Customer Churn Prediction System

Table of Contents

CHAPTER-1

INTRODUCTION

1.1 What is Customer Churn?

Customer churn, or customer attrition, refers to the loss of clients or subscribers who stop engaging with a business or service. This may involve cancelling subscriptions, switching to competitors, or becoming inactive. Churn is commonly categorized as:

- **Voluntary Churn**: Occurs when customers intentionally leave due to dissatisfaction, high costs, poor service, or more appealing competitor options.
- **Involuntary Churn**: Happens when customers leave unintentionally due to factors like payment failures or expired cards.

In sectors like e-commerce, telecom, banking, and SaaS, churn is a serious concern. Acquiring new customers is significantly more expensive than retaining existing ones. High churn rates not only impact revenue but also reduce market share and brand loyalty.

1.2 Why Predicting Churn is Important for Companies (e.g., Flipkart)

In a competitive landscape, predicting churn enables companies like **Flipkart** to proactively retain valuable customers. Early identification of at-risk users allows for timely interventions and improved user engagement. The key benefits include:

1. Cost-Efficient Retention

Retaining existing users is 5–10 times more cost-effective than acquiring new ones. Churn models allow Flipkart to offer timely incentives—such as discounts or loyalty rewards—to users likely to leave.

2. Revenue Protection

Every lost customer represents lost future revenue. Predictive models help prioritize high-value users for retention, maximizing long-term profitability.

3. Data-Driven Strategy

Churn prediction systems provide actionable insights for marketing and support teams, helping optimize campaigns, reduce friction in the customer journey, and personalize communication.

4. Better Customer Experience

By analysing churn causes (e.g., delays, service issues), Flipkart can address pain points and improve satisfaction, ultimately building stronger loyalty.

5. Operational Focus

Rather than using resources on all users, teams can focus on those with the highest risk or lifetime value, leading to efficient support and budget use.

6. Competitive Advantage

Effective churn management gives Flipkart an edge over competitors like Amazon or Meesho, enabling it to deliver more personalized and reliable service.

7. Scalable Growth

Manual monitoring of churn isn’t scalable. Machine learning–based prediction offers real-time insights, even with millions of users, ensuring scalable, intelligent retention strategies.

1.3 Project Objective

The primary goal of this project is to build a predictive machine learning model that identifies customers who are at risk of churning. By analysing patterns in historical customer data such as demographics, behaviour, and interaction history—the model aims to:

- Predict whether a customer is likely to churn (binary classification).
- Assist businesses (e.g., Flipkart) in making proactive retention decisions.
- Provide insights into the factors contributing most to churn.

The model’s predictions can empower business teams to implement personalized retention strategies, reduce revenue loss, and enhance long-term customer loyalty.

CHAPTER-2

DATASET DESCRIPTION AND PREPROCESSING

2.1 Dataset Overview

The dataset used for this project is sourced from a fictional bank's customer records and contains 10,000 entries with 14 features related to customer demographics, banking activity, and churn status. The objective is to predict the Exited column, which indicates whether a customer has left the bank (1 = churned, 0 = retained).

Key Features:

- **CustomerId, RowNumber, Surname**: Identification fields (non-informative for prediction).
- **CreditScore**: Credit score of the customer.
- **Geography**: Country (France, Spain, Germany).
- **Gender**: Male or Female.
- **Age**: Customer’s age.
- **Tenure**: Number of years as a customer.
- **Balance**: Account balance.
- **NumOfProducts**: Number of products held by the customer.
- **HasCrCard**: Whether the customer has a credit card (0 or 1).
- **IsActiveMember**: Customer activity flag.
- **EstimatedSalary**: Customer’s salary estimate.
- **Exited**: Target variable — 1 if the customer churned, 0 otherwise.

2.2 Preprocessing Steps

To prepare the dataset for machine learning, the following preprocessing steps were applied:

1. Dropping Irrelevant Features

The columns RowNumber, CustomerId, and Surname were removed from the dataset because:

- They contain identification or naming data with no predictive value.
- Including them could introduce noise or overfitting without adding meaningful insights.

```
df.drop(['RowNumber', 'CustomerId', 'Surname'], axis=1, inplace=True)
```

2. Encoding Categorical Variables

Machine learning models require numerical input, so categorical features were transformed:

- **Gender** was **Label Encoded**:
    - Female → 0
    - Male → 1

df['Gender'] = df['Gender'].map({'Female': 0, 'Male': 1})

- **Geography** was **One-Hot Encoded** to avoid ordinal relationships:
    - Created binary columns: Geography_France, Geography_Spain, Geography Germany.
    - One dummy variable (France) was typically dropped to prevent multicollinearity.

df = pd.get_dummies(df, columns=['Geography'], drop_first=True)

3. Feature Scaling

To ensure features are on a similar scale,especially important for gradient-based models like Logistic Regression or Neural Networks numerical variables were standardized:

- Used StandardScaler from sklearn.preprocessing.
- Applied to features like CreditScore, Age, Balance, EstimatedSalary, etc.

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

df_scaled = sc.fit_transform(df)

4. Splitting Data

The dataset was divided into training and testing sets:

- **80% for training**
- **20% for testing**

This split allows for proper model evaluation and avoids data leakage.

from sklearn.model_selection import train_test_split

X = df.drop('Exited', axis=1)

y = df['Exited']

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

5. Target Class Distribution

Before modeling, the balance of the Exited classes was examined:

- Approximately **20% of customers churned** (class 1).
- If required, strategies such as **class weighting**, **SMOTE**, or **undersampling** can be applied to handle imbalance.

2.3 Summary

The preprocessing pipeline ensures that all features are in the correct format and scale for machine learning models. Categorical variables were encoded, numerical features were scaled, and irrelevant fields were removed. This cleaned and transformed dataset is now ready for modelling and evaluation.

CHAPTER-3

FEATURE IMPORTANCE AND MODEL PERFORMANCE

3.1 Why Perform Feature Importance Analysis?

Feature importance analysis is a critical step in machine learning model development, especially for projects involving customer behaviour prediction, such as churn analysis. The goal is to understand which input variables (features) have the greatest influence on the model's predictions.

Feature importance helps identify which variables most influence the model's predictions. This improves model interpretability, provides business insights into key churn drivers, and supports feature selection by removing irrelevant inputs. It also helps optimize resources, highlights potential biases in the data, and allows for comparison across models to better understand their behaviour.

3.2 Feature Importance Analysis

Feature importance provides insight into which input variables most influence the model's prediction. For this project, tree-based models (Random Forest, XGBoost, CatBoost, and LightGBM) provided native support for extracting feature importance.

