know which customers are at the risk of leaving. Consequently, they will put more efforts so that customer will not switch.

Churning in any industry is a major concern as this is highly related to their profits. Companies make lot of wealth from their customers; especially Banking industry. Hence, it regularly keeps check on customer's behaviour by tracking their activities. Many companies have started adapting Machine Learning techniques in order to predict customer churn. This paper presents a model to predict customer churn in Banking industry using Supervised Machine Learning Techniques which are Random Forest Classifier, Decision Trees and Linear Regression Classifier using Pyspark. This model will help in analysing whether the customer will leave or exist in the same bank using some of the features. The analysis would be done on the basis of historical transaction data of customers. The objective of this study is an in-depth Exploratory Data Analysis and visualization of difference between churning and non-churning customers, especially all features, other than income. The initial setup of the dataset will consist of cleaning of the data, part of the pre-processing procedure. Once the data is processed and ready for analysis, various functions and machine learning techniques will be applied in order to gain a better understanding of the data I am working with and any distinctive patterns or trends. Analysing the data will be accomplished by visualising datasets in order to identify any correlation, patterns or trends.

This model will facilitate bank managers in predicting customers who are expected to unsubscribe their banking services(going to churn). After knowing the results, they can approach the client proactively; providing them better services and offers. They will try to revert the customers' decisions and prevent them from switching. The advantage of this dataset is it is quite clean, so I will be able to spend more time on modelling it in comparison to data cleaning. In contrast to this, data is very unbalanced; only 16.07% of customers churned. Thus,

making it little bit cumbersome to train the model in order to predict attrition of customers.

Machine Learning Techniques have been applied using Pyspark. Pyspark is a Python API that supports Apache Spark, a distributed framework made for handling big data analysis. It's an amazing framework to use when you are working with huge datasets, and it's becoming a must-have skill for any data scientist (How to Use Pyspark For Your Machine Learning Project, 2020). I will use documentation from Pyspark's website as reference because the Pyspark code is relevantly new to me. While using Spark, most data engineers recommends to develop either in Scala (which is the "native" Spark language) or in Python through complete PySpark API. Python for Spark is obviously slower than Scala. However like many developers, I love Python because it's flexible, robust, easy to learn, and benefits from all my favorites libraries. In my opinion, Python is the perfect language for prototyping in Big Data/Machine Learning fields (Bochet, 2021). Pyspark can be paralytically deployed with other programs, one being Jupyter. Jupyter Notebook is a popular application that enables you to edit, run and share Python code into a web view. It allows you to modify and re-execute parts of your code in a very flexible way. That's why Jupyter is a great tool to test and prototype programs. Jupyter is an interactive web-based environment used for the development of live code, visualisation and modelling (Project Jupyter, 2021). This open source application has sub frameworks such as Jupyter Notebook, Jupyter Lab and Jupyter Hub to help provide coherent development along with the magnitude of plugins and modules.

I have also used Tableau software for Data Visualisation. Tableau tool software is very user-friendly and straightforward to use, which allows for the production of interactive data visualization in a very effective manner at fast speed. Complicated charts and graphs can be easily created in Tableau Software. Datasets are calculated at a quicker rate, and massive data are handled very efficiently (Sharma, 2019).

Installation of required programs:

Machine used to perform this study has the Mac HD with the capacity of 512 GB, RAM with 8 GB and the used operating system is MacOS Big Sur Version 11.4 (20F71). The installation steps followed are explained below:

Step 1: Initially, it is important to check whether java is installed within the machine through looking at the current java version. If the java version is installed the following command is given:

```
java –version
```

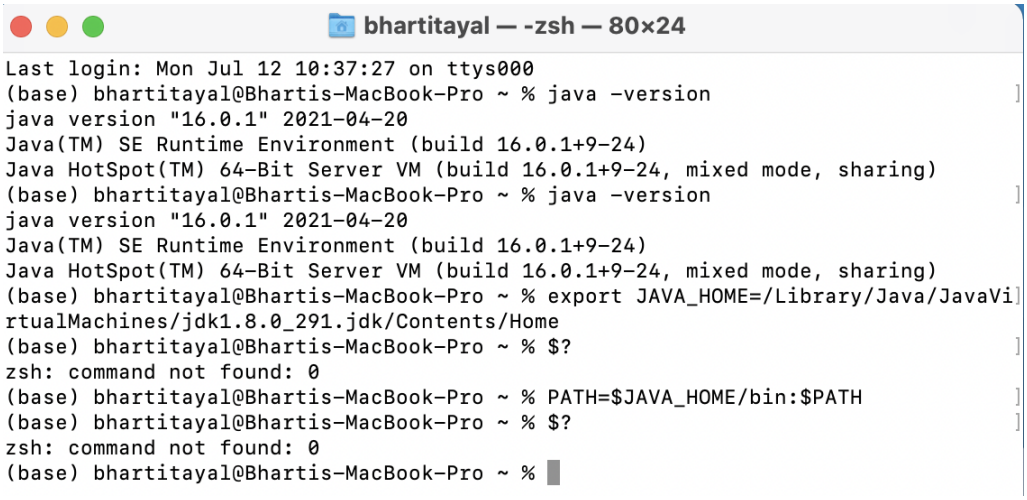


Fig. 1. Checking Java Version

And if the java is not installed the following command needs to be used:

sudo apt install default-jdk

Step 2:- Move the downloaded unzip spark file into Home folder. The command for this is given below:

tar -xzf spark-2.3.0-bin-hadoop2.7.tgz

Step 3:- Setting up the Spark Environment. The command for this shown below:

export SPARK_HOME=Users/bhartitayal/Desktop/spark-2.3.0-bin-hadoop2.7

PATH=$SPARK_HOME/bin:$PATH

Step 4:- Setting up the java environment. The command for this is displayed below:

export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0_291.jdk/Contents/Home
PATH=$JAVA_HOME/bin:$PATH

```
(base) bhartitayal@Bhartis-MacBook-Pro ~ % export SPARK_HOME=/Users/bhartitayal/Desktop/spark-2.3.0-bin-hadoop2.7
(base) bhartitayal@Bhartis-MacBook-Pro ~ % PATH=$SPARK_HOME/bin:$PATH
(base) bhartitayal@Bhartis-MacBook-Pro ~ % spark-shell
2021-07-12 11:07:49 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://172.16.1.204:4040
Spark context available as 'sc' (master = local[*], app id = local-1626084485036).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.3.0
      /_/

Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_291)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Fig. 2. Installing Spark

Step 6:- After Spark installation, to install PySpark it is important to check whether Python is installed in the machine. The following command will be used to check the Python version.

python3 --version

Step7:- Once Python is installed Jupyter notebook can be installed with the following command. User interface within Jupyter notebook allows to easily write the Python programming.

pip3 install jupyter

Step 8:-To make sure whether Jupyter notebook is successfully installed the following command is used.

jupyter notebook

```
Last login: Mon Jul 12 10:38:49 on ttys000
(base) bhartitayal@Bhartis-MacBook-Pro ~ % jupyter notebook
[I 2021-07-12 11:11:30.618 LabApp] JupyterLab extension loaded from /opt/anaconda3/lib/python3.8/site-packages/jupyterlab
[I 2021-07-12 11:11:30.619 LabApp] JupyterLab application directory is /opt/anaconda3/share/jupyter/lab
[I 11:11:30.621 NotebookApp] Serving notebooks from local directory: /Users/bhartitayal
[I 11:11:30.622 NotebookApp] Jupyter Notebook 6.3.0 is running at:
[I 11:11:30.622 NotebookApp] http://localhost:8888/?token=49a508e9cfa0f91cca6bb29bc4c87750072a896d46b7eb4d
[I 11:11:30.622 NotebookApp]  or http://127.0.0.1:8888/?token=49a508e9cfa0f91cca6bb29bc4c87750072a896d46b7eb4d
[I 11:11:30.622 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 11:11:30.625 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///Users/bhartitayal/Library/Jupyter/runtime/nbserver-1779-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=49a508e9cfa0f91cca6bb29bc4c87750072a896d46b7eb4d
     or http://127.0.0.1:8888/?token=49a508e9cfa0f91cca6bb29bc4c87750072a896d46b7eb4d
```

Fig. 3. Launching Jupyter Notebook

Step-9: After launching Jupyter notebook, Pyspark can be installed using command:

!pip install pyspark

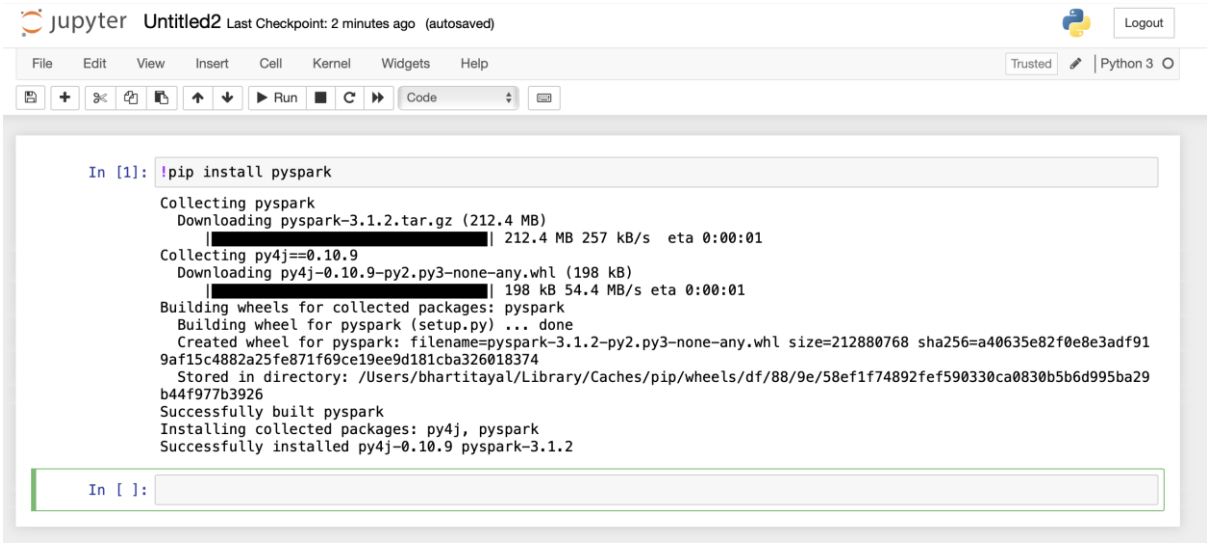


Fig. 4. Installing Pyspark

Importing necessary Libraries and Data:

Firstly, all the necessary libraries which are required for analysis have been imported.

```
from pyspark.sql import SparkSession
from pyspark.ml  import Pipeline
from pyspark.sql import SQLContext
from pyspark.ml.feature import StringIndexer, VectorAssembler
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml.feature import StandardScaler
from sklearn.metrics import roc_curve,auc
from pyspark.mllib.evaluation import BinaryClassificationMetrics as metric
from pyspark import SparkContext
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.classification import LogisticRegression

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
import plotly.express as ex
```

Fig. 5. Importies libraries

Spark Session has been created. Then, Dataset was loaded.

```
spark = SparkSession.builder.master("local[*]").getOrCreate()
dataset=spark.read.csv("BankChurners.csv",inferSchema=True, header =True)
dataset.show(5)
```

Fig. 6. Creating Spark Session

Dataset Description:

The Dataset used in this model has been acquired from an online source Kaggle. This dataset contains information about 10,127 customers with 23 features for each customer. Various features such as age, gender, income, etc have been considered to predict customer churn. The variable Attrition_Flag in the dataset tells the current status of the client whether customer had switched their bank or not. It is an unbalanced distribution (about 84% of the data belong to the class Existing Customer, whereas only 16% of clients are in the class of churned ones).

Here, printSchema( ) command has been used to print out the schema of dataset in the tree format. It is always better to work with data if you know the columns present and their data types.

```
print('Data overview')
dataset.printSchema()
print('Columns overview')
pd.DataFrame(dataset.dtypes, columns = ['Column Name','Data type'])
```

0	CLIENTNUM	int
1	Attrition_Flag	string
2	Customer_Age	int
3	Gender	string
4	Dependent_count	int
5	Education_Level	string
6	Marital_Status	string
7	Income_Category	string
8	Card_Category	string
9	Months_on_book	int
10	Total_Relationship_Count	int
11	Months_Inactive_12_mon	int
12	Contacts_Count_12_mon	int
13	Credit_Limit	double
14	Total_Revolving_Bal	int
15	Avg_Open_To_Buy	double
16	Total_Amt_Chng_Q4_Q1	double
17	Total_Trans_Amt	int
18	Total_Trans_Ct	int
19	Total_Ct_Chng_Q4_Q1	double
20	Avg_Utilization_Ratio	double
21	Naive_Bayes_Classifier_Attrition_Flag_Card_Cat...	double
22	Naive_Bayes_Classifier_Attrition_Flag_Card_Cat...	double

Fig. 7. Columns Overview of Dataset

In the dataset we can find three main classes of features:

Anagraphical Features: Customer_Age, Gender, Education_Level, Marital_Status, Income_Category. Their meaning is straightforward;

Customer-bank relationship features:

- Dependent_count: It gives you a number how many people are dependent on a credit card user for financial support. A higher count tells us that the expenditures can be high.
- Card_Category: It tells if the account is Basic or Premium depending upon Card types which are Blue, Gold, Silver.
- Months_on_book: It tells the number of months of the relationship till now;
- Total_Relationship_Count: Total number of products held by the customer. In other words, client could have other products like debit card, loans, and so on;
- Contacts_Count_12_mon: It tells you how many times contacts between the customer and the bank were there from last 12 months. It could be a key indicator of the satisfaction level of the client.

Credit Card utilization features:

- Months_Inactive: It determines the number of months in which the client was inactive.
- Credit_Limit: This tells you the maximum amount which client is allowed to use.
- Total_Revolving_Bal: The debt amount.
- Avg_Open_To_Buy: suppose a client has used 500£, and its credit limit is 2500£. The custormer is thus open to buy 2500-500=2000£. Avg_Open_To_Buy is the average over the last 12 months of the Open To Buy value;
- Total_Trans_Amt: It gives you the total transactional amount of the last 12 months;
- Total_Amt_Chng_Q4_Q1: It is the ratio of transactional amount of first quarter and the same amount for fourth quarter. Hence, a value smaller than 1 means that the customer has spent less in this quarter with respect to the last one;
- Total_Trans_Ct, Total_Ct_Chng_Q4_Q1: Their meaning is analogous to the last two variables. Of course, these differ on the underlying reference variable, since in this case it is the number of transactions instead of the amount;
- Avg_Utilization_Ratio: Avg_Utilization_Ratio is the average proportion of the credit used with respect to the credit limit in the last 12 months.

There are two more features:

- CLIENTNUM**:** It is the primary key of the dataset. It is not useful for analysis.
- Attrition_Flag**:** It tells whether the customer is an attrited one or not.

Data Preprocessing:

As this dataset has been derived from an online source, there can be lot of inconsistencies in this dataset which do not make it suitable for data visualisation and analysis. Hence, dataset should be preprocessed to perform Exploratory Data Analysis and modelling it, in order to get, unbiased and meaningful conclusions from the analysis.

Data preprocessing is a crucial phase in the process of data mining. Data-gathering methods are often loosely controlled, resulting in out-of-range values (e.g., Income: −100), impossible data combinations (e.g., Sex: Male, Pregnant: Yes), and missing values, etc. Analyzing data that has not been carefully screened for such problems can produce misleading results. Thus, the representation and quality of data is first and foremost before running any analysis. If there is much irrelevant and redundant information present or noisy and unreliable data, then knowledge discovery during the training phase is more difficult. Data preparation and filtering steps can take considerable amount of processing time. Data preprocessing includes cleaning, Instance Selection, Normalization, Transformation, Feature Extraction and Selection, etc. The product of data preprocessing is the final training set (Data pre-processing - Wikipedia, 2020).

Firstly, I have removed last 2 columns from the dataset as it was already specified in dataset description on website that they are useless for analysis. So, I removed them from my dataset using drop command.

