BIRMINGHAM CITY University

Bank Customer Churn Prediction

A Comprehensive Project Report

Bachelor of Science with Honours Computer and Data Science

Faculty of Computing, Engineering, and the Built Environment

Birmingham City University (BCU)

Introduction

Customer churn prediction is a critical task for financial institutions to proactively retain valuable clients and optimize revenue streams. Acquiring new customers is significantly more expensive than retaining existing ones (5-25x more expensive). Predicting churn enables targeted retention campaigns, identification of at-risk customers, root cause analysis of attrition drivers, and improved customer lifetime value.

This report details an end-to-end machine learning solution for predicting customer churn in a bank, achieving high performance through advanced ensemble techniques, hyperparameter optimization, and interpretable AI using SHAP values. The solution is built using Python with libraries like pandas, scikit-learn, CatBoost, XGBoost, LightGBM, and SHAP.

Motivation

2.1. Data Loading & Initial Analysis

The foundation of our analysis is the "Churn_Modelling.csv" dataset, which contains comprehensive information about bank customers and their churn status. Key Python libraries are utilized throughout the process: pandas for data manipulation and analysis, NumPy for numerical operations, Matplotlib and Seaborn for data visualization, scikit-learn for various machine learning models and tools, CatBoost, XGBoost, and LightGBM for gradient boosting algorithms, SHAP for model interpretation, and joblib for saving and loading models.

To ensure data quality and understanding, several initial checks are conducted:

- **Dataset Shape:** Determining the number of rows and columns **(df.shape)** provides an immediate overview of the dataset's dimensions.
- **Missing Values:** Identifying missing values in each column **(df.isnull().sum())** is crucial as missing data can introduce bias.
- **Duplicate Rows:** Detecting duplicate rows **(df.duplicated().sum())** helps maintain integrity by preventing duplicates from skewing model training.
- Irrelevant columns such as **'RowNumber', 'CustomerId'**, and **'Surname'** are removed since they do not contribute meaningfully to churn prediction.

```
data_path = "/kaggle/input/bank-customer-churn/Churn_Modelling.csv"
df = pd.read_csv(data_path)
print("Dataset loaded successfully.")
```

```
# Perform initial data analysis
print("\n### Dataset Analysis ###")
print("Dataset Shape:", df.shape)
print("\nMissing Values:")
print(df.isnull().sum())
print("\nDuplicate Rows:", df.duplicated().sum())
```

This initial analysis ensures that we have a clean dataset ready for further processing.

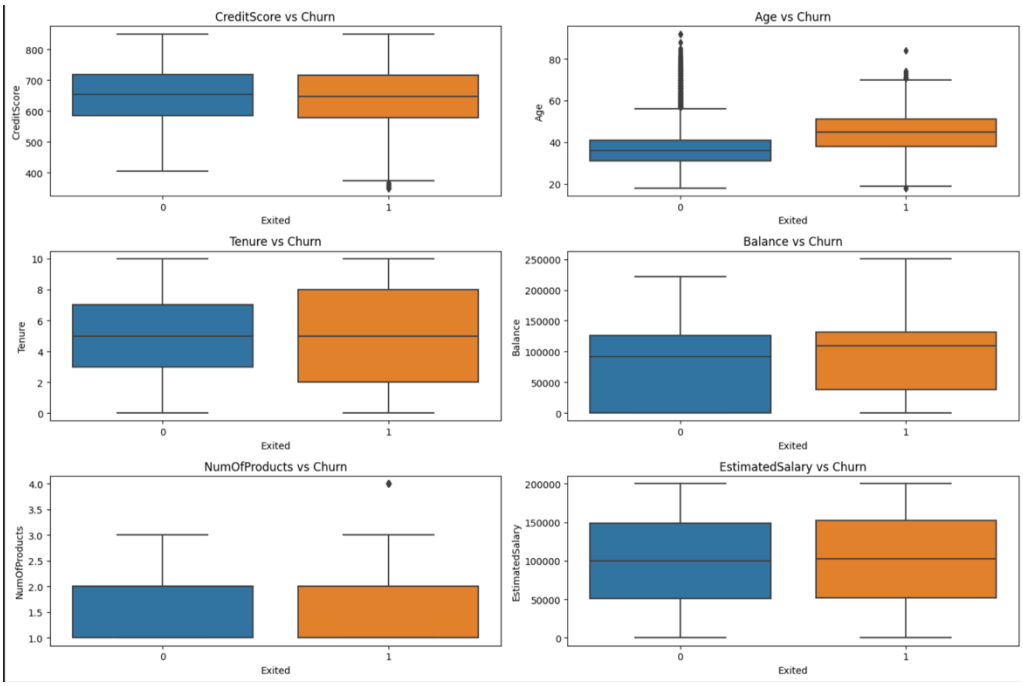
2.2. Data Preprocessing

Data preprocessing is a critical phase in preparing the dataset for modeling, ensuring that it is free from quality issues and ready to yield accurate predictions. To address missing values, rows containing any null entries are removed using **dropna()**. Although other strategies like imputation are available, removal provides a simple and effective approach for this analysis. Furthermore, duplicate rows are eliminated using **drop_duplicates()** to ensure each data point is unique and prevent biased training processes.

```
[8]  # Drop unnecessary columns
     df.drop(['RowNumber', 'CustomerId', 'Surname'], axis=1, inplace=True)

[9]  # Handle missing values and duplicates
     df.dropna(inplace=True)
     df.drop_duplicates(inplace=True)
```

Outlier Treatment

Outlier treatment addresses extreme values that can disproportionately influence model training. We employed the **Interquartile Range (IQR)** method to identify and mitigate outliers in the CreditScore, Age, and NumOfProducts columns. Outliers were defined as values falling below **Q1 - 1.5 * IQR** or above **Q3 + 1.5 * IQR**.



