Bank Customer Churn Prediction Analysis

By Vamsi Krishna Midathala

INDEX

Introduction

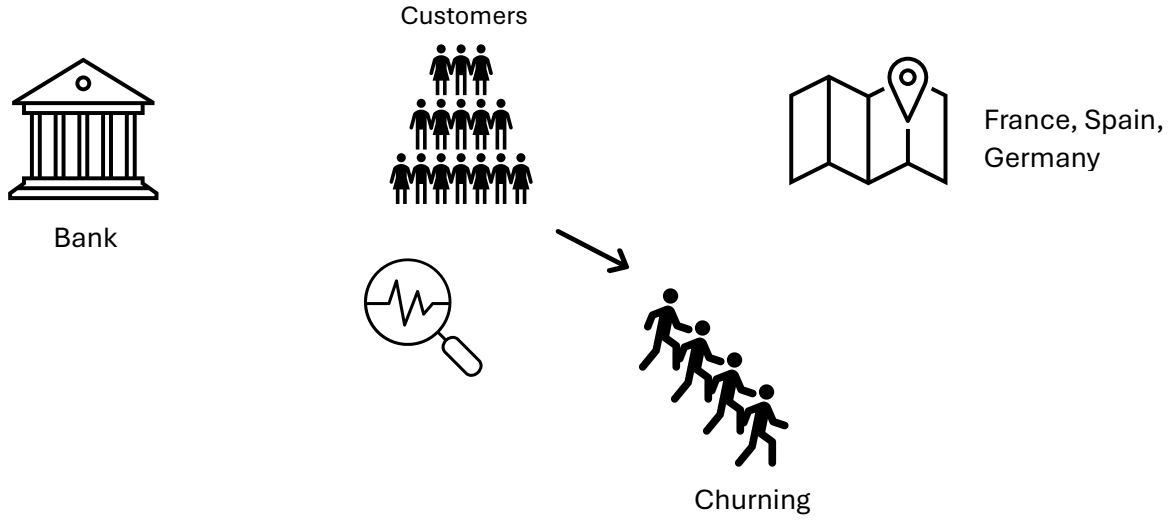
This project is the Capstone Project for the Customer Analytics Case Studies course, at the University of the Pacific, San Francisco, Under the guidance of Professor Arun Saksena. As a master's student in Data Science, I undertook this individual project over 30 days.

The dataset I chose for this project is from Kaggle which is a famous data repository for diverse datasets for free. The dataset downloaded was posted by Saurabh Badole and is licensed and owned by Creative Commons. Creative Commons provides this dataset as a Noncommercial Share alike version 4.0 international (**CC BY-NC-SA 4.0).** This license allows the user to use it for noncommercial purposes as long as credit is given and can be shared under the same terms.

I used Jupyter Notebooks as my environment to develop this project. Major resources for research and inspiration are from Google, YouTube, and Microsoft Edge platforms. Additionally, Sklearn documentation provided useful guidance during the project. Coming to project flow, I chose to follow the CRISP-DM (Cross Industry Standard Process for Data Mining) Methodology. I systematically explored and uncovered valuable insights from the dataset. CRISP-DM structured approach allowed me to effectively guide the project through various phases, from business understanding to all-the-way model evaluation.

1. Business Understanding

1.1 Problem Definition



A Fictitious Bank, serving around 10,000 customers across 3 European countries, faces a crucial challenge with the customer leaving their business which has gradually increased over time. The bank objects to addressing this issue proactively by developing and deploying a predictive model that can forecast and identify at-risk customers before they decide to leave. This leads to the enhancement of techniques and strategies for customer retention with low-cost efficiency.

1.2 Project Goal

The Primary Objective of this project is to deploy a robust and generalized churn prediction model that helps banks to forecast customer churn from both existing and new customer data. This will allow the bank to take pointed actions to reduce churn ratios. In addition, the project also aims to understand and disclose the underlying patterns and statistics about factors influencing customer churn. Specifically, will explore relationships between churn and demographic factors for instance age, Income, and region. Behavioral factors like the count of products registered by customers, and the number of years of being a customer in the bank. To achieve these objectives, I will employ a combination of three machine learning algorithms to predict the target variable. Those are a linear model, a Non-linear model, and an Ensemble method to optimize the prediction rate.

2. Data Understanding

Let's Dive into the dataset after importing it into the Jupyter Notebooks.

2.1 Statistical Analysis

Import Pandas and NumPy libraries to explore the dataset. The dataset contains 10,000 rows and 14 columns. Each row represented a unique customer ID and the other following 13 columns are features. Fig 1.1 image shows the first 5 rows in the dataset.

Columns and their descriptions.

Column name	**Description**
RowNumber	The sequential number assigned to each row
CustomerId	unique Identifier for each customer
Surname	the surname of the customer
CreditScore	credit score of the customer
Geography	geographical Location of the customer
Gender	gender of the customer
Age	age of the customer
Tenure	number of years the customer has been with the bank
Balance	account balance of the customer
NumOfProducts	number of bank products the customer has
HasCrCard	Indicates whether the customer has a credit card (binary: yes/no)
IsActiveMember	Indicates whether the customer is an active member (binary: yes/no)
EstimatedSalary	estimated salary of the customer
Exited	Indicates whether the customer has exited the bank or not (binary: yes/no)

In the Next step understand the feature data types, and its non-null count shown in Fig 1.2 so that we can validate types of the available for our further analysis.

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10

Fig 1.1

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   RowNumber        10000 non-null  int64
 1   CustomerId       10000 non-null  int64
 2   Surname          10000 non-null  object
 3   CreditScore      10000 non-null  int64
 4   Geography        10000 non-null  object
 5   Gender           10000 non-null  object
 6   Age              10000 non-null  int64
 7   Tenure           10000 non-null  int64
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64
 10  HasCrCard        10000 non-null  int64
 11  IsActiveMember   10000 non-null  int64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

Fig 1.2

As observed, there are no null values in the columns, and datatypes are correctly indicated.
In addition, as stated above each row represents a single customer from the bank having duplicates will contribute to overfitting, addressing this issue is very important. Using pandas function duplicated () to check duplicates. There are no duplicates in the data frame.

We can validate whether the data frame has duplicate values with the simple if else logic shown in Fig 1.3

```
# uniqueness of data available

a = dfm['CustomerId'].nunique()
b = dfm.shape[0]

if a == b:
    print('Each row in the dataframe represents individual customers. Proceed further :)')
else:
    print(' There are duplicate customers in the datase. Please Check!!')
```

Fig 1.3

We can observe that the index in the data frame is the same as the row number. So, setting Row Number as the index for our data frame brings our column count to 13 from 14.

Now we glance at the statistics of the data frame using the pandas function describe ().

	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000
mean	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.203700
std	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769
min	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000
25%	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000
50%	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000
75%	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000
max	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000

Fig 1.4

We can look at basic statistics for each column such as mean, standard deviation, and quartiles. This summary of statistics can be a good starting point for our churn analysis. Now let's interpret observations, The Mean value Target variable Exited is 0.20 which indicates that around 20% of customers are churned. Most customers are using 1 or 2 products from the bank and 75% of them are active members enrolled in the service and have credit cards with the bank. The average number of years customers stay with the bank is 5. The bank has customers whose ages range from 18 to 92. The standard deviation of the estimated salary is greater than 25% quartile.

By default, the describe function in Pandas only displays numerical columns. To display statistics of categorical features we input (include = ‘object) in the describe function.

	Surname	Geography	Gender
count	10000	10000	10000
unique	2933	3	2
top	Smith	France	Male
freq	32	5014	5457

Fig 1.5

In the data frame, we have more customers from France and mostly male.

2.2 Feature Distributions and Visualization

We now see first numerical feature distributions using histograms, then Categorical features followed by target variable distribution visualizations.