```
#in data description, it says ignore last 2 columns
#So, I am dropping last 2 columns
dataset=dataset.drop('Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_1',
        'Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_2')
```

Fig. 8. Dropping last two columns

My next step was to check for null or missing values as they create problems in visualisation and modelling. So, in order to check for null values, I have converted my dataset to pandas data frame and using isnull( ) function, I have detected there were no missing values in my dataset.

```
[23] #Converting Dataset to Pandas Dataframe
     df=dataset.toPandas()
     df.isnull().sum()
```

```
CLIENTNUM                   0
Attrition_Flag              0
Customer_Age                0
Gender                      0
Dependent_count             0
Education_Level             0
Marital_Status              0
Income_Category             0
Card_Category               0
Months_on_book              0
Total_Relationship_Count    0
Months_Inactive_12_mon      0
Contacts_Count_12_mon       0
Credit_Limit                0
Total_Revolving_Bal         0
Avg_Open_To_Buy             0
Total_Amt_Chng_Q4_Q1        0
Total_Trans_Amt             0
Total_Trans_Ct              0
Total_Ct_Chng_Q4_Q1         0
Avg_Utilization_Ratio       0
dtype: int64
```

Fig. 9. Checking for null values

Explorative analysis and visualizations:

In this section, I want to learn as much as possible about the data and visualise some of the most important/interesting findings using clear and data appropriate plotting techniques. Most of the computation for the plotting was done in Tableau. However, I have found some patterns between features using Pyspark also. I have used matplotlib, seaborn and plotly.express libraries.

Although I explored all possible features and their combinations, I reported only the patterns which I found. However, all created visualisations are included in the appendix B.

- **Based On Frequency of Use and Numeral Characteristics:**

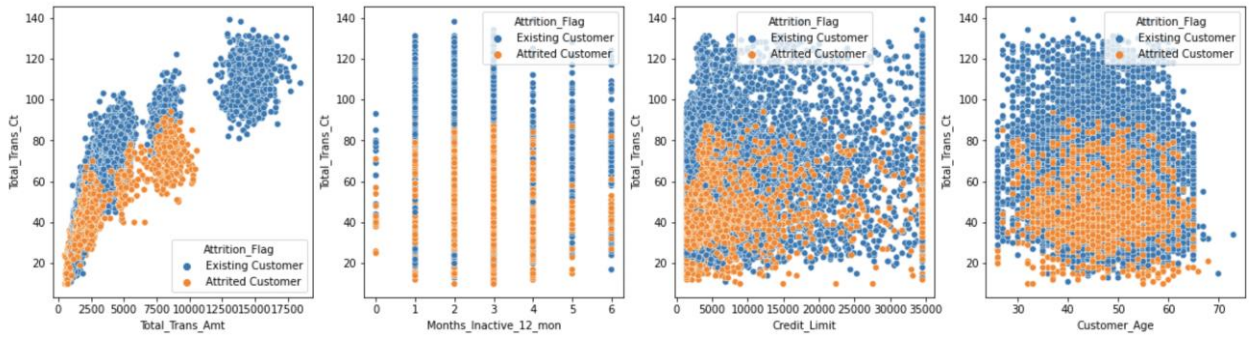


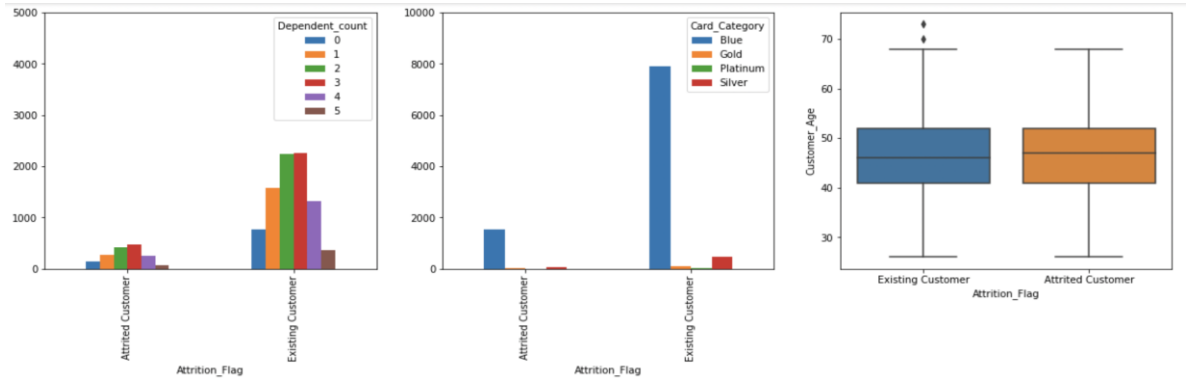
Fig. 10. Visualizations of Numerical Features

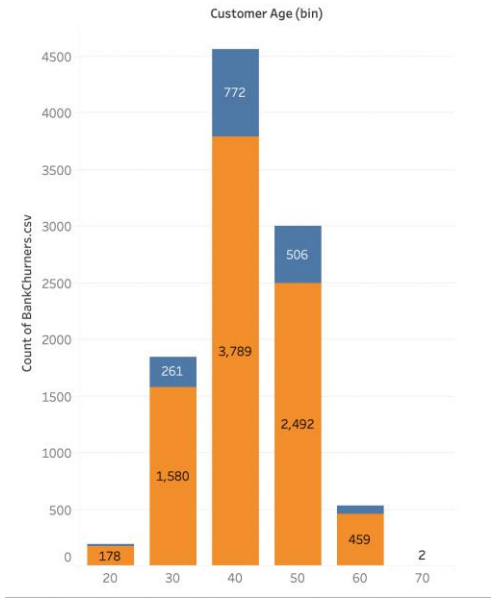
From above scatterplots, it can be clearly observed that:

1. If customers are spending more annually, they are more likely to remain.
2. After 2-3 months of inactivity, the customers are more likely to leave.
3. The higher the credit limit is, the customers are more likely to remain.
4. Age distribution does not really matter in this case, because the clusters are largely overlapped.
5. Almost all churned customers used their cards below 100 times.

- **Based On Demographics:**

Though, the graphs here show discrepancy in numbers between loyal and churned customers, the distribution of each category is alike





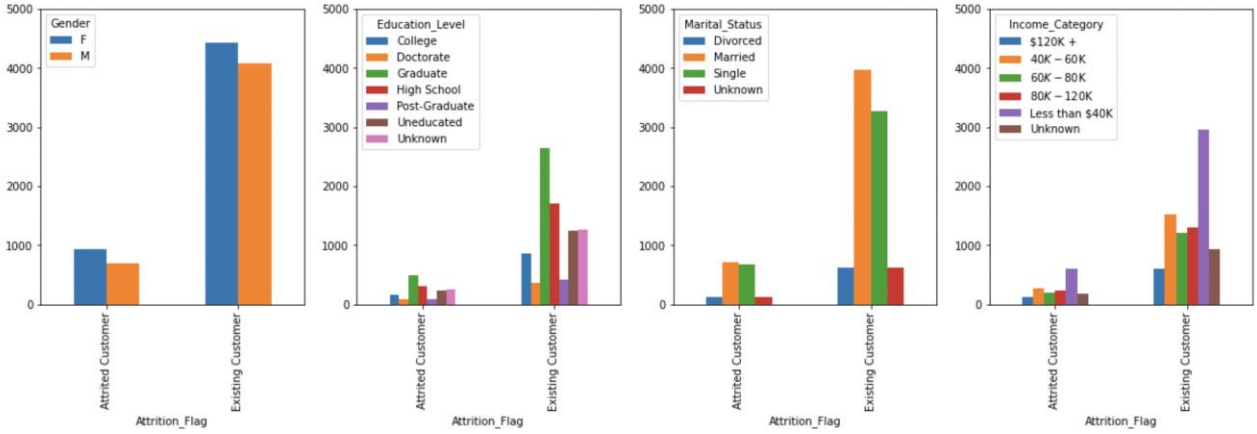


Fig. 11. Visualizations of Categorical Features

From above visualisations, following hypothesis can be made:

- **Attrited Customers** are about 6% of clients' database. They represent 1627 of the 10127 entries. A major chunk of churned customers are female, who are nearly 47 years old, holding a graduate degree, married, with a Blue-Level card, and having 3 dependents.
- The most crucial factors leading to churned clients' decision to switch their banks are Total Transaction Amount, followed by credit limit and revolving balance.
- **Active customers** account for 8500 of the 10127 entries (the majority). They make larger transactions, possess higher credit limits, and revolving balance than churned clients
- Like the other group, major part of the customers are females, holding graduate degrees, about 46 years old, married, who have Blue-level cards, and with an income less than $40K. However, In contrast to attrited clients, active clients have one less dependent.

Correlation:

Here, Standard Correlation function (Pearson coefficient) can not be used directly as my dataset comprises of mixed features: Numerical and Categorical. Firstly, I will divide my features into separate tables.

```
categorical_features = [t[0] for t in dataset.dtypes if t[1]=='string']
df_cat_fea=dataset.select(categorical_features)
df_cat_fea.describe().show()
df_cat_fea=df_cat_fea.toPandas()
numeric_features = [t[0] for t in dataset.dtypes if t[1] == 'int' or t[1] =='double']
df_num_fea=dataset.select(numeric_features)
df_num_fea.describe().show()
df_num_fea=df_num_fea.toPandas()
```

Fig. 12. Splitting Categorical Features and Numerical Features

I will create one more column, Attrition_Flag, in my numerical data which will contain categorical values whether the customer will churn or not. As it has categorical values, so, I have created its dummy variables and then added it to my numerical data frame. At this step, I have deleted one column, Clientnum, because it is not related to predict customer churn.

```
df_num_fea['Attrition_Flag']=df.loc[:,'Attrition_Flag']
oh=pd.get_dummies(df_num_fea['Attrition_Flag'])
df_num_fea=df_num_fea.drop(['Attrition_Flag'],axis=1)
df_num_fea=df_num_fea.drop(['CLIENTNUM'],axis=1)
df_num_fea=df_num_fea.join(oh)
df_num_fea.head()
```

Fig. 13. Creation of dummy variables of column Attrition_Flag

In next step, I will use Cramer's V function for categorical data. Pearson coefficient will be applied to measure correlation for numerical data. From fig. 14 Correlation heatmap**,** it can be concluded that categorical columns are NOT CORRELATED with customer churn by themselves.

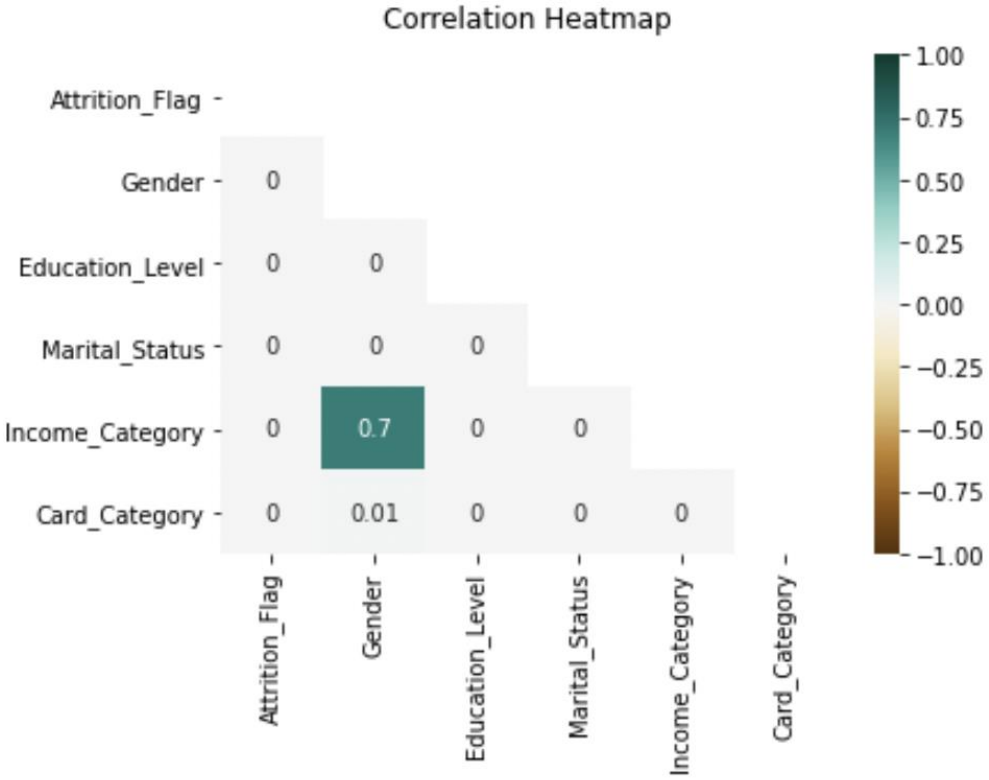


Fig. 14. Correlation heatmap of Categorical Features

From Fig. 15 ,we can now see better correlation measurements to the customer churn.

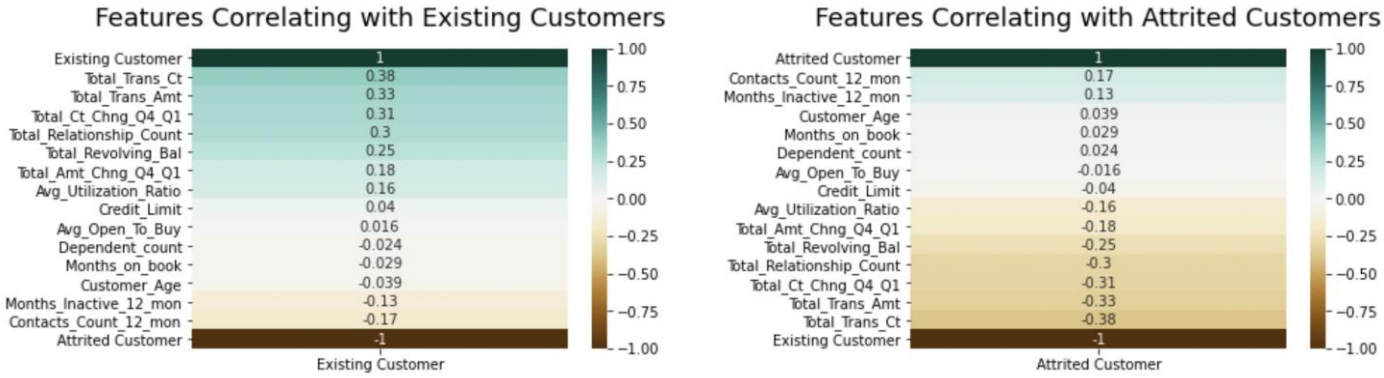


Fig. 15. Correlation heatmap of Numerical Features

Dropping Irrelevant Columns:

From above correlations, I have found that the following features are not correlated with customer churn (between -0.1 and +0.1) :

- Credit Limit
- Average Open to Buy
- Months on book
- Age
- Dependent Count