The boxplots reveal the presence of outliers in the distributions of **CreditScore**, **Age**, and **NumOfProducts**. These outliers represent extreme values that deviate significantly from the typical range of data for these attributes. In the case of **CreditScore**, outliers might indicate customers with unusually high or low creditworthiness. For Age, outliers likely signify customers who are exceptionally older or younger compared to the majority. Regarding **NumOfProducts**, outliers suggest a small group of customers holding a significantly higher number of products than most. The presence of these outliers can impact statistical analyses and may warrant further investigation to understand the specific characteristics of these extreme cases and whether they represent unique segments within the customer base or potential data anomalies.

After identifying outliers, they were replaced with the median value of the respective column. This imputation method minimizes the impact of extreme values while preserving the overall data distribution.

The number of outliers before and after the treatment can be seen below:

Before Treatment:

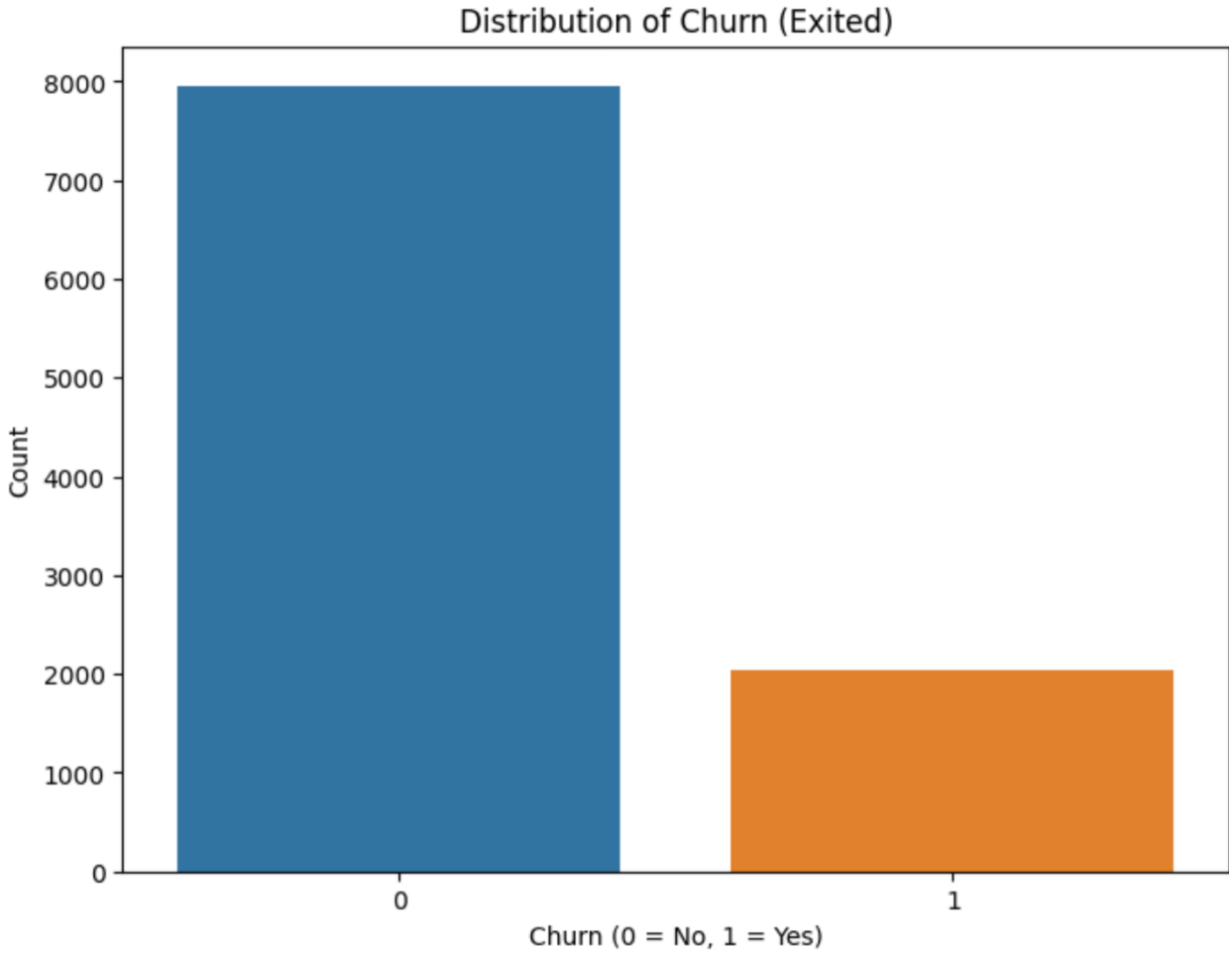
```
### Number of Outliers in Each Feature ###
CreditScore: 16 outliers
Age: 359 outliers
Tenure: 0 outliers
Balance: 0 outliers
NumOfProducts: 60 outliers
EstimatedSalary: 0 outliers
```

