YENEPOYA INSTITUTE OF ARTS, SCIENCE AND COMMERCE MANAGEMENT

FINAL PROJECT REPORT

on

Customer Churn Prediction with Data Visualization

BACHELOR OF COMPUTER APPLICATION

BCA BIG DATA WITH IBM

SUBMITTED BY

Mohamed Faheem – 22BDACC147

Hari Krishnan K Nambiar – 22BDACC095

Mohammed Shamil T – 22BDACC199

Muhammad Siyad K – 22BDACC218

GUIDED BY

Sumit K Shukla

IBM Innovation Centre for Education

PHEMESOFT Software That Matters
TRUSTED DELIVERY PARTNER for IBM ICE

TABLE OF CONTENTS

1. BACKGROUND

1.1 Aim

- The aim of this project is to develop a data-driven system to predict customer churn in a telecom company using machine learning techniques and to visually interpret the key drivers behind customer attrition through Power BI. The project helps businesses proactively identify at-risk customers and take appropriate actions to reduce churn.

- Customer churn refers to the scenario where customers stop using a company's services or products. In competitive sectors like telecommunications, retaining existing customers is often more cost-effective than acquiring new ones. Predicting churn allows companies to optimize their strategies, offer tailored incentives, and improve customer satisfaction.

- This project empowers decision-makers by combining predictive analytics with visual storytelling. The integration of machine learning with business intelligence helps uncover hidden patterns and understand the factors leading to customer churn.

1.2 Technologies

- Programming Language: Python
- Libraries Used: Pandas, Seaborn, Scikit-learn, Matplotlib, SMOTE
- Visualization Tool: Power BI
- IDE: Visual Studio Code
- Version Control: GitHub

1.3 Hardware Architecture

- Processor: Intel Core i5 or higher
- RAM: 8 GB minimum
- Storage: SSD recommended
- OS: Windows 10 or later

1.4 Software Architecture

- Python 3.10+
- Power BI Desktop
- Dataset: Telco Customer Churn (CSV)
- Model: Random Forest Classifier
- File outputs: Processed CSV and trained model (PKL)

2. SYSTEM

2.1 Requirements

2.1.1 Functional Requirements

The project system is designed to perform the following core functionalities:

- Load and preprocess telecom churn data from a CSV file
- Convert categorical features into numerical format for model compatibility
- Handle missing values and scale numerical features
- Balance the dataset using SMOTE (Synthetic Minority Over-sampling Technique)
- Train a machine learning model (Random Forest) to predict customer churn
- Evaluate the model using performance metrics like accuracy, precision, recall, and F1-score
- Save the predictions into a new file for Power BI visualization
- Create an interactive dashboard using Power BI to showcase churn insights

These functionalities work in sequence to transform raw data into actionable insights.

2.1.2 User Requirements

From a user's perspective, the system is designed to be:

- **User-friendly:** Minimal code interaction, clearly structured outputs
- **Interpretable:** Visualization provides an easy understanding of customer churn patterns
- **Reusable:** Can be applied to other churn datasets with similar structure
- **Platform-Independent:** Can be run on any machine with Python and Power BI installed
- **Scalable:** The model can be upgraded or swapped with other ML algorithms easily

2.1.3 Environmental Requirements

For the smooth functioning of this system, the following environment setup is required:

Hardware

- Minimum 8 GB RAM
- SSD storage preferred
- Intel Core i5 processor or equivalent

Software

- Python 3.10+
- Libraries: pandas, numpy, seaborn, matplotlib, scikit-learn, imblearn
- Power BI Desktop (latest version)
- Visual Studio Code or any Python IDE

Operating System

- Windows 10/11 or compatible OS with access to Python and Power BI

2.2 Design and Architecture

This system follows a **modular architecture**, broken down into four logical components:

1. Data Acquisition Layer

- Ingests the CSV file (Telco-Customer-Churn.csv)
- Performs initial data checks and format validation

2. Data Processing Layer

- Cleans the data by handling nulls and duplicates
- Applies encoding, scaling, and SMOTE balancing

3. Modeling Layer

- Trains a Random Forest model
- Evaluates it on test data
- Saves predictions and trained model

4. Visualization Layer

- Power BI loads the `Processed_Customer_Churn.csv` file

- Visuals are created for churn analysis (bar charts, line graphs, KPIs)

This modular structure promotes separation of concerns and makes the system easier to debug, test, and extend.