```
# Droping CLIENTNUM columns because Client number is unique identifier for the customer holding the account
# So this number doesn't really effect our churn prediction.
dataset=dataset.drop('CLIENTNUM','Credit_Limit','Avg_Open_To_Buy','Months_on_book','Dependent_count','Customer_Age')
```

Fig. 16. Dropping Irrelevant Columns

String Indexing:

String Indexing can also be considered as Label encoding. StringIndexer encodes a string column of labels to a column of label indices. If the input column is numeric, we cast it to string and index the string values. The indices are in [0, numLabels) (Role of StringIndexer and Pipelines in PySpark ML Feature, 2020).

```
SI_att_flag = StringIndexer(inputCol="Attrition_Flag", outputCol="Attrition_Flag_encoding")
dataset = SI_att_flag.fit(dataset).transform(dataset)

SI_Gender = StringIndexer(inputCol="Gender", outputCol="Gender_encoding")
dataset = SI_Gender.fit(dataset).transform(dataset)

SI_Education_Level = StringIndexer(inputCol="Education_Level", outputCol="Education_Level_encoding")
dataset = SI_Education_Level.fit(dataset).transform(dataset)

SI_Marital_Status = StringIndexer(inputCol="Marital_Status", outputCol="Marital_Status_encoding")
dataset = SI_Marital_Status.fit(dataset).transform(dataset)

SI_Income_Category = StringIndexer(inputCol="Income_Category", outputCol="Income_Category_encoding")
dataset = SI_Income_Category.fit(dataset).transform(dataset)

SI_Card_Category = StringIndexer(inputCol="Card_Category", outputCol="Card_Category_encoding")
dataset = SI_Card_Category.fit(dataset).transform(dataset)


dataset.show(30)
```

Fig. 17. String Indexing of Features

Vector Assembler:

In this step, I have combined all of the columns containing features into a single column. This has to be done before modelling can take place because every PySpark modeling routine expects the data to be in this form. I stored each of the values from a column as an entry in a vector. Then, from the model's point of view, every observation is a vector that contains all of the information about it and a label that tells the modeler what value that observation corresponds to (Assemble a vector | Python, 2021). I have used pyspark.ml.feature submodule which contains a class called VectorAssembler. This Transformer takes all of the columns which are specified and combines them into a new vector column.

```
[ ]  # Assemble all the features with VectorAssembler

     required_features = ['Total_Relationship_Count', 'Months_Inactive_12_mon', 'Contacts_Count_12_mon',
                          'Total_Revolving_Bal', 'Total_Amt_Chng_Q4_Q1','Total_Trans_Amt',
                          'Total_Trans_Ct','Total_Ct_Chng_Q4_Q1','Avg_Utilization_Ratio',
                          'Gender_encoding','Education_Level_encoding', 'Marital_Status_encoding',
                          'Income_Category_encoding','Card_Category_encoding'
                         ]

     assembler = VectorAssembler(inputCols=required_features, outputCol='Outcome')

     transformed_data = assembler.transform(dataset)

     transformed_data.show()
```

Fig. 18. Creation of Vectors from Features

Feature Scaling:

Feature scaling in machine learning is one of the most critical steps during the pre-processing of data before creating a machine learning model. Scaling can make a difference between a weak machine learning model and a better one. Machine learning algorithm just sees number — if there is a vast difference in the range say few ranging in thousands and few ranging in the tens, and it makes the underlying assumption that higher ranging numbers have superiority of some sort. So these more significant number starts playing a more decisive role while training the model (Roy, 2020).

```
# StandardScaler
scaler = StandardScaler(inputCol="Outcome", outputCol="scaledFeatures")

# Compute summary statistics by fitting the StandardScaler
scalerModel = scaler.fit(transformed_data)

# Normalize each feature to have unit standard deviation.
scaledData = scalerModel.transform(transformed_data)
scaledData.show()
```

Fig. 19. Scaling of Features

Down Sampling:

From Fig. It is clear that the data set is highly skewed, So , I am going to use one of the simplest method called down-samplings, which means some of the majority cases would be just randomly filtered out. It is suitable where dataset has large values which is applicable in my project.

```
scaledData.select("Attrition_Flag_encoding").groupBy("Attrition_Flag_encoding").count().collect()
```

```
[Row(Attrition_Flag_encoding=0.0, count=8500),
 Row(Attrition_Flag_encoding=1.0, count=1627)]
```

Fig. 20. Checking count of each class

After performing Down-Sampling, samples are nearly balanced.

```
# Down-sampling
scaledData_ds, _ = scaledData.filter(scaledData["Attrition_Flag_encoding"] == 0.0).randomSplit([0.4, 0.6])
scaledData_ds = scaledData.filter(scaledData["Attrition_Flag_encoding"] == 1.0).union(scaledData_ds)
scaledData_ds.select("Attrition_Flag_encoding").groupBy("Attrition_Flag_encoding").count().collect()
```

```
[Row(Attrition_Flag_encoding=0.0, count=3434),
 Row(Attrition_Flag_encoding=1.0, count=1627)]
```

Fig. 21. Unsampling and count of each class

Test and Train Data:

Next, the dataset has been split into training data and test data. I have provided 80% of my dataset to train the model and 20% for testing purposes.

```
# Split the data
(training_data, test_data) = scaledData_ds.randomSplit([0.8,0.2], seed=1)
```

Fig. 22. Splitting dataset into training and testing data

Classification:

Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention ("Machine Learning: What it is and why it matters", 2021). Machine Learning has various applications such as image recognition, Self-driving cars, Virtual Personal Assistance, Computational Finance, Search engine etc.

There are many types of learnings available in Machine Learning; but broadly classified into two types: Supervised Learning and Unsupervised Learning.

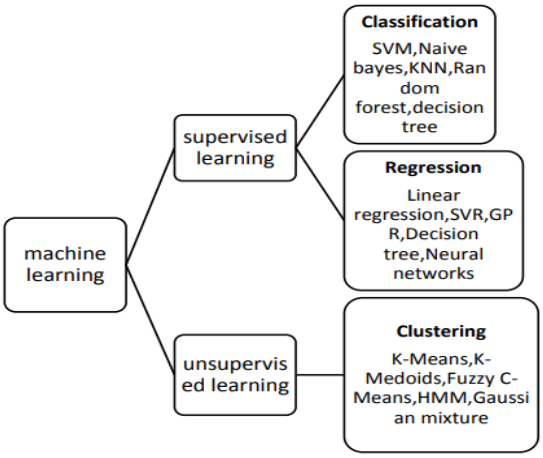


Fig. 23. Some Important Machine Learning Algorithms

From the above mentioned machine learning models, I have used Classification Techniques: Random Forest Classifier, Decision tree Classifier, and Logistic Regression Classifier .

Random forest Classifier:

I have used Random forest algorithm as it is considered as an 'Ensemble Method' in machine learning. It makes use of Decision Trees beneath and forms multiple trees and eventually takes majority vote out of it. It also reduces variance and overfitting problem of decision trees, which improves its accuracy. This algorithm also handles non-linear parameters efficiently. Using Random Forest Classifier, I have fit my training data into model. I have also evaluated Accuracy and Precision of my model.

```
#Define the model
rf = RandomForestClassifier(labelCol='Attrition_Flag_encoding',
                            featuresCol='scaledFeatures',
                            maxDepth=5)

# Fit the model
model = rf.fit(training_data)
# Predict with the test dataset
rf_predictions = model.transform(test_data)
from pyspark.ml.evaluation import MulticlassClassificationEvaluator


multi_evaluator_a = MulticlassClassificationEvaluator(labelCol = 'Attrition_Flag_encoding', metricName = 'accuracy')
print('Random Forest classifier Accuracy:', multi_evaluator_a.evaluate(rf_predictions))

multi_evaluator_p = MulticlassClassificationEvaluator(labelCol = 'Attrition_Flag_encoding', metricName = 'weightedPrecision')
print('Random Forest classifier Precision:', multi_evaluator_p.evaluate(rf_predictions))
```

```
Random Forest classifier Accuracy: 0.9042769857433809
Random Forest classifier Precision: 0.903524766986527
```

Fig. 24. Random Forest Classifier

After accuracy and precision, I have also evaluated ROC score and visualised AUC metrics.

```
sc =SparkContext.getOrCreate()   # We need to create SparkContext
results = rf_predictions.select(['probability', 'Attrition_Flag_encoding'])

## prepare score-label set
results_collect = results.collect()
results_list = [(float(i[0][0]), 1.0-float(i[1])) for i in results_collect]
scoreAndLabels = sc.parallelize(results_list)

metrics = metric(scoreAndLabels)
print("The ROC score is : ", metrics.areaUnderROC)
```

```
The ROC score is :  0.9631368260819305
```

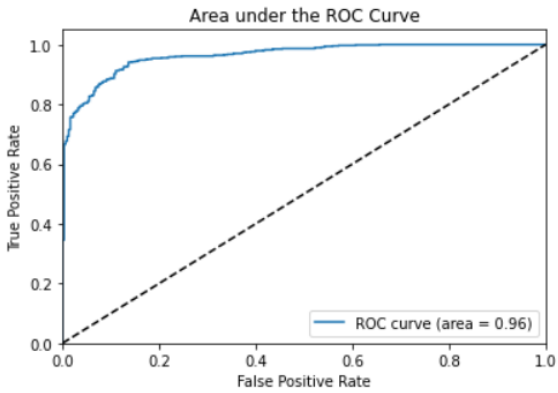


Fig. 25. Checking ROC score and Plotting ROC Curve of Random Forest Classifer

Decision Tree Classifier:

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation ("1.10. Decision Trees — scikit-learn 0.24.2 documentation", 2007). I have used Decision Tree Classifier as decision trees require very less effort for data preparation during data pre-processing in comparison to other algorithms. It has several other advantages also like not requires scaling of data. Furthermore, it can also ignore null values to a some extent. The best part is it is very intuitive and easily explainable.

I have fit my training data into model using Decision Tree Classifier. I have also evaluated Accuracy and Precision of my model.

```
dt = DecisionTreeClassifier(featuresCol = 'scaledFeatures', labelCol = 'Attrition_Flag_encoding', maxDepth = 3)
dtModel = dt.fit(training_data)
dt_predictions = dtModel.transform(test_data)
multi_evaluator_a = MulticlassClassificationEvaluator(labelCol = 'Attrition_Flag_encoding', metricName = 'accuracy')
print('Decision Tree Accuracy:', multi_evaluator_a.evaluate(dt_predictions))
multi_evaluator_p = MulticlassClassificationEvaluator(labelCol = 'Attrition_Flag_encoding',
                                                      metricName = 'weightedPrecision')
print('Decision Tree Precision:', multi_evaluator_p.evaluate(dt_predictions))
```

```
Decision Tree Accuracy: 0.8767820773930753
Decision Tree Precision: 0.8757127376931148
```

Fig. 26. Decision Tree Classifier

After accuracy and precision, I have also evaluated ROC score and visualised AUC metrics.

```
results = dt_predictions.select(['probability', 'Attrition_Flag_encoding'])

## prepare score-label set
results_collect = results.collect()
results_list = [(float(i[0][0]), 1.0-float(i[1])) for i in results_collect]
scoreAndLabels = sc.parallelize(results_list)

metrics = metric(scoreAndLabels)
print("The ROC score is : ", metrics.areaUnderROC)
```

```
The ROC score is :  0.8986666152838261
```

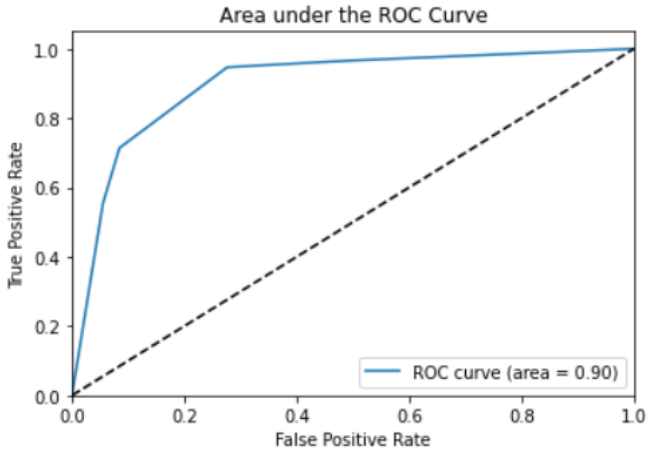


Fig. 25. Checking ROC score and Plotting ROC Curve of Decision Tree Classifer

Logistic Regression:

I have used Logistic Regression model next as it is a Machine Learning algorithm which is used for the classification problems. It is suitable for models when target values are discrete

which are suitable for my model. It is a predictive analysis algorithm and based on the concept of probability. Logistic regression transforms its output using the logistic sigmoid function to return a probability value (Pant, 2019). I have fit my training data into model using Logistic Regression Classifier. I have also evaluated Accuracy and Precision of my model.