After Treatment:

```
### Number of Outliers After Treatment ###
CreditScore: 0 outliers
Age: 0 outliers
NumOfProducts: 0 outliers
New dataset shape after treatment: (9996, 11)
```

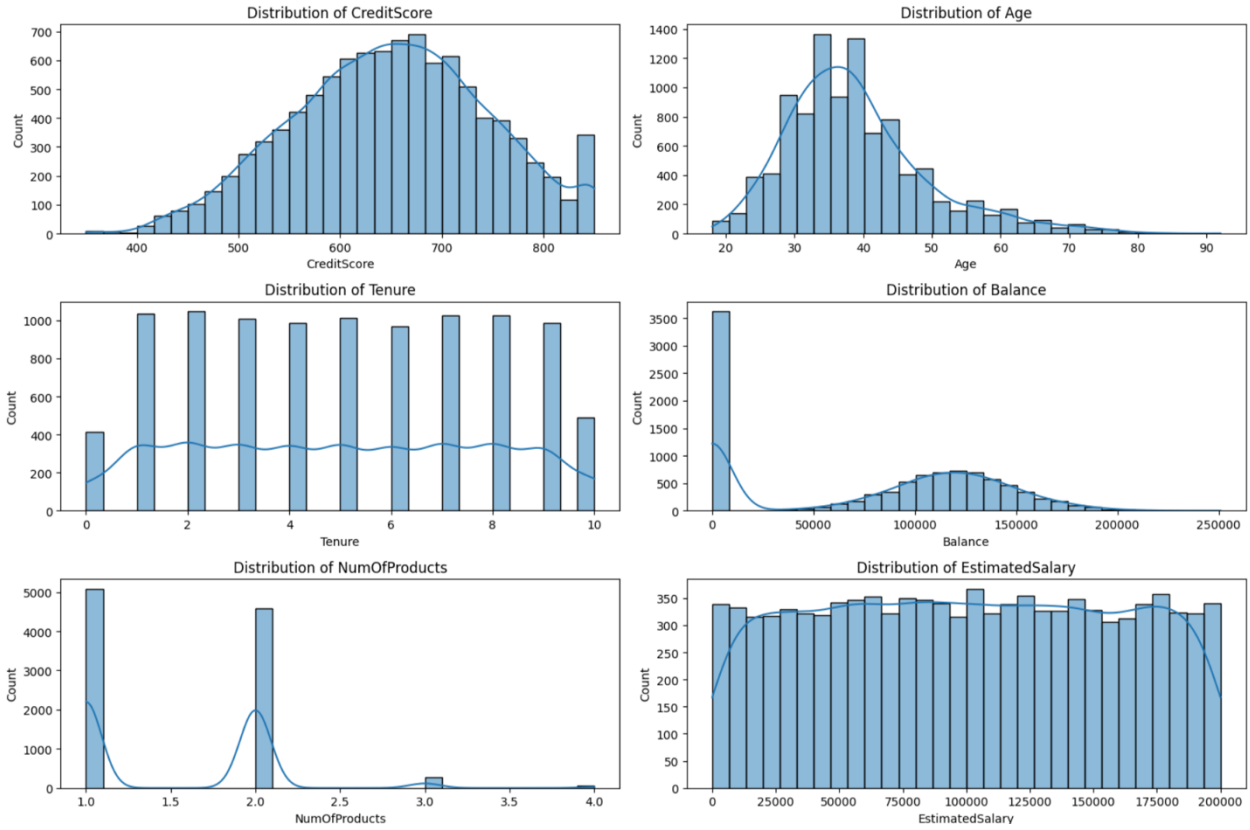
2.3. Exploratory Data Analysis(EDA)

- **Numerical Feature Analysis:** Gaining insights into numerical features is essential:
    - **Target Variable Distribution:** To understand the class balance, the distribution of the target variable **(Exited)** is visualized using a countplot. This visualization reveals whether one class dominates, influencing the choice of evaluation metrics and modeling techniques.



