رواد مصر الرقمية

BANK CUSTOMER CHURN

CASE STUDY

MEET OUR TEAM

SUPERVISOR: ENG. MAHMOUD ELSAYED

MOHAMED ELASSYOUTY

MOHAMED FEKRY

MOHAMED MESELHY

HASSAN WAKED

MUSTAFA MAHMOUD

AGENDA

- Introduction
- Problem Statement & Objectives
- Data Sources & Description
- Preprocessing & Feature Engineering
- Exploratory Data Analysis (EDA)
- Power BI
- Machine Learning Model
- Deployment

INTRODUCTION

DATASET OVERVIEW

Dataset Purpose:

This dataset contains information about bank customers and their account activities. It is primarily used for customer churn analysis — predicting whether a customer will leave the bank.

Number of Attributes:

14 columns describing customer demographics, account status, and banking activity.

Key Features:

Customer Information: Customer ID, Surname, Age, Gender, Geography.

Banking Behavior: Credit Score, Tenure, Balance, Number of Products, Has Credit Card, Is Active Member.

Financial Data: Estimated Salary.

Target Variable: Exited (1 = Customer left, 0 = Customer stayed).

CUSTOMER CHURN DEMOGRAPHIC TABLE

CustomerId	Surname	Gender	Age	Geography	CreditScore	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
15634602	Hargrave	Female	42	France	619	2	0.0	1	1	1	101348.88	1
15647311	Hill	Female	41	Spain	608	1	83807.86	1	0	1	112542.58	0
15619304	Onio	Female	42	France	502	8	159660.8	3	1	0	113931.57	1
15701354	Boni	Female	39	France	699	1	0.0	2	0	0	93826.63	0
15737888	Mitchell	Female	43	Spain	850	2	125510.82	1	1	1	79084.1	0
15574012	Chu	Male	44	Spain	645	8	113755.78	2	1	0	149756.71	1
15592531	Bartlett	Male	50	France	822	7	0.0	2	1	1	10062.8	0
15656148	Obinna	Female	29	Germany	376	4	115046.74	4	1	0	119346.88	1
15792365	He	Male	44	France	501	4	142051.07	2	0	1	74940.5	0
15592389	H?	Male	27	France	684	2	134603.88	1	1	1	71725.73	0

PROBLEM STATEMENT

OBJECTIVES

PROBLEM STATEMENT

Customer **churn** poses a significant threat to banks by reducing **revenues** and increasing **customer acquisition costs**. Despite having access to extensive **demographic**, **financial**, and **behavioral data**, identifying customers at risk of leaving remains a challenge.

This project addresses the need to analyze **customer data** to accurately **predict churn** and develop effective **retention strategies** that enhance **customer loyalty** and improve overall **bank performance**.

OBJECTIVES

- ANALYZE
- DEVELOP
- IDENTIFY
- PROVIDE

METADATA

	Column Name	Description
Churn Modelling	Customer ID	A unique identifier for each customer.
	Surname	The customer's surname or last name.
	Credit Score	A numerical value representing the customer's credit score.
	Geography	The country where the customer resides (France, Spain, or Germany).
	Gender	The customer's gender (Male or Female).
	Age	The customer's age.
	Tenure	The number of years the customer has been with the bank.
	Balance	The customer's account balance.
	NumOfProducts	The number of bank products the customer uses (e.g., savings account, credit card).
	HasCrCard	Whether the customer has a credit card (1 = yes, 0 = no).
	IsActiveMember	Whether the customer is an active member (1 = yes, 0 = no).
	EstimatedSalary	The estimated salary of the customer.
	Exited	Whether the customer has churned (1 = yes, 0 = no).

```python
# our own libriaies
import MyMachineLearningLib as ml
import MyDataUitlsLib as ul
import MyVisualizationLib as vl
```
✓ 0.0s

MyMachineLearningLib > __init__.py > ...

```python
from sklearn.model_selection import train_test_split, GridSea
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Ridge, Las
from sklearn.tree import DecisionTreeRegressor, DecisionTreeC
from sklearn.ensemble import (
    RandomForestRegressor, RandomForestClassifier,
    GradientBoostingRegressor, GradientBoostingClassifier
)
from sklearn.svm import SVR, SVC
from sklearn.neighbors import KNeighborsRegressor, KNeighbors
from sklearn.naive_bayes import GaussianNB, MultinomialNB
from sklearn.neural_network import MLPRegressor, MLPClassifie
from sklearn.metrics import mean_squared_error, mean_absolute

from xgboost import XGBRegressor, XGBClassifier
from lightgbm import LGBMRegressor, LGBMClassifier

import numpy as np
import joblib
from datetime import datetime


def calculate_regression_metrics(y_test, y_predict):...

# Function to save a trained model and its scaler
def save_model_and_scaler(model, scaler, model_name, base_dir

# Function to load a saved model and its scaler
def load_model_and_scaler(model_filename, scaler_filename):...

# Function to make predictions using a saved model and scaler
def predict_with_model(model, scaler, x_new):...

def create_linear_regression_model(x, y, test_size=0.3, shuff


def create_svm_model(x, y, test_size=0.3, shuffle=True, rando


def create_random_forest_model(x, y, test_size=0.3, shuffle=T



def create_decision_tree_model(x, y, test_size=0.3, shuffle=T


def evaluate_model_performance(y_true, y_pred, task_type='reg
    """
```

src > MyVisualizationLib > __init__.py > ...

```python
import matplotlib.pyplot as plt
import seaborn as sns
import os
import random
import numpy as np
import pandas as pd

def plot_boxplots(df, features, save_folder="Milestone 1/boxp


def plot_histograms(data, features, colors=None, save_folder=


def plot_pairplots(data, features, hue=None, save_folder="pai

def plot_heatmap(data, features, save_folder="heatmap_images"

def plot_model_performance(y_true, y_pred, task_type='regress

def plot_feature_importance(model, feature_names, save_folder

def plot_correlation_heatmap(df, save_folder="correlation", f

def plot_time_series(data, date_column, value_column, save_fo
```

src > MyDataUitlsLib > __init__.py > ...

```python
import numpy as np
import joblib
import streamlit as st
import pandas as pd
import logging
from sklearn.preprocessing import LabelEncoder
import os
from sklearn.preprocessing import PolynomialFeatures
import matplotlib.pyplot as plt
import seaborn as sns

def make_prediction(user_input, ModelPath, ScalerPath):...

def load_data(file_path: str):...

def check_data_for_preprocessing(...

class DataFrameStatistics:...

def standardize_column_headers(df: pd.DataFrame) -> pd.DataFr

def identify_outliers(df: pd.DataFrame) -> pd.DataFrame:...

def drop_columns(df: pd.DataFrame, columns_to_drop: list) ->

def drop_duplicates(df: pd.DataFrame) -> pd.DataFrame:...

def handle_missing_values(df: pd.DataFrame, strategy: str = '

def encode_column(data, column_name, encoding_type="onehot"):

def encode_by_ranges(df: pd.DataFrame, column: str, new_colum

def save_to_csv(data, filename):...

def write_to_text_file(data, filename='output.txt'):...

def feature_engineering(df: pd.DataFrame, target_column: str

def validate_data(df: pd.DataFrame, schema: dict) -> dict:...

def detect_anomalies(df: pd.DataFrame, method: str = 'zscore'

def create_time_series_features(df: pd.DataFrame, date_column
```

DATA COLLECTION

DESCRIPTION

DATA COLLECTION

- Load the dataset (CSV, SQL, API, etc.)
- Understand the source and quality of the data

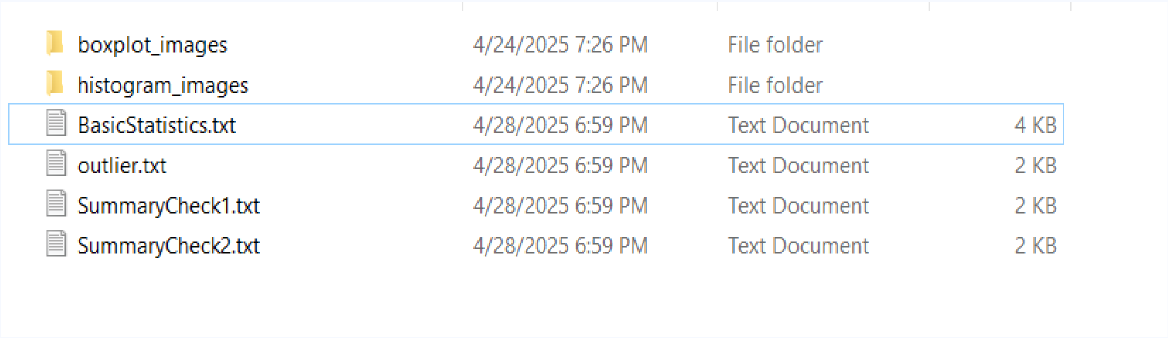
```
In [7]: FilePath= r'1-DataCollection\DataSet Before Cleanig.csv'
        df=ul.load_data(FilePath)

==============================================================================
column names: ['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography', 'Gender', 'Age', 'Tenure', 'Balance', 'NumOf
Products', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary', 'Exited']
==============================================================================
Updated column names(strip, lowercase, and standardize spaces): ['rownumber', 'customerid', 'surname', 'creditscore', 'geogra
phy', 'gender', 'age', 'tenure', 'balance', 'numofproducts', 'hascrcard', 'isactivemember', 'estimatedsalary', 'exited']
==============================================================================
Dataset loaded successfully with 10002 rows and 14 columns.
==============================================================================
```

Conclusion

Description	Details
Original Column Names	['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography', 'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary', 'Exited']
Updated Column Names	['rownumber', 'customerid', 'surname', 'creditscore', 'geography', 'gender', 'age', 'tenure', 'balance', 'numofproducts', 'hascrcard', 'isactivemember', 'estimatedsalary', 'exited']
Rows in Dataset	10002
Columns in Dataset	14
Notes	Column names were stripped, lowercased, and standardized for consistency.

DATA UNDERSTANDING

- View dataset structure (rows, columns)
- Check data types and sample values
- Identify target and feature variables
- Understand the business/domain context



FEATURES

Feature	Unique Count	Unique Percentage
gender	2	0.02%
hascrcard	2	0.02%
isactivemember	2	0.02%
exited	2	0.02%
geography	3	0.03%
numofproducts	4	0.04%
tenure	11	0.11%
age	73	0.73%
creditscore	460	4.60%
surname	2,932	29.31%
balance	6,382	63.81%
estimatedsalary	9,999	99.97%
rownumber	10,000	99.98%
customerid	10,000	99.98%

UNIQUE VALUES PER FEATURE

Rows	10,002
Columns	14

DATASET SHAPE

FEATURES

Feature	Missing Count	Missing Percentage
rownumber	0	0.00%
customerid	0	0.00%
surname	0	0.00%
creditscore	0	0.00%
geography	1	0.01%
gender	0	0.00%
age	1	0.01%
tenure	0	0.00%
balance	0	0.00%
numofproducts	0	0.00%
hascrcard	1	0.01%
isactivemember	1	0.01%
estimatedsalary	0	0.00%
exited	0	0.00%

UNIQUE VALUES PER FEATURE

Metric	Value
Duplicate Count	2
Duplicate Percentage	0.02%

DATASET SHAPE

INITIAL SUMMARY CHECK

```
--------------------------------------------------------------------------------
Dataset Shape: (10002, 14)
Duplicate Rows: 2 (0.02%)
================================================================================
Handle Missing Values:
- geography: 1 missing (0.01%)
- age: 1 missing (0.01%)
- hascrcard: 1 missing (0.01%)
- isactivemember: 1 missing (0.01%)
================================================================================
No constant columns.
================================================================================
Encode Categorical Columns:
- surname
- geography
- gender
================================================================================
Mixed Type Columns:
- geography
================================================================================
High Cardinality Columns:
- rownumber: 10000 unique values
- customerid: 10000 unique values
- surname: 2932 unique values
- creditscore: 460 unique values
- age: 73 unique values
- balance: 6382 unique values
- estimatedsalary: 9999 unique values
================================================================================
Skewed Numeric Columns:
- exited: Skewness = 1.47
- age: Skewness = 1.01
================================================================================
Suggested Next Steps:
- Handle missing data.
- Encode categorical features.
- Consider binning/embedding for high cardinality.
- Apply transformations to skewed features.
- Resolve inconsistent data types.
```

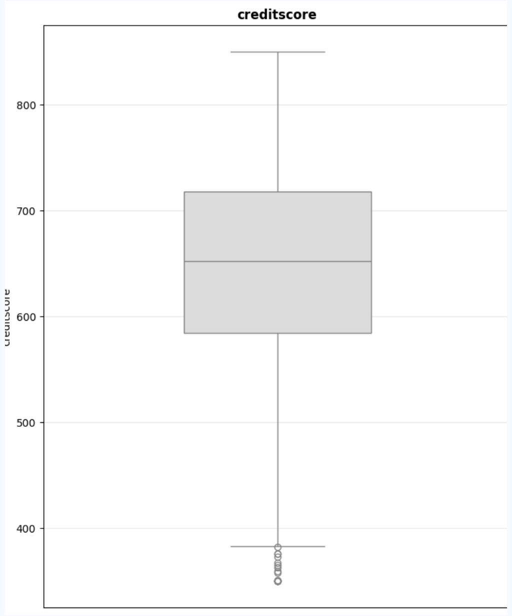
DESCRIPTIVE STATISTICS (NUMERICAL COLUMNS)

Metric	rownumber	customerid	creditscore	age	tenure
Count	10002	10002	10002	10001	10002
Mean	5001.5	15,690,930	650.56	38.92	5.01
Std	2887.47	71,931.77	96.66	10.49	2.89
Min	1	15,565,700	350	18	0
25% (Q1)	2501.25	15,628,520	584	32	3
Median (Q2)	5001.5	15,690,730	652	37	5
75% (Q3)	7501.75	15,753,230	718	44	7
Max	10000	15,815,690	850	92	10