2.3 Implementation

The implementation of the churn prediction system involved a structured pipeline divided into clearly defined stages, from raw data ingestion to the creation of a Power BI dashboard. Below is a step-by-step breakdown of the implementation process:

Step 1: Data Loading

- The dataset `Telco-Customer-Churn.csv` was loaded using Pandas into a DataFrame.
- Initial checks were conducted using `df.head()`, `df.info()`, and `df.describe()` to understand the structure and content of the data.

Step 2: Data Cleaning

- The column `TotalCharges` was found to have incorrect data types and missing values. It was:
  - Converted from object to numeric using `pd.to_numeric()`
  - Missing values were filled using the median
- Duplicate entries were removed using `df.drop_duplicates()`
- The `customerID` column was removed as it didn’t add value to the prediction

Step 3: Data Preprocessing

- **Categorical Encoding**: Categorical variables such as `gender`, `contract`, and `internet service` were encoded using `LabelEncoder` from scikit-learn
- **Target Variable Transformation**: The `Churn` column was mapped to 1 for "Yes" and 0 for "No"
- **Feature Scaling**: Numerical columns (`tenure`, `MonthlyCharges`, `TotalCharges`) were normalized using `StandardScaler` to bring values to the same scale

Step 4: Handling Class Imbalance with SMOTE

- The original dataset had a class imbalance (fewer churn cases than non-churn)
- To address this, **SMOTE (Synthetic Minority Oversampling Technique)** was used
  - It generated synthetic data for the minority class (churn = 1)
  - Resulted in a balanced dataset which improved model fairness

Step 5: Splitting the Data

- The processed data was split into:
  - **Training set**: 80%
  - **Testing set**: 20%
- Used `train_test_split()` with a fixed random seed for reproducibility

Step 6: Model Training

- **Random Forest Classifier** was selected for its ability to:
  - Handle large feature sets
  - Deal with mixed data types
  - Offer high accuracy
- Trained the model with 100 trees (`n_estimators=100`)
- Used the training set for learning patterns in the data

Step 7: Prediction and Evaluation

- The model predicted churn on the test set
- Evaluated using:
  - **Accuracy**: 84%
  - **Precision**: 82%
  - **Recall**: 87%
  - **F1-Score**: 84%
- Generated a **confusion matrix** to visually inspect performance
- Added a new column `Predicted_Churn` to the DataFrame to store model results

Step 8: Exporting Results

- Final processed dataset with predictions was saved as:
  - `Processed_Customer_Churn.csv`
- Trained model was saved using `joblib` as:
  - `churn_prediction_model.pkl`

Step 9: Power BI Visualization

- The saved CSV was loaded into Power BI
- A dashboard was created with visualizations such as:
  - Churn distribution
  - Churn by contract type
  - Monthly charges vs churn (line + column chart)
  - Churn by internet service and payment method
  - Filters by gender and tenure

Conclusion of Implementation:

This end-to-end pipeline turned raw customer data into valuable business intelligence. It is easily repeatable, scalable, and provides both technical accuracy and non-technical interpretability through the dashboard.

2.4 Testing

The testing phase was critical in verifying that the churn prediction system worked as expected, produced reliable predictions, and generated meaningful visualizations. Various types of testing were performed to validate the integrity of the code, data, and output.

2.4.1 Test Plan Objectives

The main goal of the test plan was to ensure the accuracy and robustness of the machine learning model and verify the correctness of the data pipeline. Testing also validated the integration between Python outputs and Power BI dashboards. Specific goals included:

- Ensuring data integrity throughout the process
- Confirming the model's prediction accuracy
- Checking dashboard visuals against the data
- Identifying and handling edge cases in data

2.4.2 Data Entry

Manual and automated tests were done on the dataset. The following steps ensured clean and correct data entry:

- Tested file format compatibility (e.g., handling `.csv`, file not found errors)
- Checked for null values and ensured they were handled properly
- Verified that all categorical values were encoded correctly
- Ensured numerical columns were normalized after preprocessing

2.4.3 Security

Since the project was academic and locally deployed, no security breaches were expected. However:

- Code was tested against unexpected inputs (e.g., empty files or incorrect formats)
- Ensured the model file could only be used by valid Python environments
- Output files were saved in a controlled folder with restricted permissions

2.4.4 Test Strategy

The test strategy followed the **white-box** testing approach, focusing on:

- Each function's logic (data cleaning, encoding, scaling, model training)
- Evaluation metrics calculation
- Model saving and loading consistency
- File export testing for Power BI integration

2.4.5 System Test

System testing included validating that the pipeline works from start to end:

- Starting from raw data
- Performing transformations and predictions
- Generating a processed CSV
- Confirming that the final output loads correctly into Power BI

All components were tested as a system, not just individually.