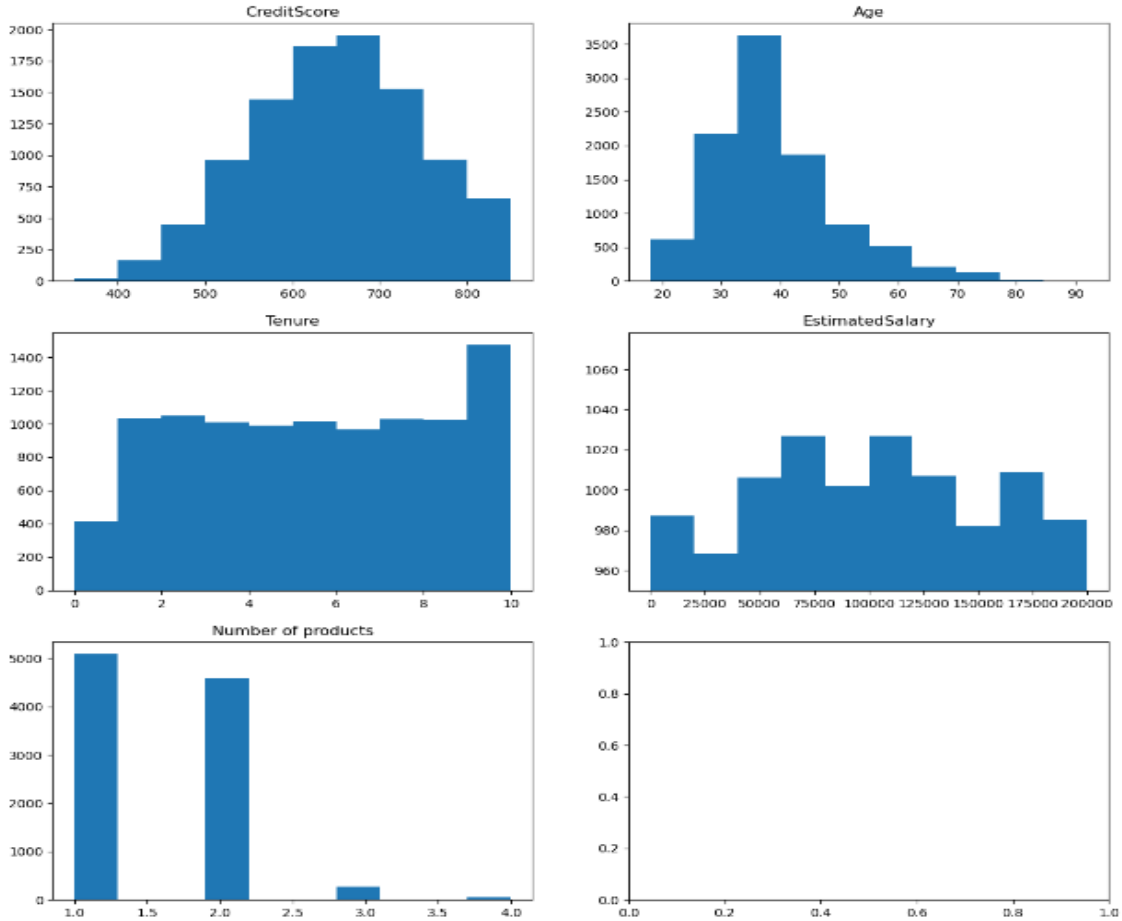
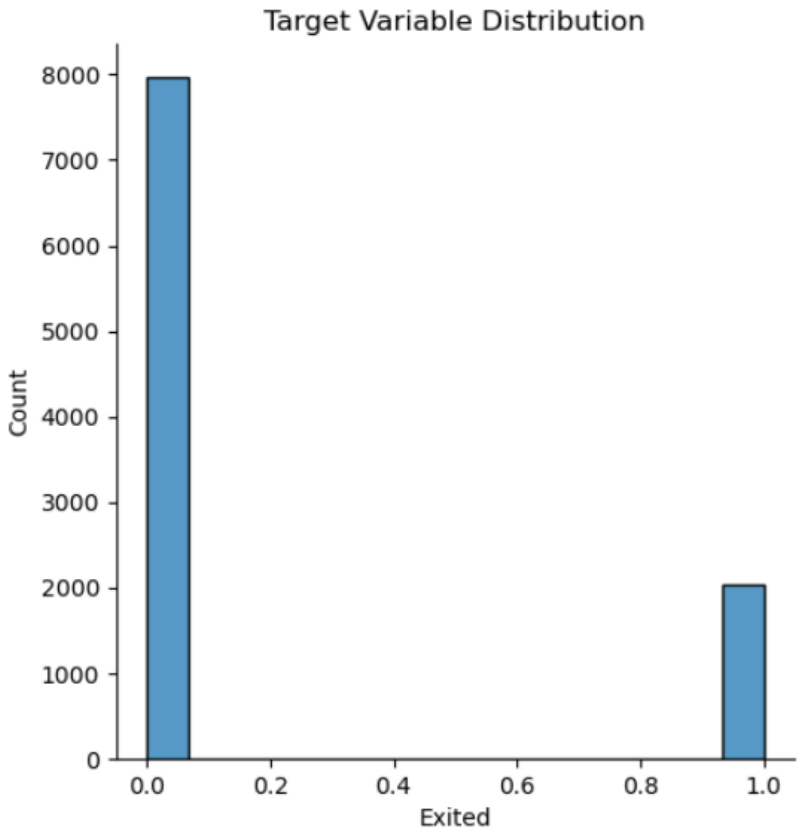


Fig 1.6

From the above plots of histograms,

The credit score is right skewed, indicating most of the values are in the lower range having lower credit scores. Age is also right-skewed, which is considered as normal as most of the bank customers would be young people. Tenure follows a uniform distribution. The estimated salary is a bimodal distribution with two peaks at 75,000 and 125,000 indicating the dataset has most of the customers have near these estimated salaries. As observed in Fig 1.4 most of the bank customers use at least 1 product showing unimodal distribution.



Target variable distribution Fig 1.7

The minority class of the dataset is customers Exited from the bank, which has 20.37%.

0 7963

1 2037

Name: Exited, type: int64

The above Distribution plot for a target variable is heavily skewed towards 0, meaning more customers are not exited in the dataset. This suggests that there is a class imbalance that needs to be addressed or handled while modeling.

Let’s see the distribution of categorical columns of our data.