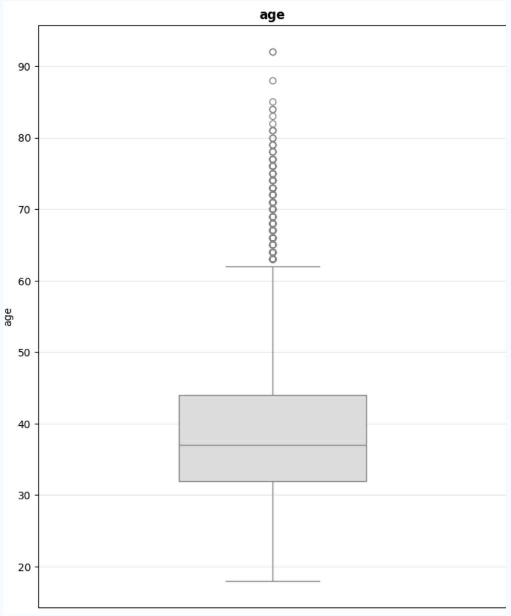
DESCRIPTIVE STATISTICS (NUMERICAL COLUMNS)

Metric	balance	numofproducts	hascrcard	isactivemember	estimatedsalary
Count	10002	10002	10001	10001	10002
Mean	76,491.11	1.53	0.71	0.51	100,083.33
Std	62,393.47	0.58	0.46	0.5	57,508.12
Min	0	1	0	0	11.58
25% (Q1)	0	1	0	0	50,983.75
Median (Q2)	97,198.54	1	1	1	100,185.24
75% (Q3)	127,647.84	2	1	1	149,383.65
Max	250,898.09	4	1	1	199,992.48

BOX PLOT

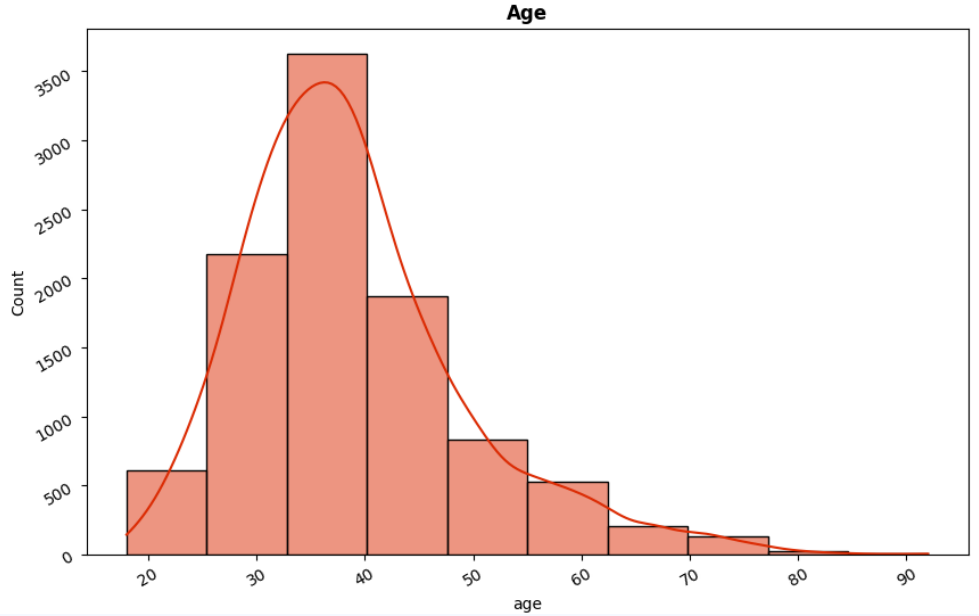


CREDIT SCORE

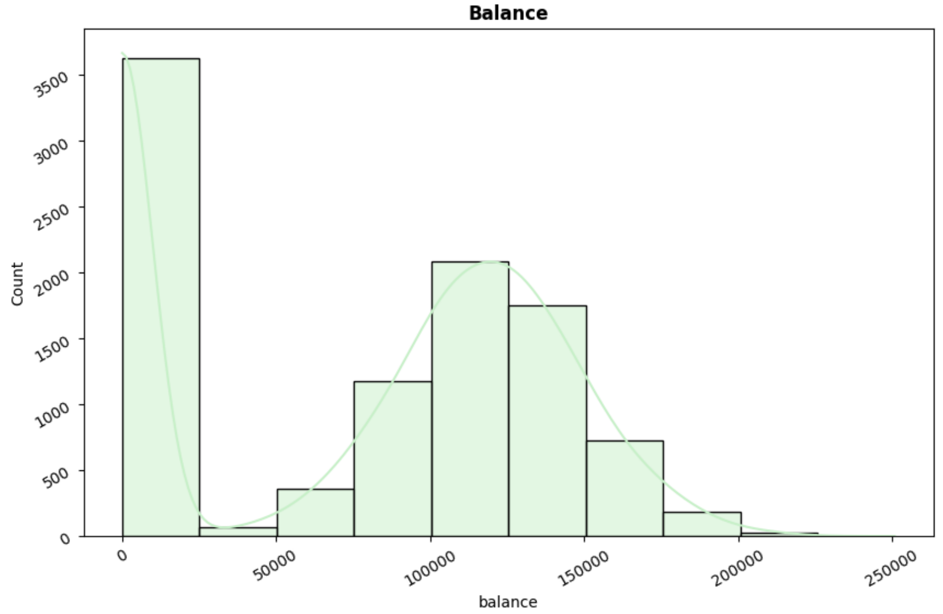


AGE

HISTOGRAM

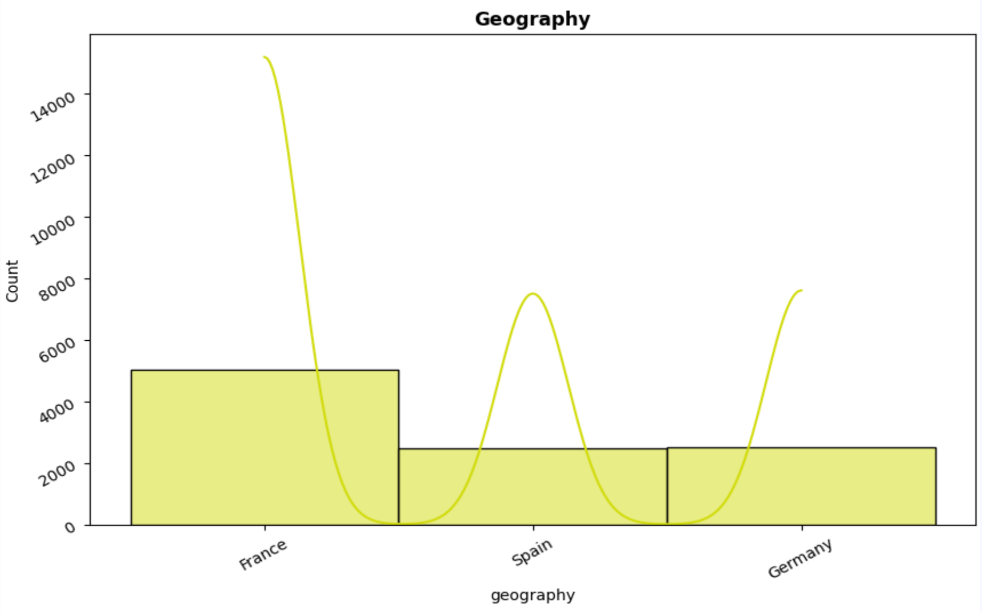


AGE

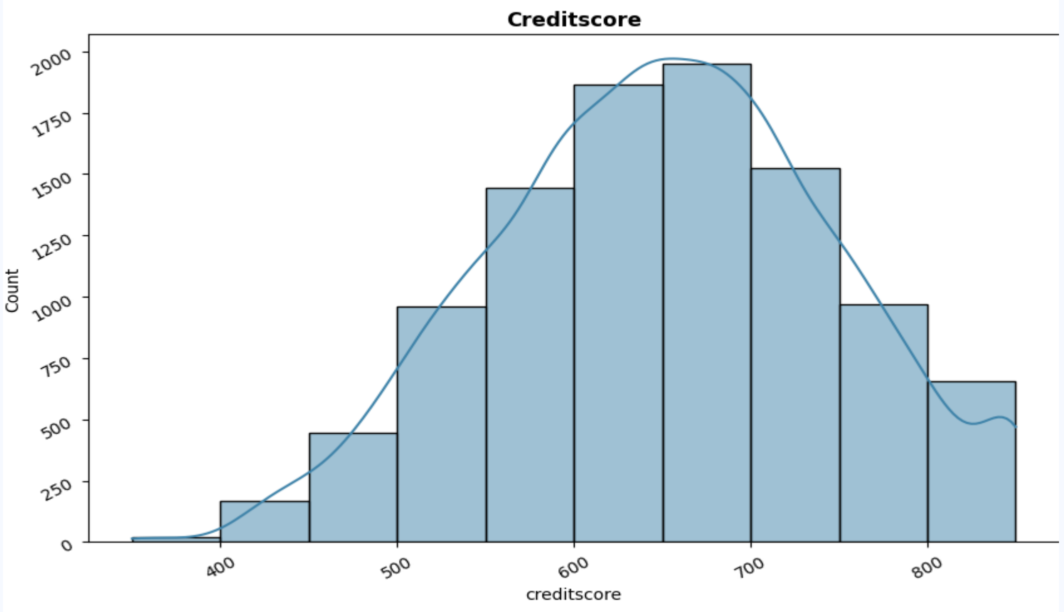


BALANCE

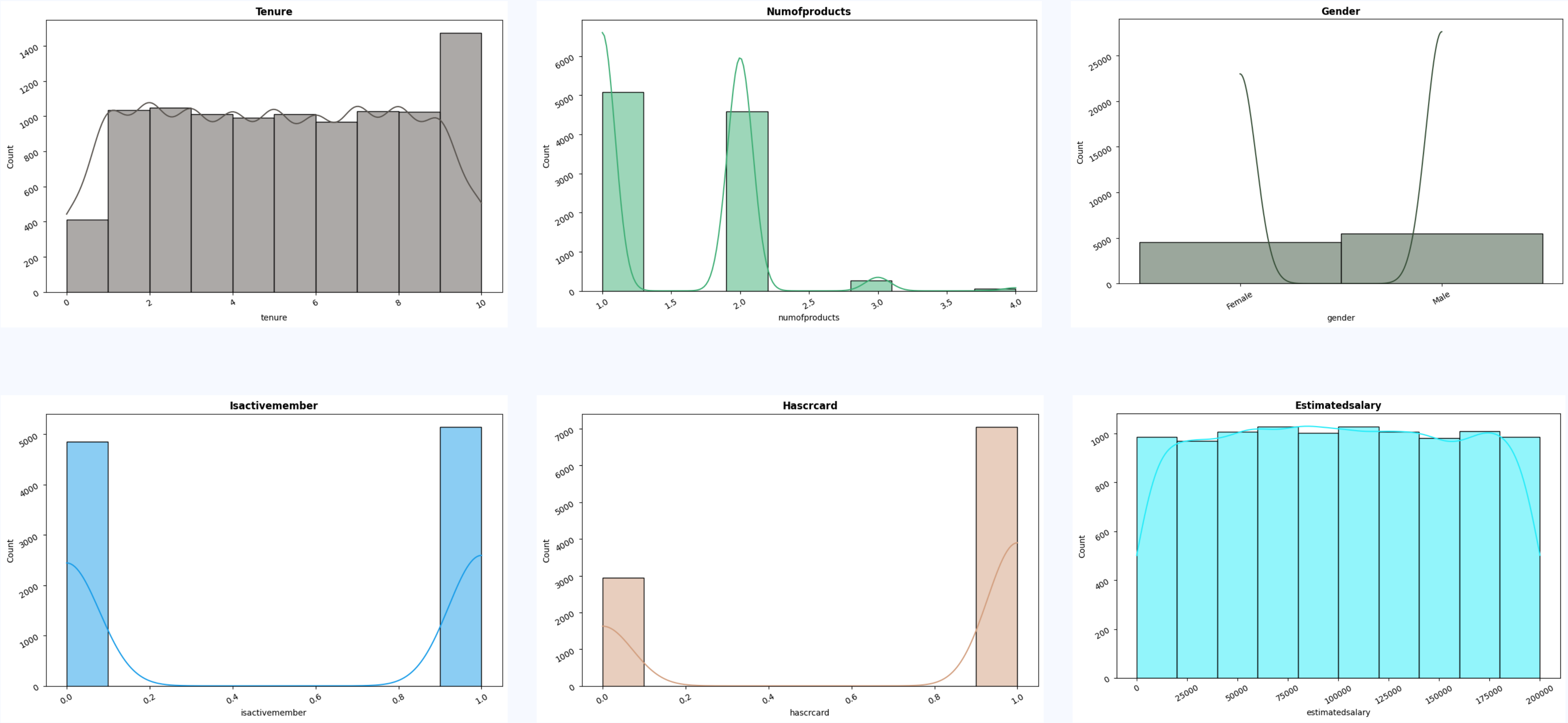
HISTOGRAM



GEOGRAPHY



CREDIT SCORE


Tenure
Count
tenure
Numofproducts
Count
numofproducts
Gender
Count
Female
Male
gender
Isactivemember
Count
isactivemember
Hascrcard
Count
hascrcard
Estimatedsalary
Count
estimatedsalary

CONCLUSION

Data Cleaning:

1. Remove unnecessary columns (rownumber, customerid, surname).
2. Eliminate duplicate rows.
3. Handle missing values.

Data Preprocessing:

1. Apply OneHot encoding for gender and Label encoding for geography.
2. Handle high cardinality for features like creditscore, balance, and estimatedsalary.
3. Apply transformations to skewed features like age.