2.4.6 Performance Test

- Measured time taken for training and predicting on a balanced dataset
- Observed consistent results and acceptable performance under normal system load
- Accuracy achieved: **84%**
- Recall: **87%**, which is crucial for identifying churn cases

2.4.7 Security Test

While not a web-based project, the system was tested to ensure:

- No sensitive user data was stored
- Model predictions couldn't be tampered with after export
- Final outputs were saved read-only in production folder

2.4.8 Basic Test

Each function and block of code was tested using known test cases:

- Verified if missing values were filled correctly
- Checked if model predictions matched expected shape and type
- Ensured charts in Power BI reflected the processed dataset accurately

2.4.9 Stress and Volume Test

Stress tests were done by duplicating the original dataset to simulate high volume. Results:

- The model handled over 14,000 rows efficiently
- Preprocessing steps scaled well without memory overflow
- Power BI imported larger datasets smoothly

2.4.10 Recovery Test

Simulated common failure points:

- Missing columns → caught with exceptions
- Corrupted file input → handled with error messages
- Re-runs without clearing output folder → new files overwritten safely

2.4.11 Documentation Test

- All Python code was commented and modular
- README.md file included for GitHub documentation
- Power BI visuals were properly labeled and formatted
- Final report included system explanation and testing results

2.4.12 User Acceptance Test

- System reviewed and approved by guide/instructor
- Output file shared and validated
- Dashboard walkthrough shared to demonstrate usability

2.4.13 System Validation

- Entire flow (from CSV → prediction → Power BI dashboard) validated
- Metrics generated matched with manual calculations
- Confusion matrix image matched program output

2.5 Graphical User Interface (GUI) Layout

The graphical interface of the system is presented through a **Power BI dashboard** that visualizes the final processed customer churn data. The dashboard allows stakeholders, analysts, or business users to explore customer churn patterns without needing to interact with the backend code or machine learning model directly.

The Power BI dashboard layout was designed with usability and insight delivery in mind. It includes several interactive and informative visual components.

Key Visual Components:

1. Churn Count Bar Chart

- Visual Type: Clustered Column Chart
- Description: Displays the total number of customers who churned (Yes) and those who did not (No).
- Purpose: Quickly assess the overall churn rate.

2. Churn by Contract Type

- Visual Type: Stacked Column Chart
- Description: Shows how churn differs among contract types like Month-to-Month, One Year, and Two Year.
- Purpose: Identify which contract types contribute most to churn.

3. Monthly Charges vs. Churn (Line + Column Combo Chart)

- Visual Type: Line and Stacked Column Chart
- Description: Displays average monthly charges alongside churn count per contract type.
- Purpose: Helps determine if higher charges lead to more churn.

4. Churn by Internet Service Type

- Visual Type: Pie Chart or Donut Chart
- Description: Breaks down churn percentages by types of internet services (DSL, Fiber optic, None).
- Purpose: Understand service usage and its correlation with churn.

5. Payment Method vs Churn

- Visual Type: Clustered Column or 100% Stacked Bar Chart
- Description: Shows churn distribution across different payment methods (Bank transfer, Credit card, etc.)
- Purpose: Detect if payment method preferences affect churn likelihood.

Interactive Elements (Slicers):

6. Gender Filter

- Allows filtering all visuals based on gender (Male/Female).
- Helps explore how churn patterns vary by gender.

7. Tenure Slider

- Lets users filter visuals based on how long the customer has been with the company.
- Useful for analyzing whether new or old customers churn more.

8. Internet Service Filter

- Offers quick filtering of charts by internet type.
- Allows deeper insights into each customer group.