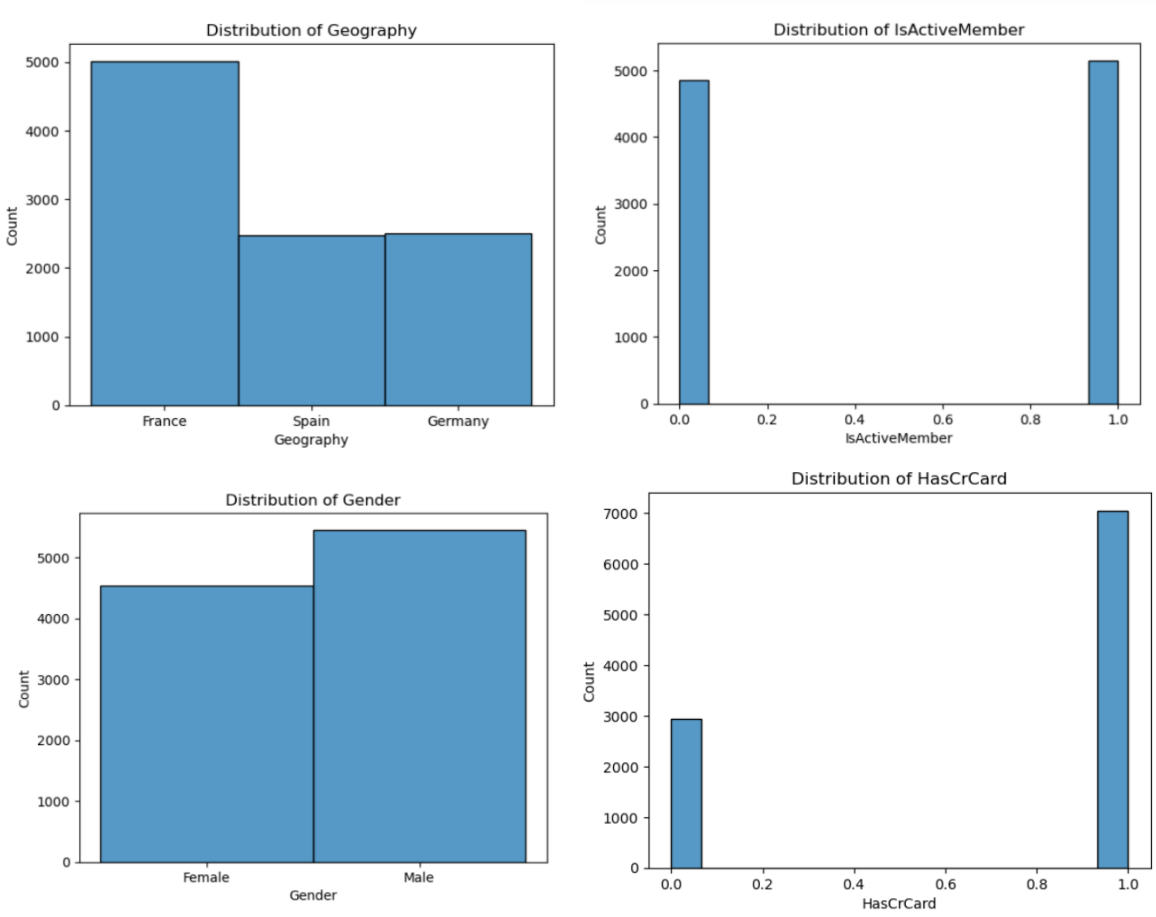


Fig 1.8

Columns HasCrCard and IsActiveMember are considered categorical data columns because of the available binary data.

From the above distribution plots, it is observed that France has the highest number, two times the number of customers to the bank when compared to Germany and Spain. Gender distribution is almost normal having a difference of around 1000 instances between males and females.

Observing IsActiveMembers distribution will request a proper description of that column. For this project, we assume that customers who are not enrolled in any product and have not made any transaction for the past 90 days and customers who are exited are considered as Not active members denoted with number 0. Active customers were denoted with the number 1. Interestingly some customers are active and not active and have almost has same number of instances in the dataset with a minor difference of around 800 which is less than 1%.

70% of Customers are using credit card services, high in number when compared who are not using any credit card services with the bank. These distribution insights can uncover interesting things in the next steps of EDA.

2.3 Data Validations

In this section, we see data validations of column distributions writing a simple code in Python.

Categorical distributions validation to cross-check whether the distributions visualized are absolute or not.

```
# Data validations

catcols09 = ['Gender', 'Geography','HasCrCard', 'IsActiveMember']

for t in catcols09:
    count = dfm[t].value_counts()

    unq_valper = count / len(dfm) * 100
    print(f'Percentage of unique values of column {t}')
    print(unq_valper)
    print('-----------------------------')
```

```
Percentage of unique values of column Gender
Male      54.57
Female    45.43
Name: Gender, dtype: float64
-----------------------------
Percentage of unique values of column Geography
France     50.14
Germany    25.09
Spain      24.77
Name: Geography, dtype: float64
-----------------------------
Percentage of unique values of column HasCrCard
1    70.55
0    29.45
Name: HasCrCard, dtype: float64
-----------------------------
Percentage of unique values of column IsActiveMember
1    51.51
0    48.49
Name: IsActiveMember, dtype: float64
-----------------------------
```

Fig 1.9

These percentages perfectly match with the data visualized above.

For Numerical data type columns, it is hard to see every distribution percentage because of the large number of unique values.so, we print only the top 3 highest percentage values for numerical columns if those match with the peaks of the distribution data will be validated.

```
# Data Validations
numcols09 = ['CreditScore', 'Age','Tenure', 'NumOfProducts','Balance','EstimatedSalary' ]

for m in numcols09:
    count = dfm[m].value_counts()

    unq_valper = (count / len(dfm) * 100 ).nlargest(3)
    print(f'Percentage of unique values of column {m}')
    print(unq_valper)
    print('------------------------------')
```

```
Percentage of unique values of column CreditScore
850    2.33
678    0.63
655    0.54
Name: CreditScore, dtype: float64
------------------------------
Percentage of unique values of column Age
37    4.78
38    4.77
35    4.74
Name: Age, dtype: float64
------------------------------
Percentage of unique values of column Tenure
2    10.48
1    10.35
7    10.28
Name: Tenure, dtype: float64
------------------------------
Percentage of unique values of column NumOfProducts
1    50.84
2    45.90
3     2.66
Name: NumOfProducts, dtype: float64
------------------------------
Percentage of unique values of column Balance
0.00         36.17
130170.82     0.02
105473.74     0.02
Name: Balance, dtype: float64
------------------------------
Percentage of unique values of column EstimatedSalary
24924.92     0.02
101348.88    0.01
55313.44     0.01
Name: EstimatedSalary, dtype: float64
------------------------------
```

Fig 1.10

2.4 Exploratory Data Analysis

In this Section, we will perform EDA, Exploratory Data Analysis is supposed to give an initial understanding of the data and to uncover valuable information that could lead or assist in subsequent modeling and analytical steps. In EDA, we examine the configuration, trends, and correlation of the data using visualization plots and summary statistics. These steps will help us identify important characteristics such as the distribution of variables, missing values, and outliers that might impact our analysis. Further, it visually inspects the correlation and trend for a better understanding of relationships between features and locates features that may provide substantial contributions to the research question. These various insights obtained during EDA formed as basis of informed decision-making, directing toward the right strategy regarding data preprocessing and feature engineering.

The initial Exploration of data was to check whether any missing/na values were present or not. If Present framing the strategies in advance of how to impute those values is important.

In the observation, we found that we have no missing or na values in our dataset as it is delivered clean. We proceed to the next step of correlation analysis.

2.4.1 Correlation Analysis

For the correlation analysis, we start using a pair plot from the seaborn library to glance at the whole picture of relations between columns.

Seaborn and matplotlib are the visualization libraries used in this project which are efficient to graph almost any type of data.

A pair plot is a scatter plot of all pairs of variables in the dataset.
Looking at the target variable, which says whether the customer has left the bank or not, some interesting observations can be drawn:
Age: More customers have left the bank in the 25-35 years old relatively younger age group. This may suggest that it is the relatively younger customers who will leave for better deals or because their needs are not being met as well by a particular bank.
Estimated Salary: There is no pattern established between estimated salary and the churn rate, hence probably the level of income is not important when considering customer churn.
Balance: Customers with higher account balances do seem less likely to leave the bank. That could be because richer customers are more satisfied with the institution's services, or, for whatever financial reason, their propensity to leave is lower.

Relations between features:

Balance and CreditScore: The balance increases with credit score, the higher the credit score is the larger the amount one can keep in balance. There is a positive correlation between them.
Age and balance: there is a positive correlation between age and financial balance, indicating the greater the age of a customer, the higher the balance is likely to be.
Tenure and balance: the variable of tenure is directly proportional to the balance in the customer account. This will imply that the longer the customer stays with the bank, the higher the account balance is bound to be. In all, the pair plot provides a full view of the relationship between the different variables involved in the dataset. This plot will also be used to emphasize any potentially improving areas in customer retention and to develop focused and segmented marketing campaigns.