✅ DATA CLEANING & ⚙️ DATA PREPROCESSING

No.	Credit Quality	Score Range	Encoded Value
1	Poor	300–579	0
2	Fair	580–669	1
3	Good	670–739	2
4	Very Good	740–799	3
5	Excellent	800–850	4

No.	Age Group	Age Range	Encoded Value
1	Younger	18–35	0
2	Middle	35–50	1
3	Older	50 and above	2

No.	Salary Group	Salary Range	Encoded Value
1	Low Salary	0 < Salary < 40,000	0
2	Middle Salary	40,000 ≤ Salary < 70,000	1
3	High Salary	Salary ≥ 70,000	2

No.	Tenure Group	Tenure Range	Encoded Value
1	New Client	0 ≤ Tenure ≤ 1	0
2	Short Client	1 < Tenure ≤ 3	1
3	Mid Client	3 < Tenure ≤ 6	2
4	Long Client	Tenure > 6	3

No.	Balance Group	Balance Range	Encoded Value
1	Low Balance	0 < Balance < 40,000	0
2	Middle Balance	40,000 ≤ Balance < 120,000	1
3	High Balance	Balance ≥ 120,000	2

UNIQUE VALUES PER FEATURE

Column	Unique Count	Unique Percentage
isactivemember	2	0.020008
geographyspain	2	0.020008
gender	2	0.020008
geographygermany	2	0.020008
geographyfrance	2	0.020008
genderlabel	2	0.020008
exited	2	0.020008
hascrcard	2	0.020008
balancerange	3	0.030012
estimatedsalaryrange	3	0.030012
geography	3	0.030012
tenurerange	4	0.040016
numofproducts	4	0.040016
creditscorerange	5	0.05002
tenure	11	0.110044
age	73	0.730292
ageskewed	73	0.730292
creditscore	460	4.601841
balance	6379	63.815526
estimatedsalary	9995	99.989996

DESCRIPTIVE STATISTICS (NUMERICAL COLUMNS)

Metric	creditscore	age	tenure	balance	numofproducts	hascrcard	isactivemember
Count	9996	9996	9996	9996	9996	9996	9996
Mean	650.503301	38.921071	5.013305	76476.26322	1.530212	0.705482	0.514906
Std	96.624668	10.488421	2.892353	62397.11882	0.581684	0.455849	0.499803
Min	350	18	0	0	1	0	0
25% (Q1)	584	32	3	0	1	0	0
Median (Q2)	652	37	5	97173.29	1	1	1
75% (Q3)	717.25	44	7.25	127639.3725	2	1	1
Max	850	92	10	250898.09	4	1	1

DESCRIPTIVE STATISTICS (NUMERICAL COLUMNS)

Metric	estimatedsalary	exited	genderlabel	geographyfrance	geographygermany	geographyspain	ageskewed
Count	9996	9996	9996	9996	9996	9996	9996
Mean	100106.7012	0.203782	0.545618	0.501301	0.251	0.247699	3.65468
Std	57513.3144	0.402829	0.49794	0.500023	0.43361	0.431698	0.251657
Min	11.58	0	0	0	0	0	2.944439
25% (Q1)	51002.11	0	0	0	0	0	3.496508
Median (Q2)	100238.11	0	1	1	0	0	3.637586
75% (Q3)	149400.1075	0	1	1	1	0	3.806662
Max	199992.48	1	1	1	1	1	4.532599

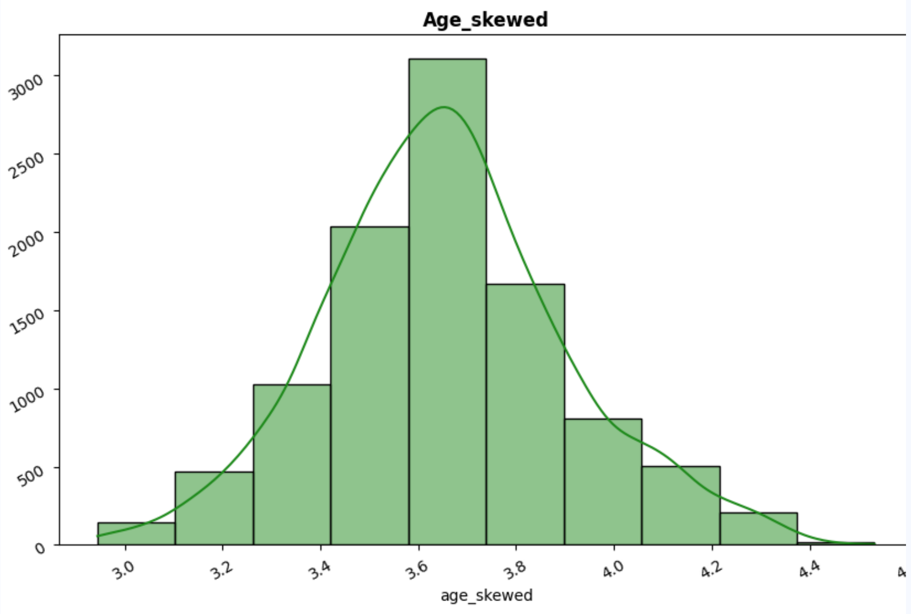
FINAL SUMMARY CHECK

```
==============================================================================
Dataset Shape: (9996, 20)
Duplicate Rows: 0 (0.00%)
==============================================================================
No missing values.
==============================================================================
No constant columns.
==============================================================================
Encode Categorical Columns:
- geography
- gender
- creditscorerange
- balancerange
- estimatedsalaryrange
- tenurerange
==============================================================================
No mixed-type columns.
==============================================================================
High Cardinality Columns:
- creditscore: 460 unique values
- age: 73 unique values
- balance: 6379 unique values
- estimatedsalary: 9995 unique values
- ageskewed: 73 unique values
==============================================================================
Skewed Numeric Columns:
- exited: Skewness = 1.47
- geographyspain: Skewness = 1.17
- geographygermany: Skewness = 1.15
- age: Skewness = 1.01
==============================================================================
Suggested Next Steps:
- Encode categorical features.
- Consider binning/embedding for high cardinality.
- Apply transformations to skewed features.
==============================================================================
```