```
lr = LogisticRegression(featuresCol = 'scaledFeatures', labelCol = 'Attrition_Flag_encoding', maxIter=10)
lrModel = lr.fit(training_data)
lr_predictions = dtModel.transform(test_data)

multi_evaluator_a = MulticlassClassificationEvaluator(labelCol = 'Attrition_Flag_encoding', metricName = 'accuracy')
print('Logistic Regression Accuracy:', multi_evaluator_a.evaluate(lr_predictions))

multi_evaluator_p = MulticlassClassificationEvaluator(labelCol = 'Attrition_Flag_encoding', metricName = 'accuracy')
print('Logistic Regression Accuracy:', multi_evaluator_p.evaluate(lr_predictions))
```

```
Logistic Regression Accuracy: 0.8767820773930753
Logistic Regression Accuracy: 0.8767820773930753
```

Fig. 26. Logistic Regression Classifier

After accuracy and precision, I have also evaluated ROC score and visualised AUC metrics.

```
results = lr_predictions.select(['probability', 'Attrition_Flag_encoding'])

## prepare score-label set
results_collect = results.collect()
results_list = [(float(i[0][0]), 1.0-float(i[1])) for i in results_collect]
scoreAndLabels = sc.parallelize(results_list)

metrics = metric(scoreAndLabels)
print("The ROC score is : ", metrics.areaUnderROC)
```