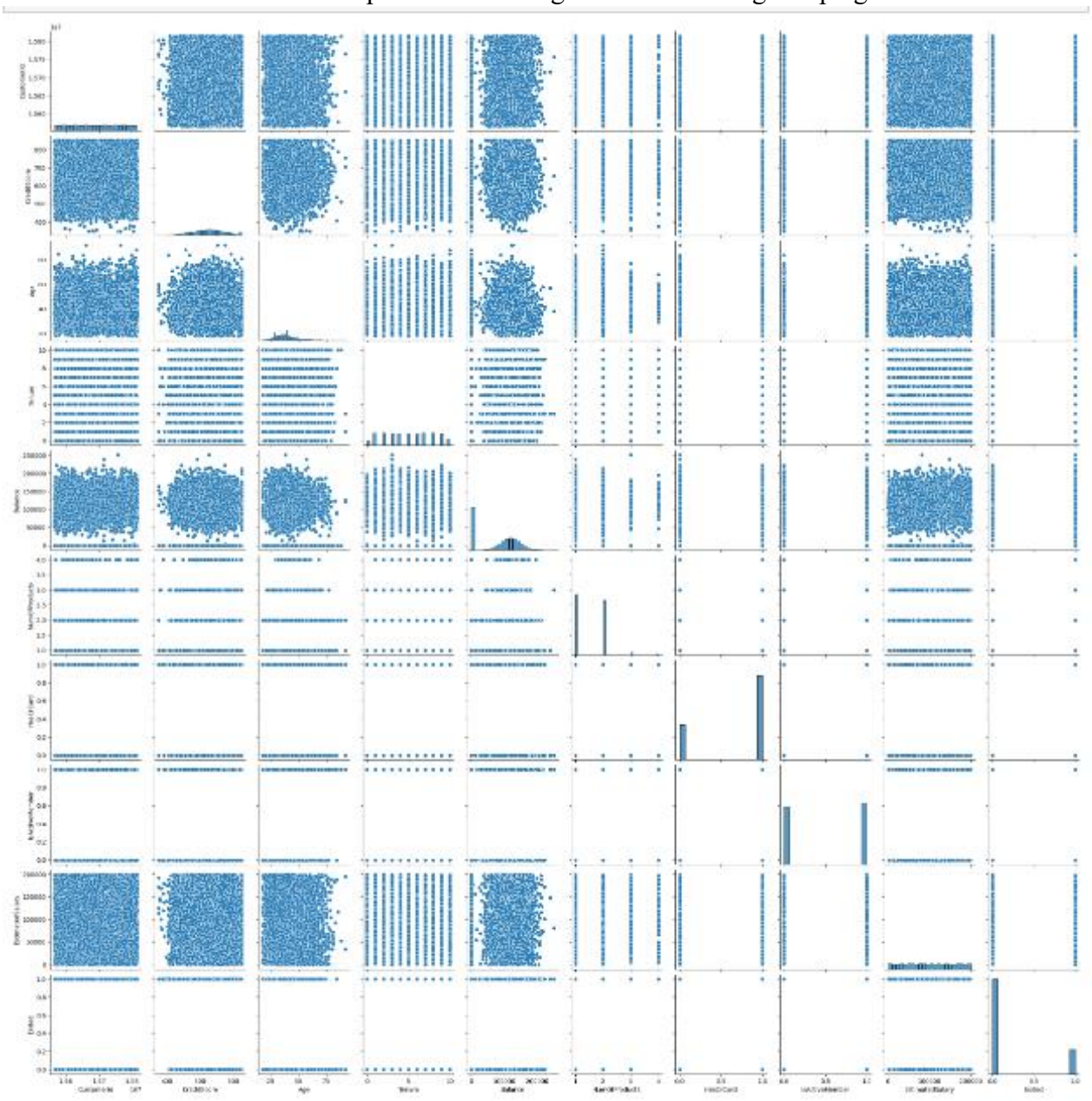


Fig 1.11

In the Next step, we will see the correlation coefficient matrix between features using the Spearman technique.

Spearman Correlation:

Spearman's rank correlation is the measure of the strength and direction of the monotonic relationship between two continuous variables. Concerning this, the attributes are ranked or placed in order of their preference. It is represented by the symbol "rho" denoted as *p* and may range between -1 and +1. A positive value of *p* indicates that there is a positive association between the two variables, and a negative value of *p* indicates a negative association. A *p*-value of 0 indicates that there is no association between the two variables.

Spearman correlation can be calculated manually using the formula below.

Spearman's Correlation formula

p = correlation coefficient

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$$

di = difference in the ranks given to the two variables

n = total number of observations

	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
CustomerId	1.000000	0.005967	0.008775	-0.015072	-0.013932	0.019297	-0.014023	0.001682	0.015246	-0.006264
CreditScore	0.005967	1.000000	-0.007974	0.001133	0.005687	0.012568	-0.003802	0.024262	0.001237	-0.023289
Age	0.008775	-0.007974	1.000000	-0.010405	0.033304	-0.058566	-0.015278	0.039839	-0.002431	0.323968
Tenure	-0.015072	0.001133	-0.010405	1.000000	-0.009513	0.012908	0.022354	-0.028673	0.007778	-0.013978
Balance	-0.013932	0.005687	0.033304	-0.009513	1.000000	-0.316627	-0.009835	-0.011497	0.011778	0.111110
NumOfProducts	0.019297	0.012568	-0.058566	0.012908	-0.316627	1.000000	0.003859	0.016292	0.012570	-0.125282
HasCrCard	-0.014023	-0.003802	-0.015278	0.022354	-0.009835	0.003859	1.000000	-0.011866	-0.010041	-0.007138
IsActiveMember	0.001682	0.024262	0.039839	-0.028673	-0.011497	0.016292	-0.011866	1.000000	-0.011469	-0.156128
EstimatedSalary	0.015246	0.001237	-0.002431	0.007778	0.011778	0.012570	-0.010041	-0.011469	1.000000	0.012081
Exited	-0.006264	-0.023289	0.323968	-0.013978	0.111110	-0.125282	-0.007138	-0.156128	0.012081	1.000000

Fig 1.12

One of the advantages of using Spearman correlation is it does not assume a specific distribution and is non-parametric.

To generate the above table, we can use Panda's library correlation function. Looks like DataFrame.corr(). What is observed from the above correlation table is that there is a strong positive correlation between the variables Age and Exited, indicating the significance this significance can lead to age-segmented marketing and can be used while building our predictive model. Variable No of products is negatively correlated to the target variable with the value of -0.125 meaning that active members are less likely to exit. Balance and No of products variables have a negative correlation of 0-0.3166, suggesting that customers with higher balances may be registered in fewer products.

2.4.3 Relations between Features and Target Variables

Now, we glance at a visualization of a heatmap showing the correlation between variables.