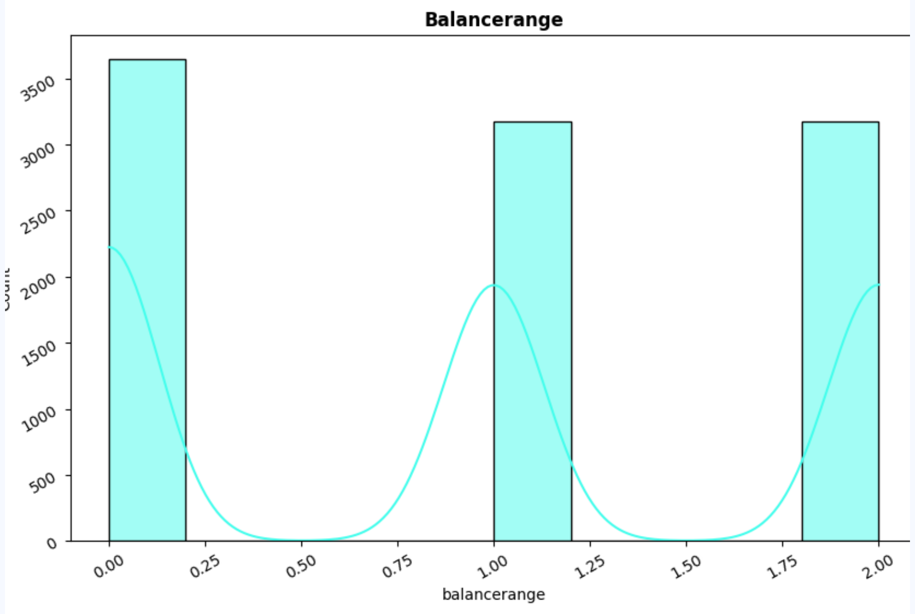
EDA

EXPLORATORY DATA ANALYSIS

HISTOGRAMS

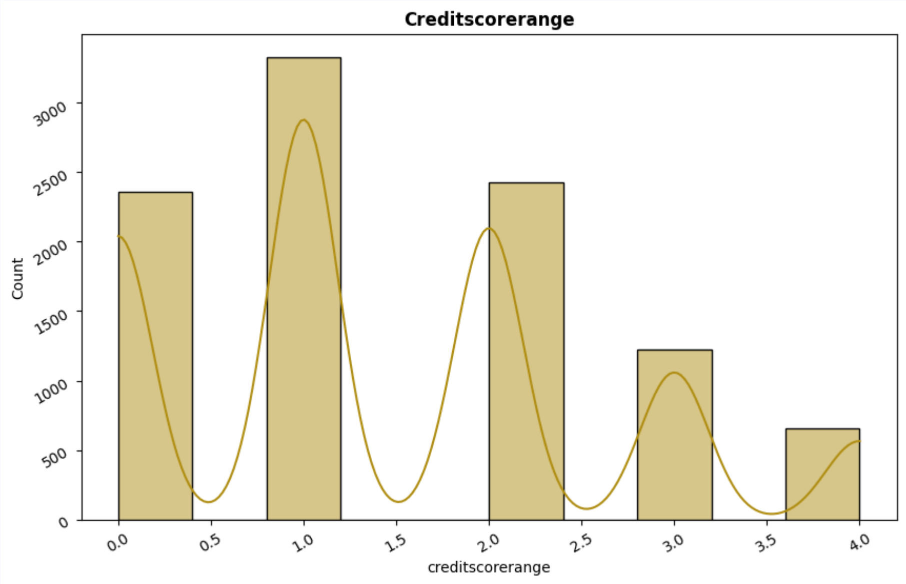


AGE SKEWED

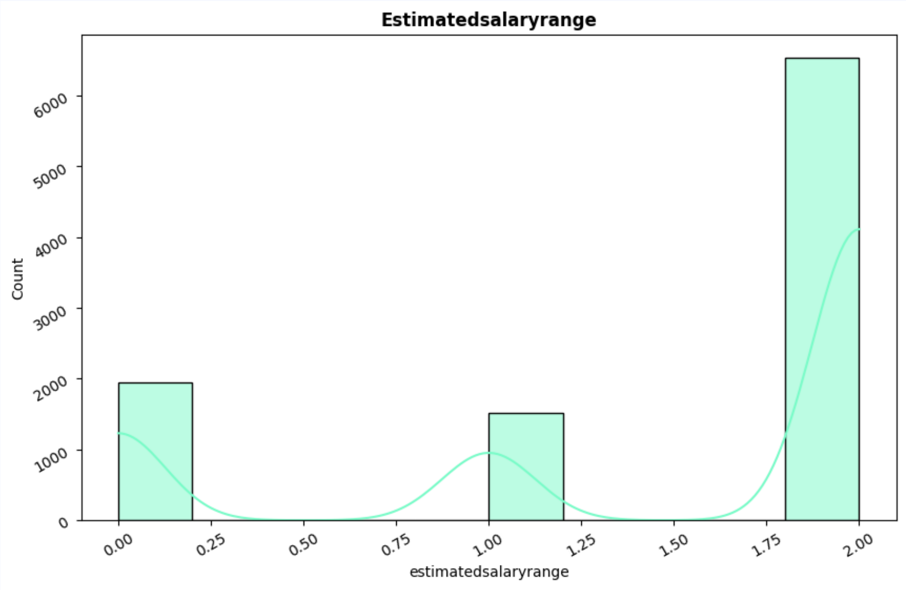


BALANCE RANGE

HISTOGRAMS

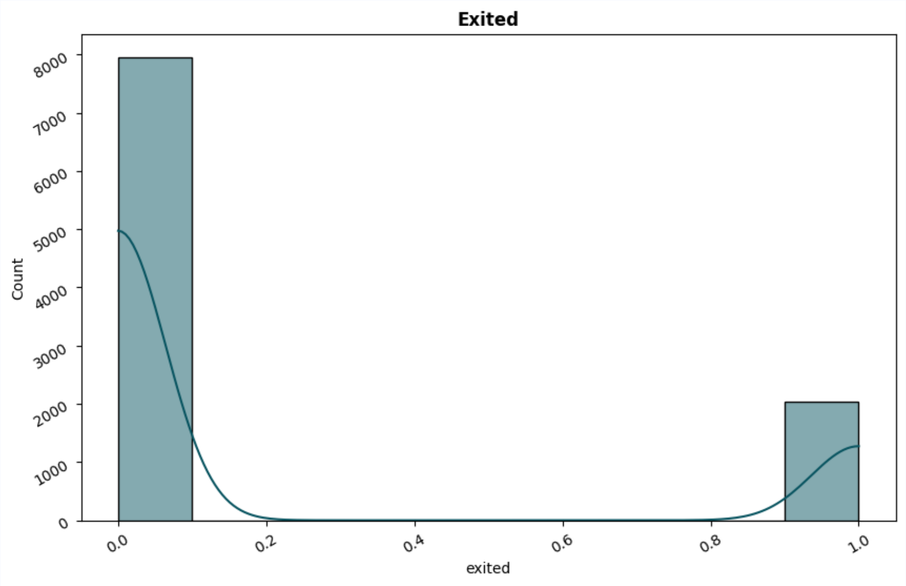


CREDIT SCORE RANGE

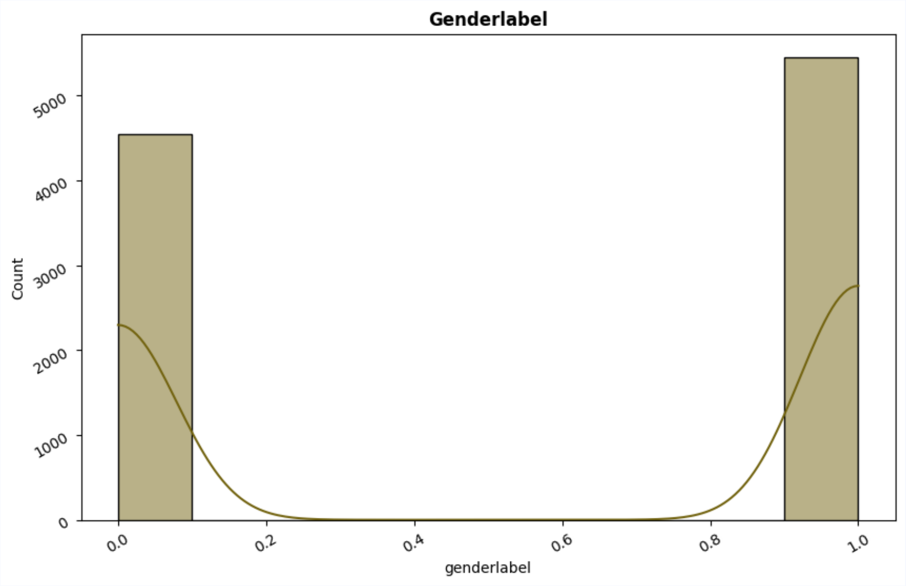


ESTIMATED SALARY RANGE

HISTOGRAMS

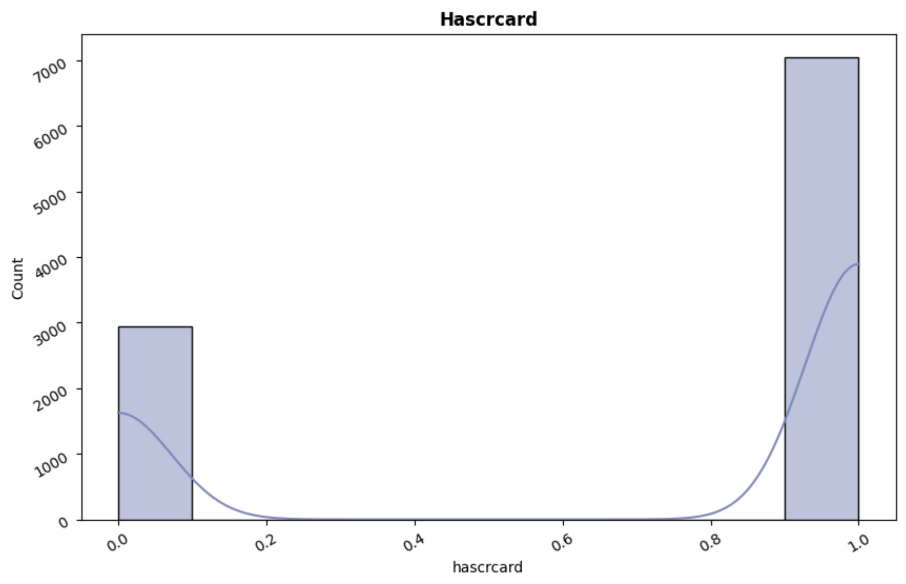


AGE SKEWED

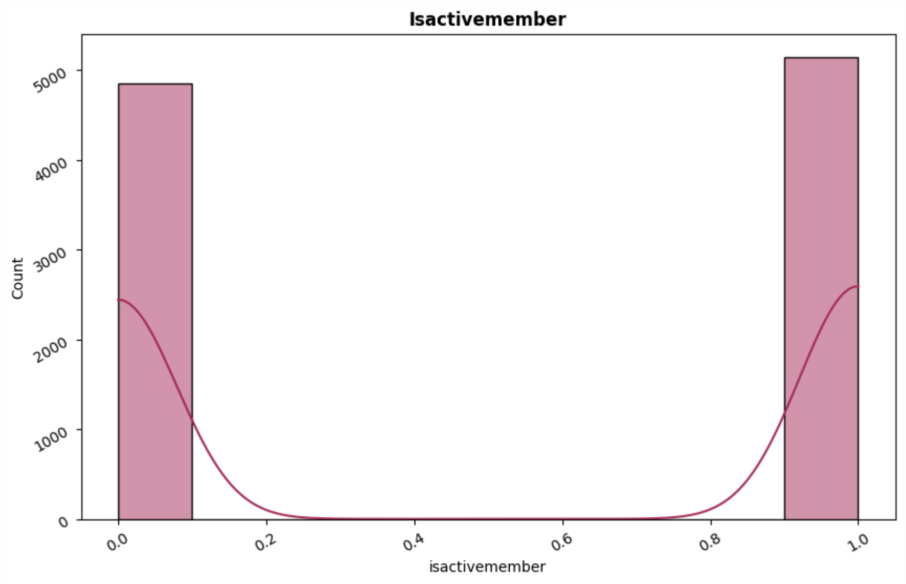


GENDER

HISTOGRAMS

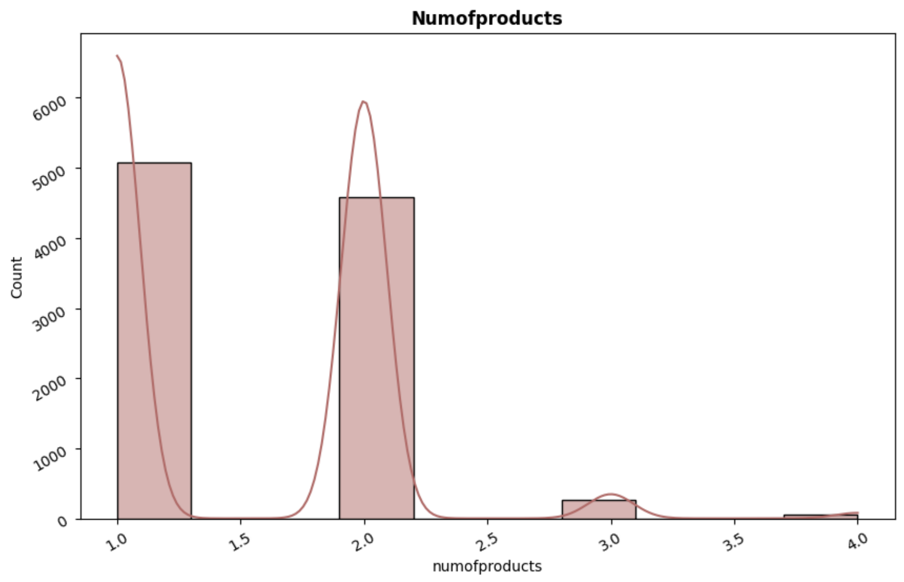


HAS CREDIT CARD

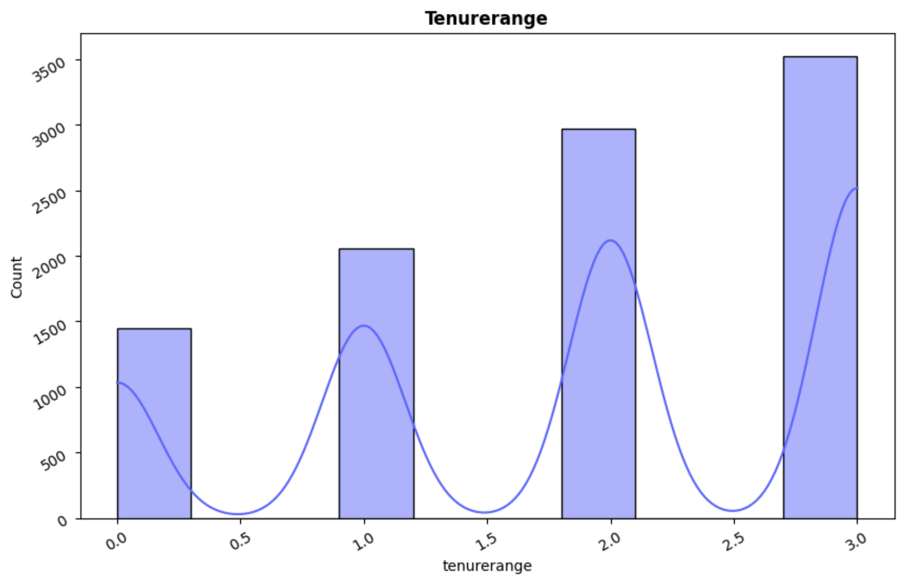


ACTIVE MEMEBER

HISTOGRAMS



NUMBER OF PRODUCTS

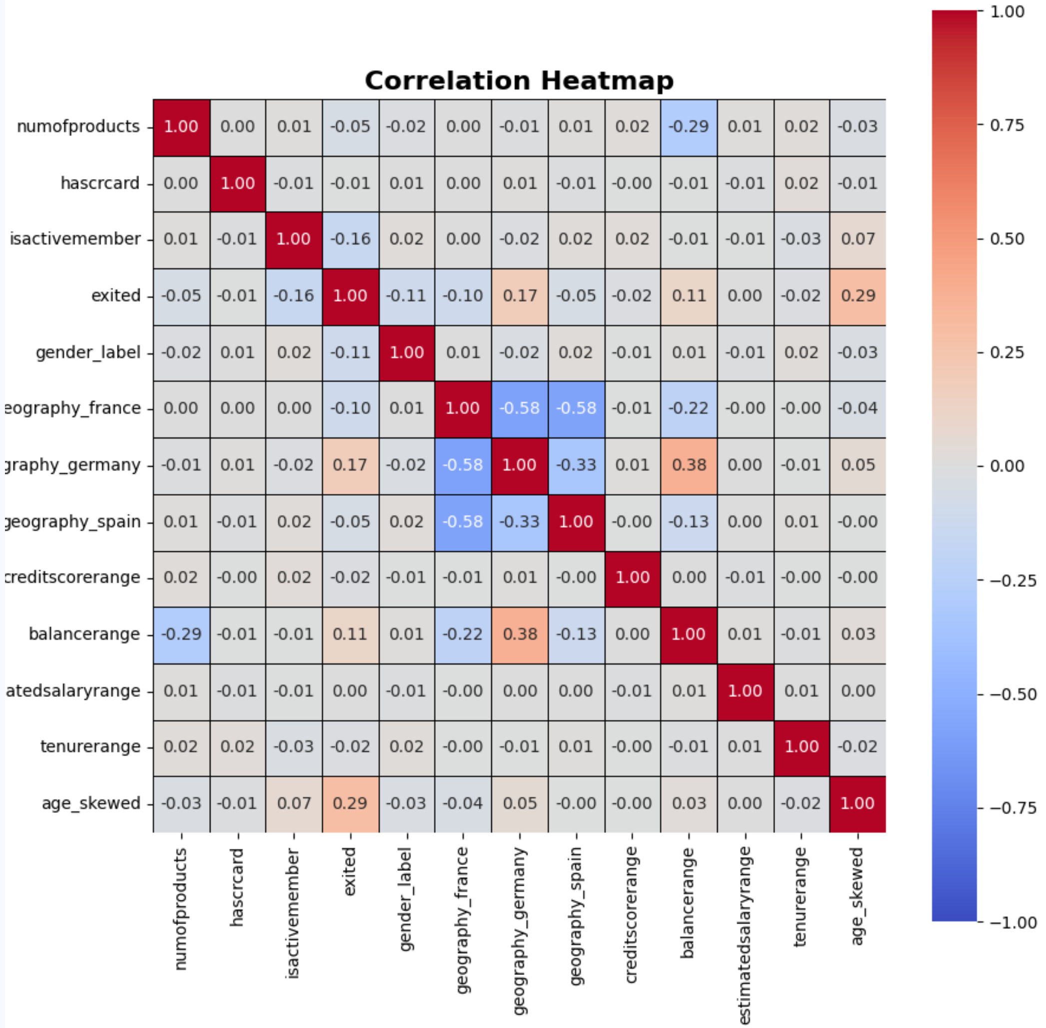


TENURE RANGE

HEAT MAP

- IsActiveMember →
  moderate negative correlation (0.16):
  active members are less likely to exit.

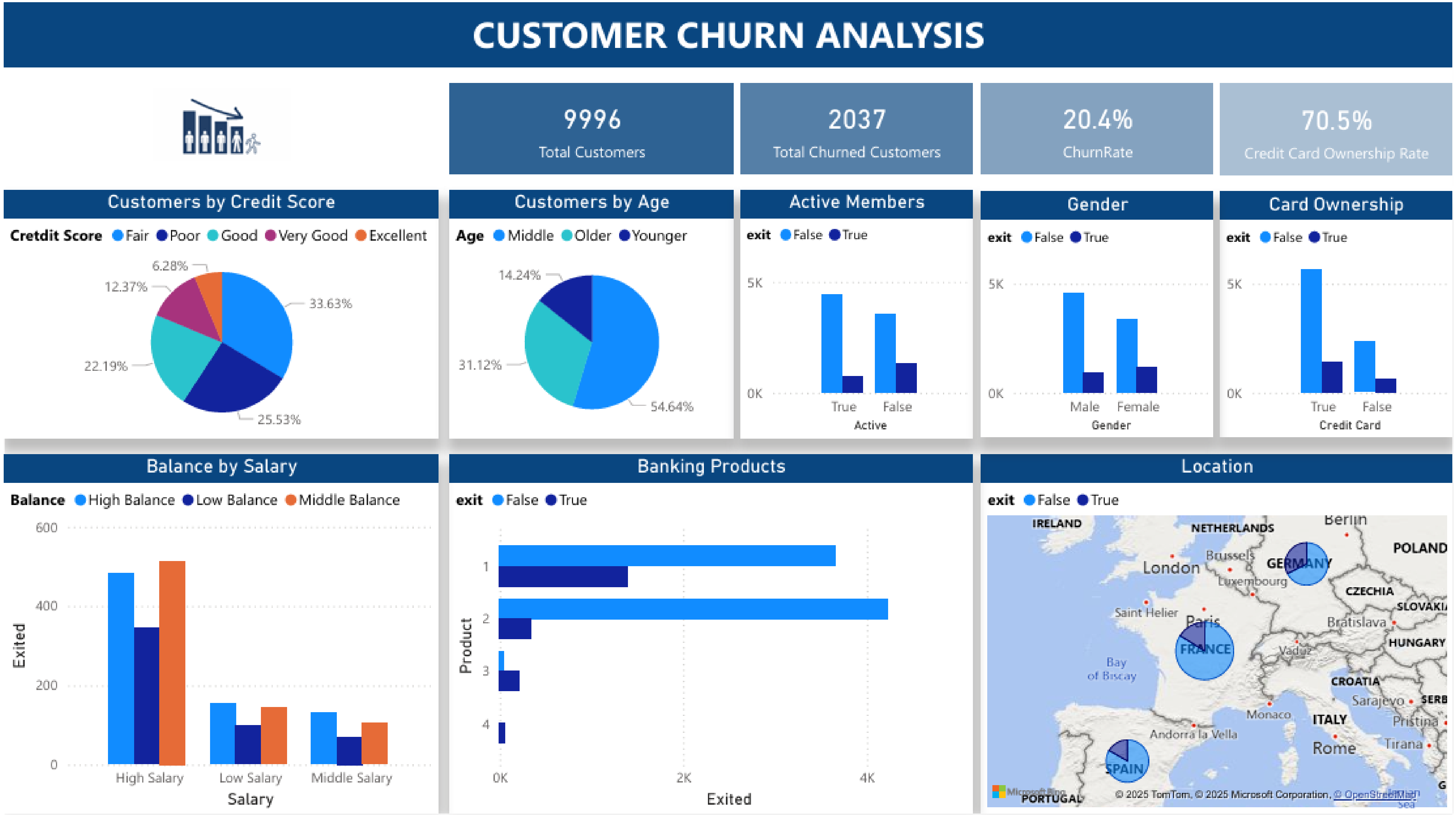
- AgeSkewed →
  moderate positive correlation (0.29):
  older customers are more likely to exit.