```
The ROC score is :  0.8986666152838261
```

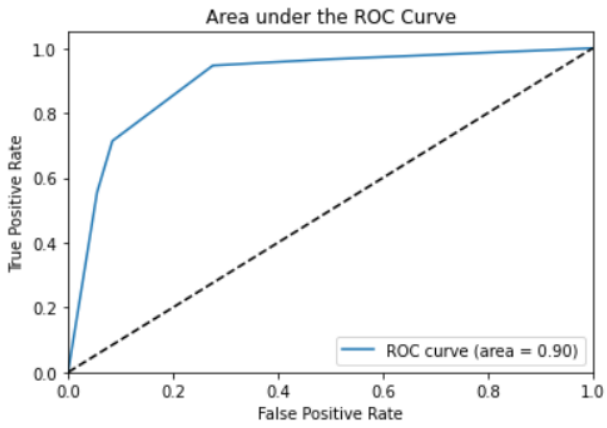


Fig. 27. Checking ROC score and Plotting ROC Curve of Logistic Regression Classifier

Discussion and Conclusion:

The analysis was appropriate for the current dataset and values based on past historical data of some customers, however part of the project was to predict if customer will churn or not. Therefore, machine learning was adopted and we used a supervised approach to it. Models were trained using parameters that deliver the best results. The results of models used can be summarized as:

Model	Precision	Accuracy	ROC Score
Random Forest	0.9035	0.9042	0.9631
Decision Tree	0.8757	0.8767	0.8986
Logistic Regression	0.8757	0.8767	0.8988

Table 1. Comparison of results of models used

Without any doubt, Random Forest outperformed Decision Tree and Logistic Regression with the accuracy of 90%. Higher the accuracy, the more reliable the output will be. However, there is slight difference in performances of these models, especially between Decision Tree and Logistic Regression; which reflected nearly same results.

Customer churn is a good indicator for industries, in which, clients pay for services on a continual basis. That is the reason why clients always get so many advantageous and interesting offers by company, in order to retain them. Customers have a chance to switch every time their commitment ends. Undoubtedly, some natural attrition is unavoidable, but if attrition rate is high, it becomes matter of concern for any industry. Therefore, this model will be useful for many industries who are concerned with their attrition rates so that they approach their clients and prevent them from switching.

Pyspark is a great language for data scientists to use as it helps in Machine Learning pipelines as well as for scalable analysis. I have also applied my knowledge of Python and Pandas to Spark to analyse. As pyspark was new to me, so it was very educational and challenging from my part, but I took in my stride to achieve these and research new literature to expand my knowledge and abilities. I have learnt Exploratory Data Analysis and machine learning applications using Pyspark. I used three algorithms in this study.

Appendix :

Appendix A:

The link to dataset used is:

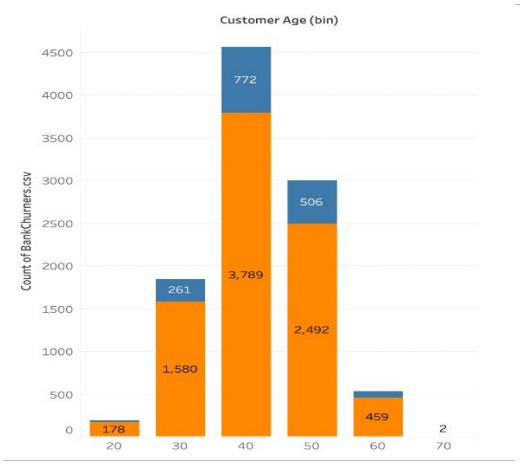
https://www.kaggle.com/sakshigoyal7/credit-card-customers

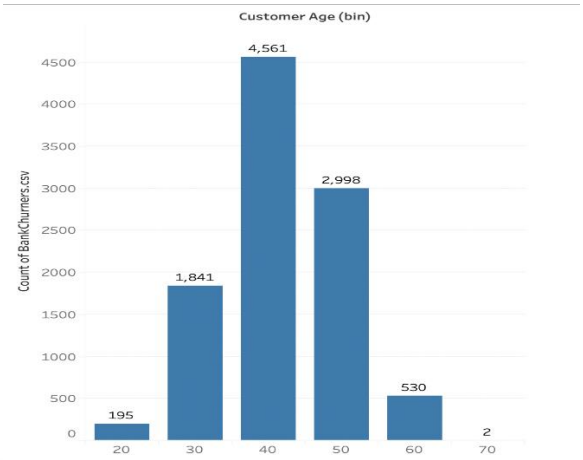
The complete source code for executing and producing all plots can be found on following link:

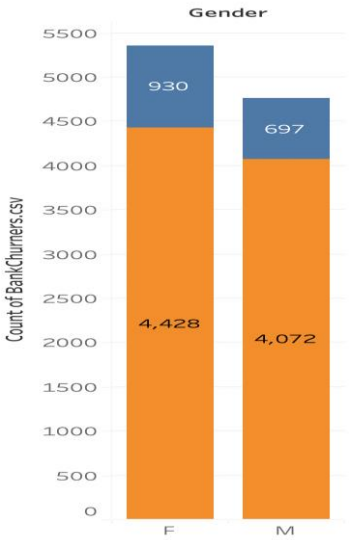