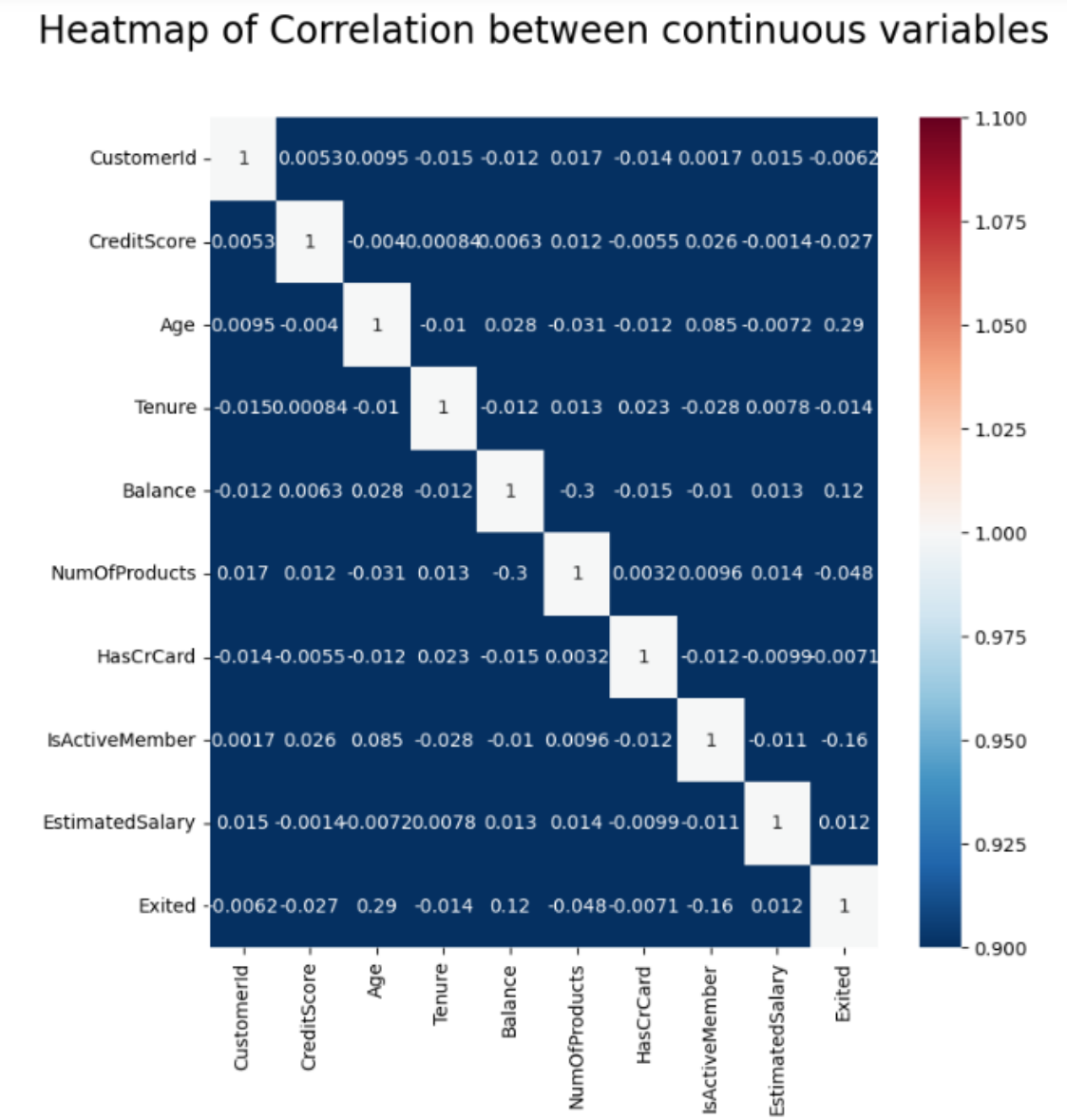


Fig 1.13

As observed in the above heatmap there are some weak correlations, around zero. That means they do not tend to be strongly monotonically related. For example, Credit Score hardly shows any correlation with any of the features and hence is relatively independent. Is active member variable shows weak correlations with most variables but has a negative correlation with the target variable, making it potentially useful in churn prediction.

Overall, the heatmap reflects low correlations of most features amongst themselves and also with the target variable, which is Exited. The strongest relationships, with Age, IsActiveMember, and Balance, are those that might be important in the prediction model.

We now visualize the categorical variable relations with the target variable as hue using a box plot.

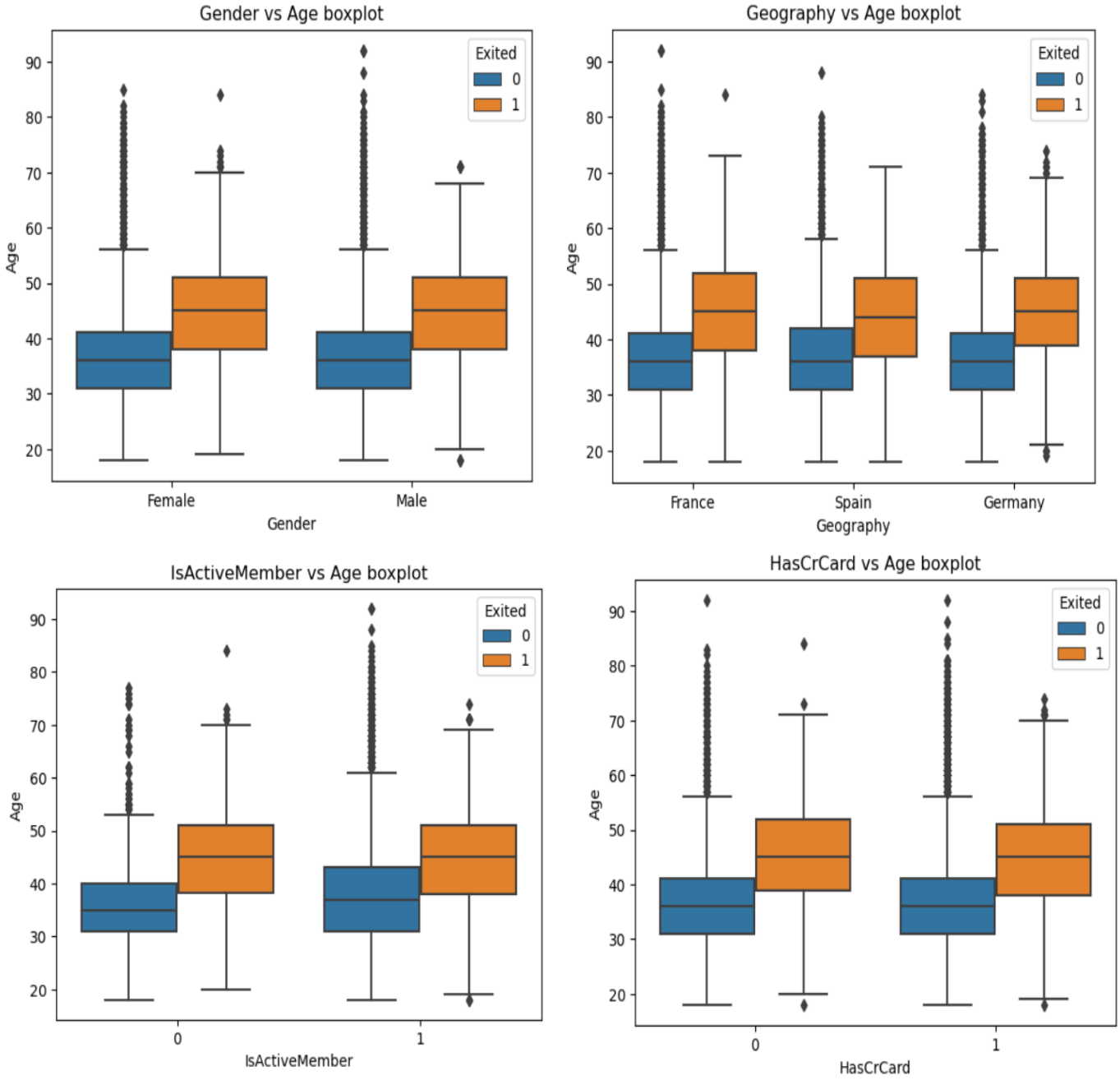


Fig 1.14

These box plots describe how the categorical variables of Gender, Geography, IsActiveMember, and HasCrCard are related to Age when the customer has exited or not since age is the most positively correlated continuous variable to the target variable.

Gender vs Age: Women who exited also tend to be older on average than those who have stayed. The median age of women customers who have exited catches upwards. For Men the influence of age seems a bit weak, the age difference between those who left and those who remained is less pronounced but still slightly higher in clients who left. Observed women customers may be more likely to churn at an older age compared to a younger age.

Geography vs Age: Another feature that distinguishes Germany and France is that their exited customers are generally older than their customers who stayed with the bank. In Germany, age is still strongly related to churning, which again indicates the difference in the behavior of customers in different regions and perhaps suggests a need for a differential approach toward their engagement.