CONCLUSION

Feature Name	Type	Recommendation	Reason for Inclusion/Exclusion
age_skewed	Numerical	✅ Keep	Older customers show significantly higher churn **(corr ≈ 0.29);** strong behavioral indicator.
isactivemember	Categorical	✅ Keep	Active customers are less likely to churn (**corr ≈ −0.16**); crucial behavioral flag.
geography_germany	Categorical	✅ Keep	Customers from Germany churn more frequently (**corr ≈ 0.17**); useful regional feature.
geography_spain	Categorical	✅ Keep	Contrasts with Germany; adds diversity and comparative signal.
geography_france	Categorical	✅ Keep	Used as base category to avoid dummy variable trap in one-hot encoding.
balancerange	Numerical	✅ Keep	Financial indicator; shows bimodal pattern, possibly linked with churn behavior.
creditscorerange	Numerical	🔷 Optional	**Weak or no correlation**, but valuable in risk-based financial modeling. Useful for trees.
tenurerange	Numerical	✅ Keep	Loyalty indicator; bimodal pattern could be informative for churn prediction.
numofproducts	Numerical	✅ Keep	Discrete but meaningful; customers with more products behave differently. Helps trees.
estimatedsalaryrange	Numerical	🔷 Optional	**Flat distribution**; weak predictor, but might support tree models after feature importance check.
gender_label	Categorical	🔷 Optional	**Slight imbalance**; **very weak churn correlation** (−0.10); could be tested but not critical.
hascrcard	Categorical	🔷 Optional	**Almost zero correlation** with churn; keep for testing, drop if model doesn't improve.
exited (target)	Categorical	🎯 Target	Target variable (imbalanced); apply class balancing methods during training.

DASHBOARD


CUSTOMER CHURN ANALYSIS
9996
Total Customers
2037
Total Churned Customers
20.4%
ChurnRate
70.5%
Credit Card Ownership Rate
Customers by Credit Score
Cretdit Score Fair Poor Good Very Good Excellent
6.28%
12.37%
33.63%
22.19%
25.53%
Customers by Age
Age Middle Older Younger
14.24%
31.12%
54.64%
Active Members
exit False True
5K
0K
True
False
Active
Gender
exit False True
5K
0K
Male
Female
Gender
Card Ownership
exit False True
5K
0K
True
False
Credit Card
Balance by Salary
Balance High Balance Low Balance Middle Balance
600
400
200
0
Exited
High Salary
Low Salary
Middle Salary
Salary
Banking Products
exit False True
1
2
3
4
Product
0K
2K
4K
Exited
Location
exit False True
IRELAND
NETHERLANDS
Berlin
POLAND
London
Brussels
GERMANY
Luxembourg
CZECHIA
SLOVAKIA
Saint Helier
Paris
Bratislava
FRANCE
Vaduz
HUNGARY
Bay of Biscay
CROATIA
Sarajevo
Monaco
ITALY
Pristina
Andorra la Vella
Rome
Tirana
SPAIN
PORTUGAL
© 2025 TomTom, © 2025 Microsoft Corporation, © OpenStreetMap

DATA VISUALIZATION FACTORS – POWER BI

Geography France, Germany, Spain

Age Younger, Middle, Older

Gender Male, Female

Credit Score Poor, Fair, Good, Very Good, Excellent

Credit Card Ownership Yes, No

DATA VISUALIZATION FACTORS – POWER BI

Bank Balance Low, Middle, High

Estimated Salary Low, Middle, High

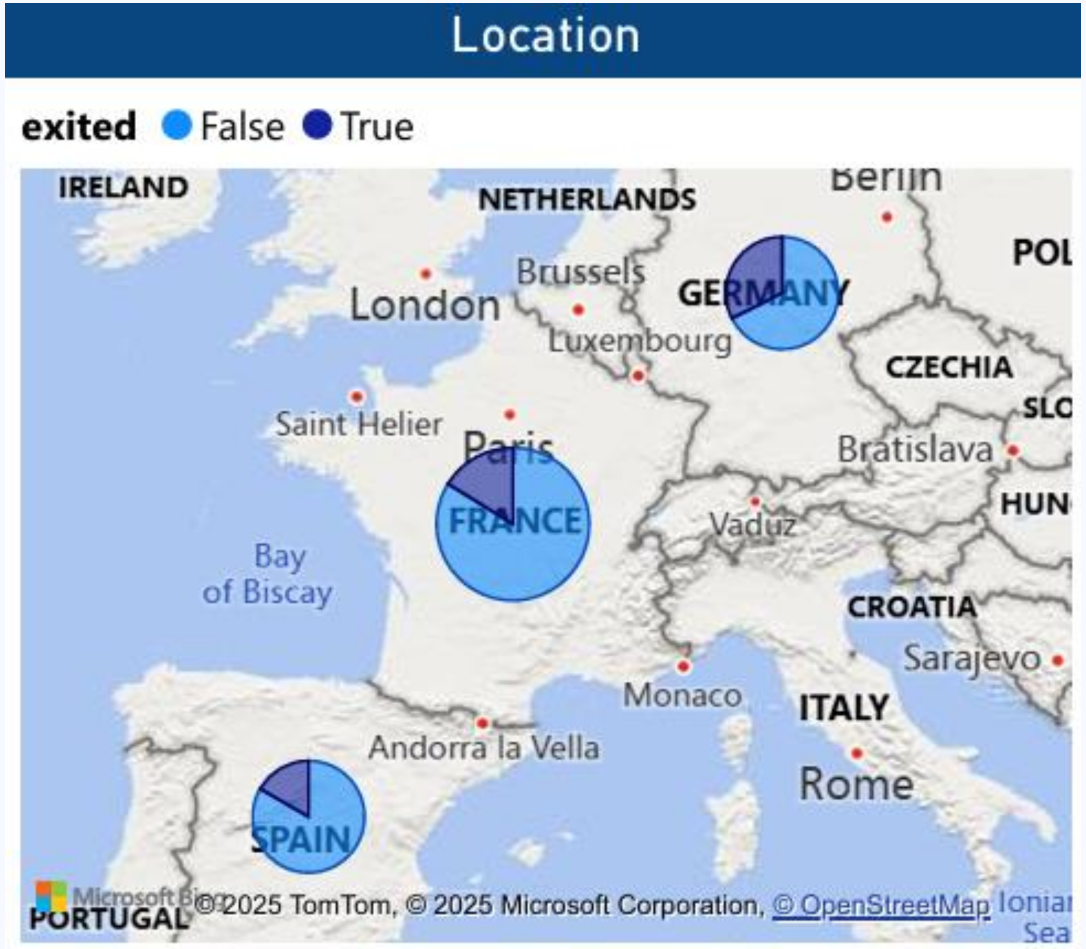
Banking Products 1 Product, 2 Product, 3 Product, 4 Product

Customer Activity Active, Inactive

DEMOGRAPHICS – POWER BI

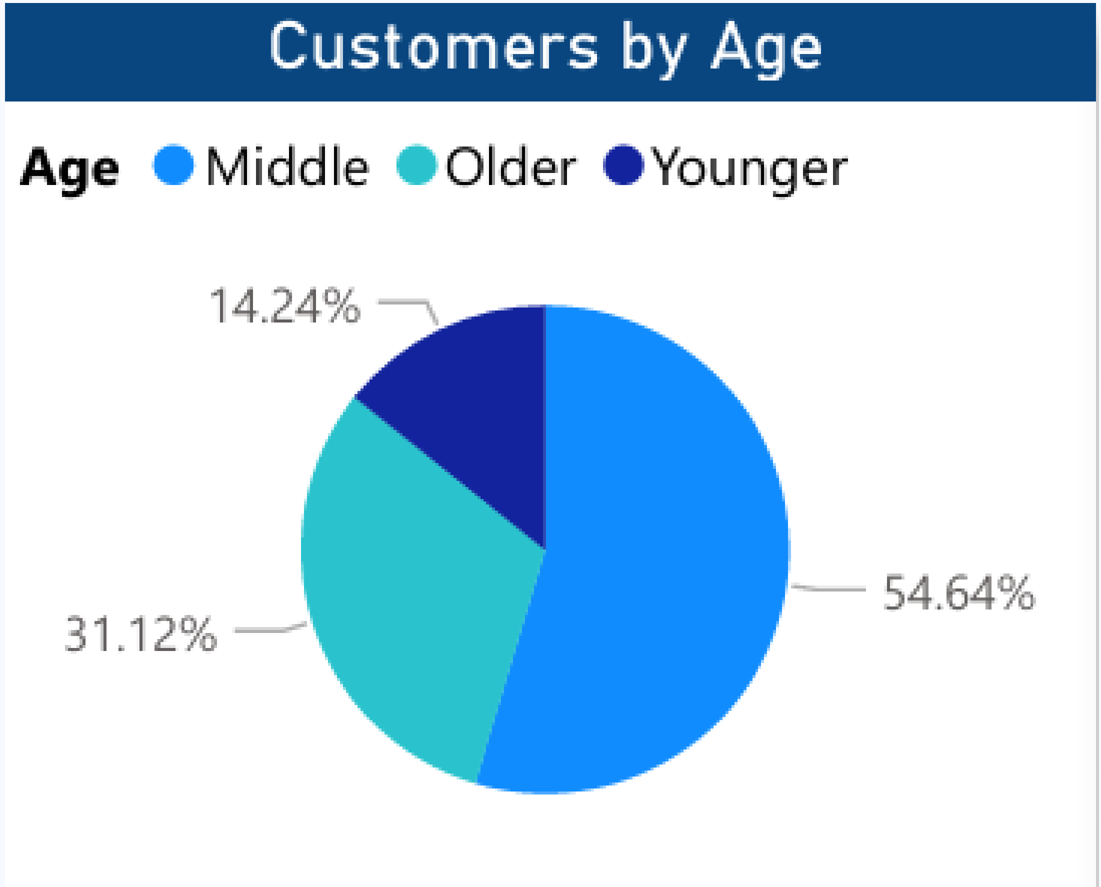
GEOGRAPHY

	France	Spain	Germany
Total Customers	5011	2476	2509
Total Churned Customers	810	413	814
Churn Rate Per Total Customers	16.2%	16.7%	32%



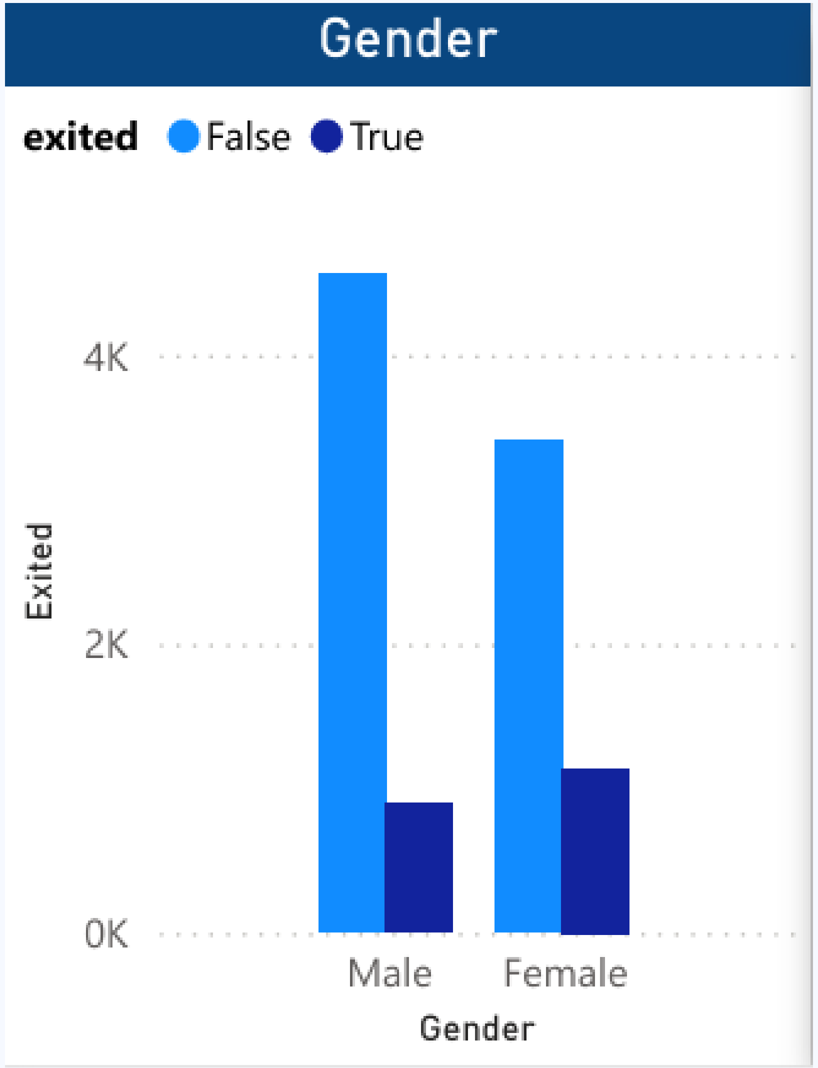
AGE

	Young	Middle	Old
Total Customers	3678	4924	1394
Total Churned Customers	290	1113	634
Churn Rate Per Total Customers	7.9%	22.6%	45.5%