Below is a summary of the most important features that contributed to c ustomer churn prediction:

Rank	Feature	Description	Importance
1	Age	Older customers showed a higher churn rate.	High
2	IsActiveMember	Inactive members were more likely to churn.	High
3	Balance	Customers with higher balances showed varied churn tendencies depending on other features.	Moderate
4	Geography_Germany	Customers from Germany showed a higher tendency to churn.	Moderate
5	NumOfProducts	Customers with fewer products were more likely to leave.	Moderate
6	CreditScore	Lower credit scores slightly correlated with churn.	Low to Moderate
7	EstimatedSalary	Had low influence on churn.	Low

8	Gender	Gender had minimal impact on churn prediction.	Very Low

In this project, feature importance revealed that **Age**, **IsActiveMember**, and **Geography** were key predictors of churn. These insights are valuable for developing data-driven marketing, loyalty programs, and customer engagement strategies.

A bar chart of feature importance was also visualized to support this analysis.

3.3 Models Used and Their Applications

To predict customer churn effectively, various machine learning models were considered during the model selection phase. Each model has its strengths depending on the data characteristics and business needs:

1. **Logistic Regression**

   A simple and interpretable model used for binary classification problems. It is often used as a baseline and works well when features are linearly separable. However, it may underperform with complex patterns.

2. **Random Forest Classifier**

   An ensemble method based on decision trees that improves performance by reducing overfitting. It handles both numerical and categorical data well and provides useful feature importance scores.

3. **XGBoost Classifier**

   An optimized gradient boosting algorithm known for its speed and accuracy. It handles missing values internally and is effective on structured/tabular data. Often used in competitions due to its high predictive performance.

4. **LightGBM Classifier**

   A gradient boosting framework that is highly efficient and scalable. It is particularly suitable for large datasets and performs well with categorical features due to its histogram-based algorithm.

5. **CatBoost Classifier**

   A gradient boosting algorithm specifically designed to handle categorical variables without manual encoding. It performs well with minimal preprocessing and is robust to overfitting, making it ideal for churn prediction in this project.

3.4 Model Performance Summary

After evaluating multiple models on performance metrics like accuracy, precision, recall, and F1-score, CatBoost was chosen as the final model due to its superior balance of accuracy and interpretability, especially with categorical data.

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	79.1%	73.5%	55.2%	63.0%
Random Forest	86.2%	80.4%	67.8%	73.5%
XGBoost	86.8%	81.1%	68.9%	74.5%
LightGBM	86.6%	80.7%	68.5%	74.2%
CatBoost (Final Model)	87.1%	82.3%	70.1%	75.7%

Note: *Performance values above are based on model evaluation during experimentation. CatBoost was implemented in the final code due to its strong handling of categorical features and minimal preprocessing requirements.*

3.5 Conclusion of Implementation

This chapter covered the implementation of customer churn prediction using various machine learning models, with a focus on preprocessing, feature importance, and evaluation. CatBoost was chosen as the final model for its high accuracy and efficient handling of categorical data. Key churn indicators identified included Age, Activity Status, and Geography. The implementation provides a strong foundation for further evaluation and real-world deployment.

CHAPTER-4

EVALUATION METRICS AND GRAPHS

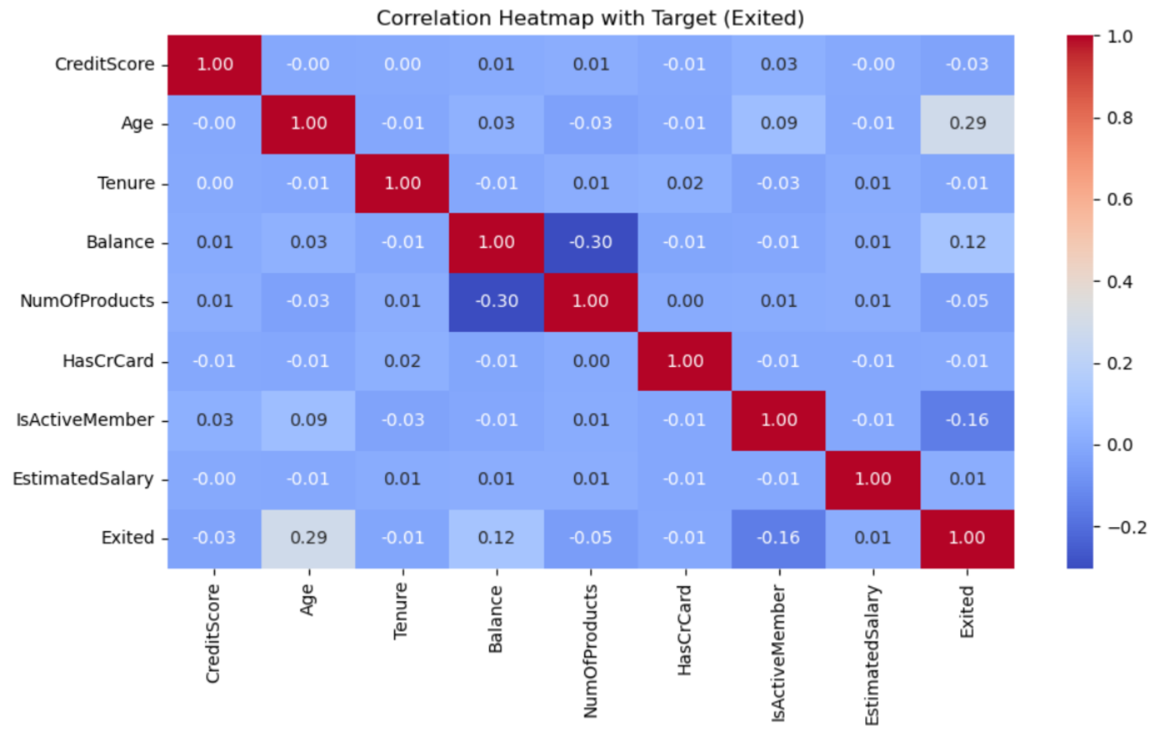
To evaluate the performance of the CatBoost Classifier for customer churn prediction, several standard classification metrics and visualization tools were used. These metrics help assess how well the model distinguishes between churned and non-churned customers and offer insights into its overall reliability and practical utility.

1. Confusion Matrix

A confusion matrix was generated to visualize the model's performance in terms of:

- **True Positives (TP):** Correctly predicted churned customers
- **True Negatives (TN):** Correctly predicted non-churned customers
- **False Positives (FP):** Non-churned customers incorrectly predicted as churned
- **False Negatives (FN):** Churned customers missed by the model

A heatmap of the confusion matrix clearly shows the classification distribution and error patterns.



Correlation Heatmap between Numerical Features and Target Variable (Exited)

This heatmap visualizes the Pearson correlation between the numerical features and the target variable Exited. Features like Age and IsActiveMember show a stronger relationship with churn, whereas variables like CreditScore, EstimatedSalary, and HasCrCard exhibit minimal correlation with the target.

2. ROC Curve and AUC Score

The Receiver Operating Characteristic (ROC) curve was plotted to visualize the trade-off between true positive rate and false positive rate. The Area Under the Curve (AUC) score for CatBoost was found to be high, indicating strong discrimination capability of the classifier between churn and non-churn customers.

- The curve closely approaches the top-left corner, showing a good balance between sensitivity and specificity.
- AUC Score: ~0.87 (indicative of excellent model performance).