IsActiveMember vs Age: Inactive customers are older on average for those who have exited compared to those who stayed, older and non-active customers have more churn probability, which may indicate that the risk of not being active could accumulate over time.

HasCrCard vs Age: Clients with no credit card denoted with 0 are slightly older on average than those that stay. There seems to be among customers who possess a credit card a noticeable yet smaller age difference between those who left and those remaining. Understanding that could involve an older customer who does not use a credit card and has a high churn rate, indicating an uninterested or unattached group from the bank services.

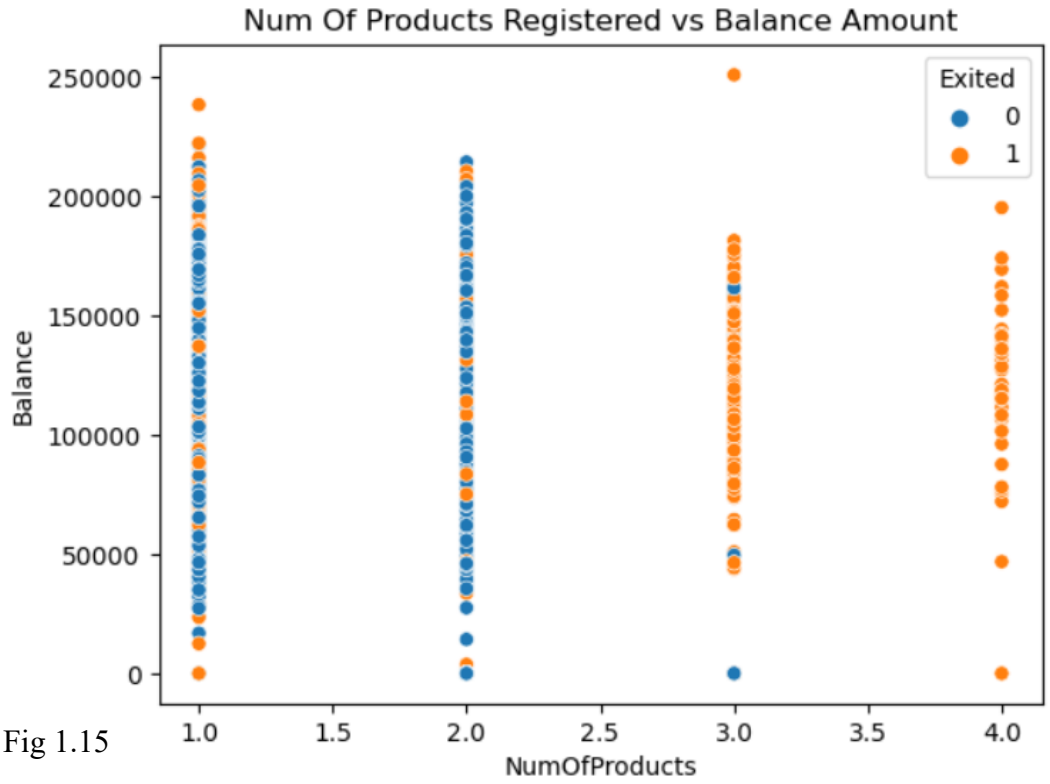


Fig 1.15

Validating visualizations by looking at actual numbers.

customers exited from the Bank having an average balance of 72745.0 $

customers stayed loyal with the bank having an average balance of 91109.0 $

Customers using Number of products Not exited

2 4242

1 3675

3 46

Name: NumOfProducts, dtype: int64

Customers using Number of products Exited

1 1409

2 348

3 220

4 60

Name: NumOfProducts, dtype: int64

3. Data Preparation

3.1 Data Quality

Data Preparation is the step to convert raw data which is to be analyzed. Primary activities undertaken in this stage are a group of tasks like collection, cleaning, labeling, and preprocessing the raw data into consumable form by machine learning algorithms or other procedures for exploration and visualization.

In this Project, the dataset we downloaded from Kaggle doesn't have any missing or Na values to impute. Proceeding further with removing variables 'CustomerId' and 'Surname' in consideration of Both Data Governance and Data leakage.

Data Governance informs that information used to identify individuals either needs to be transformed or removed as an input variable for the machine learning model.

Data leakage means any information that is directly linked to the output and identifies the output should be given as input. If given model may produce a high accuracy score but low performance in the long term.

The data set is now loaded into a new variable. 'cleaned_dfm'

3.2 Data Transformation

Data Transformation is the process of converting and organizing variables that can be understood by machine algorithms and supporting the decision-making process. One of the popular methods to convert variables is one hot encoding for categorical data and label encoding for continuous data,
In this project, we are going to use a robust method of transformation that is from the pandas library under the function named 'df.get_dummies()'.
Using Pandas get dummies converts each variable in as many 0 or 1 variables as there are different values. Columns in the output are named after a value. If the input is a data frame, the name of the original variable is prepended to the value.
Setting the parameter drop first to 'True' removes first-level dummies which means it creates only one column when a column has two unique variables. In regular one hot encoding, it creates a new column for every unique value in the variable that may cause dimensionality issues for a predictive model. Drop first reduces the risk of the model falling into dimensionality issues creating few features that can be understood by machines.

```
# encoding categorical features
df_encoded= pd.get_dummies(cleaned_dfm, drop_first = True)
df_encoded.head()
```

reditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Geography_Germany	Geography_Spain	Gender_Male
619	42	2	0.00	1	1	1	101348.88	1	0	0	0
608	41	1	83807.86	1	0	1	112542.58	0	0	1	0
502	42	8	159660.80	3	1	0	113931.57	1	0	0	0
699	39	1	0.00	2	0	0	93826.63	0	0	0	0
850	43	2	125510.82	1	1	1	79084.10	0	0	1	0

Fig 2.1

In Fig 2.1 you can observe that the get dummies function created only one column for gender rather than creating two and the same with the geography variable because we provided the parameter drop first.
Here, in the Gender male column assumes the numbers 0 is Female and 1 means Male.

4. Data Modelling

4.1 Build Model

We build predictive models on a famous machine-learning library in Python called Scikit Learn in short it is also called Sklearn. It is an open-source machine learning library that is free to use and built on NumPy, SciPy, and Matplotlib. It features various data analysis techniques like classification, regression, and clustering algorithms.