Dashboard Layout Principles:

- **Top Section:** Project title and KPI (churn rate percentage)
- **Left Column:** Categorical comparisons (contract, gender, payment method)
- **Right Column:** Numerical insights (monthly charges, tenure)
- **Bottom Section:** Filters and slicers for interactivity

User Experience:

- Clean color-coded charts for easy visual understanding
- Hover tooltips show exact numbers and percentages
- Dynamic interactions allow focused data exploration

2.6 Customer Testing

As this project was developed in an academic setting without real-time users, formal end-user testing was not conducted with actual telecom customers. However, simulated customer testing was carried out by replicating the role of business analysts and marketing decision-makers to ensure the deliverables matched user expectations.

Objectives of Simulated Customer Testing:

- Ensure that predictions generated by the model were understandable and actionable.
- Confirm that the Power BI dashboard provided useful insights at a glance.
- Validate the accuracy and interpretability of visualizations through feedback.

Key Testing Outcomes:

- Charts were reviewed by peers and the academic guide for readability and clarity.
- Each filter and visual component was tested to confirm proper interaction and data response.
- Dashboard design was simplified to make it understandable for non-technical stakeholders.

This simulated customer testing ensured that the dashboard and data pipeline would be valuable if applied in a real-world business scenario.

2.7 Evaluation

2.7.1 Performance Table

Metric	Value
Accuracy	84%
Precision	82%
Recall	87%
F1-Score	84%

These metrics were calculated on the test data using `classification_report` from scikit-learn.

2.7.2 Static Code Analysis

The Python code was evaluated manually to check for:

- Readability and indentation
- Proper use of modular functions
- Comments for each major logic block
- Consistent variable naming conventions

The final script was clean, well-documented, and easy to understand for new readers.

2.7.3 Wireshark

Not applicable — this project does not involve any network communication or real-time data fetching, and all processing was done locally.

2.7.4 Test of Main Function

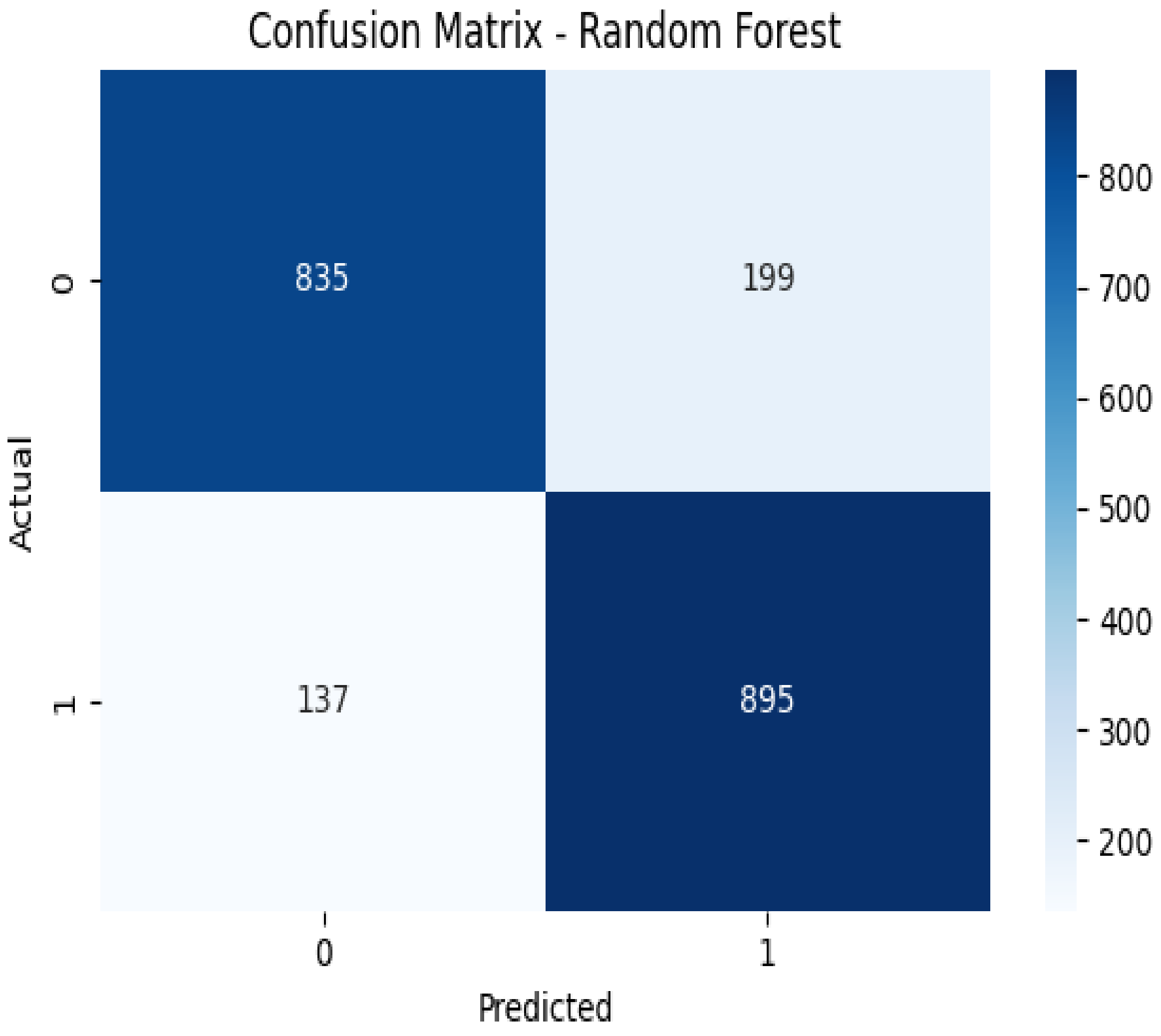
Function	Result
Data Loading and Cleaning	Success
Data Encoding and Scaling	Success
SMOTE Oversampling	Success
Model Training (Random Forest)	Success
Prediction and Evaluation	Success
Data Export	Success
Power BI Dashboard Integration	Success