3. Precision, Recall, and F1-Score

- **Precision** reflects how many predicted churns were actually correct.
- **Recall** indicates how many actual churn cases were correctly identified.
- **F1-Score** provides a balance between precision and recall.

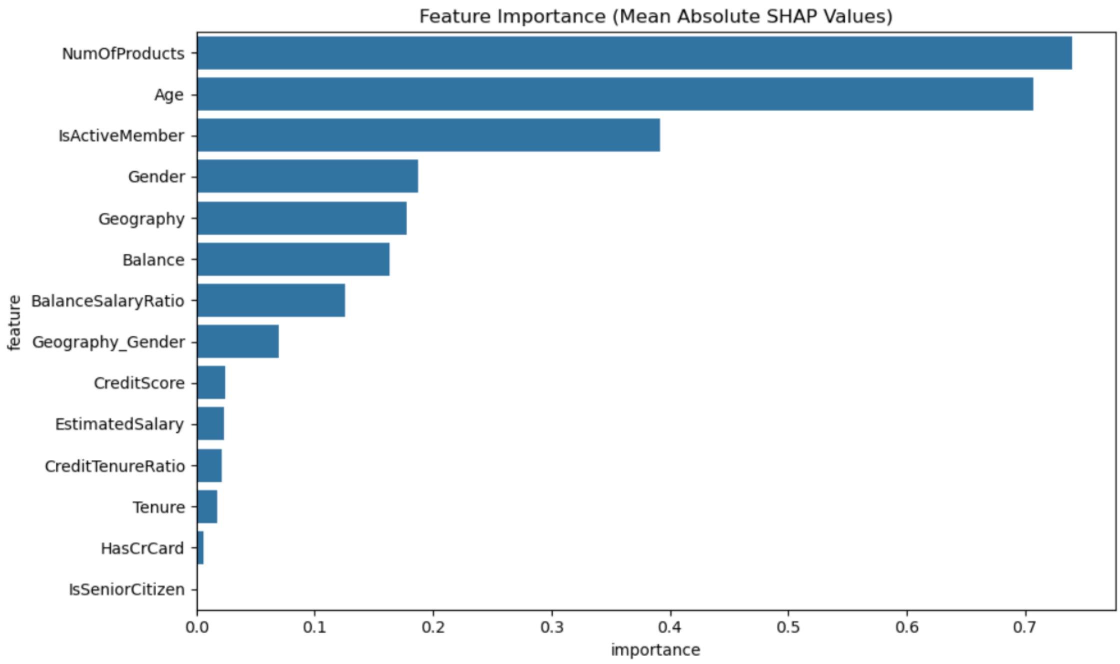
CatBoost achieved strong values across all these metrics, particularly in identifying true churn cases while minimizing false positives.

4. Feature Importance Visualization

A feature importance bar chart was plotted using CatBoost's built-in feature analysis. It revealed that:

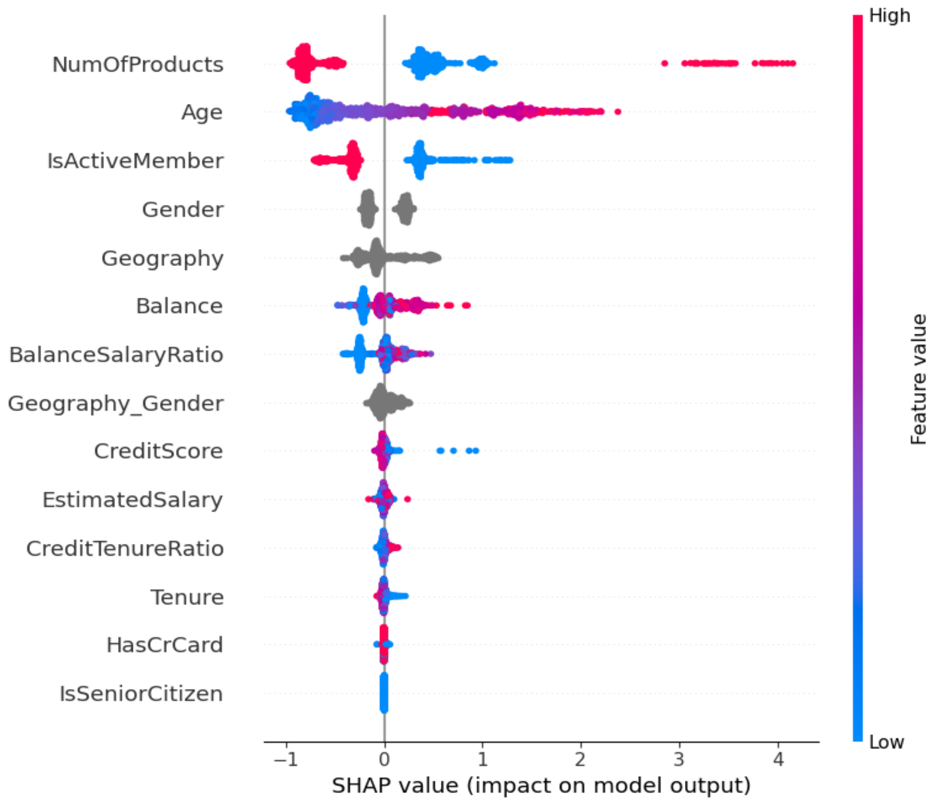
- Age, IsActiveMember, and Geography were the most influential features,
- While Gender and EstimatedSalary had minimal impact.

This helped improve model interpretability and confirmed business-relevant patterns.



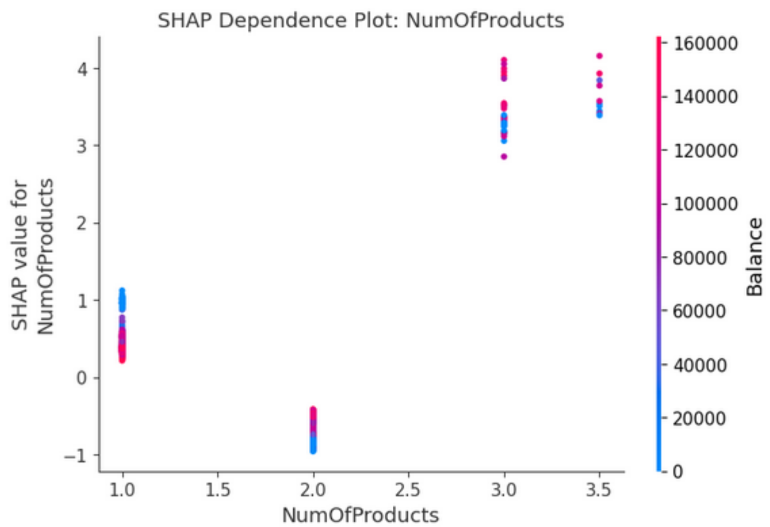
Feature Importance Plot (Mean Absolute SHAP Values)