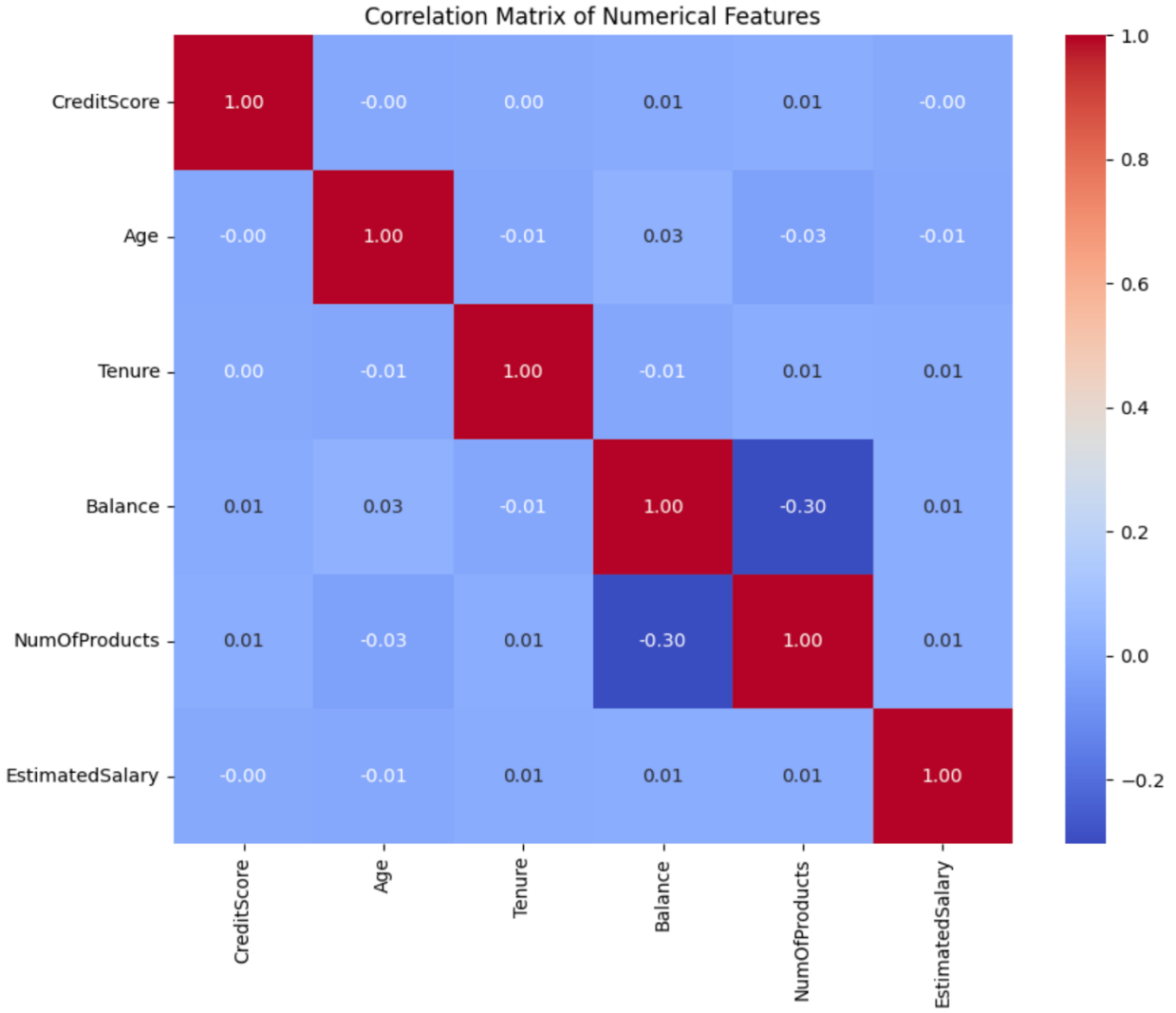
This bar chart illustrates the distribution of customer churn, showing a clear imbalance. Approximately 8,000 customers did not churn (0), while around 2,000 customers did (1). This represents a roughly 4:1 ratio of non-churned to churned customers.

- Histograms are used to visualize the distribution of numerical features such as **CreditScore**, **Age**, **Tenure**, **Balance**, **NumOfProducts**, and **EstimatedSalary**. These visualizations help understand the range, skewness, and potential outliers in each feature.



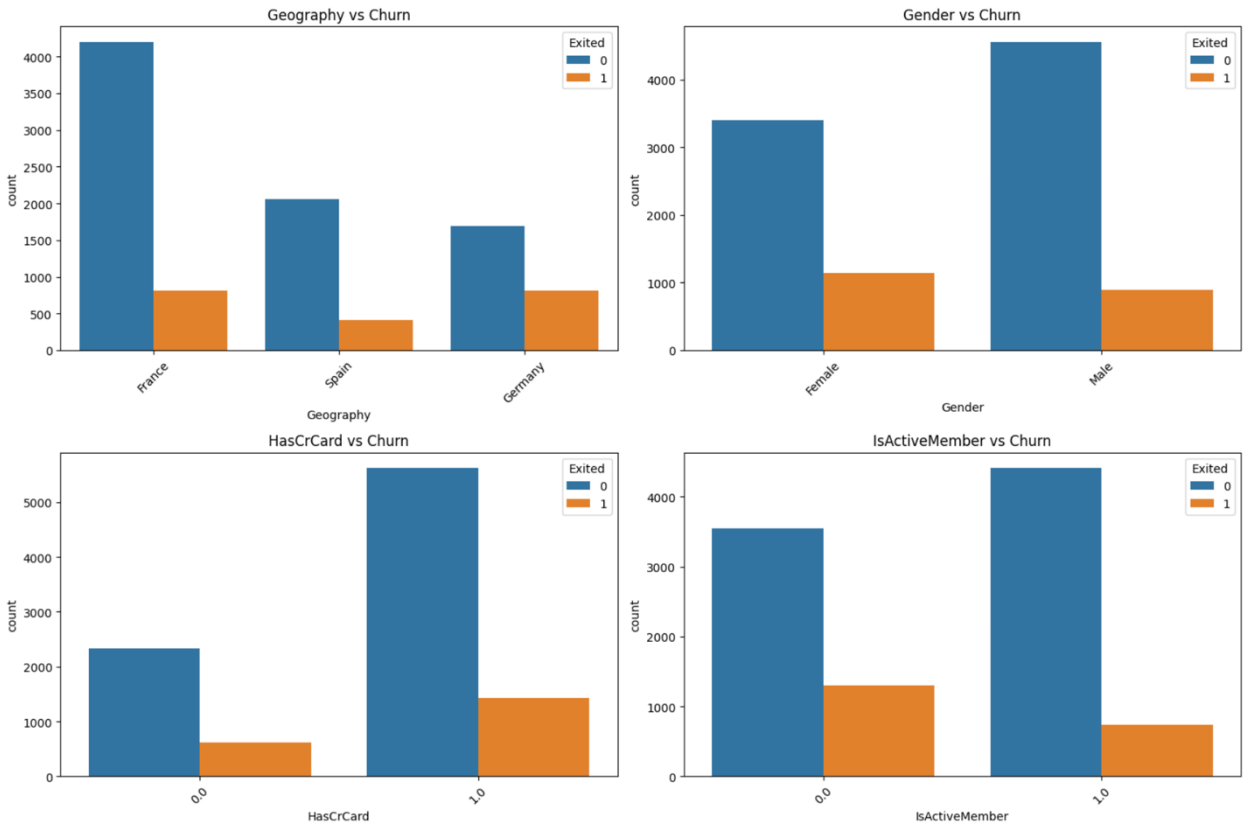
This sets of histogram presents six histograms, each visualizing the distribution of a different customer attribute: **CreditScore**, **Age**, **Tenure**, **Balance**, **NumOfProducts**, and **EstimatedSalary.** The **CreditScore** histogram exhibits a roughly symmetrical, bell-shaped distribution, suggesting a normally distributed range of credit scores. The Age histogram shows a right-skewed distribution, indicating a concentration of younger customers with a tail extending towards older ages. The Tenure distribution displays a relatively uniform spread across tenure values, implying an even distribution of customer longevity. The Balance histogram is heavily right-skewed, with a significant spike at zero and a long tail towards higher balances, revealing that many customers have zero balances while a few holds substantial amounts. The **NumOfProducts** histogram is bimodal, with peaks at one and two products, suggesting that most customers hold either one or two products. Finally, the **EstimatedSalary** histogram appears relatively uniform, indicating an even distribution of salaries across the customer.