https://colab.research.google.com/drive/1QJISUHqwKwh5-lpMjZof05Z0WDhwwrMl?usp=sharing

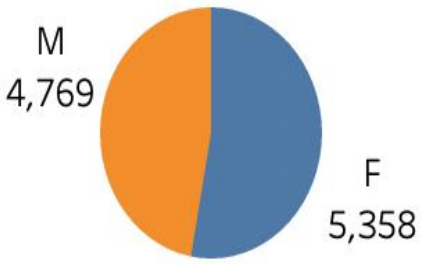
Appendix B:

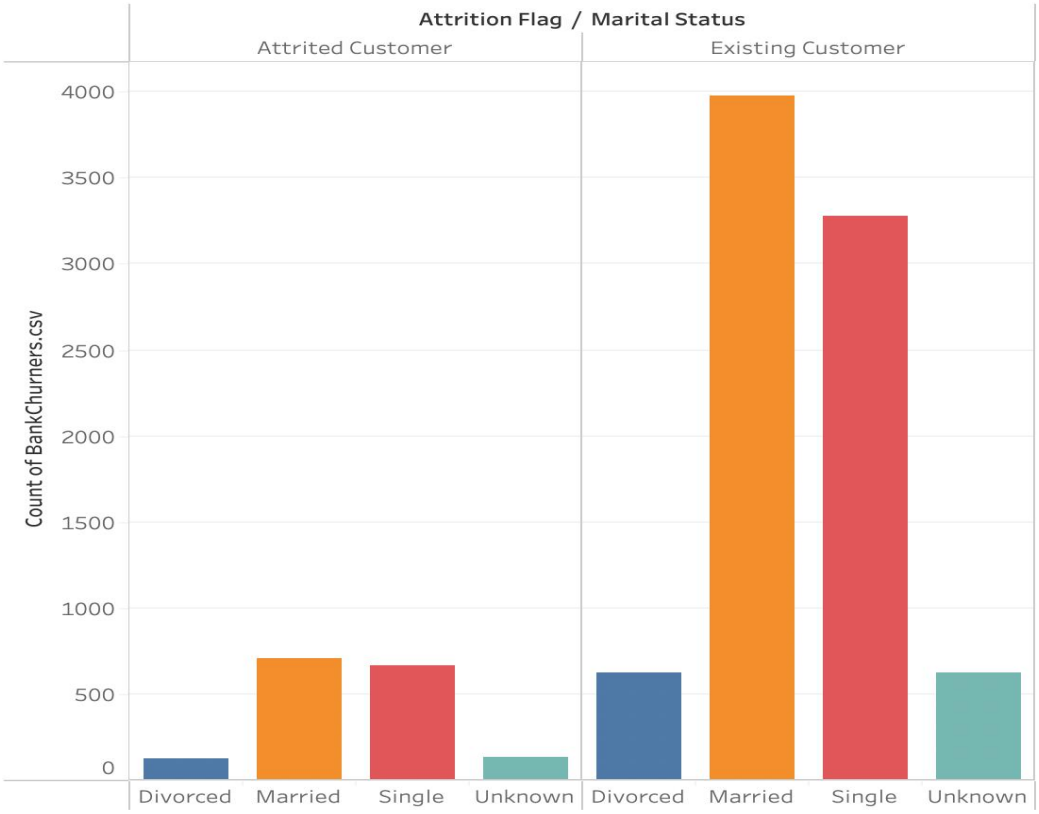
These are all the visualisations produced for this project using Tableau:

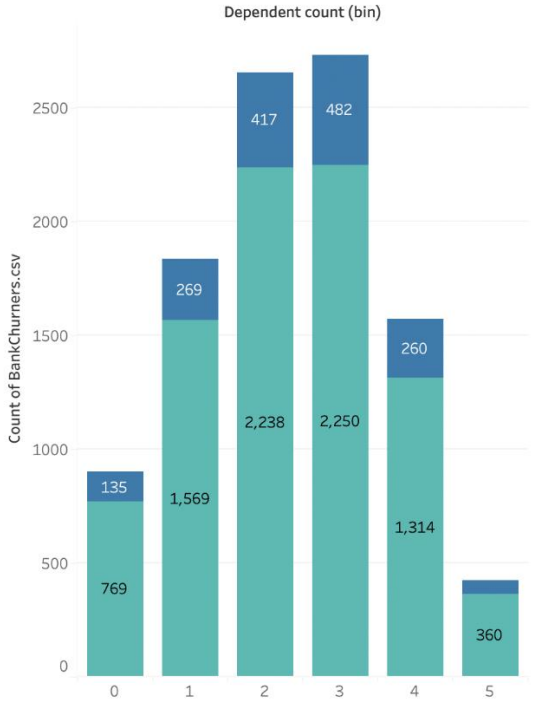


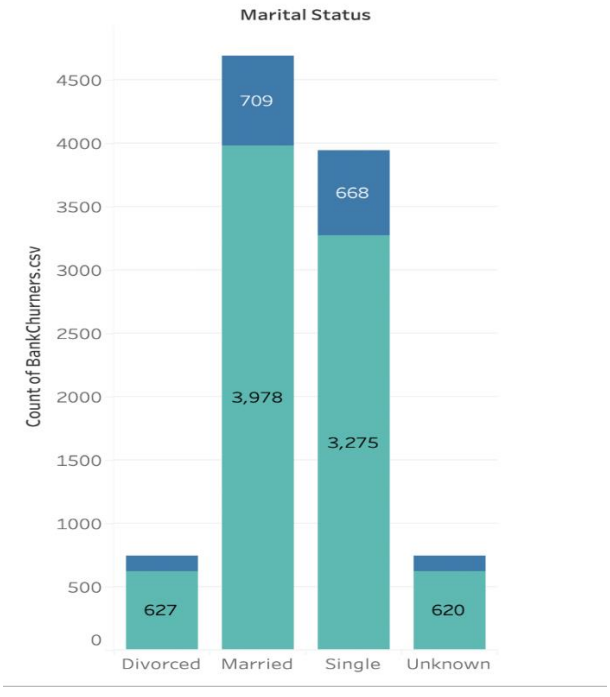


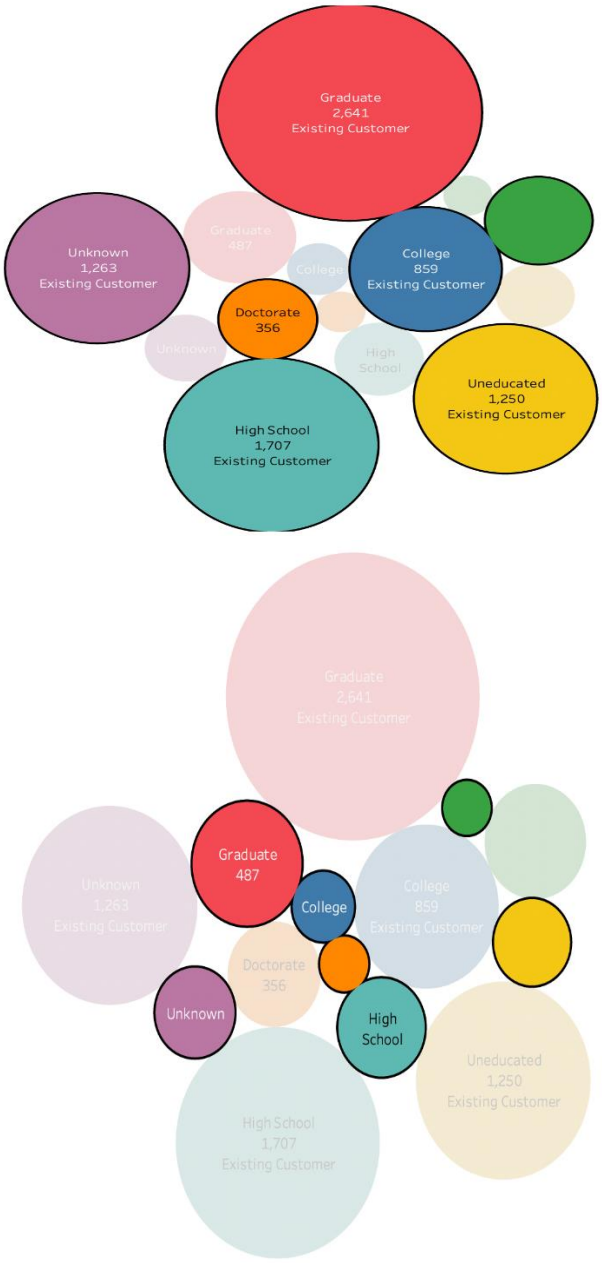


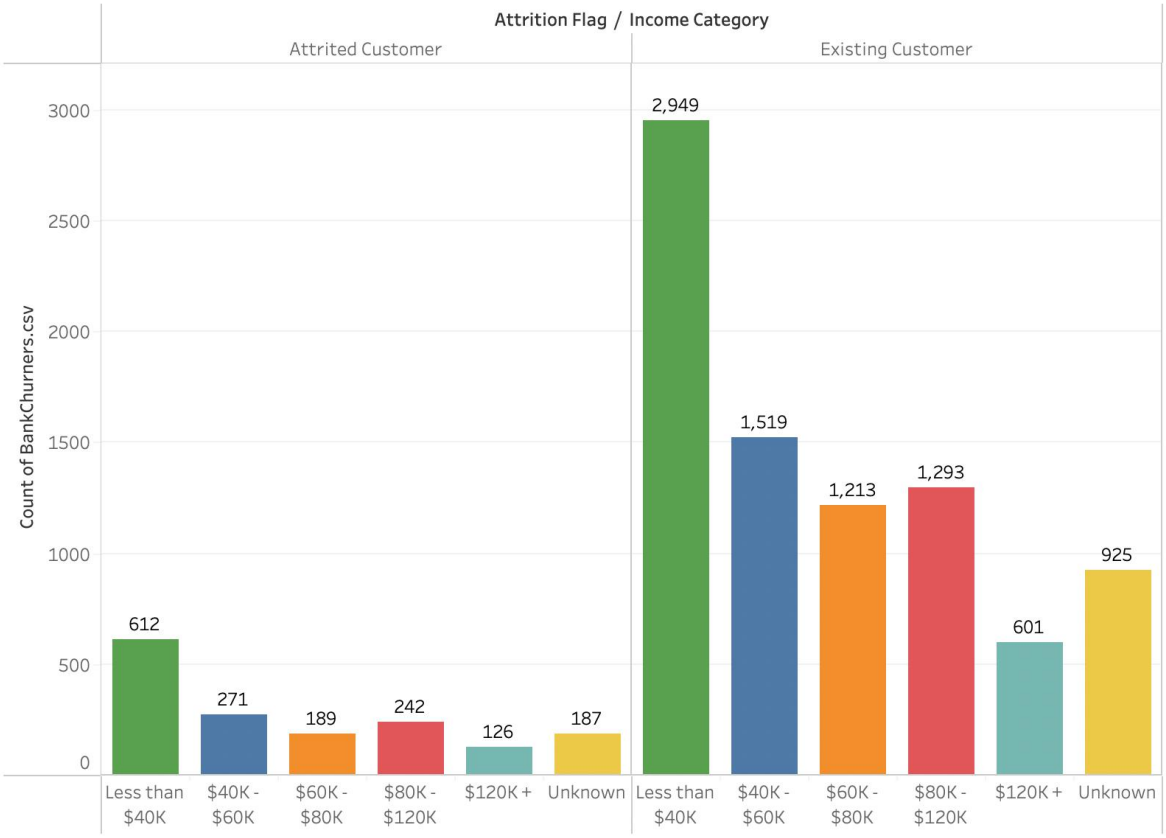


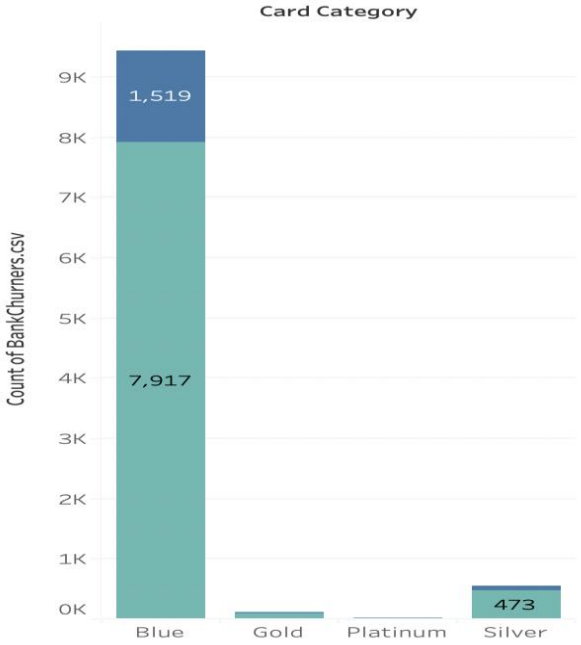

Attrition Flag / Marital Status
Attrited Customer
Existing Customer
Count of BankChurners.csv
4000
3500
3000
2500
2000
1500
1000
500
0
Divorced
Married
Single
Unknown
Divorced
Married
Single
Unknown

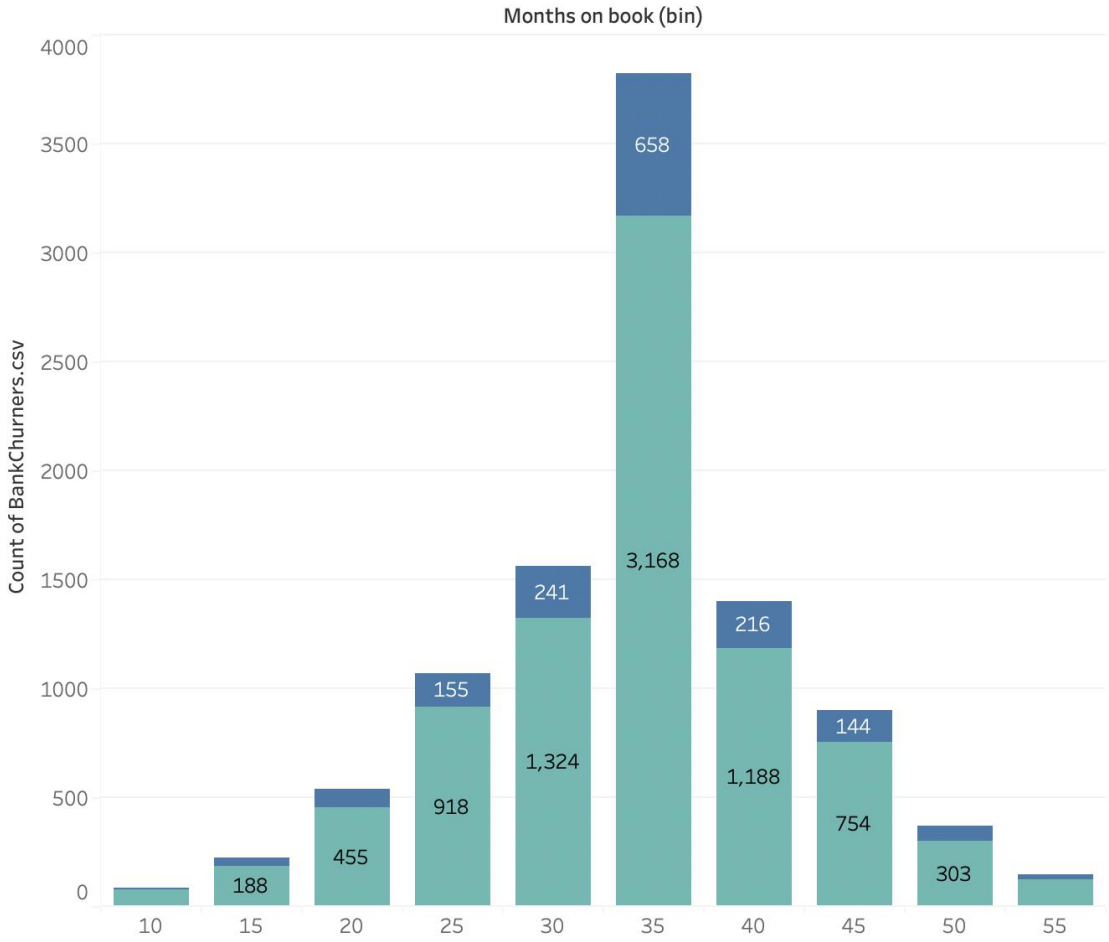

Dependent count (bin)
Count of BankChurners.csv
2500
2000
1500
1000
500
0
135
769
269
1,569
417
2,238
482
2,250
260
1,314
360
0
1
2
3
4
5

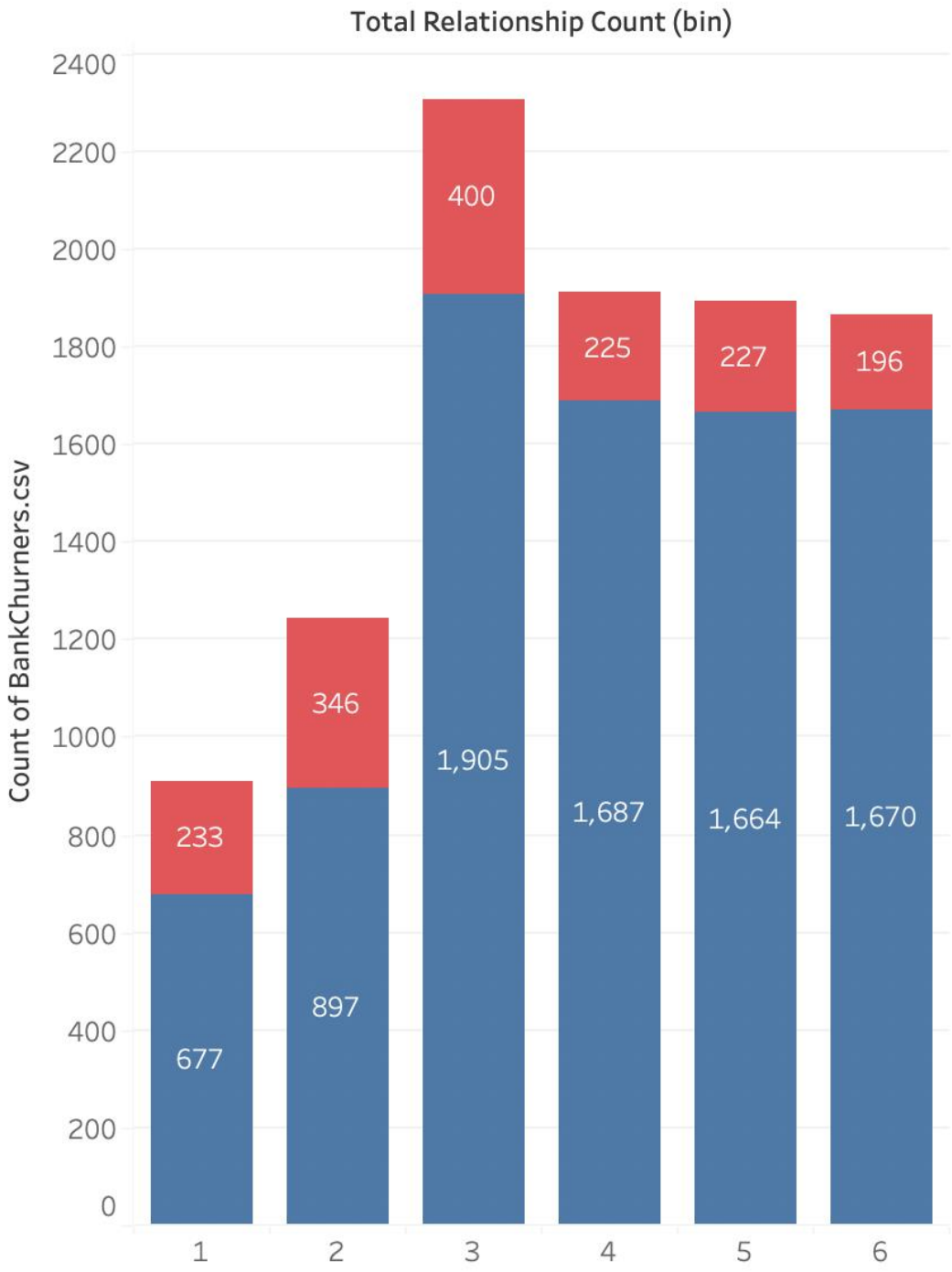

Marital Status
Count of BankChurners.csv
4500
4000
3500
3000
2500
2000
1500
1000
500
0
627
709
3,978
668
3,275
620
Divorced
Married
Single
Unknown

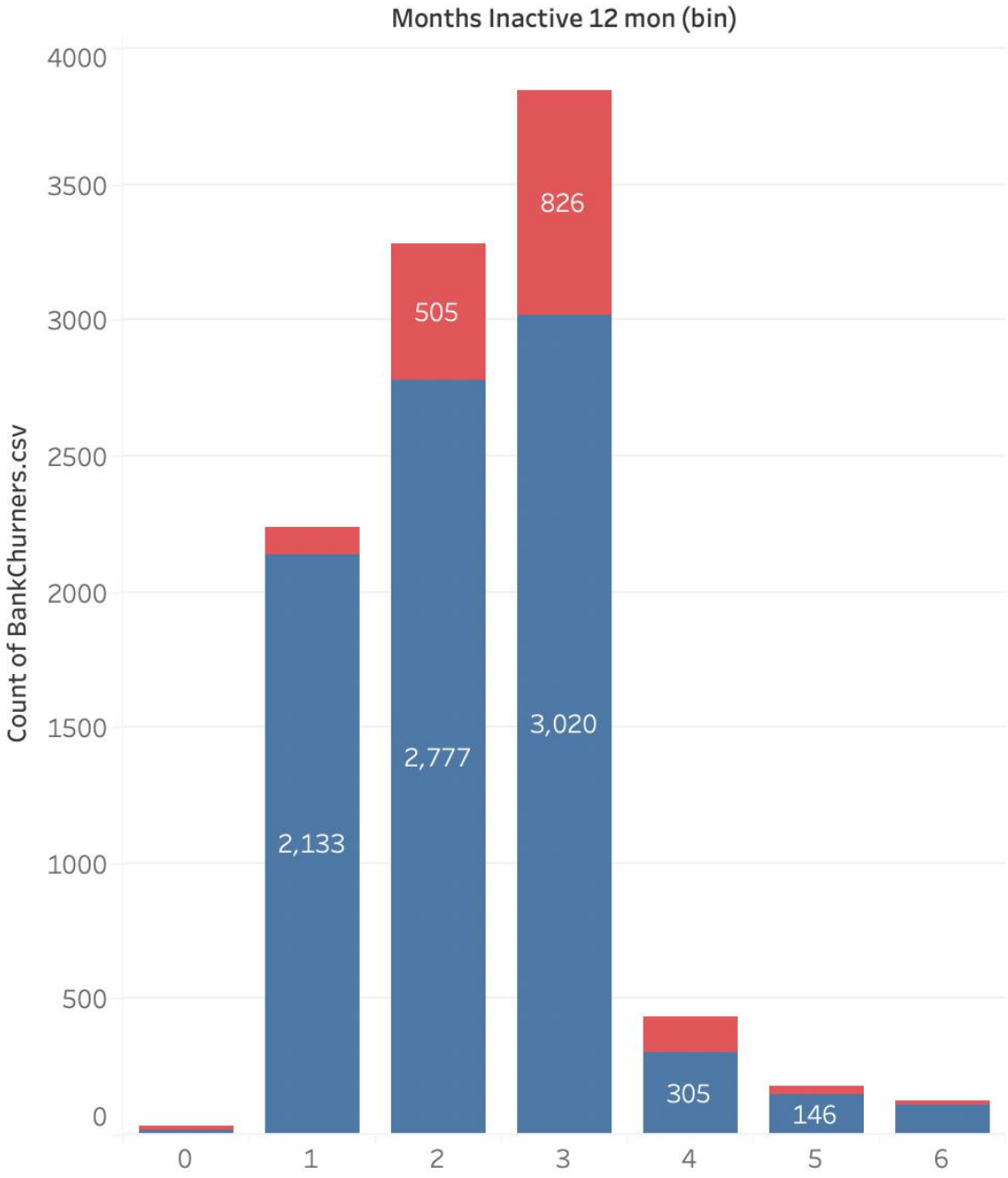

Graduate
2,641
Existing Customer
Unknown
1,263
Existing Customer
College
859
Existing Customer
Doctorate
356
High School
1,707
Existing Customer
Uneducated
1,250
Existing Customer
Graduate
487
College
Unknown
High School

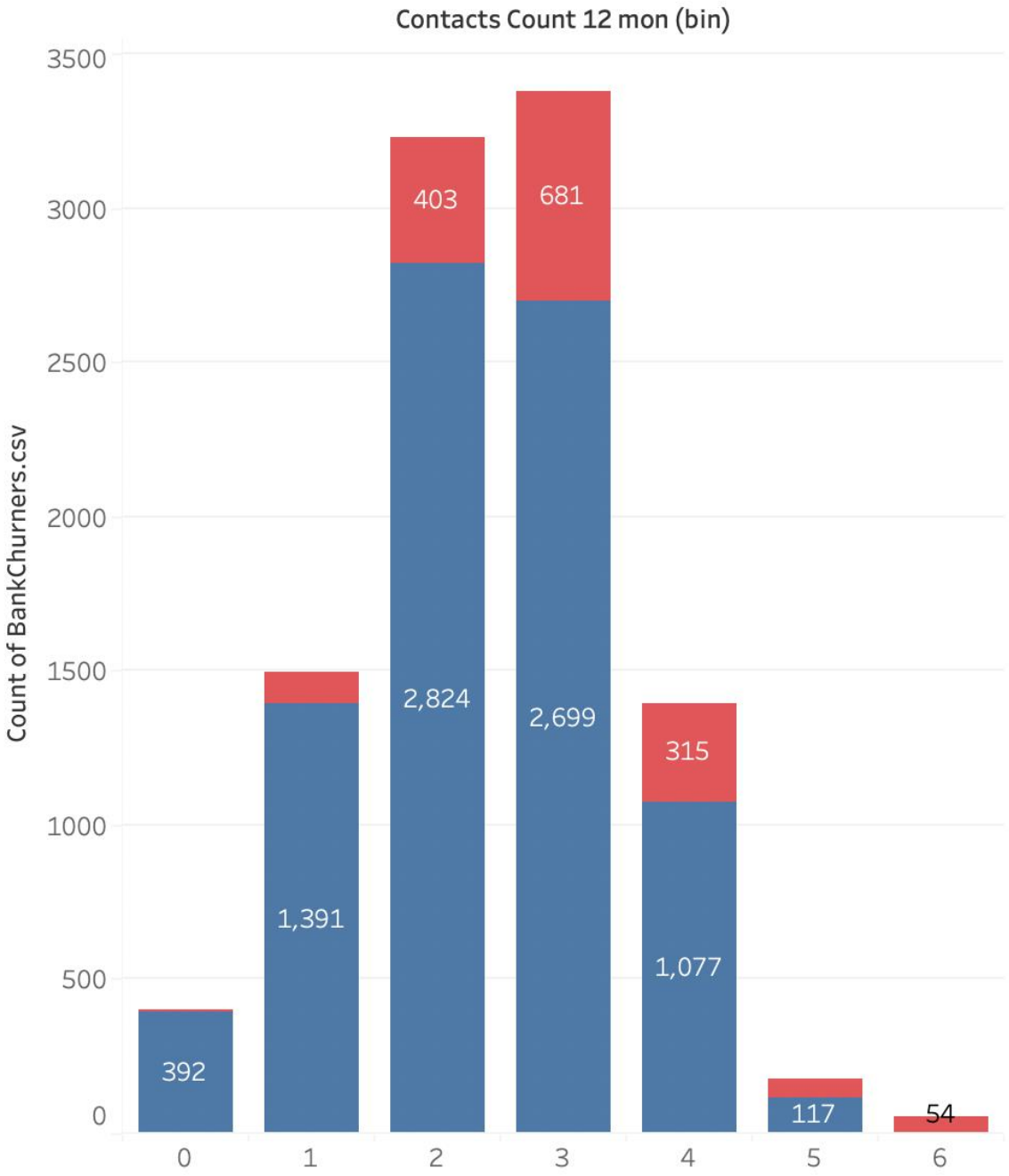

Attrition Flag / Income Category
Attrited Customer
Existing Customer
Count of BankChurners.csv
3000
2500
2000
1500
1000
500
0
612
271
189
242
126
187
2,949
1,519
1,213
1,293
601
925
Less than $40K
$40K - $60K
$60K - $80K
$80K - $120K
$120K +
Unknown

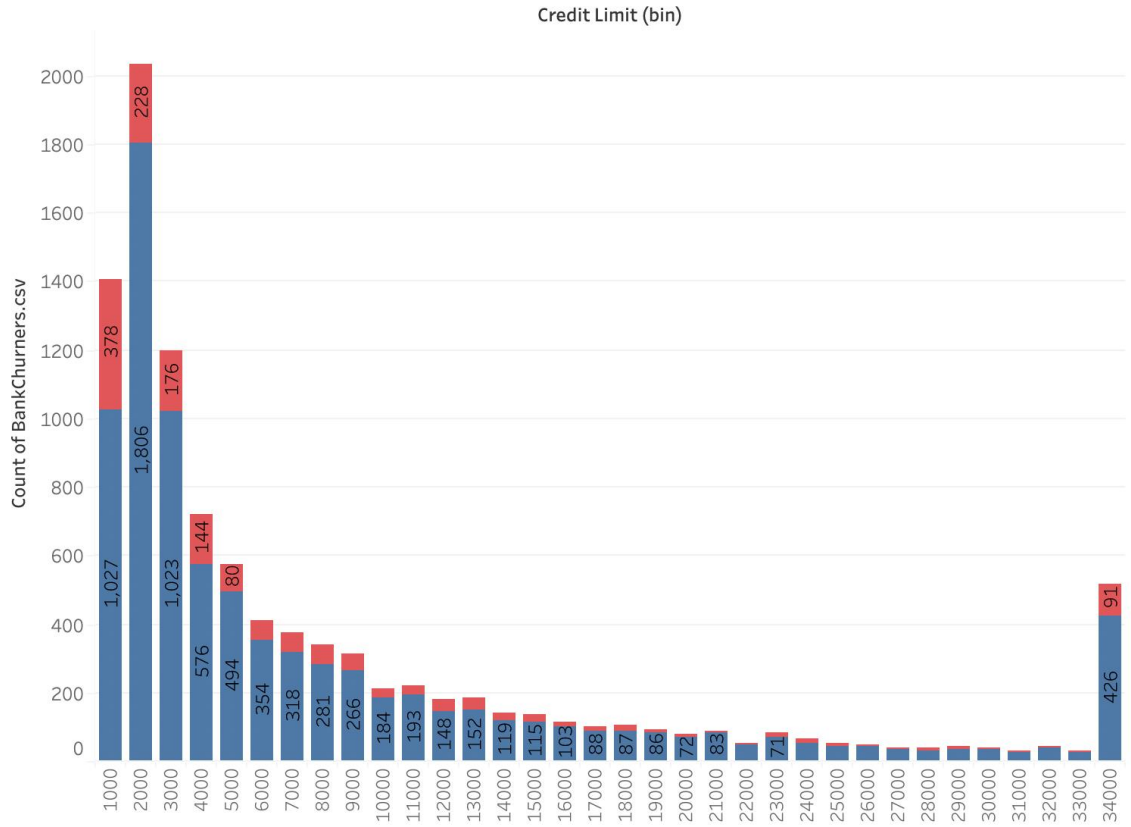

Card Category
Count of BankChurners.csv
9K
8K
7K
6K
5K
4K
3K
2K
1K
0K
1,519
7,917
473
Blue
Gold
Platinum
Silver

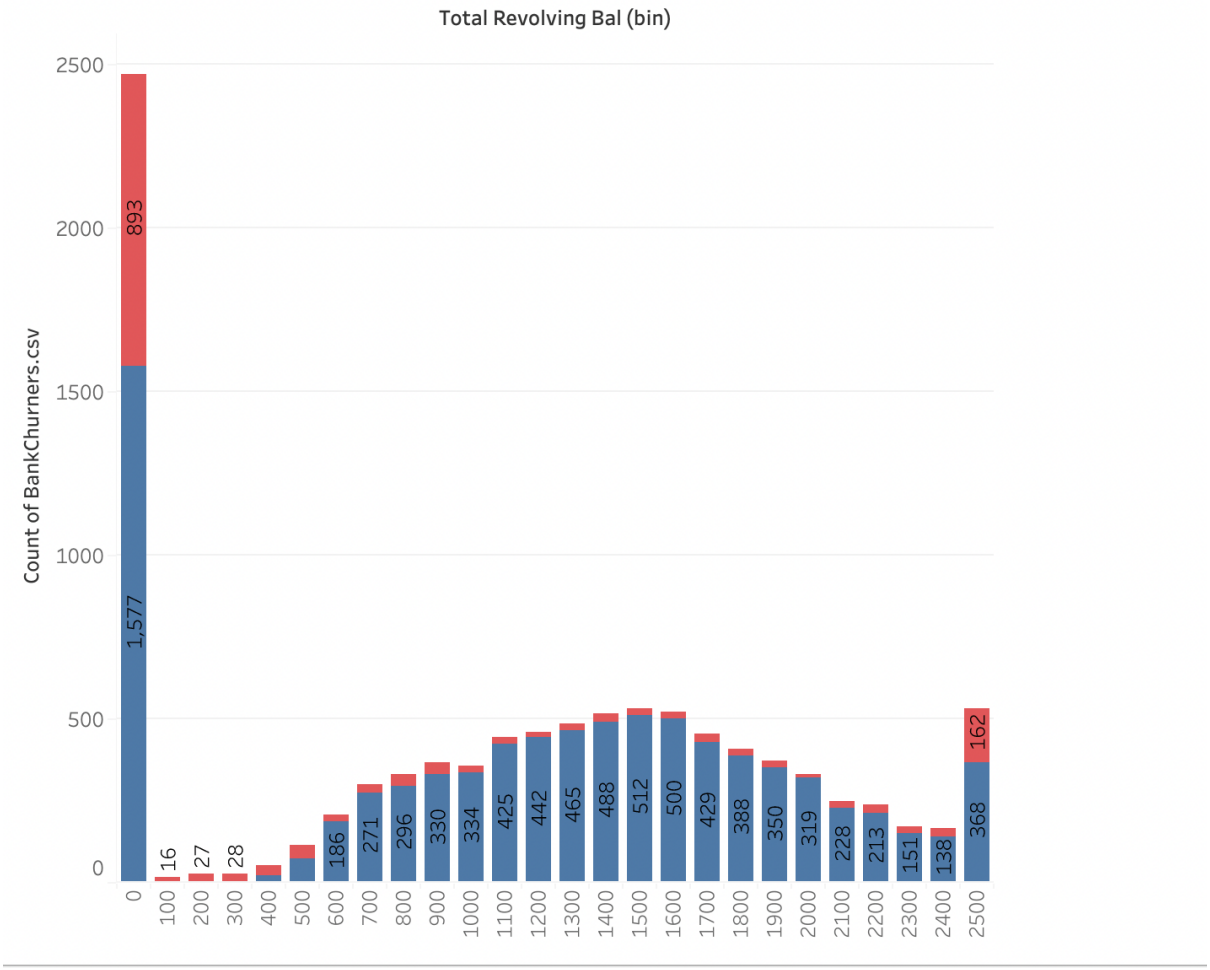

Months on book (bin)
Count of BankChurners.csv
4000
3500
3000
2500
2000
1500
1000
500
0
658
3,168
241
1,324
155
918
216
1,188
144
754
455
188
303
10
15
20
25
30
35
40
45
50
55

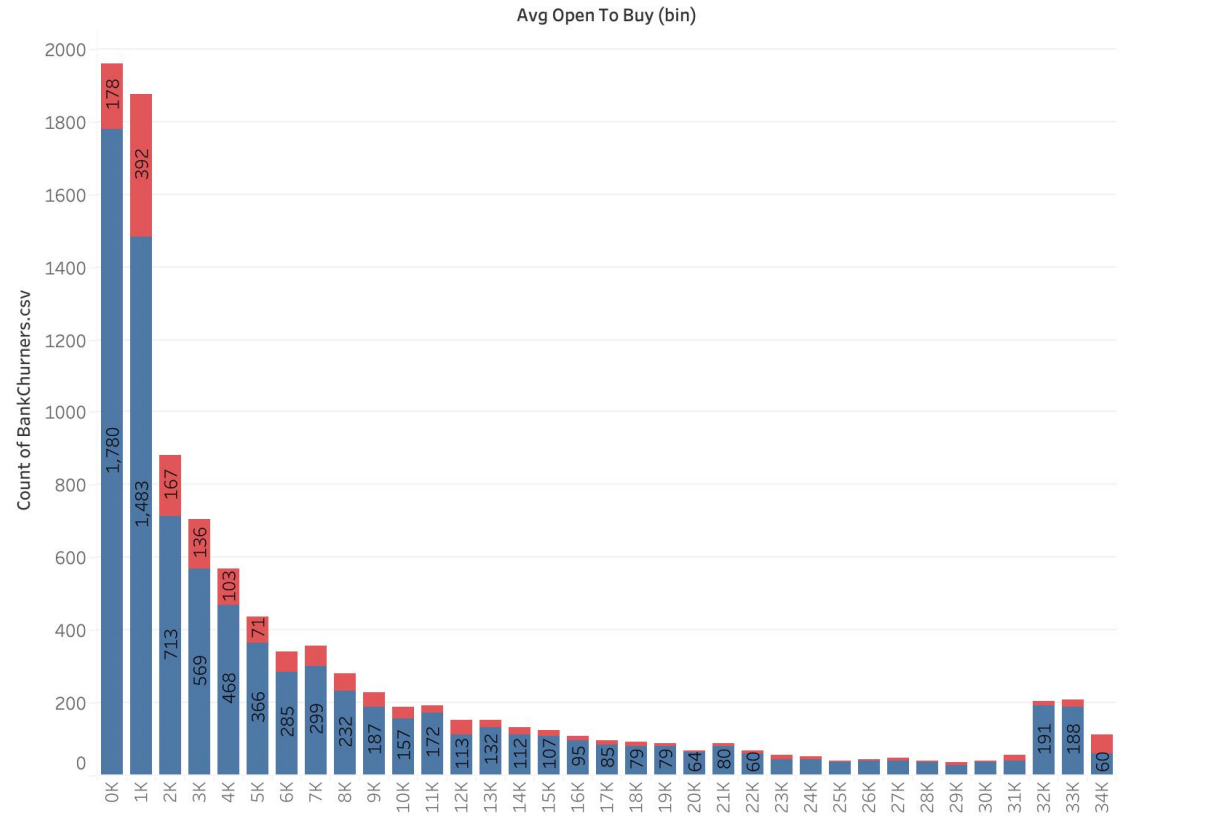

Total Relationship Count (bin)
Count of BankChurners.csv
2400
2200
2000
1800
1600
1400
1200
1000
800
600
400
200
0
233
677
346
897
400
1,905
225
1,687
227
1,664
196
1,670
1
2
3
4
5
6

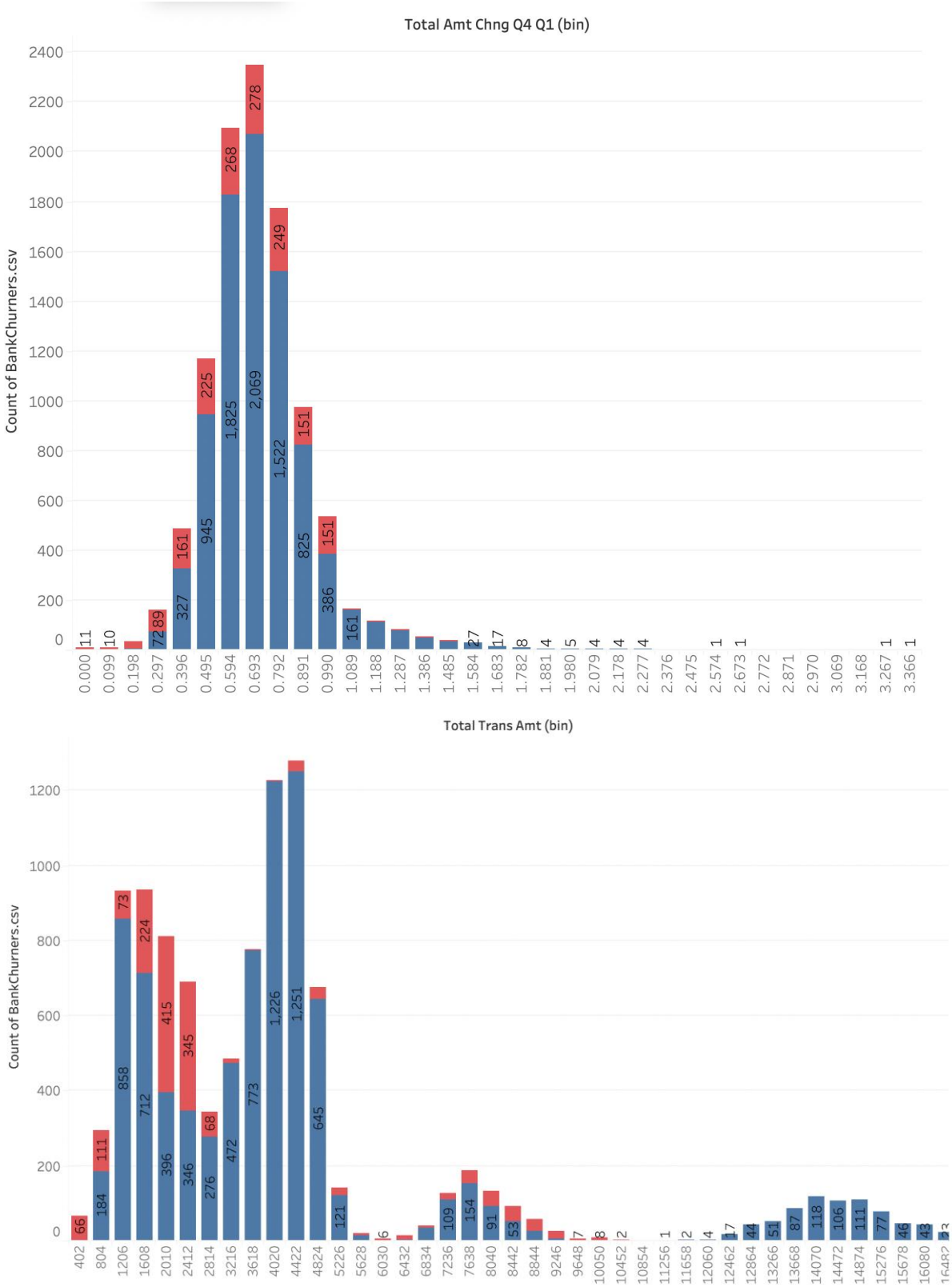

Months Inactive 12 mon (bin)
Count of BankChurners.csv
4000
3500
3000
2500
2000
1500
1000
500
0
826
505
2,133
2,777
3,020
305
146
0
1
2
3
4
5
6

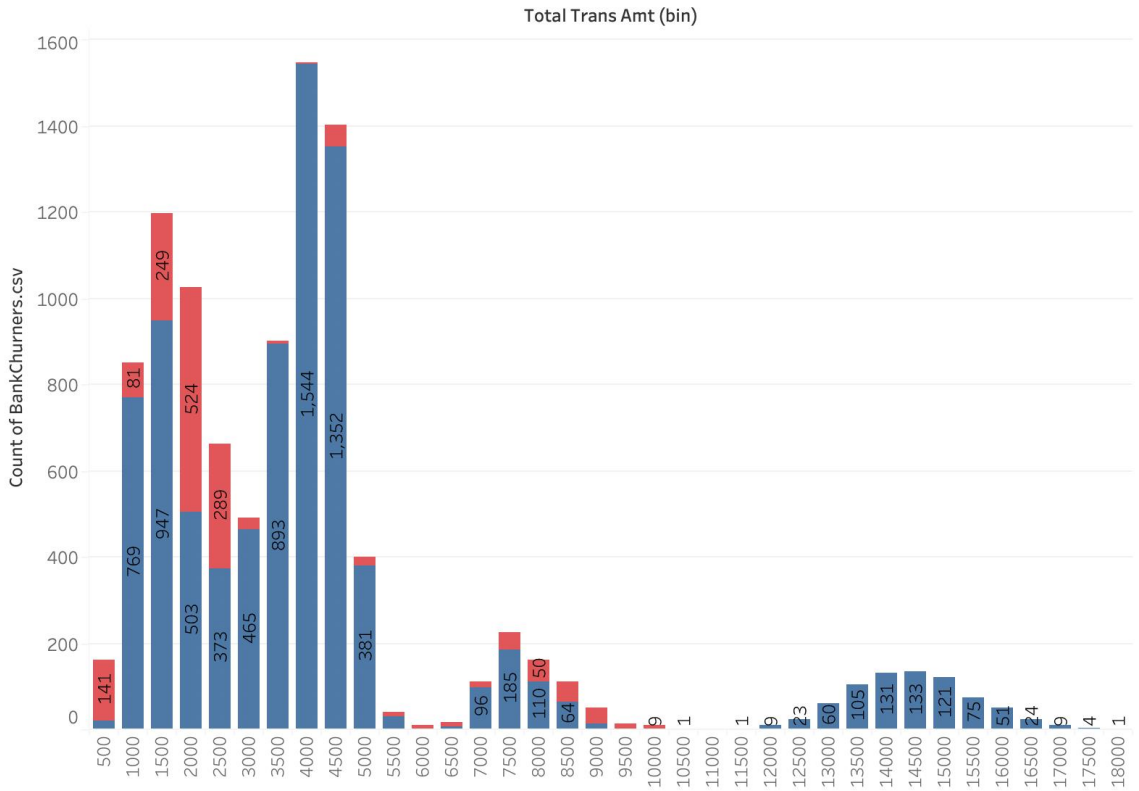

Contacts Count 12 mon (bin)
Count of BankChurners.csv
3500
3000
2500
2000
1500
1000
500
0
392
1,391
2,824
403