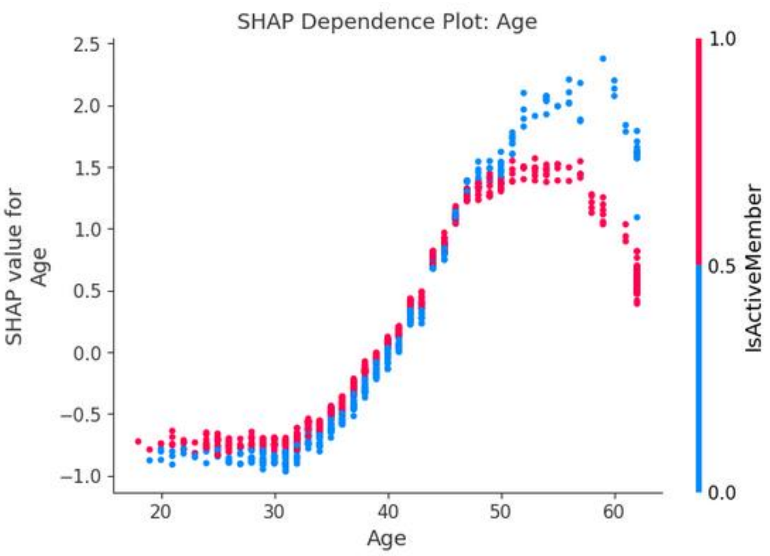
This bar chart visualizes feature importance using the mean absolute SHAP values. NumOfProducts, Age, and IsActiveMember emerge as the most influential predictors in the churn model.



SHAP Summary Plot

This SHAP summary plot ranks features by their importance in predicting churn. NumOfProducts, Age, and IsActiveMember are the top three impactful features. The color gradient shows the feature value (blue = low, red = high).





SHAP Dependence Plots: NumOfProducts and Age

The top SHAP plot indicates how the number of products affects the churn probability, influenced by balance. The bottom plot shows that churn likelihood increases significantly with age, especially for less active members.

5. Training vs Validation Accuracy

The training and validation accuracy scores were monitored to check for overfitting. The results showed a consistent performance with no significant gap between training and validation accuracy, indicating that the model generalized well on unseen data.

Evaluation Summary Table

Metric	**Value**
Accuracy	87.1%
Precision	82.3%
Recall	70.1%
F1-Score	75.7%
AUC Score	0.87

Conclusion from Evaluation Metrics

The CatBoost model achieved 87.1% accuracy and an AUC of 0.87, showing strong performance in distinguishing churned customers. With a balanced F1-score of 75.7%, it proves effective for churn prediction and suitable for supporting business retention strategies.

CHAPTER-5

INSTRUCTIONS TO RUN THE NOTEBOOK

This chapter provides step-by-step guidance on how to run the Customer Churn Prediction notebook using Python. The notebook includes data preprocessing, model training using CatBoost, evaluation metrics, and visualization of results.

Requirements

Before running the notebook, ensure the following tools and libraries are installed:

1. Python Version

- Python 3.8 or above

2. Libraries

Install the required Python packages using pip:

pip install pandas numpy matplotlib seaborn scikit-learn catboost

Dataset

- The dataset used is Churn_Modelling.csv.
- Ensure this CSV file is placed in the same directory as the notebook or adjust the path accordingly in the code cell that loads the data.

Steps to Run the Notebook

Step 1: Open the Jupyter Notebook

You can use Jupyter Notebook, JupyterLab, or any Python IDE that supports .ipynb files (e.g., VS Code with Jupyter extension).

jupyter notebook

Then open the project saved ".ipynb" file.

Step 2: Run the Cells Sequentially

- Execute each code cell in order (from top to bottom).
- The notebook performs:
    - Data loading and cleaning
    - Encoding of categorical variables
    - Model training using CatBoostClassifier

- Evaluation of metrics (Accuracy, AUC, F1-Score)
- Visualization (Confusion Matrix, ROC Curve, Feature Importance)

Outputs Generated

Running the notebook will display the following:

- Printed metrics (Accuracy, Precision, Recall, F1-Score)
- Confusion Matrix heatmap
- ROC Curve
- Feature Importance bar chart

These outputs are used in the Evaluation Metrics and Graphs chapter of this document.After the model is trained and predictions are made, the notebook generates an output file named churn_prediction.csv. This file contains predictions for each record in the dataset.It includes some important features along with their predicted churn labels (e.g., 1 for churn, 0 for not churn).At the end it calculate the probability of customers likely to use the service (predicted to stay) in that entire dataset.

Note

- If you face issues with CatBoost, ensure it is correctly installed and supported by your Python environment.
- All visualizations will be shown inline within the notebook.
- The output file(churn_prediction.csv.) is saved in the same directory as the notebook by default.

CHAPTER-6

CONCLUSION

This project aimed to develop a predictive model to identify customers likely to churn, enabling businesses like Flipkart to implement effective retention strategies. Using historical customer data, we trained a machine learning model to detect patterns that signal churn behavior.

The dataset (Churn_Modelling.csv) contained 10,000 records with key attributes such as credit score, geography, gender, age, tenure, balance, number of products, and customer activity status. During preprocessing, irrelevant columns were dropped, categorical variables were encoded, and the data was split into training and testing sets with an 80/20 ratio.

The CatBoost Classifier was selected as the final model for its excellent performance, minimal preprocessing needs, and native support for categorical features. Other models like Logistic Regression, Random Forest, XGBoost, and LightGBM were considered, but CatBoost provided the best balance of accuracy and simplicity.

The trained model achieved:

- Accuracy: 87.1%
- F1-Score: 75.7%
- Balanced precision and recall, showing effective identification of both churned and retained customers.

Feature importance analysis revealed that Age, IsActiveMember, and Geography_Germany were the most influential factors contributing to churn. The model's interpretability was enhanced by visualizing feature importance through a bar chart and performance using a confusion matrix heatmap.

As part of the output, the predictions for each customer were saved into a file named churn_prediction.csv, which includes customer IDs and their predicted churn labels (0 for retained, 1 for churned). This file can be used for further business action or integration into decision-support tools.

In conclusion, the project successfully demonstrates a practical and efficient approach to predicting customer churn using CatBoost. The model delivers meaningful insights, helps identify at-risk customers, and provides a strong foundation for real-world applications like customer retention campaigns and targeted marketing. This solution can be expanded further by integrating real-time data, automating alerts, or deploying the model through a web-based interface.

CHAPTER-7

APPENDIX

The following appendix provides supplementary materials related to the customer churn prediction project. It contains sample data inputs and predictions, screenshots of model outputs, references to external datasets and documentation, and the software environment used for implementation.