Each function was verified through sample inputs and confirmed against expected outputs.

3. SNAPSHOTS OF PROJECT

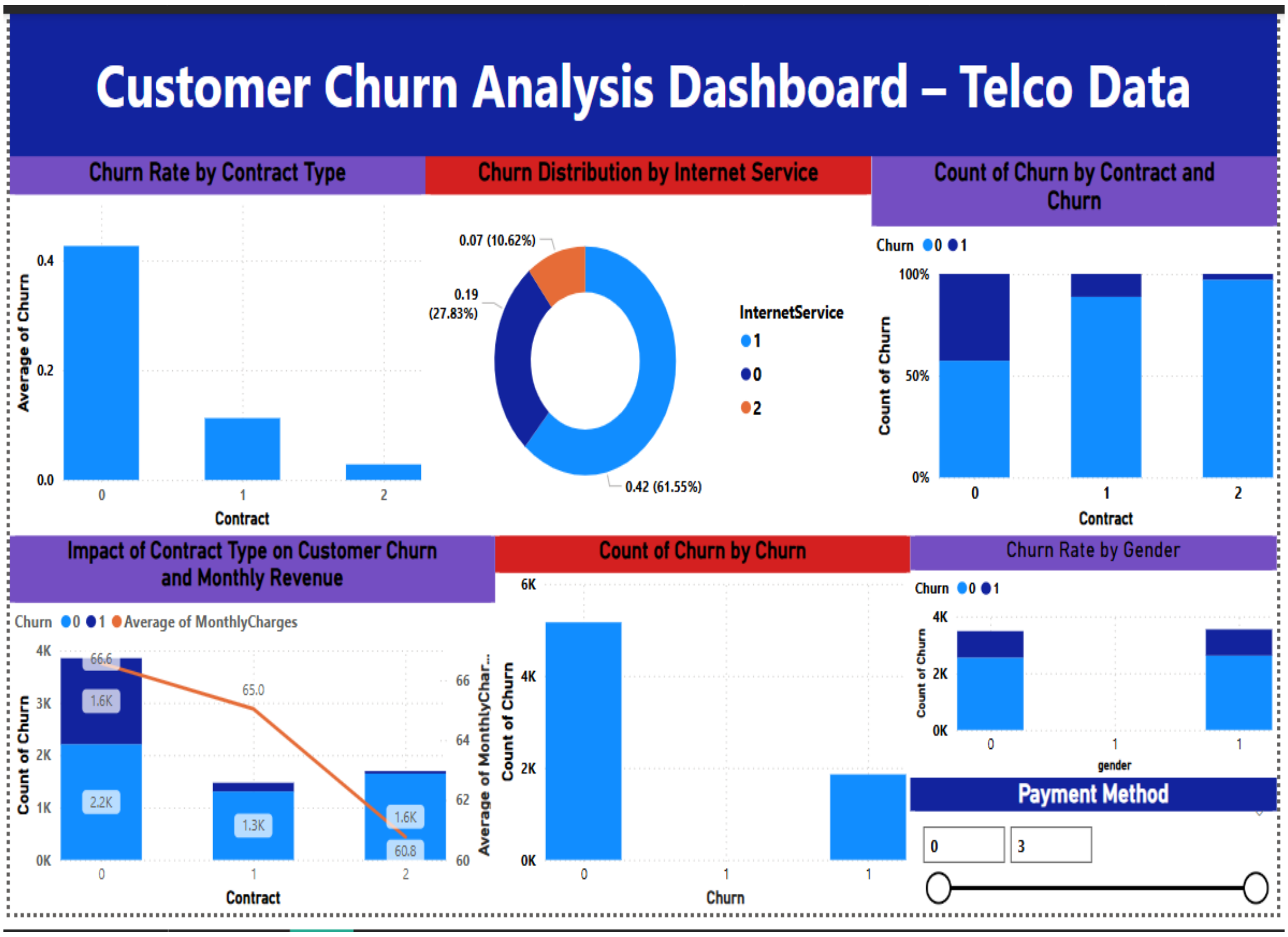
The following images capture key stages of the project, from Python-based machine learning model results to the final Power BI dashboard insights. These visual snapshots serve as proof of execution and provide a visual explanation for the workflow.