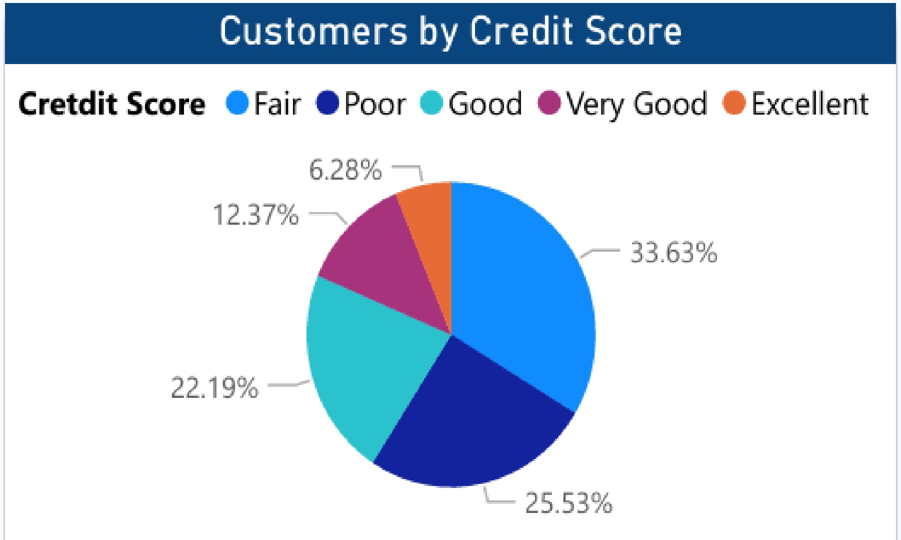
GENDER

	Male	Female
Total Customers	5454	4542
Total Churned Customers	898	1139
Churn Rate Per Total Customers	16.5%	25.1%

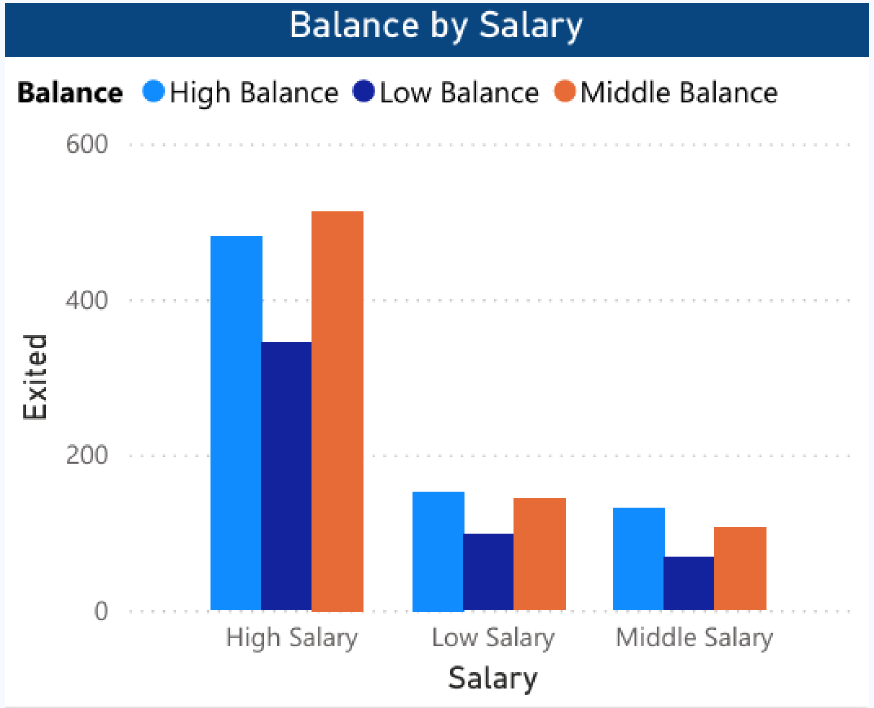


CREDIT SCORE

	Poor	Fair	Good	Very Good	Excellent
Score Range	300-579	580-669	670-739	740-799	800-850
Total Customers	2361	3331	2427	1224	653
Total Churned Customers	520	685	452	252	128
Churn Rate Per Total Customers	22%	20.6%	18.6%	20.6%	19.6%



BANK BALANCE
&
ESTIMATED SALARY



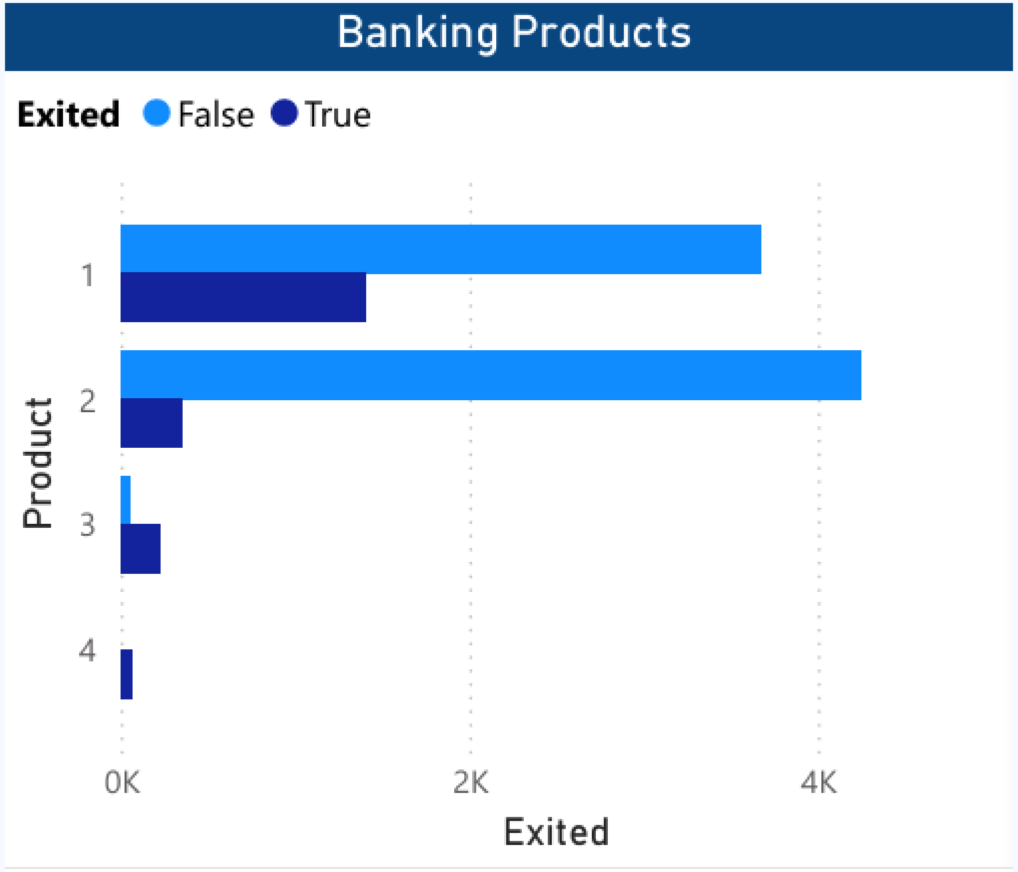
	Low Salary			Middle Salary			High Salary		
Balance	**Low**	**Middle**	**High**	**Low**	**Middle**	**High**	**Low**	**Middle**	**High**
Total Customers	735	627	592	545	459	509	2365	2087	2077
Total Churned Customers	98	143	153	69	105	131	344	513	481
Churn Rate Per Total Customers	13.3%	22.8%	25.8%	12.7%	22.9%	25.7%	14.5%	24.6%	23.2%

BANKING PRODUCTS

	1	2	3	4
Total Customers	5082	4588	266	60
Total Churned Customers	1409	348	220	60
Churn Rate Per Total Customers	27.7%	7.6%	82.3%	100%

Example of Banking Products:

- Deposit Products
- Loan Products
- Credit Products
- Investment Products

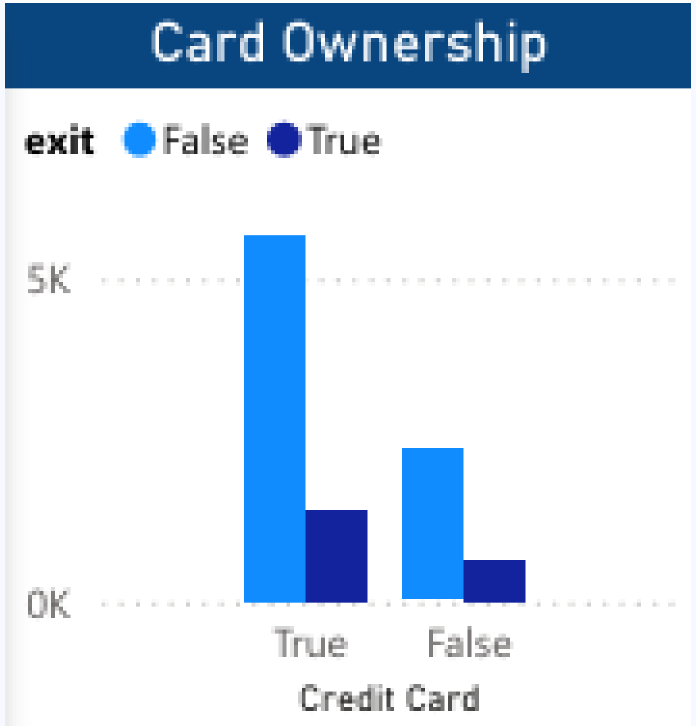


CREDIT CARD OWNERSHIP

	Yes	No
Total Customers	7052	2944
Total Churned Customers	1424	613
Churn Rate Per Total Customers	20.2%	20.8%

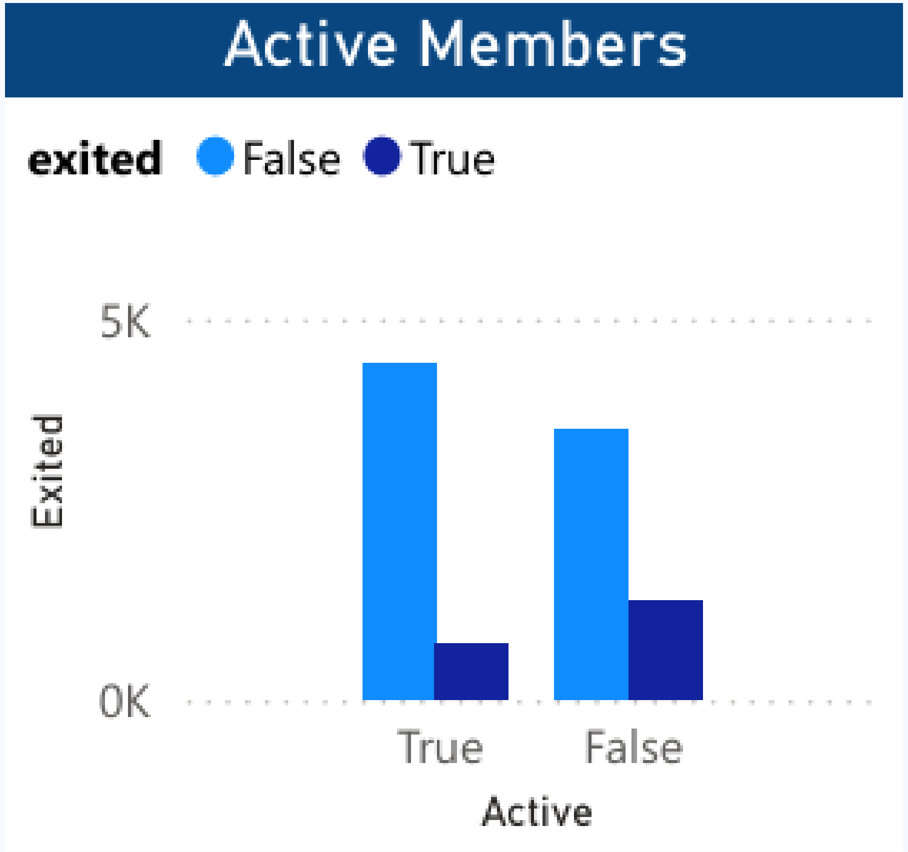
70.5%

Credit Card Ownership Rate



CUSTOMER ACTIVITY

	Active	Inactive
Total Customers	5147	4849
Total Churned Customers	735	1302
Churn Rate Per Total Customers	14.3%	26.9%



MODEL BUILDING

MACHINE LEARNING

DATA SPLITTING

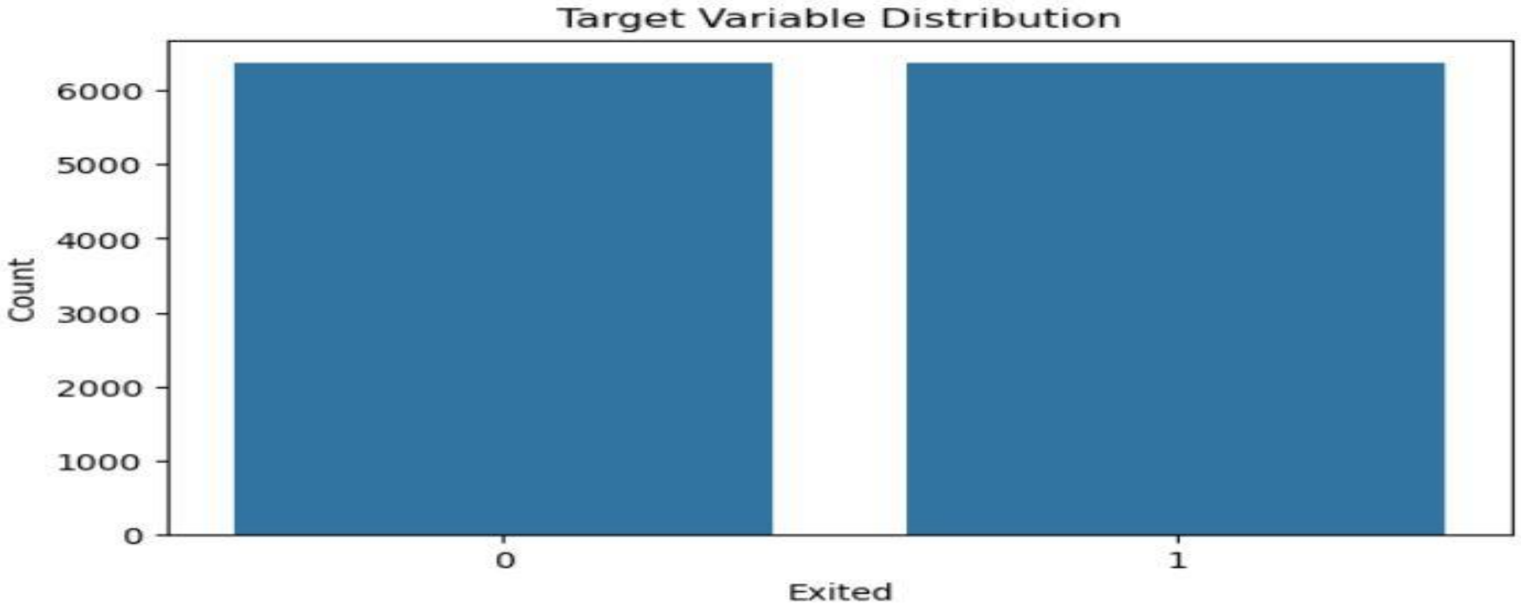
5.1- Spliting data

[61]:
```
# import function of train_test_splite to splite dataset
from sklearn.model_selection import train_test_split

# Features of data
X = df[["creditscore","numofproducts","balance","genderlabel","ageskewed","isactivemember","geographyfrance","geographygermany","geographyspain"]]

#X = df.drop(columns=["creditscore","geography", "gender", "age","tenure","hascrcard","estimatedsalary","exited","creditscorerange","balancerange","estim

y = df["exited"]

# Splite dataset to train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,shuffle=True, stratify=y, random_state=60)

# Display dataset after spliting
display(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
```

```
(7996, 9)
(7996,)
(2000, 9)
(2000,)
```