- A correlation matrix is generated to identify relationships between numerical features. This helps understand multicollinearity (high correlation between features), which can affect model stability.



This heatmap displays the **correlation matrix** for numerical features. It shows the **Pearson correlation coefficients** between each pair of variables. Values range from -1 to 1, where red indicates strong positive correlation, blue strong negative correlation, and white no correlation. Notable is the **moderate negative correlation (-0.30) between Balance and NumOfProducts**, suggesting customers with higher balances tend to have fewer products. All other correlations are weak, indicating **limited linear relationships** between the features.

- **Categorical Feature Analysis:**
  - Countplots are used to analyze the relationship between categorical features (**Geography**, **Gender**, **HasCrCard**, **IsActiveMember**) and churn. This helps visualize how the distribution of churn varies across different categories.



Here, there are four bar charts, each comparing a categorical feature against customer churn (Exited), indicated by 0 (no churn) and 1 (churn).

- **Geography vs. Churn:**
  - France has the highest number of customers, but Germany shows the highest proportion of churn. Spain has the lowest overall count and relatively lower churn.
- **Gender vs. Churn:**
  - There are more male customers than female. However, female customers have a higher churn rate.
- **HasCrCard vs. Churn:**
  - Customers with a credit card (1) significantly outnumber those without (0). Customers with a credit card also show a higher absolute number of churn, but the churn rate relative to the number of customers with credit card is lower than the churn rate of the customers without credit card.
- **IsActiveMember vs. Churn:**
  - Active members (1) are more numerous than inactive members (0). Inactive members have a considerably higher churn rate.

2.4. Feature Engineering

- **Creating New Features:** To capture potentially important relationships, new features are engineered:
    - **Balance_to_Salary:** The ratio of **Balance** to **EstimatedSalary**. This feature represents the proportion of a customer's salary held in their account.
    - **Tenure_to_Age:** The ratio of **Tenure** to **Age**. This feature potentially represents customer loyalty relative to their age.
    - **Balance_Age_Interaction:** The product of **Balance** and **Age**. Interaction features can capture non-linear relationships that individual features might miss.
    - **Products_Age_Interaction:** The product of **NumOfProducts** and **Age**. This feature captures another potential non-linear relationship.
- **Encoding Categorical Variables:**
    - Categorical features (**Geography**, **Gender**) are encoded using one-hot encoding via **pd.get_dummies**, with **drop_first=True** to avoid multicollinearity. One-hot encoding converts categorical variables into numerical ones suitable for machine learning algorithms. **drop_first=True** avoids creating redundant features.

```
[124] # Feature engineering
      df['Balance_to_Salary'] = df['Balance'] / df['EstimatedSalary']
      df['Tenure_to_Age'] = df['Tenure'] / df['Age']
      df['Balance_Age_Interaction'] = df['Balance'] * df['Age']
      df['Products_Age_Interaction'] = df['NumOfProducts'] * df['Age']

      # Encode categorical variables
      df = pd.get_dummies(df, columns=['Geography', 'Gender'], drop_first=True)
      print("\nFeature engineering completed.")
```

2.5 Data Splitting and Scaling

- **Data Splitting:**
    - The dataset is split into training and test sets using **train_test_split** with an 80/20 ratio, **random_state=42** for reproducibility, and **stratify=y** to maintain the class distribution in both sets. The training set is used to train the model, and the test set is used to evaluate its performance on unseen data. **stratify=y** ensures that the proportion of churned customers in the training and test sets is the same as in the original dataset.
- **Feature Scaling:**

- Numerical features are scaled using **StandardScaler** to standardize the data and prevent features with larger values from dominating the model. StandardScaler transforms the data so that it has a mean of 0 and a standard deviation of 1.