Next importing essential libraries to perform a baseline model on our dataset.

```
# modelling
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

Fig 3.1

Divide data frame into x and y, x consists of variables that are inputs for the algorithm and y consists of only the target variable which is predicted by the algorithm.
Train, test split library is used to slice data into random train and test subsets.
For this project, we split the dataset into a **70:30** ratio to train and test sets considering class imbalance in the target variable provided stratify parameter with the output variable y.

As a next step, these train and test datasets need to be normalized using techniques like standard scaler or min max scaler. For this project chose to perform a standard scaler as a Data normalization technique removing the mean and scaling to unit variance.

The standard scaler of x is calculated as

$z = (x - u) / s$

u - mean of the sample
s - standard deviation

After initializing the standard scaler into a pipeline or to a new variable, use the fit and fit transform method to normalize data. We only use the fit method on input test data and fit transform the input train data. Standardizing target variables is not preferred and provides poor model performance scores.

For this project we chose a linear model, a nonlinear model, and an ensemble algorithm we try to explain those in brief and why they were chosen.

Logistic regression:
Logistic regression is the machine learning regression-based methodology that is used to foresee the binary classes. Used for classification problems. The outcome or target variable is dichotomous. Dichotomous means there are only two possible classes. It computes the probability of an event occurrence. One of the advantages of logistic regression is it can handle dense and sparse input.

It is a special type of regression where the target variable happens to be categorical. The log of odds against forms the dependent variable. Logistic regression predicts the occurrence probability of a binary event through a logit function.

Sigmoid Function:

$$p = \frac{1}{1 + e^{-y}}$$

Fig 3.2

Decision Tree:
Decision Tree is a non-linear model. It represents one of the nonparametric supervised learning algorithms, whereby it finds its applications in classifications in classification and regression problems. A decision tree is a tree-like diagram holding a hierarchical representation comprising a root node, branches, internal nodes, and leaf nodes.

The decision tree starts with some form of root node, which has no incoming branches. The outgoing branches are from the root node, which has no incoming branches. The outgoing branches from the root node in turn get fed into internal nodes, also referred to as decision nodes. Those two kinds of nodes are from homogenous subsets, which are then represented by leaf nodes, or so-called terminal nodes. Decision trees have a number of characteristics they can handle missing values and different types of data types.

Random Forest:
Random Forest algorithm, trademarked by Leo Breiman and Adele Cutler, constitutes one of the most important ones in machine learning. It does so by taking the output from many decision trees to produce one single result. Its simplicity and flexibility increased the growth in it, since it handles both classification and regression problems, in that regard, this approach involves drawing a random sample with replacement that is, the same data point may get picked many times from the data in the training set. From this, once multiple samples of the data are available, each independently trained, and depending on the task at hand, whether regression or classification, the average or majority of those predictions give a better estimate. This typically can be done to reduce the variance of a noisy dataset.

The Random Forest algorithm extends the bagging method by incorporating the principles of both bagging and feature randomness in constructing an uncorrelated forest of decision trees. Feature randomness also referred to as feature bagging or 'the random subspace method', creates for each sample a random subset of features that are then used to ensure low correlation among the produced decision trees. An important distinction here compared with decision trees is the introduction of this feature randomness. While decision trees take into consideration all possible feature splits, random forests select only a subset of those features.

```
# Baseline model performance


algorithms = {"logistic Regression": LogisticRegression(),
       "Decision Tree": DecisionTreeClassifier(),
       "Random Forest Classifier": RandomForestClassifier()}


for key , algo in algorithms.items():

    #fits the data in to the model
    algo.fit(X_train1, y_train1)

    #Prediction
    prediction = algo.predict(X_test1)

    #accuracy
    acc = accuracy_score(y_test1, prediction)
    print(f'Accuracy of {key} {acc:.2f} ')

    #Classification Report
    report = classification_report(y_test1, prediction,zero_division=0)
    print(f'Classification Report of  {key}')
    print(report)


    #confusion matrix
    conf_mtrx = confusion_matrix(y_test1, prediction)
    print(f"Confusion Matrix {key}:")
    print(conf_mtrx)
    print('-------------------------------------------------------------')
```

Fig 3.3

4.2 Test Score and Interpretation

From the above code, snipped Fig 3.3 performed model fit and prediction for all the three chosen algorithms using for loop function. This loop function fits data to the model initializes prediction prints evaluation metrics using a classification report and prints the confusion matrix as output for the baseline models.

Let's have a look at the output of our model and frame to see if there are any possibilities to develop.

```
Accuracy of logistic Regression 0.81
Classification Report of  logistic Regression
              precision    recall  f1-score   support

           0       0.83      0.97      0.89      2389
           1       0.63      0.20      0.30       611

    accuracy                           0.81      3000
   macro avg       0.73      0.58      0.60      3000
weighted avg       0.79      0.81      0.77      3000

Confusion Matrix logistic Regression:
[[2318   71]
 [ 491  120]]
------------------------------------------------------------
Accuracy of Decision Tree 0.79
Classification Report of  Decision Tree
              precision    recall  f1-score   support

           0       0.87      0.87      0.87      2389
           1       0.49      0.50      0.50       611

    accuracy                           0.79      3000
   macro avg       0.68      0.69      0.68      3000
weighted avg       0.80      0.79      0.79      3000

Confusion Matrix Decision Tree:
[[2075  314]
 [ 304  307]]
------------------------------------------------------------
Accuracy of Random Forest Classifier 0.86
Classification Report of  Random Forest Classifier
              precision    recall  f1-score   support

           0       0.87      0.96      0.92      2389
           1       0.76      0.46      0.57       611

    accuracy                           0.86      3000
   macro avg       0.82      0.71      0.74      3000
weighted avg       0.85      0.86      0.85      3000

Confusion Matrix Random Forest Classifier:
[[2303   86]
 [ 332  279]]
```

Fig 3.4

Here is a summary of the classifier report in Fig 3.4

Logistic regression yielded 81% percent accuracy; for class 0, the precision was 83% and the recall was 97%, but class 1 had a very poor recall of 20% and an f1 score of 30% indicating the model performed poorly for the minority class.

The decision tree had 79% accuracy, balanced by the precision and recall in both classes, 87% for class 0 and 50% for class 1, although with somewhat low accuracy compared to logistic regression.

Random Forest classifier performed the best with an overall accuracy of 86%, with high precision and recall for class 0 at 87% and 96% respectively, and reasonably balanced precision and recall for class 1 at 76% and 46% respectively among other models.

The random forest model outperforms the best and yields the highest performance with a higher balanced class recall, although class 1 recall can be further improved by addressing the class imbalance in the target variable. Less number of data point support is observed in all the models.

5. Model Evaluation

5.1 Cross Validation

Cross-validation in general is a resampling procedure that provides an evaluation of a machine learning model on limited data.
We are running three different types of models for this project, there is only a test set is given for all three models to fit and provide model accuracy. Cross-validation helps to evaluate the model using a given number of test sets to provide a confident accuracy as output.
This function takes only one parameter, k, which represents the number of sets that a given data sample is to be divided into; hence, the procedure is often referred to as k-fold cross-validation. Cross-validation mainly has wide applications in machine learning for producing a less biased or less optimistic estimation of the model.
K fold cross-validation is to evaluate the model design, not a particular training. Because you re-trained the model of the same design with different training sets.

```
# k fold  cross validation
# used for model who have class imbalance in dataset

kf = KFold(n_splits = 5)
kf.get_n_splits(X1)

print(kf)
```

```
KFold(n_splits=5, random_state=None, shuffle=False)
```

```
for key, algo in algorithms.items():
    kfoldscores = cross_val_score(algo, X_train1,y_train1 , cv = kf)
    print(f'kfold Cross validation of {key} {kfoldscores}')
    print('************************')
    print(f'Kfold cross validation score mean {kfoldscores.mean()}')
    print('-----------------------------------------')
```

```
kfold Cross validation of logistic Regression [0.78642857 0.81714286 0.83928571 0.79714286 0.80785714]
************************
Kfold cross validation score mean 0.8095714285714287
-----------------------------------------
kfold Cross validation of Decision Tree [0.76071429 0.78857143 0.79214286 0.78642857 0.78571429]
************************
Kfold cross validation score mean 0.7827142857142857
-----------------------------------------
kfold Cross validation of Random Forest Classifier [0.83928571 0.86       0.87285714 0.84357143 0.85714286]
************************
Kfold cross validation score mean 0.8545714285714284
-----------------------------------------
```

Fig 4.1

We initialize k fold with 5 number of splits which means 5 folds are performed on dataset.

Logistic regression:
Scores obtained are not so different from each other going from 78.64% to 83.93%. That means the performances were stable indicating the model is not overfitting. The average score value is 80.96% is good for general performance.
Decision Tree:
Decision tree performing poorly compared to logistic regression with an average score of 78.27%
Random Forest:
Random Forest is the best performer overall with the top mean score of validations 85.46% percent, which is best generalized to new data and is most robust on this dataset.

5.2 Up Sampling

As a next step of model development, we up-sample the distribution classes of the target variable using the SMOTE technique.

SMOTE (Synthetic Minority Oversampling Technique) is an oversampling technique from the imblearn library to address imbalanced datasets. Smote creates random samples for the minority class increasing its instances in the dataset.