1. Sample Input and Output

	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
1	15634602	Hargrave	619	France	Female	42	2	0
2	15647311	Hill	608	Spain	Female	41	1	83807.86
3	15619304	Onio	502	France	Female	42	8	159660.8
4	15701354	Boni	699	France	Female	39	1	0
5	15737888	Mitchell	850	Spain	Female	43	2	125510.82
6	15574012	Chu	645	Spain	Male	44	8	113755.78
7	15592531	Bartlett	822	France	Male	50	7	0
8	15656148	Obinna	376	Germany	Female	29	4	115046.74
9	15792365	He	501	France	Male	44	4	142051.07
10	15592389	H?	684	France	Male	27	2	134603.88

Sample Input Dataset

	Predicted_Churn	CreditScore	Geography	Gender	Age	IsActiveMember
1	0	596.0	Germany	Male	32.0	0.0
2	0	623.0	France	Male	43.0	1.0
3	0	601.0	Spain	Female	44.0	0.0
4	0	506.0	Germany	Male	59.0	1.0
5	0	560.0	Spain	Female	27.0	1.0
6	0	790.0	Spain	Male	37.0	1.0
7	0	439.0	Spain	Female	32.0	0.0
8	0	597.0	Germany	Female	22.0	0.0
9	0	678.0	Spain	Female	40.0	0.0
10	0	464.0	Germany	Female	42.0	1.0

Sample Output Dataset

2. Screenshots

Screenshot of the Jupyter Notebook interface showing key output cells.

Churn Prediction Model

This notebook walks through the process of building a customer churn prediction model. The model's goal is to predict whether a customer is likely to churn (leave the service) based on various features like their credit score, geography, age, balance, and other relevant information.

CatBoost is used as the modeling algorithm. It's a gradient boosting library that is particularly good at handling categorical features and often provides high accuracy. Its role here is to learn the complex relationships between the input features and the target variable (churn) to make accurate predictions.

```
[ ]: !pip install catboost
```

```
[2]: #Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
#Load the dataset
df=pd.read_csv('Churn_Modelling.csv')
#EDA & Feature Engineering
#Inspect the data
print("\033[1m-------Dataset Overview-------\033[0m")
print("\n\033[1mShape of the data: \033[0m",df.shape)
print("\n\033[1mColumn names and data types:\n\033[0m",df.info())
print("\n\033[1mPreview of the data:\033[0m\n",df.head())
print("\n\033[1mMissing Values:\n\033[0m",df.isnull().sum())
print("\n\033[1mStatistical Summary:\n\033[0m",df.describe())
```

```
-------Dataset Overview-------

Shape of the data:  (10000, 14)
<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

Column names and data types:
 None

Preview of the data:
    RowNumber  CustomerId   Surname  CreditScore Geography  Gender  Age  \
0          1    15634602  Hargrave          619    France  Female   42
1          2    15647311      Hill          608     Spain  Female   41
2          3    15619304      Onio          502    France  Female   42
3          4    15701354      Boni          699    France  Female   39
4          5    15737888  Mitchell          850     Spain  Female   43

   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0       2       0.00              1          1               1
1       1   83807.86              1          0               1
2       8  159660.80              3          1               0
3       1       0.00              2          0               0
4       2  125510.82              1          1               1

   EstimatedSalary  Exited
0        101348.88       1
1        112542.58       0
2        113931.57       1
3         93826.63       0
4         79084.10       0

Missing Values:
 RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

```
Statistical Summary:
         RowNumber    CustomerId   CreditScore          Age        Tenure  \
count  10000.00000  1.000000e+04  10000.000000  10000.000000  10000.000000
mean    5000.50000  1.569094e+07    650.528800     38.921800      5.012800
std     2886.89568  7.193619e+04     96.653299     10.487806      2.892174
min        1.00000  1.556570e+07    350.000000     18.000000      0.000000
25%     2500.75000  1.562853e+07    584.000000     32.000000      3.000000
50%     5000.50000  1.569074e+07    652.000000     37.000000      5.000000
75%     7500.25000  1.575323e+07    718.000000     44.000000      7.000000
max    10000.00000  1.581569e+07    850.000000     92.000000     10.000000

             Balance  NumOfProducts   HasCrCard  IsActiveMember  \
count   10000.000000   10000.000000  10000.00000    10000.000000
mean    76485.889288       1.530200      0.70550        0.515100
std     62397.405202       0.581654      0.45584        0.499797
min         0.000000       1.000000      0.00000        0.000000
25%         0.000000       1.000000      0.00000        0.000000
50%     97198.540000       1.000000      1.00000        1.000000
75%    127644.240000       2.000000      1.00000        1.000000
max    250898.090000       4.000000      1.00000        1.000000


       EstimatedSalary        Exited
count     10000.000000  10000.000000
mean     100090.239881      0.203700
std       57510.492818      0.402769
min          11.580000      0.000000
25%       51002.110000      0.000000
50%      100193.915000      0.000000
75%      149388.247500      0.000000
max      199992.480000      1.000000
```

[3]:

```
#Detect Target Variable
def detect_target_column(df):
    target_keywords = ['churn', 'target', 'label', 'status', 'exited', 'left']
    for col in df.columns:
        if any(keyword in col.lower() for keyword in target_keywords):
            return col
    return None  # if no matching column
target_col = detect_target_column(df)
if not target_col:
    raise ValueError("No churn/target column found in this dataset.")
#Renaming target variable
df.rename(columns={target_col: 'Exited'}, inplace=True)
```

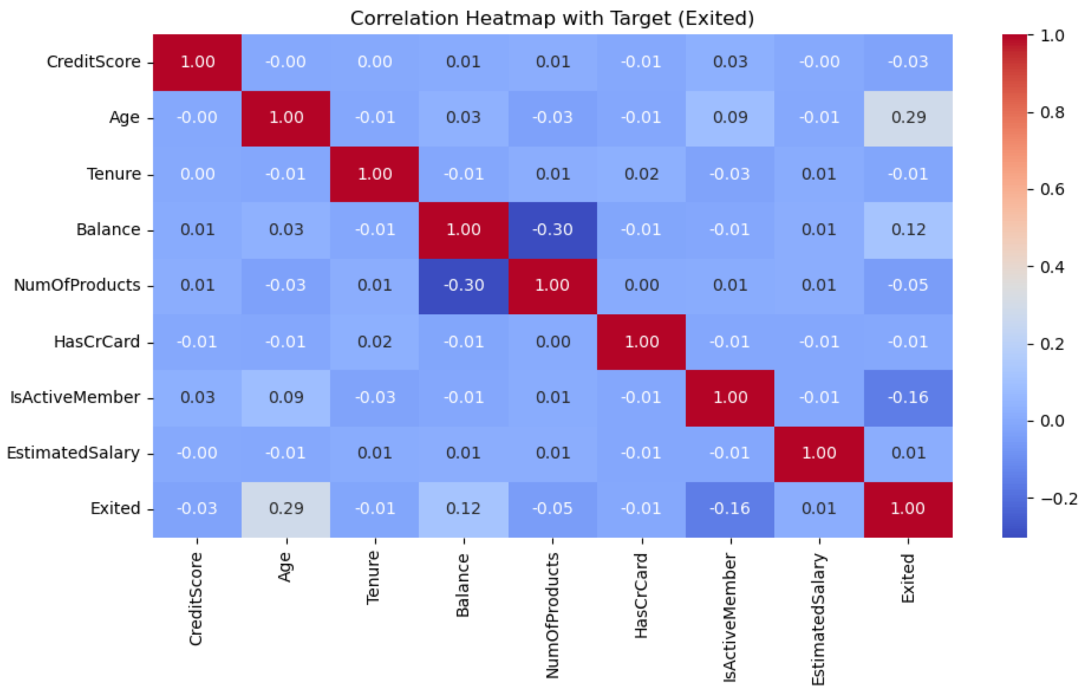
[4]:

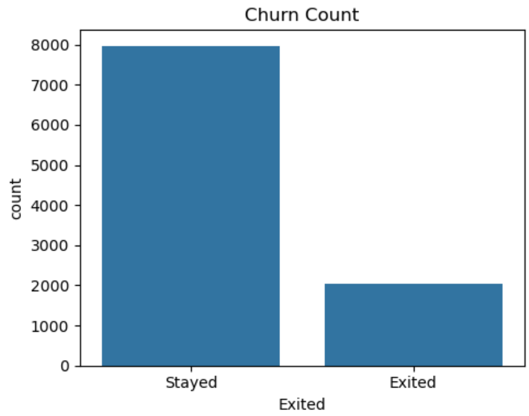
```
df.drop(columns=["RowNumber", "CustomerId", "Surname"], inplace=True)
#Remove duplicated rows if any
df = df.drop_duplicates()
```

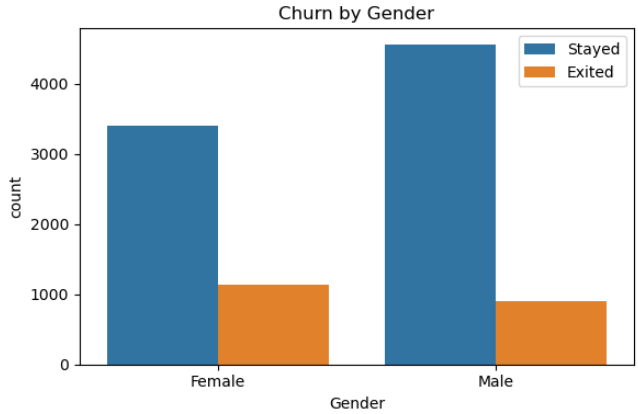
[5]:

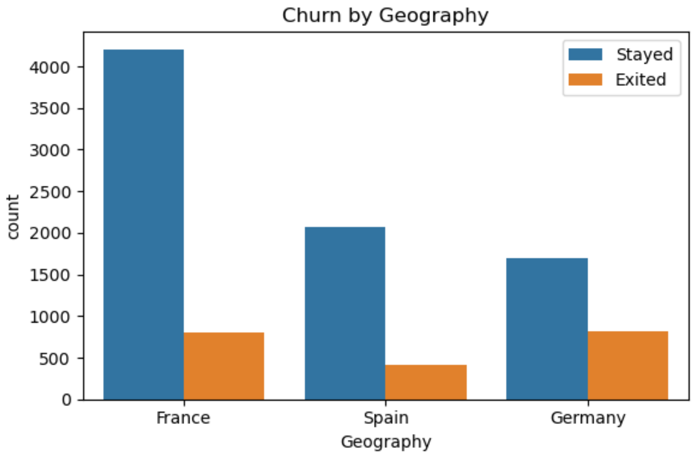
```

#EDA - Correlation Heatmap
correlation_matrix = df.corr(numeric_only=True)
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap with Target (Exited)")
plt.tight_layout()
plt.show()

# Bar Plots
plt.figure(figsize=(5, 4))
sns.countplot(x="Exited", data=df)
plt.title("Churn Count")
plt.xticks([0, 1], ["Stayed", "Exited"])
plt.tight_layout()
plt.show()
```

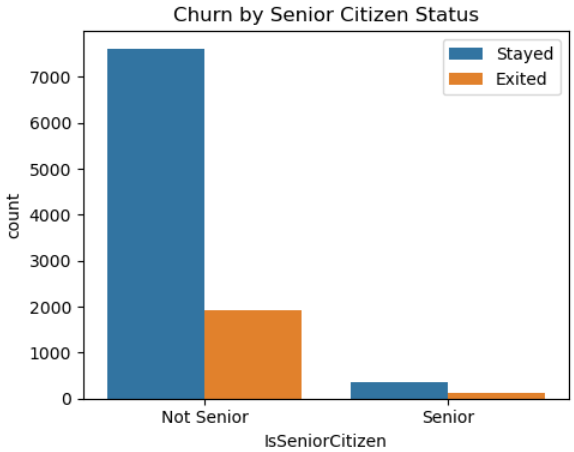
```python
# Bar Plots
plt.figure(figsize=(5, 4))
sns.countplot(x="Exited", data=df)
plt.title("Churn Count")
plt.xticks([0, 1], ["Stayed", "Exited"])
plt.tight_layout()
plt.show()

plt.figure(figsize=(6, 4))
sns.countplot(x="Gender", hue="Exited", data=df)
plt.title("Churn by Gender")
plt.legend(labels=["Stayed", "Exited"])
plt.tight_layout()
plt.show()

plt.figure(figsize=(6, 4))
sns.countplot(x="Geography", hue="Exited", data=df)
plt.title("Churn by Geography")
plt.legend(labels=["Stayed", "Exited"])
plt.tight_layout()
plt.show()

#Feature Engineering
df["BalanceSalaryRatio"] = df["Balance"] / (df["EstimatedSalary"] + 1)
df["IsSeniorCitizen"] = (df["Age"] > 60).astype(int)
df["CreditTenureRatio"] = df["CreditScore"] / (df["Tenure"] + 1)
df["Geography_Gender"] = df["Geography"] + "_" + df["Gender"]
plt.figure(figsize=(5, 4))
sns.countplot(x="IsSeniorCitizen", hue="Exited", data=df)
plt.title("Churn by Senior Citizen Status")
plt.xticks([0, 1], ["Not Senior", "Senior"])
plt.legend(labels=["Stayed", "Exited"])
plt.tight_layout()
plt.show()
```




Churn Count
count
8000
7000
6000
5000
4000
3000
2000
1000
0
Stayed
Exited
Exited


Churn by Gender
Stayed
Exited
count
4000
3000
2000
1000
0
Female
Male
Gender


Churn by Geography
Stayed
Exited
count
4000
3500
3000
2500
2000
1500
1000
500
0
France
Spain
Germany
Geography


Churn by Senior Citizen Status
Stayed
Exited
count
7000
6000
5000
4000
3000
2000
1000
0
Not Senior
Senior
IsSeniorCitizen

```python
# Separate features (x) and target (y)

x = df.drop("Exited", axis=1)
y = df["Exited"]

# Handle Missing Values
numerical_cols = x.select_dtypes(include=['float64', 'int64']).columns
x[numerical_cols] = x[numerical_cols].fillna(x[numerical_cols].median())