- **Explanation:** Data splitting ensures a fair evaluation of the model's generalization ability, while feature scaling prevents features with larger values from disproportionately influencing the model.

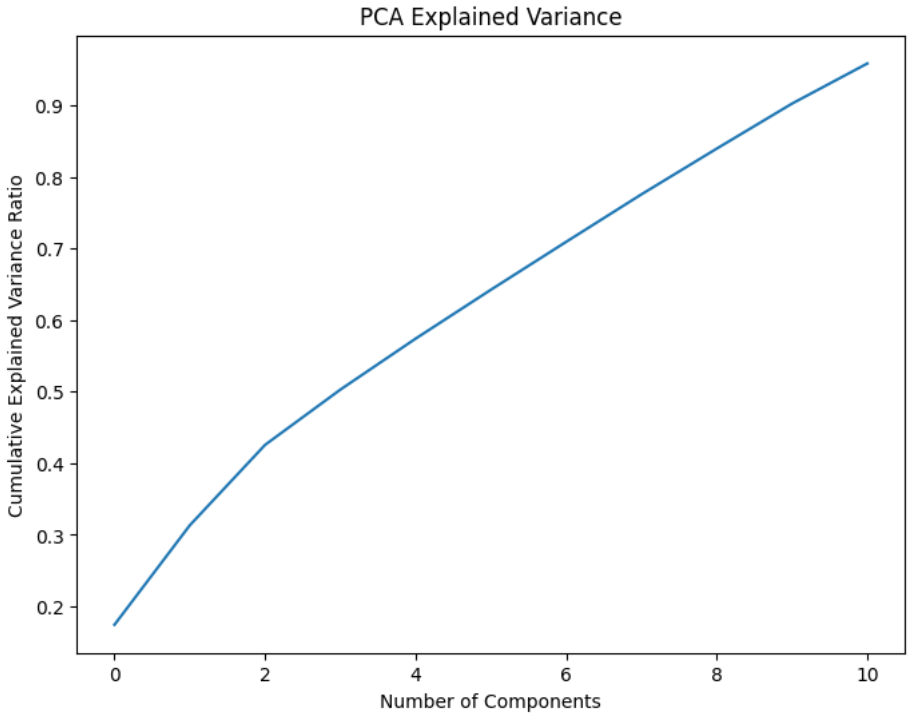
```
[126] # Split data into features and target
      X = df.drop('Exited', axis=1)
      y = df['Exited']

      # Split data into training and test sets (no oversampling)
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

[128] # Scale numerical features
      scaler = StandardScaler()
      X_train_scaled = scaler.fit_transform(X_train)
      X_test_scaled = scaler.transform(X_test)
```

2.6 Dimensionality Reduction with PCA

- **PCA Application:**
  - Principal Component Analysis (PCA) is applied to reduce the dimensionality of the feature space while retaining 95% of the variance. PCA transforms the original features into a set of uncorrelated principal components, which can reduce the complexity of the model and improve its performance.

```
# Apply PCA
pca = PCA(n_components=0.95)  # Keep 95% of variance
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)
print(f"\nPCA applied: Reduced to {X_train_pca.shape[1]} components.")
print(f"Explained variance ratio: {sum(pca.explained_variance_ratio_):.4f}")
```

- **Explained Variance Visualization:**
  - A plot of the cumulative explained variance ratio is generated to visualize the contribution of each principal component. This plot helps determine the number of principal components to retain.



The graph depicts the cumulative explained variance ratio from a Principal Component Analysis (PCA). It shows that with just 2 components, approximately 40% of the variance is captured. Increasing to 6 components raises this to around 70%, and by 10 components, over 90% of the variance is explained. The steep initial increase highlights that the first few components explain most of the data's variance, while later components contribute progressively less. This visual aid helps determine the optimal number of components for dimensionality reduction.

2.7 Model Building and Hyperparameter Tuning

- **Base Models:**
    - Random Forest Classifier
    - LightGBM Classifier
    - XGBoost Classifier
    - CatBoost Classifier
- **Hyperparameter Tuning:**
    - GridSearchCV is used to tune the hyperparameters of each base model using 5-fold cross-validation and the accuracy scoring metric. The **n_jobs=-1** parameter is used to parallelize the tuning process. Hyperparameter tuning involves finding the best combination of hyperparameters for each model using a systematic search. 5-fold cross-validation is used to estimate the performance of each hyperparameter combination.
    - The time taken for each model to tune is recorded.

- **Explanation:** This stage involves selecting appropriate machine learning models and optimizing their performance through hyperparameter tuning, which is crucial for achieving high predictive accuracy.