```
# Upsampling distributions using SMOTE

sm = SMOTE(sampling_strategy='auto', k_neighbors= 5)

x_resamp , y_resamp = sm.fit_resample(X1,y1)

X_train2, X_test2, y_train2, y_test2 = train_test_split(x_resamp,y_resamp, test_size = 0.20, random_state = 42, stratify = y_resa


X_train2 = scaler.fit_transform(X_train2)
X_test2 = scaler.transform(X_test2)
```

```
# checks classes counts after upsampling and compares it with count of before upsampling
y_resamp.value_counts(), y1.value_counts()
```

```
(1    7963
 0    7963
 Name: Exited, dtype: int64,
 0    7963
 1    2037
 Name: Exited, dtype: int64)
```

Fig 4.2

For the SMOTE, we will provide parameters of the sampling strategy set to auto and k neighbors to 5. Next fit and resample the input and output variables and split them into train and test data.

Validating the smote technique by checking the value counts for both classes. We can observe that minority class 1 instances have increased to the same count as class 0.

```
Accuracy of logistic Regression 0.78
Classification Report of  logistic Regression
              precision    recall  f1-score   support

           0       0.78      0.78      0.78      2389
           1       0.78      0.79      0.78      2389

    accuracy                           0.78      4778
   macro avg       0.78      0.78      0.78      4778
weighted avg       0.78      0.78      0.78      4778

Confusion Matrix logistic Regression:
[[1860  529]
 [ 511 1878]]
------------------------------------------------------------
Accuracy of Decision Tree 0.79
Classification Report of  Decision Tree
              precision    recall  f1-score   support

           0       0.80      0.78      0.79      2389
           1       0.78      0.80      0.79      2389

    accuracy                           0.79      4778
   macro avg       0.79      0.79      0.79      4778
weighted avg       0.79      0.79      0.79      4778

Confusion Matrix Decision Tree:
[[1859  530]
 [ 472 1917]]
------------------------------------------------------------
Accuracy of Random Forest Classifier 0.86
Classification Report of  Random Forest Classifier
              precision    recall  f1-score   support

           0       0.86      0.86      0.86      2389
           1       0.86      0.86      0.86      2389

    accuracy                           0.86      4778
   macro avg       0.86      0.86      0.86      4778
weighted avg       0.86      0.86      0.86      4778

Confusion Matrix Random Forest Classifier:
[[2044  345]
 [ 339 2050]]
```

Fig 4.3

Fitted the resampled data to the models. Showed success in the resampling smote technique as it produced the relatively same accuracy scores before resampling. Increase in minority class recall from 46% to 86 % and F1 score from 57% to 86% of the random forest classifier which is our best model. In the confusion matrix, we see an increase in true negative predictions indicating that our model is more generalized and robust and can easily predict customers who are Exited class 1.

```
# visualize confusion matrix

cm_display = ConfusionMatrixDisplay(confusion_matrix = confu_matrix, display_labels = randforest.classes_)

cm_display.plot()

plt.show()
```

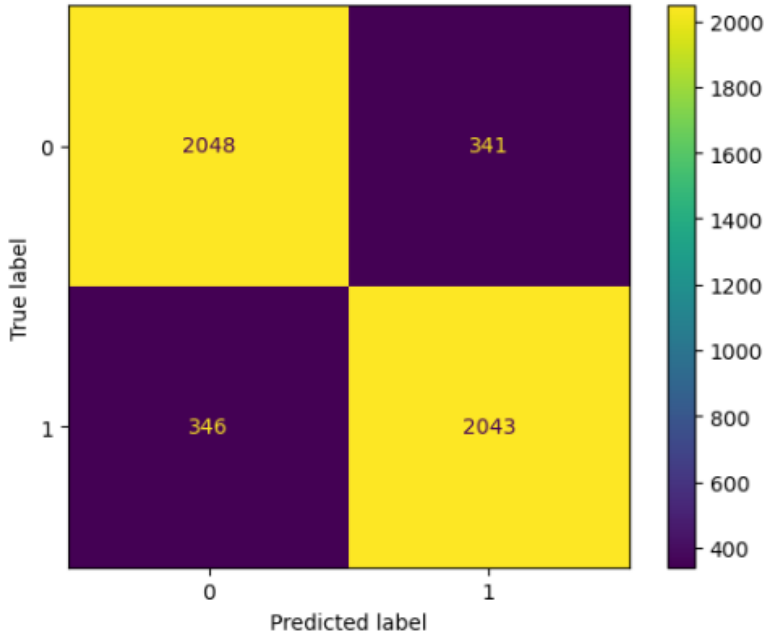


Fig 4.4 Confusion Matrix Visualization

As a last step, we now look at which features contributed most to predicting output variables in the random forest model.

```
Feature importances for Model Random Forest Classifier
------------------------------------------------------
Age                0.225
EstimatedSalary    0.123
Balance            0.120
CreditScore        0.116
IsActiveMember     0.104
NumOfProducts      0.101
Gender_Male        0.074
Tenure             0.071
Geography_Spain    0.029
HasCrCard          0.022
Geography_Germany    0.015
```

Fig 4.5

As assumed or validated from the EDA age is the highest predictor variable with 22.5%, surprisingly, estimated salary captures the second highest predictor variable with 12.3% even if it is less correlated with the target variable that has been shown in the above correlation analysis.

Summary

Customer churn prediction analysis therefore achieved its major goal of identifying customers who were more likely to leave the bank so that proactive retention efforts could be carried out on them. The final model was the Random Forest classifier with the best-performing accuracy score of 86% and an F1 score of 86%, meeting the predefined success criteria. Key drivers of churn are likely Age, Balance and estimated salary, and credit score. Among them, age stands as the highest predictor of a predictive model.

These insights uncovered in EDA and combining important variables like age and credit score can lead to the development of segmenting and targeting marketing strategies and also provide insight into the behaviors and characteristics of those who are at risk for example we uncovered an interesting fact that customers old age people having a low credit score is a normal observation, while customers having low credit score and age 25 – 35 are likely to churn more when compared to others. The greater the balance, the less likely to leave, probably due to greater satisfaction with the bank services. More number of products registered a less likely to churn these inversely proportional indicates that organizations should focus more on registering their customers more than 2 products. Customers who are not regularly active can leave the bank for this kind of customers Send an alert or notification mentioning the number of inactive days and having person-to-person will surely provide useful information to develop strategies for upcoming customers.

Regional differences in churn might include higher churn for German customers, which could necessitate more localized marketing strategies or product customization to meet regional preferences. The relatively sedentary older customers have a higher churn ratio. Periodical check in calls or even financial wellness calls for this segment of customers may be considered in establishing the potential hidden causes of dissatisfaction.

Investing in model improvement because the random forest model was quite successful, consider using ensemble models or fine-tuning that can provide even better results for churn predictions.

References:

Dataset: https://www.kaggle.com/datasets/saurabhbadole/bank-customer-churn-prediction-dataset

License: https://creativecommons.org/licenses/by-nc-sa/4.0/

https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.grid.html

https://en.wikipedia.org/wiki/List_of_European_countries_by_life_expectancya

https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/

https://scikitlearn.org/stable/auto_examples/compose/plot_column_transformer_mixed_types.html

https://medium.com/@rithpansanga/choosing-the-right-size-a-look-at-the-differences-between-upsampling-and-downsampling-methods-daae83915c19#:~:text=If%20the%20focus%20is%20on,may%20be%20a%20better%20option.

https://youtu.be/4SivdTLIwHc?feature=shared

https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

https://www.geeksforgeeks.org/spearmans-rank-correlation/

https://aws.amazon.com/what-is/data-preparation/

https://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html

https://scikit-learn.org/dev/modules/generated/sklearn.preprocessing.StandardScaler.html

https://www.datacamp.com/tutorial/understanding-logistic-regression-python

https://www.ibm.com/topics/decision-trees

https://www.ibm.com/topics/random-forest