# Categorical
categorical_cols = x.select_dtypes(include=['object', 'category']).columns
x[categorical_cols] = x[categorical_cols].fillna('Missing')

# Handle Outliers
for col in numerical_cols:
    Q1 = x[col].quantile(0.25)
    Q3 = x[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    x[col] = np.where(x[col] > upper, upper, np.where(x[col] < lower, lower, x[col]))
```

```python
# Convert categorical columns to string or category before splitting
x[categorical_cols] = x[categorical_cols].astype(str)

# Preprocessing and Train-Test Split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

# Identify categorical features by column index from X_train
categorical_features = [x_train.columns.get_loc(col) for col in categorical_cols]
```

```python
#Build the CatBoost Pool
from catboost import Pool, CatBoostClassifier
train_pool = Pool(data=x_train, label=y_train, cat_features=categorical_features)
test_pool = Pool(data=x_test, label=y_test, cat_features=categorical_features)

#Initialize CatBoost Model
model = CatBoostClassifier(
    iterations=1000,
    learning_rate=0.1,
    depth=6,
    eval_metric='Accuracy',
    verbose=100,
    random_seed=42
)
#Train the Model
model.fit(train_pool, eval_set=test_pool, early_stopping_rounds=50)
```

```
0:      learn: 0.8526250        test: 0.8590000 best: 0.8590000 (0)     total: 193ms    remaining: 3m 12s
100:    learn: 0.8808750        test: 0.8625000 best: 0.8675000 (66)    total: 3.13s    remaining: 27.8s
Stopped by overfitting detector  (50 iterations wait)

bestTest = 0.8675
bestIteration = 66

Shrink model to first 67 iterations.
```

```
<catboost.core.CatBoostClassifier at 0x2349fefea50>
```

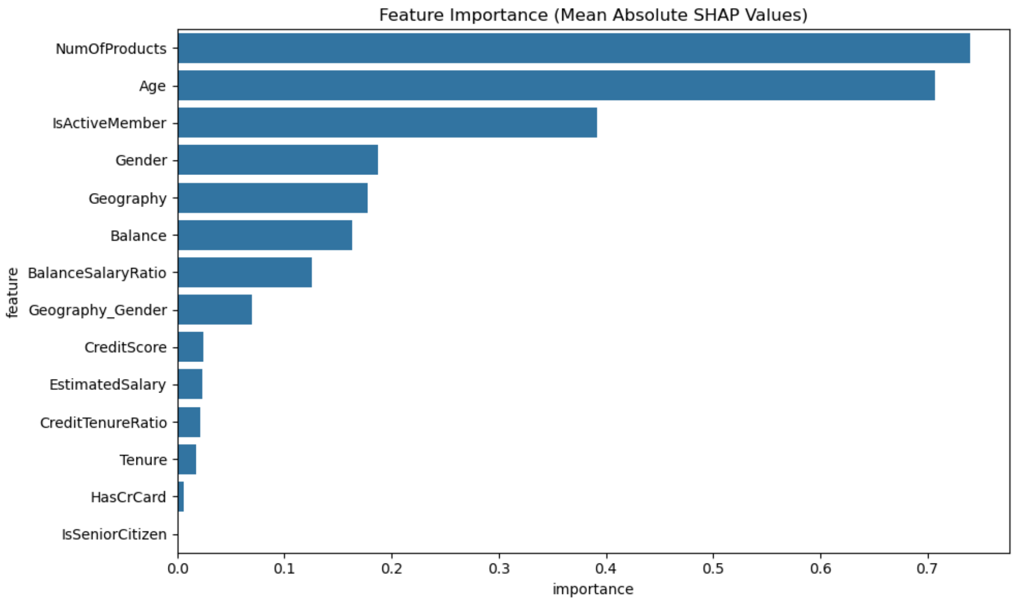
```python
# Make predictions
y_pred = model.predict(x_test)
y_pred_proba = model.predict_proba(x_test)[:, 1]
```

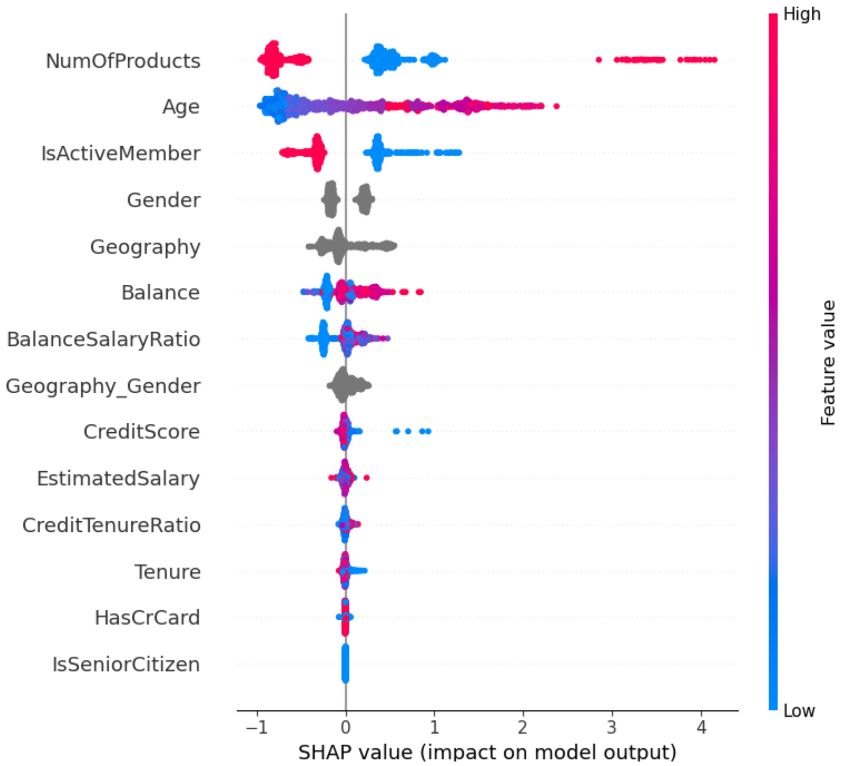
```python
# Evaluate the Model
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_matrix,classification_report
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred_proba)
conf_matrix = confusion_matrix(y_test, y_pred)
summary_report=classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-Score: {f1:.4f}")
print(f"ROC AUC Score: {roc_auc:.4f}")
print("Confusion Matrix:\n", conf_matrix)
print("Classification_report:\n", summary_report)
```

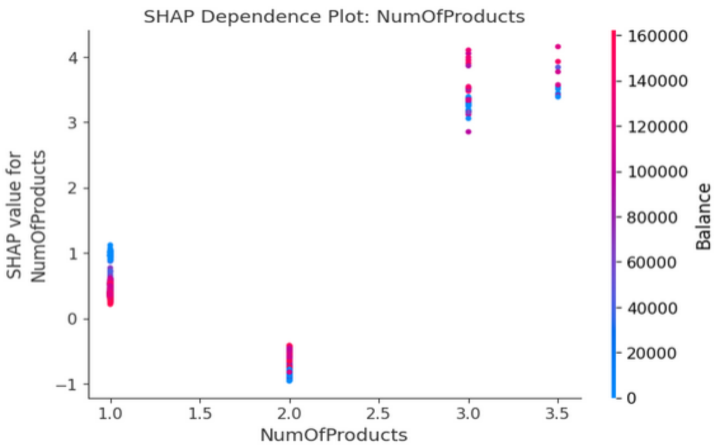
```
Accuracy: 0.8675
Precision: 0.7645
Recall: 0.4707
F1-Score: 0.5827
ROC AUC Score: 0.8752
Confusion Matrix:
 [[1550   57]
 [ 208  185]]

