Customer Churn Analysis and Prediction in Banking Sector

Software Test Document (STD)

Volkan Çamlı

Table of Contents

1. INTRODUCTION

1.1. Purpose

This document has been prepared to define the testing process of the software system developed for the “Customer Churn Analysis in the Banking Sector” project. The purpose is to plan the testing activities required to evaluate the system’s accuracy, reliability, and compliance with its requirements.

1.2. Scope

The testing process covers the data preprocessing module, training and evaluation of machine learning models, output generation, and visualization using Power BI. The tests aim to verify whether each component of the system functions as expected.

1.3. Features to Be Tested

- Data cleaning operations (missing value handling, categorical encoding)
- Feature engineering outputs
- Model training and prediction performance (logistic regression, random forest, LightGBM, neural network)
- Evaluation metrics (accuracy, F1-score, AUC)
- Model comparison and selection process
- Consistency of Power BI output visuals

1.4. Features Not to Be Tested

- Real-time system integration
- Web-based user interface
- Distributed system performance testing

1.5. Pass/Fail Criteria

For a test to be considered successful, the system must produce the expected output and the evaluation metrics must exceed acceptable thresholds.

- Model accuracy ≥ 80%
- AUC score ≥ 0.85
- Data files processed without corruption

2. TEST ITEMS

2.1. Data Preprocessing Module

- Checking for missing values
- Encoding of categorical variables (Label Encoding, One-Hot Encoding)
- Proper application of customer segmentation (K-means)
- Accurate generation and integration of new features (age group, risk level, income level)

2.2. Model Training and Evaluation Module

- Successful training of logistic regression, decision tree, random forest, lightGBM and neural network models
- Accurate prediction generation
- Proper calculation of evaluation metrics (Accuracy, F1-score, AUC)
- Valid output of confusion matrices

2.3. Model Comparison and Selection

- Correct identification of the best-performing model
- Detection of anomalies or unexpected results

2.4. Visualization (Power BI)

- Accurate rendering of prediction results in Power BI
- Correct display of segment-based analysis (churn rate, risk distribution, etc.)
- Responsive updates of visuals when filters are applied

3. TEST APPROACH

The testing process follows a modular and phased approach, evaluating each component separately and then as part of the integrated system. The following strategies will be applied:

3.1. Unit Testing

- Each module (preprocessing, modeling, output) will be tested independently.
- Code cells will be executed step-by-step in Jupyter Notebook to verify correct functionality.
- New features created through feature engineering will be checked for correctness.

3.2. Functional Testing

- Each model will be tested to ensure it provides correct and valid predictions.
- Consistency will be checked between training and testing datasets.
- Predicted Exited values will be compared to actual values to measure performance.

3.3. Comparative Model Testing

- Evaluation metrics (F1-score, AUC) will be compared across models to verify proper model selection.
- Outlier or abnormal results will be analyzed in detail.

3.4. Visualization Testing

- The correctness, clarity, and responsiveness of visuals created in Power BI will be tested.
- For example, when applying filters like “Female customers in Germany”, the dashboard visuals should respond accordingly.

3.5. Tools Used

- Jupyter Notebook (manual cell-based testing)
- Python assert statements for error handling
- sklearn.metrics for automated evaluation
- Manual inspection of interactive visuals in Power BI

4. TEST DELIVERABLES

The following outputs will be produced as a result of the testing process:

4.1. Test Scenarios and Test Cases

- Step-by-step testing instructions and expected vs. actual results for each module
- For example: “Are missing values correctly handled?”, “Is AUC ≥ 0.85 after model training?”

4.2. Test Execution Logs

- Cell outputs and screenshots from Jupyter Notebook showing successful or failed test cases
- Any error messages encountered and the resolutions applied

4.3. Model Evaluation Report

- Comparative table of Accuracy, F1-score, and AUC metrics for all trained models
- Explanation of why the best model was selected

4.4. Power BI Visual Testing Report

- Summary of filter responsiveness and interaction test results in Power BI
- Screenshots or notes verifying visual consistency

5. TEST ENVIRONMENT REQUIREMENTS

To ensure successful execution of testing activities, the following software and hardware environments are required:

5.1. Software Requirements

- Operating System: Windows 10 or later / macOS 12+
- Development Environment:
  - Jupyter Notebook (Anaconda Distribution recommended)
  - Python 3.8+
- Libraries: pandas, numpy, scikit-learn, xgboost, lightgbm, matplotlib, seaborn, keras, tensorflow
- Visualization: Microsoft Power BI Desktop

5.2. Hardware Requirements

- Processor: Minimum Intel i5 or equivalent, i7 recommended
- RAM: At least 8 GB, 16 GB recommended
- Storage: Minimum 2 GB free space (for dataset and output files)
- Screen Resolution: At least 1366x768 for effective Power BI usage

5.3. Additional Requirements

- Internet access for Power BI (for visual updates and data access)
- Internet access for Python (for installing/updating libraries)

6. RESPONSIBILITIES

Since this is an individual academic project, all test responsibilities are handled by the student. The academic advisor provides feedback and high-level guidance throughout the process.

Role	Responsible Person	Description of Duties
Test Executor	Volkan Çamlı	Plans and executes all testing activities, error analysis, and reporting
Academic Advisor	Prof. Dr. Mehmet Serdar Güzel	Provides academic feedback and guidance on testing outcomes
Visual Inspector	Volkan Çamlı	Manually checks the consistency and functionality of Power BI visualizations

7. RISKS AND ISSUES

Risk / Issue	Description	Impact Level	Mitigation / Solution
Imbalanced dataset (class imbalance)	Only ~20% of customers are churners, which may affect model performance	Medium	Use balanced metrics like AUC and F1-score; apply sampling if needed
Inconsistent model outcomes	Different models may produce conflicting predictions	Medium	Apply cross-validation and carefully compare evaluation metrics
Library version conflicts	Compatibility issues between Python package versions	Low	Use virtual environments and fix library versions
Power BI visuals not syncing with data	Visuals may not update correctly after filters are applied	Low	Manually test visuals and validate data integrity before updating
Hardware limitations	Inadequate RAM/CPU may slow down operations with large datasets	Low	Use sample data for testing or run on more powerful hardware