SMOTE OVER SAMPLE

```
(12734, 9)
(12734,)
(2000, 9)
(2000,)
```



DATA SCALING

5.3- Scaling Data

[65]:

```
from sklearn.preprocessing import StandardScaler

#  Create Scaler
scaler = StandardScaler()

X_train_scaled = pd.DataFrame(scaler.fit_transform(X_train), columns=X_train.columns)
X_test_scaled = pd.DataFrame(scaler.transform(X_test), columns=X_test.columns)
X_train_scaled.head(3)
```

[65]:

	creditscore	numofproducts	balance	genderlabel	ageskewed	isactivemember	geographyfrance	geographygermany	geographyspain
0	1.941631	1.022392	-1.338476	-0.845450	-1.284296	-0.967077	0.928569	-0.757142	-0.637413
1	-2.333735	1.022392	0.931958	1.182802	-0.401395	-0.967077	-1.076926	1.320756	-0.637413
2	2.133065	-0.723035	0.662659	1.182802	-0.289124	1.164177	-1.076926	1.320756	-0.637413

MODEL VERSION 1

- K-Nearest Neighbors (KNN)
- Support Vector Classifier (SVC)
- Logistic Regression
- Decision Tree
- Random Forest
- Optimized Random Forest

K-NEAREST NEIGHBORS (KNN)

Best Hyperparameters:

- n_neighbors: 9
- weights: Distance
- metric: Manhattan

Training Accuracy 99.99%:

Validation Accuracy: 83.80%

Test Accuracy: 83.26%

Conclusion:

- The model shows signs of overfitting.
- Generalization is not ideal.
- Performance on unseen data is lower than expected based on training accuracy

SUPPORT VECTOR CLASSIFIER (SVC)

Best Hyperparameters:

- C: 1
- kernel: RBF
- gamma: Scale

Training Accuracy: 99.99%

Validation Accuracy: 82.08%

Test Accuracy: 81.71%

Conclusion:

- The model is overfitting.
- High training accuracy but much lower validation/test accuracy.
- Needs better regularization or tuning.

LOGISTIC REGRESSION

Best Hyperparameters:

- C: 0.1
- penalty: L1
- solver: SAGA

Training Accuracy: 83.16%

Validation Accuracy: 74.30%

Test Accuracy: 73.38%

Conclusion:

- Moderate overfitting is present.
- Generalization is acceptable, but performance could be improved.

DECISION TREE

Best Hyperparameters:

- criterion: Entropy
- max_depth: 10

Training Accuracy: 75.66%

Validation Accuracy: 81.53%

Test Accuracy: 81.36%

Conclusion:

- Good generalization capability.
- No overfitting — model generalizes well to new data.
- Training performance could be slightly improved, but validation/test performance is strong.

RANDOM FOREST

Best Hyperparameters:

- n_estimators: 200
- max_depth: 20

Training Accuracy: 94.01%

Validation Accuracy: 85.80%

Test Accuracy: 86.09%

Conclusion:

- Strong generalization with minimal overfitting.
- Excellent performance on both validation and test sets.

OPTIMIZED RANDOM FOREST

Best Hyperparameters:

- n_estimators: 300
- max_depth: 12

Training Accuracy: 90.55%

Validation Accuracy: 85.80%

Test Accuracy: 85.52%

Conclusion:

- Excellent generalization.
- Best performing model overall.
- Minimal overfitting, strong and stable performance

CONCLUSION - MODEL COMPARISON

Model	Training Accuracy	Validation Accuracy	Test Accuracy	Conclusion
KNN	99.99%	83.80%	83.26%	Good generalization, slight overfitting
SVC	99.99%	82.08%	81.71%	Stable and reliable performance
Logistic Regression	83.16%	74.30%	73.38%	Moderate; scope for improvement
Decision Tree	75.66%	81.53%	81.36%	Balanced model
Random Forest	94.01%	85.80%	86.09%	High performance, great generalization
Optimized RF	90.55%	84.65%	85.52%	Best performing model overall

Optimized Random Forest demonstrated best generalization.

MODEL VERSION 2
BEFORE TUNING

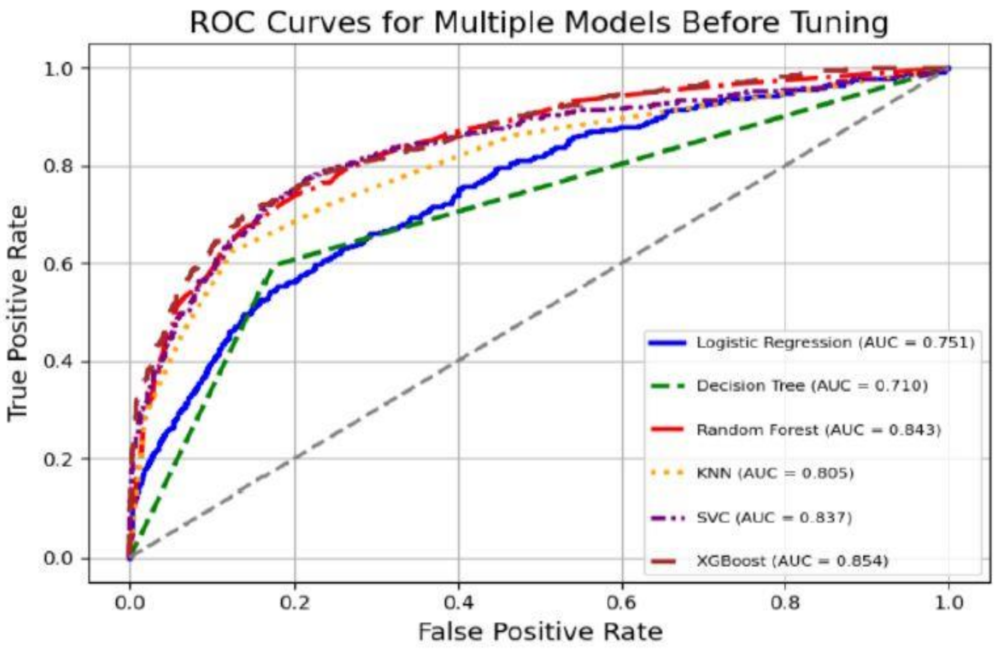
- **Logistic Regression**
- **Decision Tree**
- **Random Forest**
- **KNN**
- **SVC**
- **XGBoost**

MODEL EVALUATION BEFORE TUNING

2. Model Evaluation Before Tuning

- The following table shows the evaluation metrics before any hyperparameter tuning:

Algorithm	Train Accuracy	Test Accuracy	Precision	Recall	F1-score	AUC
Logistic Regression	0.8238	0.7830	0.4698	0.4951	0.4821	0.7506
Decision Tree	0.9998	0.7785	0.4664	0.5956	0.5231	0.7104
Random Forest	0.9998	0.8325	0.5884	0.5956	0.5920	0.8430
KNN	0.8996	0.8240	0.5619	0.6225	0.5907	0.8052
SVC	0.8677	0.8360	0.5985	0.5956	0.5971	0.8374
XGBoost	0.9473	0.8535	0.6584	0.5858	0.6200	0.8537



CONCLUSION

Performance Analysis (Before Tuning):

- XGBoost achieved the highest Test Accuracy (85.35%) and AUC (0.8537), indicating strong predictive performance.
- Decision Tree and Random Forest models showed signs of overfitting due to very high training accuracy.
- SVC and XGBoost provided the most balanced results in terms of both Precision and Recall.
- And with best AUC xgboost 0.85, random forest 0.84, and SVC 0.83.

TUNING THE BEST MODELS

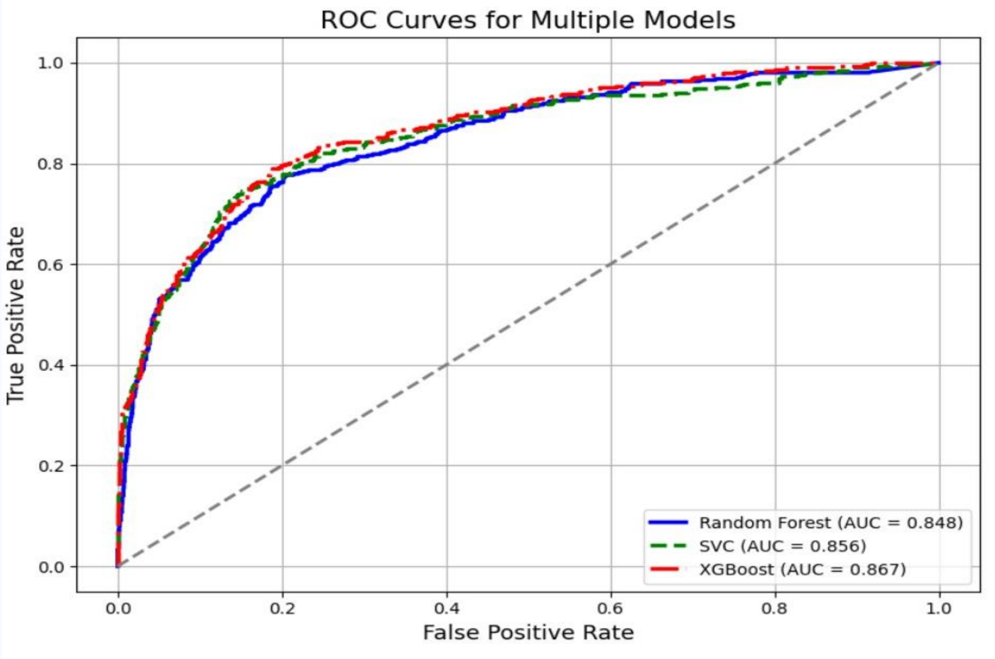
- Random Forest
- SVC
- XGBoost

MODEL EVALUATION AFTER TUNING

5. Model Evaluation After Hyperparameter Tuning

- After tuning, the performance of the top models improved. The results are summarized below:

Algorithm	Train Accuracy	Test Accuracy	Precision	Recall	F1-score	AUC
Random Forest	0.9995	0.8320	0.5857	0.6029	0.5942	0.8434
SVC	0.8784	0.8465	0.6241	0.6225	0.6233	0.8319
XGBoost	0.9109	0.8510	0.6410	0.6127	0.6266	0.8625

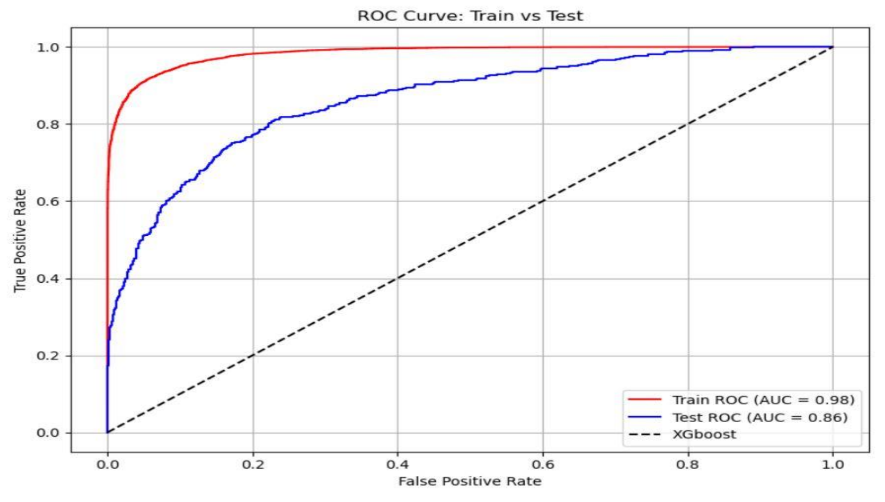


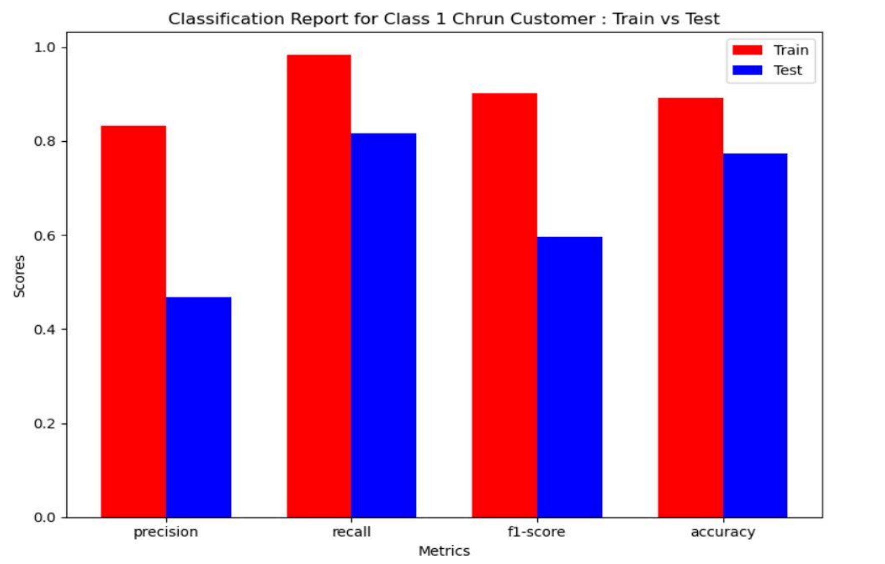
Conclusion

Final Conclusion and Selected Mode:

- Based on the evaluation after tuning, **XGBoost** remains the best-performing model with the highest AUC (0.8625) and a well-balanced Precision and Recall. Hence, it is selected as the final model for deployment.