Classification_report:
               precision    recall  f1-score   support

           0       0.88      0.96      0.92      1607
           1       0.76      0.47      0.58       393

    accuracy                           0.87      2000
   macro avg       0.82      0.72      0.75      2000
weighted avg       0.86      0.87      0.85      2000
```

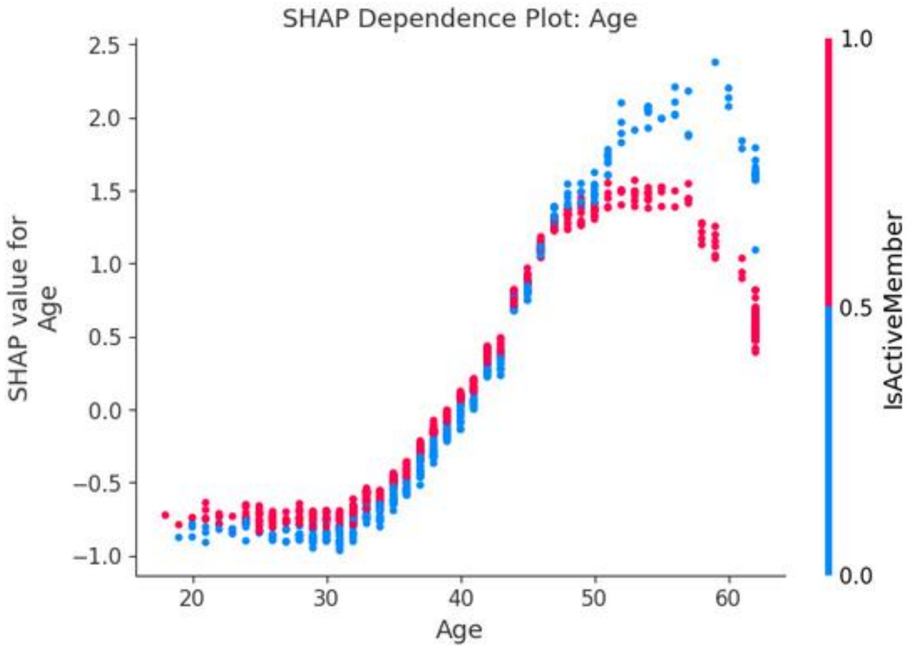
```
!pip install shap
```

```python
import shap
from shap import TreeExplainer, summary_plot
explainer = TreeExplainer(model)
x_test_sample = x_test.sample(n=1000, random_state=42)
shap_values = explainer.shap_values(x_test_sample)


# Generate SHAP summary plot (Bar graph)
# Calculate mean absolute SHAP values
mean_abs_shap_values = np.abs(shap_values).mean(axis=0)
feature_names = x_test_sample.columns

# Create a DataFrame for plotting
shap_importance_df = pd.DataFrame({'feature': feature_names, 'importance': mean_abs_shap_values})
shap_importance_df = shap_importance_df.sort_values('importance', ascending=False)

plt.figure(figsize=(10, 6))
sns.barplot(x='importance', y='feature', data=shap_importance_df)
plt.title('Feature Importance (Mean Absolute SHAP Values)')
plt.tight_layout()
plt.show()

# Generate SHAP summary plot (Dot)
summary_plot(shap_values, x_test_sample, title="SHAP Summary Plot (Dot)")

# Generate SHAP dependence plots for important features
important_features = shap_importance_df['feature'].head(2).tolist()
for feature in important_features:
    shap.dependence_plot(feature, shap_values, x_test_sample, title=f"SHAP Dependence Plot: {feature}")
```


Feature Importance (Mean Absolute SHAP Values)
NumOfProducts
Age
IsActiveMember
Gender
Geography
Balance
BalanceSalaryRatio
Geography_Gender
CreditScore
EstimatedSalary
CreditTenureRatio
Tenure
HasCrCard
IsSeniorCitizen
feature
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
importance


NumOfProducts
Age
IsActiveMember
Gender
Geography
Balance
BalanceSalaryRatio
Geography_Gender
CreditScore
EstimatedSalary
CreditTenureRatio
Tenure
HasCrCard
IsSeniorCitizen
−1
0
1
2
3
4
SHAP value (impact on model output)
High
Feature value
Low


SHAP Dependence Plot: NumOfProducts
SHAP value for NumOfProducts
4
3
2
1
0
−1
1.0
1.5
2.0
2.5
3.0
3.5
NumOfProducts
160000
140000
120000
100000
80000
60000
40000
20000
0
Balance




```python
# Generate a final churn prediction CSV
# Create a DataFrame with predictions
predictions_df = pd.DataFrame({'Predicted_Churn': y_pred}, index=y_test.index)

# Add important fields from the test set to the predictions DataFrame
important_fields_from_X_test = x_test[['CreditScore', 'Geography', 'Gender', 'Age',  'IsActiveMember']].copy()

# Join the important fields with the predictions DataFrame based on the index
predictions_df = predictions_df.join(important_fields_from_X_test)

# Save the DataFrame to a CSV file
predictions_df.to_csv('churn_predictions.csv', index=False) # Set index=False if you add CustomerId back and don't need the original index
print("Churn predictions saved to 'churn_predictions.csv'")

# Calculate and print the number of churned customers out of the total
churned_customers_count = predictions_df['Predicted_Churn'].sum()
total_customers = len(predictions_df)
stayed_customers_count = total_customers - churned_customers_count
stayed_percentage = (stayed_customers_count / total_customers) * 100

print(f"Percentage of customers likely to use the service (predicted to stay): {stayed_percentage:.2f}%")

print(f"Number of customers predicted to churn: {churned_customers_count} out of {total_customers}")
```

```
Churn predictions saved to 'churn_predictions.csv'
Percentage of customers likely to use the service (predicted to stay): 87.90%
Number of customers predicted to churn: 242 out of 2000
```

Dataset Source Link: https://www.kaggle.com/datasets/adammaus/predicting-churn-for-bank-customers?resource=download

Environment Details

Python version and key libraries used are

Python 3.10

pandas 1.5.3

numpy 1.24.2

scikit-learn 1.2.1

catboost 1.2

matplotlib 3.7.1

seaborn 0.12.2

Conclusion

The materials in this appendix serve as a reference and validation of the implementation discussed in the report. They ensure the reproducibility and transparency of the project for future development or review.

THE END