2,699
681
1,077
315
117
54
0
1
2
3
4
5
6

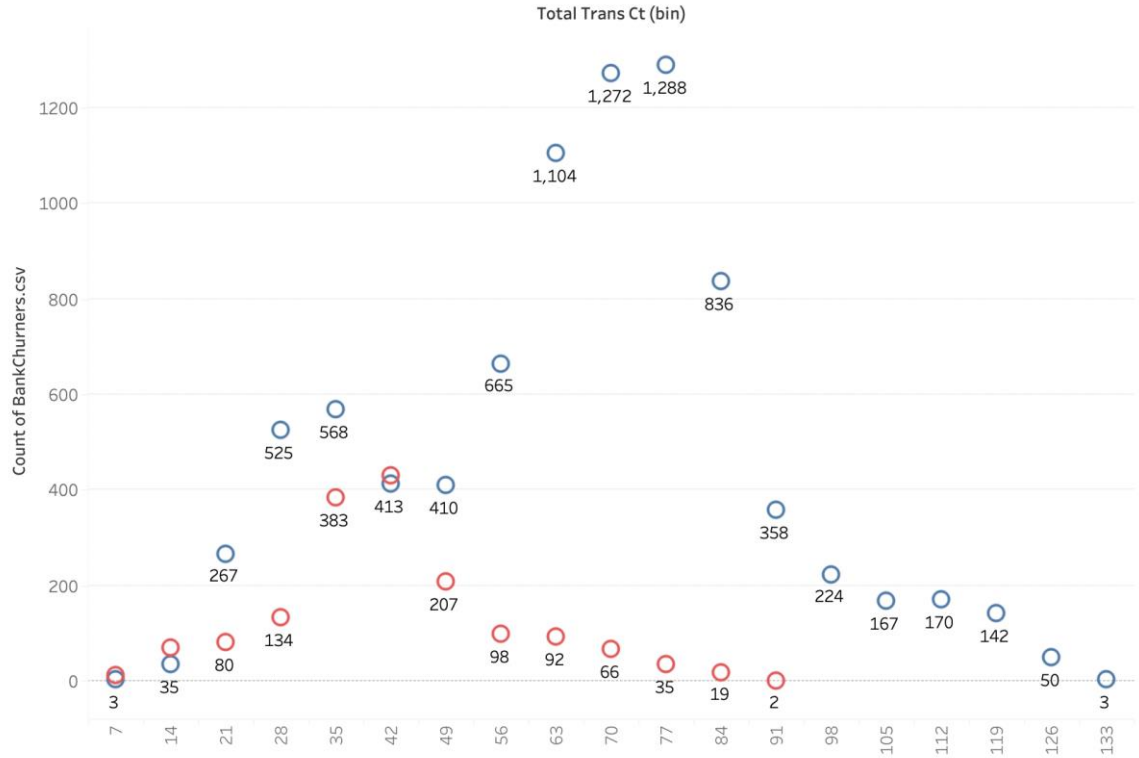

Credit Limit (bin)
Count of BankChurners.csv
2000
1800
1600
1400
1200
1000
800
600
400
200
0
1,027
378
1,806
228
1,023
176
576
144
494
80
354
318
281
266
184
193
148
152
119
115
103
88
87
86
72
83
71
426
91
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
13000
14000
15000
16000
17000
18000
19000
20000
21000
22000
23000
24000
25000
26000
27000
28000
29000
30000
31000
32000
33000
34000

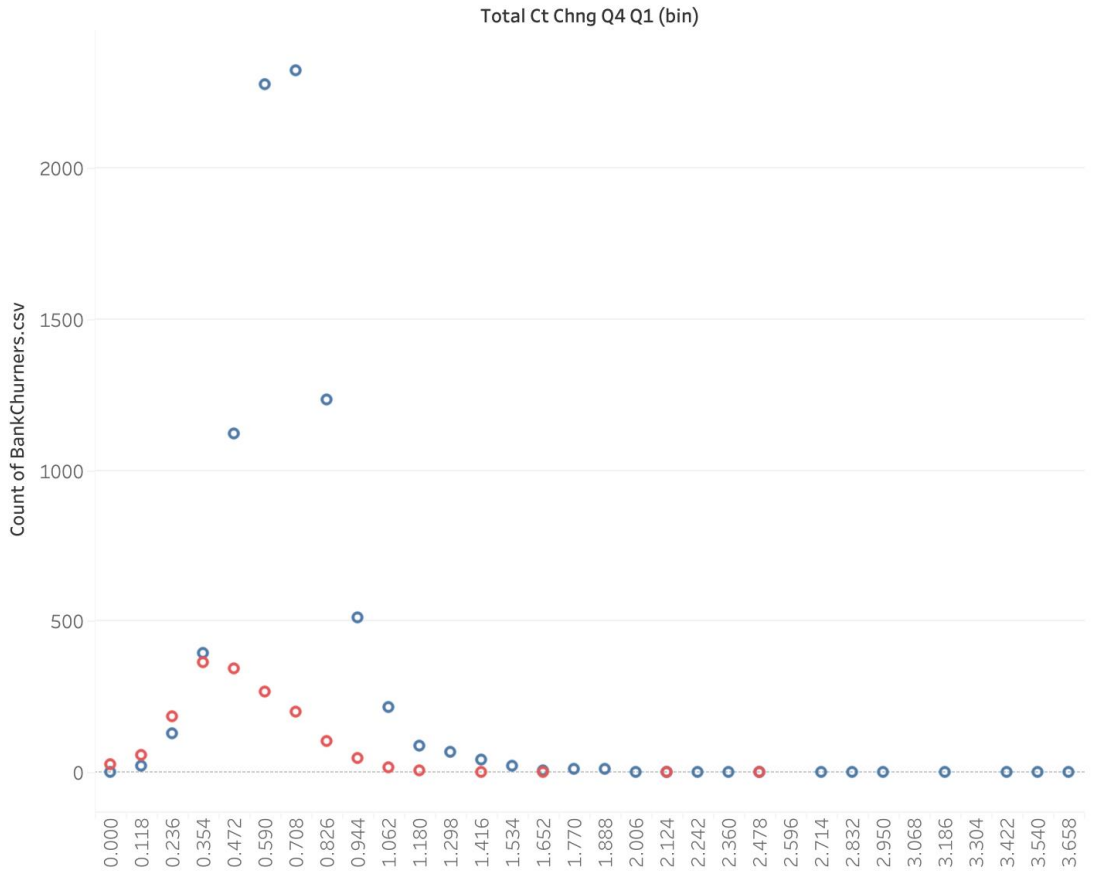

Total Revolving Bal (bin)
Count of BankChurners.csv
2500
2000
1500
1000
500
0
893
1,577
16
27
28
186
271
296
330
334
425
442
465
488
512
500
429
388
350
319
228
213
151
138
368
162
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
1400
1500
1600
1700
1800
1900
2000
2100
2200
2300
2400
2500

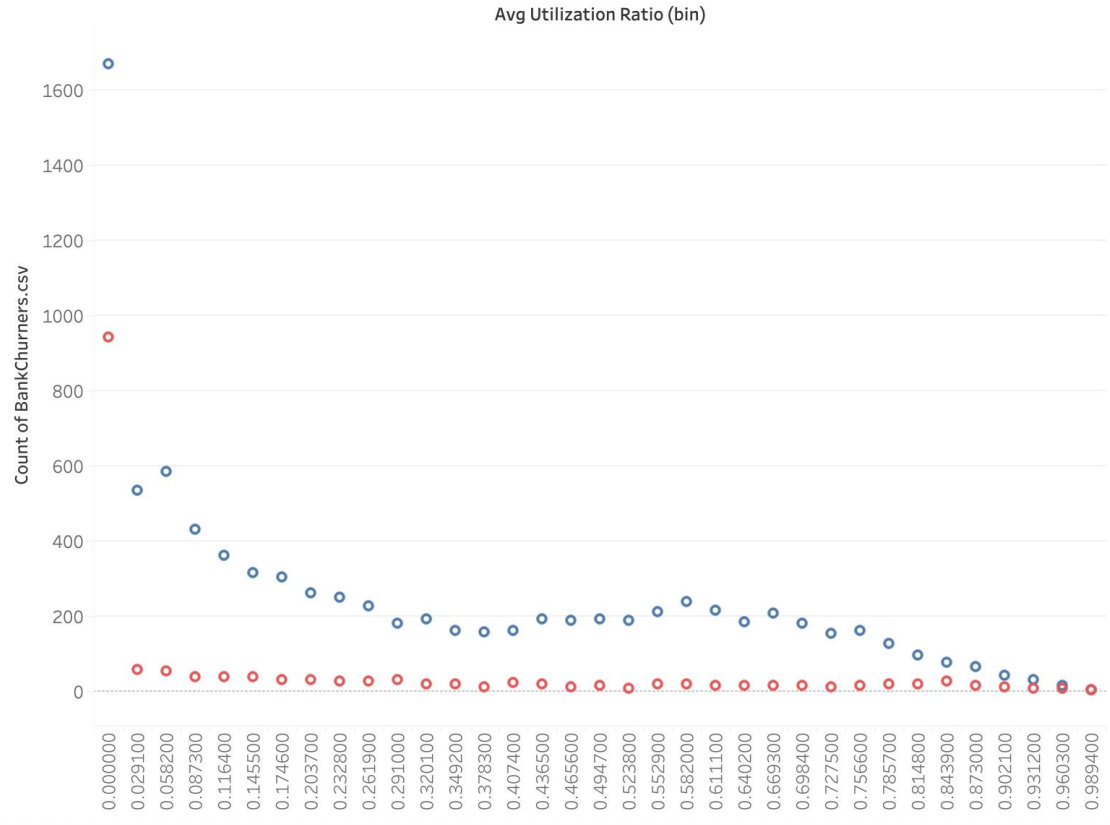

Avg Open To Buy (bin)
Count of BankChurners.csv
2000
1800
1600
1400
1200
1000
800
600
400
200
0
178
1,780
392
1,483
167
713
136
569
103
468
71
366
285
299
232
187
157
172
113
132
112
107
95
85
79
79
64
80
60
60
191
188
0K
1K
2K
3K
4K
5K
6K
7K
8K
9K
10K
11K
12K
13K
14K
15K
16K
17K
18K
19K
20K
21K
22K
23K
24K
25K
26K
27K
28K
29K
30K
31K
32K
33K
34K

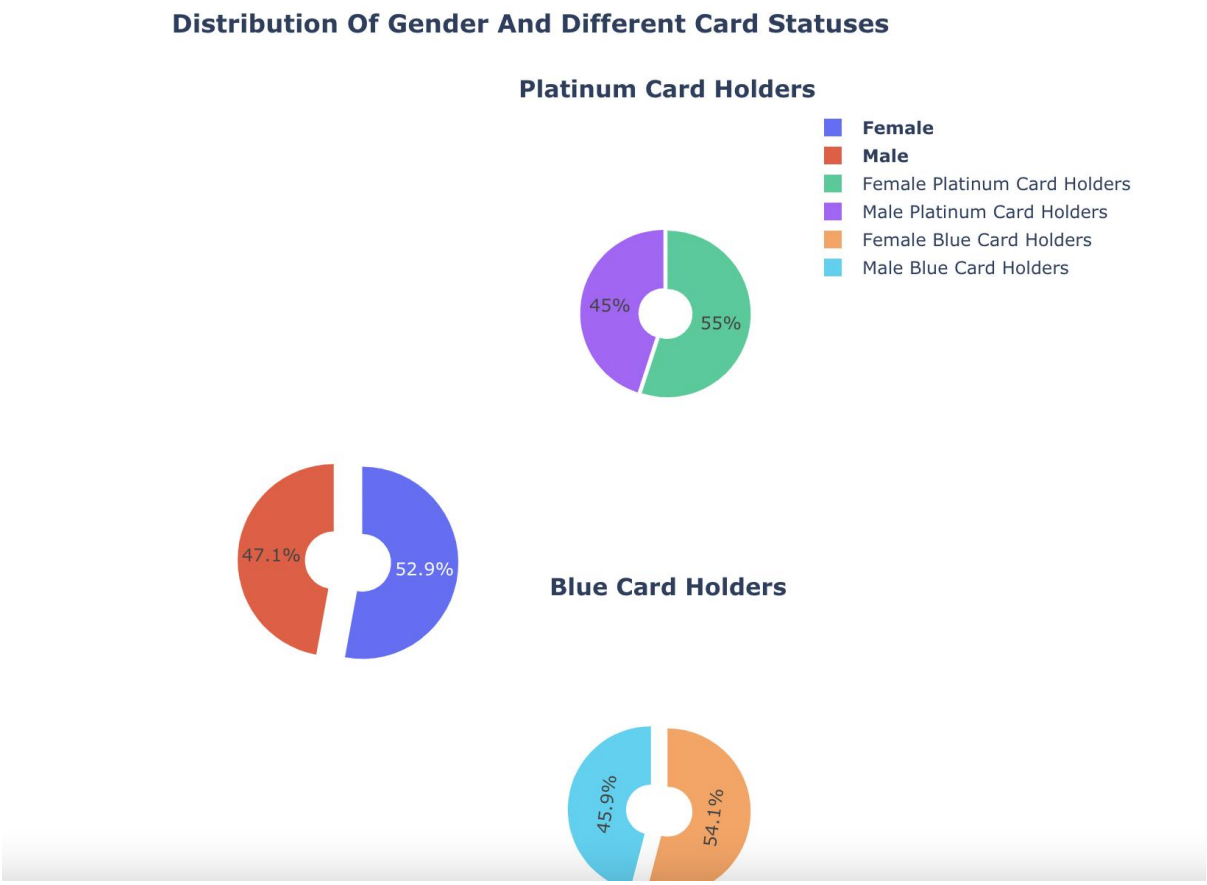

Total Amt Chng Q4 Q1 (bin)
Count of BankChurners.csv
Total Trans Amt (bin)
Count of BankChurners.csv


Total Trans Amt (bin)
Count of BankChurners.csv
1600
1400
1200
1000
800
600
400
200
0
500 1000 1500 2000 2500 3000 3500 4000 4500 5000 5500 6000 6500 7000 7500 8000 8500 9000 9500 10000 10500 11000 11500 12000 12500 13000 13500 14000 14500 15000 15500 16000 16500 17000 17500 18000
141 81 249 524 289 1,544 1,352 381 96 185 110 50 64 9 1 1 9 23 60 105 131 133 121 75 51 24 9 4 1
769 947 503 373 465 893


Total Trans Ct (bin)
Count of BankChurners.csv
1200
1000
800
600
400
200
0
7 14 21 28 35 42 49 56 63 70 77 84 91 98 105 112 119 126 133
1,272 1,288
1,104
836
665
568
525
413 410
383
358
267
224
207
167 170
142
134
98 92 66
80
50
35 19
3 35 2 3


Total Ct Chng Q4 Q1 (bin)
Count of BankChurners.csv
2000
1500
1000
500
0
0.000
0.118
0.236
0.354
0.472
0.590
0.708
0.826
0.944
1.062
1.180
1.298
1.416
1.534
1.652
1.770
1.888
2.006
2.124
2.242
2.360
2.478
2.596
2.714
2.832
2.950
3.068
3.186
3.304
3.422
3.540
3.658


Avg Utilization Ratio (bin)
Count of BankChurners.csv
1600
1400
1200
1000
800
600
400
200
0
0.000000
0.029100
0.058200
0.087300
0.116400
0.145500
0.174600
0.203700
0.232800
0.261900
0.291000
0.320100
0.349200
0.378300
0.407400
0.436500
0.465600
0.494700
0.523800
0.552900
0.582000
0.611100