- **The best hyperparameters for XGBoost are:-**
  - colsample_bytree = 0.8 - gamma = 0.1 - learning_rate = 0.05 - max_depth = 6 - n_estimators = 300 - scale_pos_weight = 5 - subsample = 0.8 - random_state = 60





DEPLOYMENT

MODEL

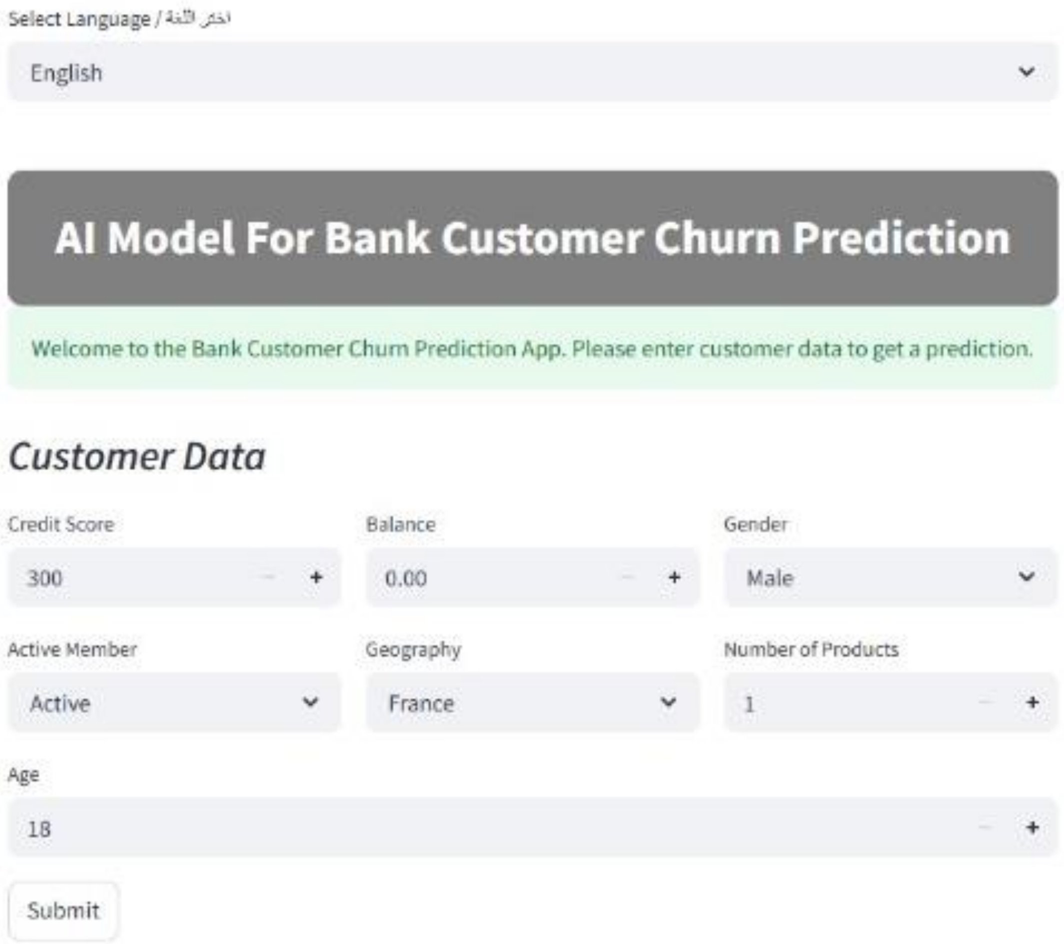
SAVING MODEL WEIGHTS

1- Saving Model Weights

```
[103]: pip install joblib

       Requirement already satisfied: joblib in c:\users\hassan\anaconda3\lib\site-packages (1.4.2)
       Note: you may need to restart the kernel to use updated packages.

[105]: from joblib import dump, load

       # Save Scaler
       dump(scaler, 'scaler.joblib')

       # Save the trained model to a file
       dump(model, 'model.joblib')

[105]: ['model.joblib']
```

STREAMLIT API & GUI

```python
import streamlit as st
import joblib

def run_app():
    # Load the trained model and scaler
    model1 = joblib.load('model.joblib')  # Load the trained model
    scaler1 = joblib.load('scaler.joblib')  # Load the fitted scaler

    # Define label translations for multilingual support
    labels = {
        'en': {
            "app_title": "AI Model For Bank Customer Churn Prediction",
            "title": "Customer Data",
            'Age': 'Age',
            'Balance': 'Balance',
            'Credit Score': 'Credit Score',
            'Gender': 'Gender',
            'Male': 'Male',
            'Female': 'Female',
            'Active Member': 'Active Member',
            'Active': 'Active',
            'Not Active': 'Not Active',
            'Geography': 'Geography',
            'Geography Options': ['France', 'Germany', 'Spain'],
            'Number of Products': 'Number of Products',
            'Credit Card Ownership': 'Credit Card Ownership',
            'Submit': 'Submit',
            'Prediction': 'Prediction',
            'Churn Prediction': 'Customer Churn Prediction'
        },
        'ar': {
            "app_title": "نموذج ذكاء اصطناعي لتوقع مغادرة عملاء البنك",
            "title": "بيانات العميل",
            'Age': 'العمر',
            'Balance': 'الرصيد',
            'Credit Score': 'درجة الائتمان',
            'Gender': 'الجنس',
            'Male': 'ذكر',
            'Female': 'أنثى',
            'Active Member': 'عضو نشط',
            'Active': 'نشط',
```

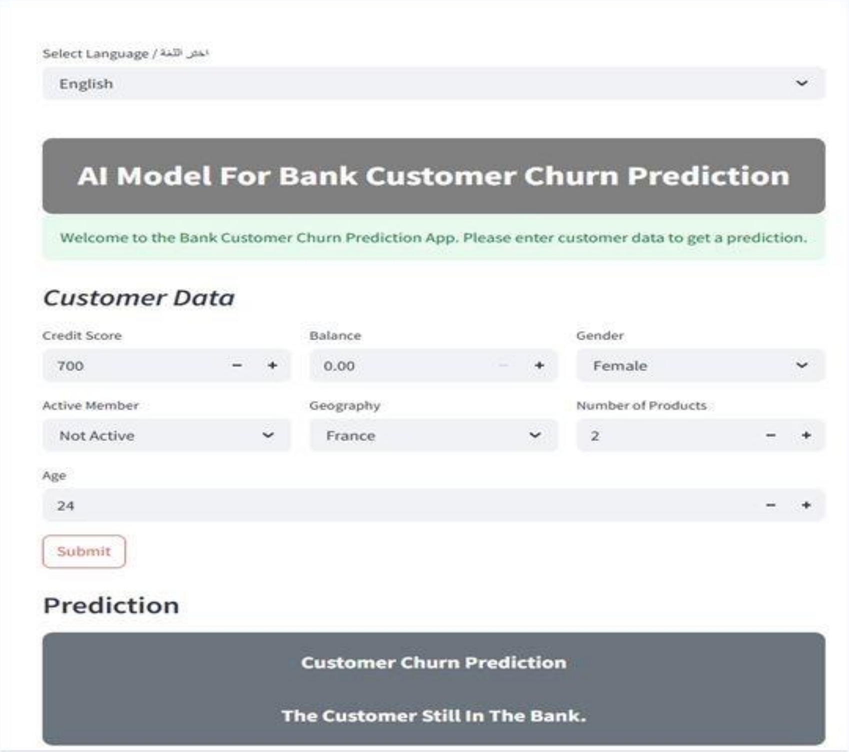


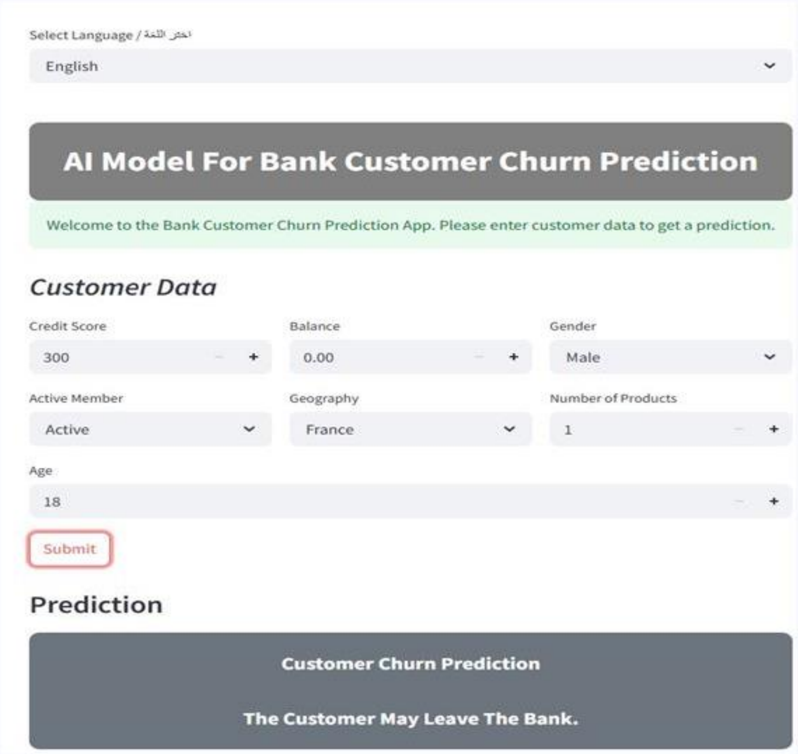
STREAMLIT CLOUD REAL TIME REQUIREMENTS

```
pandas==2.2.2
numpy==2.0.2
matplotlib==3.10.0
streamlit==1.44.1
xgboost==2.1.4
scikit-learn==1.6.0
seaborn==0.13.2
category_encoders==2.8.1
joblib==1.4.2
```

4- REAL TIME STREAMLIT CLOUD

& TEST MODEL WITH REAL DATA

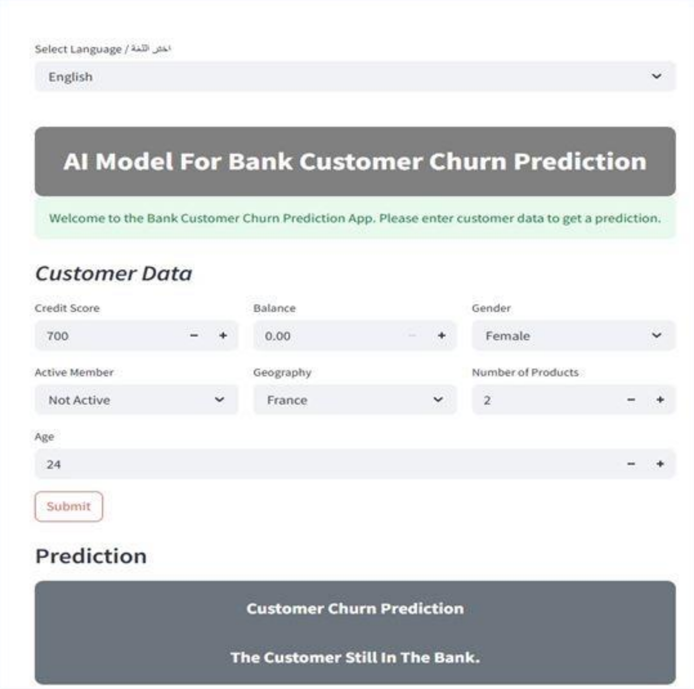




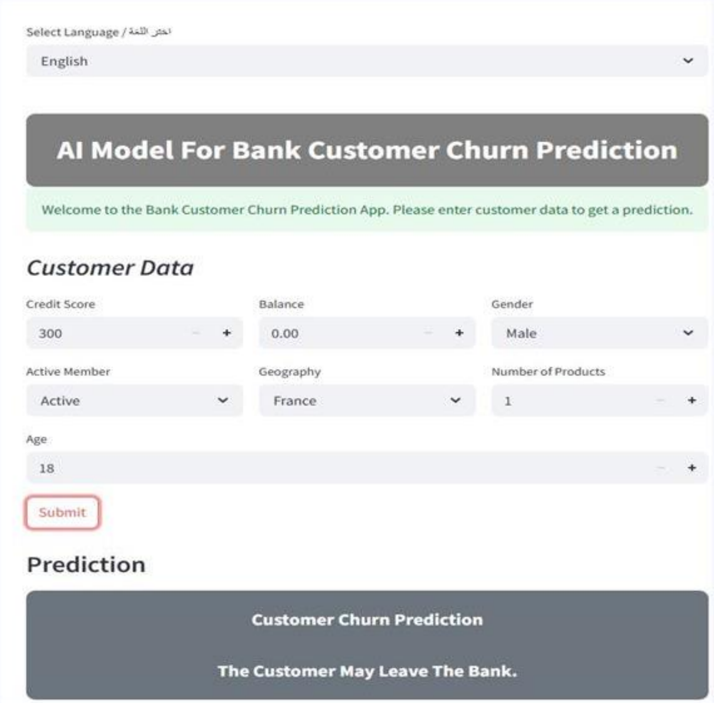
AND YOU CAN VISIT AND USE OUR PROJECT THROUGH

LINK : BANK_CUATOMER_CHURN_PREDICTION

REAL TIME STREAMLIT & MODEL TESTING WITH REAL DATA



CUSTOMER STAY



CUSTOMER EXIT

FUTURE ENHANCEMENTS/IMPROVEMENTS

1. Improve data preprocessing by adding and exploring more features.
2. Collect and prepare data for model training and make it recurring training.
3. Utilize an ensemble model for better performance.
4. Add additional functions to enhance AutoML capabilities.

Questions?

THANK YOU