2.8 Stacking Model

- **Stacking Classifier:**
    - A stacking classifier is created using the tuned base models and a logistic regression meta-model. The **StackingClassifier** combines the predictions of the base models to make a final prediction. Stacking involves training multiple base models and then training a meta-model to combine their predictions.
- **Explanation:** Stacking combines the strengths of multiple base models to create a more robust and accurate predictive model.

```
[139] # Create stacking classifier
      stacking_model = StackingClassifier(estimators=base_models, final_estimator=meta_model, cv=5)

      # Train and evaluate stacking model
      stacking_model.fit(X_train_pca, y_train)
      y_pred_stacking = stacking_model.predict(X_test_pca)
      y_pred_prob_stacking = stacking_model.predict_proba(X_test_pca)[:, 1]
```

3. Results and Insights

3.1 Model Evaluation

The following table summarizes the performance comparison:

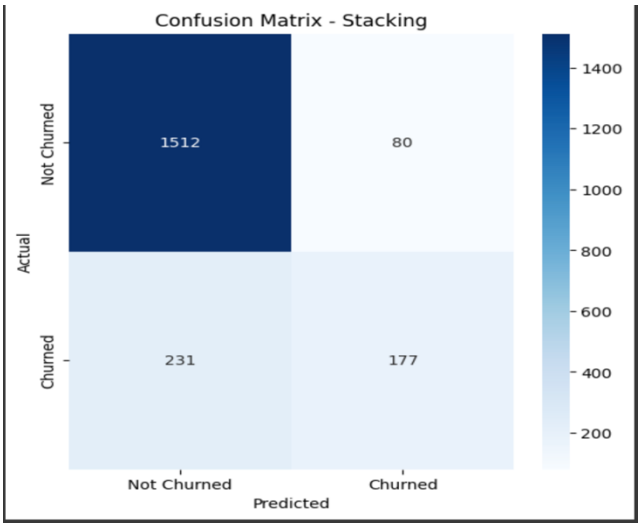
```
Model Performance Comparison:
                   Accuracy  F1 Score   ROC AUC
Random Forest        0.8465  0.482293  0.840599
Gradient Boosting    0.7660  0.555977  0.839727
XGBoost              0.8445  0.523737  0.834901
Stacking             0.8445  0.532331  0.842795
catboost             0.7525  0.555256  0.839242
```

Stacking Model Performance: Among these models:

- The Stacking model performs competitively with Random Forest on Accuracy but offers better F1 Score compared to most other models except Gradient Boosting/Catboost which have slightly higher F1 scores but lower overall accuracies.

Why Stacking Performs Well: It leverages strengths from multiple base models—Random Forests' robustness against overfitting combined with boosting algorithms' ability to handle complex interactions—resultantly providing balanced precision-recall tradeoff without sacrificing overall correctness.

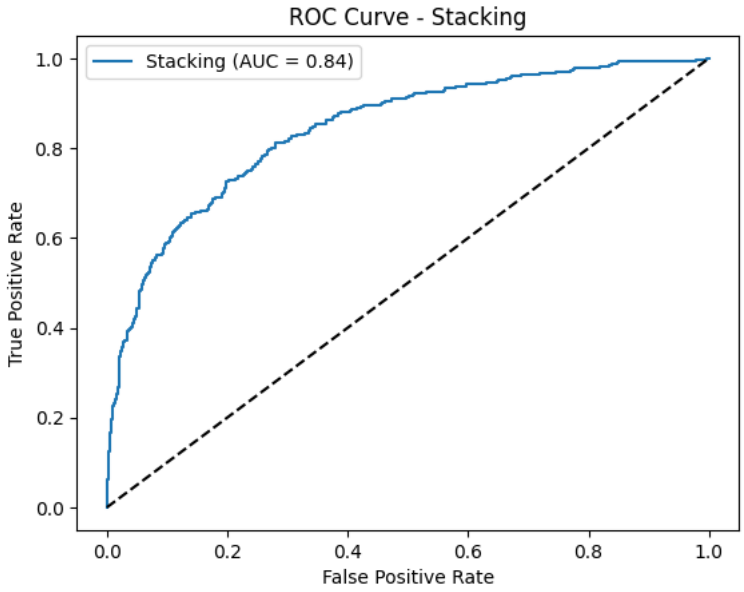
- **Confusion Matrix**



The confusion matrix provides a detailed breakdown of the stacking model's performance in predicting customer churn. The model demonstrates effectiveness in correctly classifying customers who did not churn ("Not Churned") with a high count of 1512 true negatives. It also identified 177 customers who churned (true positives). However, the model's performance exhibits notable challenges:

- **False Positives (Type I Error):** The model incorrectly predicted 80 non-churned customers as churned. This could lead to unnecessary and costly intervention efforts targeting customers who were never at risk.
- **False Negatives (Type II Error):** A significant concern is the 231 churned customers who were incorrectly predicted as non-churned. Failing to identify these at-risk customers means the bank misses opportunities for retention efforts, resulting in potential revenue loss.