0.640200
0.669300
0.698400
0.727500
0.756600
0.785700
0.814800
0.843900
0.873000
0.902100
0.931200
0.960300
0.989400



References:

[1]. churn?, W., 2020. *How do you calculate churn rate? Here are 4 formulas.*. [online] Zendesk. Available at: <https://www.zendesk.co.uk/blog/customer-churn-rate/> [Accessed 15 July 2021].

[2]. En.wikipedia.org. 2021. *Customer attrition - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/Customer_attrition> [Accessed 15 July 2021].

[3]. Medium. 2020. *How to Use Pyspark For Your Machine Learning Project*. [online] Available at: <https://towardsdatascience.com/how-to-use-pyspark-for-your-machine-learning-project-19aa138e96ec> [Accessed 15 July 2021].

[4]. Bochet, S., 2021. *Get Started with PySpark and Jupyter Notebook in 3 Minutes | Sicara*. [online] Sicara. Available at: <https://www.sicara.ai/blog/2017-05-02-get-started-pyspark-jupyter-notebook-3-minutes> [Accessed 15 July 2021].

[5]. Jupyter.org. 2021. *Project Jupyter*. [online] Available at: <https://jupyter.org> [Accessed 15 July 2021].

[6]. En.wikipedia.org. 2020. *Data pre-processing - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/Data_pre-processing> [Accessed 16 July 2021].

[7]. Sharma, R., 2019. *19 Reasons You Should Learn Tableau [An In-depth Analysis] | upGrad blog*. [online] upGrad blog. Available at: <https://www.upgrad.com/blog/reasons-to-learn-tableau/#6_Better_Integration_of_Data> [Accessed 16 July 2021].

[8]. Medium. 2020. *Role of StringIndexer and Pipelines in PySpark ML Feature*. [online] Available at: <https://medium.com/@nutanbhogendrasharma/role-of-stringindexer-and-pipelines-in-pyspark-ml-feature-b79085bb8a6c> [Accessed 17 July 2021].

[9]. Campus.datacamp.com. 2021. *Assemble a vector | Python*. [online] Available at: <https://campus.datacamp.com/courses/introduction-to-pyspark/getting-started-with-machine-learning-pipelines?ex=10> [Accessed 17 July 2021].

[10]. Roy, B., 2020. *All about Feature Scaling*. [online] Medium. Available at: <https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35> [Accessed 17 July 2021].

[11]. Machine Learning: What it is and why it matters. (2021). Retrieved 17 July 2021, from https://www.sas.com/en_gb/insights/analytics/machine-learning.html

[12]. 1.10. Decision Trees — scikit-learn 0.24.2 documentation. (2007). Retrieved 17 July 2021, from https://scikit-learn.org/stable/modules/tree.html

[13]. Pant, A. (2019). *Introduction to Logistic Regression*. Medium. Retrieved 17 July 2021, from https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148.