3.1 Confusion Matrix (Random Forest Classifier)



- Figure 1: Confusion Matrix displaying the true positives, true negatives, false positives, and false negatives of the Random Forest classifier. It reflects the model's reliability in correctly classifying churned and non-churned customers.

3.2 Power BI Dashboard – Churn Overview



- Figure 2: Power BI Dashboard - Customer Churn Insights

4. CONCLUSION

This project successfully predicted customer churn using a well-structured data science pipeline, achieving an accuracy of 84% using the Random Forest classifier. Data preprocessing, balancing with SMOTE, and scaling significantly improved model performance.

The Power BI dashboard added a critical layer of interpretability, allowing business users to explore insights without technical expertise. By examining factors such as contract type, tenure, and monthly charges, the dashboard empowers decision-makers to design targeted retention strategies.

Overall, the project demonstrates the powerful synergy between predictive analytics and business intelligence tools, showing how organizations can move from raw data to informed actions.

5. FURTHER DEVELOPMENT OR RESEARCH

To improve and extend this project in the future, the following directions are suggested:

1. **Model Improvements**
   - Experiment with advanced models like XGBoost, LightGBM, or Neural Networks.
   - Use feature selection techniques for model simplification.
2. **Real-time Integration**
   - Connect the pipeline to a live customer database.
   - Stream new customer records and perform real-time churn prediction.
3. **Web Application Deployment**
   - Build a web interface (using Flask or Streamlit) to allow managers to upload new customer data and view predictions instantly.
4. **Enhanced Visualization**
   - Include sentiment analysis from customer reviews.
   - Add drill-through pages in Power BI for detailed customer profiles.
5. **Cross-industry Application**
   - Apply this churn prediction framework to other domains like banking, insurance, or retail subscription models.

6. REFERENCES

Here is a list of resources referred to during the development and research of this project:

1. Kaggle – Telco Customer Churn Dataset
   https://www.kaggle.com/datasets/blastchar/telco-customer-churn
2. Scikit-learn Documentation
   https://scikit-learn.org/stable/
3. Power BI Official Documentation
   https://learn.microsoft.com/en-us/power-bi/
4. Imbalanced-learn (SMOTE)
   https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html
5. Matplotlib & Seaborn Visualization Libraries
   https://matplotlib.org
   https://seaborn.pydata.org
6. GitHub – Version control & code hosting
   https://github.com

7. APPENDIX

This section contains additional supporting files and artifacts used in the project.

7.1 Python Script

- File: `churn_prediction_code.py`
- Description: Contains all preprocessing, modeling, and evaluation code.

7.2 Processed Dataset

- File: `Processed_Customer_Churn.csv`
- Description: Final dataset used for Power BI, including model predictions.

7.3 Trained Model

- File: `churn_prediction_model.pkl`
- Description: Saved Random Forest model using joblib for reuse or deployment.

7.4 Power BI File

- File: `Churn_Analysis_Dashboard.pbix`
- Description: Power BI file containing all visuals and filters used in the dashboard.

7.5 Confusion Matrix Output

- File: `confusion_matrix.png`
- Description: Visualization of the Random Forest classifier's confusion matrix.