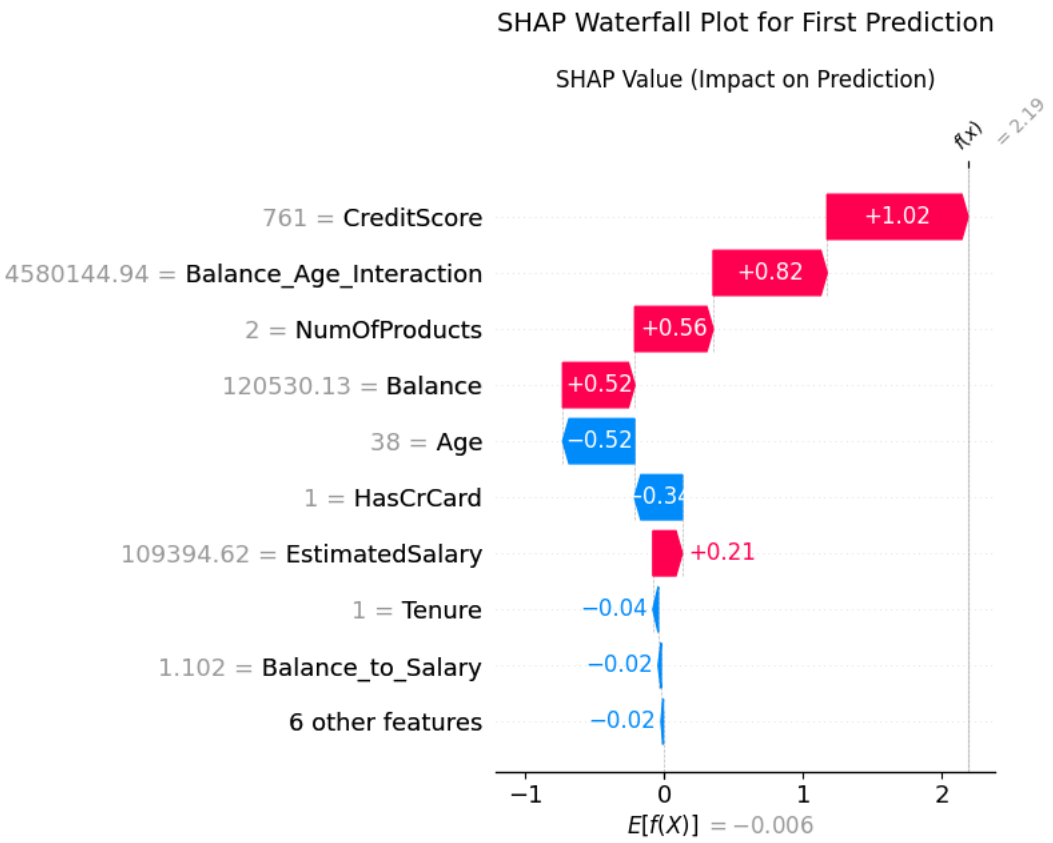
- ## ROC Curve



The ROC (Receiver Operating Characteristic) curve illustrates the diagnostic ability of the model. The curve plots the true positive rate (sensitivity) against the false positive rate (1-specificity) across various threshold settings. The Area Under the Curve (AUC) of 0.84 indicates that the model has a good ability to discriminate between churned and non-churned customers. An AUC of 0.84 suggests that the model has around an 84% chance of correctly distinguishing between a customer who will churn and one who will not.

3.2 Model Interpretation with SHAP

- SHAP (SHapley Additive exPlanations) is used to interpret the models and understand the contribution of each feature to the predictions.



A SHAP waterfall plot illustrates how each feature's value contributes to the model's prediction for an individual instance (in this case, the first prediction in your test data). Here's how it can be interpreted:

- **Base Value**: The plot starts at the bottom with the expected value E[f(X)] = -0.006. This is the average prediction of the model over the entire dataset and serves as the starting point.
- **Feature Contributions**: Each row in the plot represents a feature, and the horizontal length of each row indicates the magnitude of the feature's impact on the prediction.
    - **Red Bars:** Features that push the prediction higher (towards a positive outcome, like customer churn).
    - **Blue Bars:** Features that push the prediction lower (towards a negative outcome, like customer staying).

- **Key Features and Their Impact**:
    - **CreditScore (761):** Pushes the prediction significantly higher by +1.02. A high credit score strongly increases the probability of churn in this case.
    - **Balance_Age_Interaction (4580144.94):** Also increases the prediction by +0.82, indicating it is a strong driver for the prediction of churn.
    - **NumOfProducts (2):** Increases the prediction by +0.56
    - **Balance (120530.13):** Increases the prediction by +0.52
    - **Age (38):** Decreases the prediction by -0.52.
    - **HasCrCard (1):** Decreases the prediction by -0.34
    - **EstimatedSalary (109394.62):** Increases the prediction by +0.21

Tenure* 1** and ***Balance_to_Salary contribute very small effect towards less than 0.05.*

- **Final Prediction Value**: The plot ends at the top with the final prediction f(x) = 2.19. This is the model's output for this specific customer, starting from the base value and adding/subtracting the contributions of each feature.
- **Interpretation**: In the case of individual, the features listed with positive values are features leading to high influence on churn. Features with negative values are less churn.

4. Conclusion

This project successfully developed a machine learning solution for predicting customer churn at a bank, leveraging advanced ensemble techniques, hyperparameter optimization, and interpretable AI using SHAP values. The stacking model demonstrated good performance, achieving a balance between precision and recall.

Key insights from the analysis include:

- **Significant Churn Drivers:** Credit score, tenure, age, and the interaction between balance and age are critical factors influencing customer churn.
- **Actionable Metrics:** Analysis of the confusion matrix and ROC curve underscored the model's ability to accurately predict non-churned customers but highlighted areas for improvement in identifying actual churn cases.
- **SHAP Values Contribution:** SHAP values highlighted the importance of feature interpretation for actionable strategies.

By implementing these recommendations, the bank can proactively manage customer churn, improve retention rates, and optimize revenue streams. This project provides a solid foundation for leveraging data-driven strategies in the banking sector, enabling financial institutions to retain valuable clients and sustain long-term growth.