Project Documentation: Credit Scoring System Development Week-6

Project Overview

The project involves creating a credit scoring system for a financial institution, "Bati Bank," which is partnering with an e-commerce platform to offer a "Buy-Now-Pay-Later" service. The goal is to develop a robust machine-learning pipeline to assess customer creditworthiness, predict the likelihood of default, and deploy a production-ready REST API for real-time risk prediction.

Contributions

Task 1: Understanding Credit Risk

- Researched the concept of credit risk, the Basel II Capital Accord, and the RFMS (Recency, Frequency, Monetary, and Stability) formalism for scoring.
- Identified key parameters that influence credit risk, such as transaction patterns, historical repayment behavior, and fraud detection metrics.

Task 2: Exploratory Data Analysis (EDA)

- **Data Insights:**
  - Loaded and analyzed the dataset to understand the structure, missing values, and key variables.
  - Generated summary statistics and visualized the distribution of numerical and categorical variables.
- **Correlation Analysis:**
  - Investigated relationships between features to identify predictors for default risk.
- **Outlier and Missing Data Handling:**
  - Detected and handled outliers using boxplots.
  - Imputed missing data with the median for numerical columns and the mode for categorical columns.

Task 3: Feature Engineering

- **Aggregate Features:**
  - Created customer-level aggregate features such as `Total Transaction Amount`, `Average Transaction Amount`, `Transaction Count`, and `Standard Deviation of Transaction Amounts`.
- **Extracted Features:**

    - Extracted temporal features such as `Transaction Hour`, `Transaction Day`, `Transaction Month`, and `Transaction Year`.
- **Categorical Encoding:**
    - Used One-Hot Encoding for nominal variables and Label Encoding for ordinal variables.
- **Normalization/Standardization:**
    - Scaled numerical features using StandardScaler to ensure consistent ranges.
- **Weight of Evidence (WoE) Binning:**
    - Applied WoE encoding to create bins that separate high-risk and low-risk groups.

Task 4: Model Development

- **Data Splitting:**
    - Split the dataset into training (80%) and testing (20%) sets.
- **Model Selection:**
    - Implemented and trained Logistic Regression and Gradient Boosting Machines (GBM).
- **Hyperparameter Tuning:**
    - Conducted hyperparameter optimization using Grid Search to improve model performance.
- **Model Evaluation:**
    - Evaluated models using metrics such as Accuracy, Precision, Recall, F1-Score, and ROC-AUC.
    - Identified Gradient Boosting as the best-performing model.

Task 5: Model Serving via REST API

- **API Development:**
    - Created a REST API using FastAPI to serve the trained model for real-time predictions.
    - Developed endpoints:
        - `/status`: Health check endpoint for API availability.
        - `/predict`: Accepts transaction details as JSON input and returns risk predictions.
- **Deployment:**
    - Designed the API for deployment on cloud platforms such as AWS or Google Cloud for production use.
    - Ensured scalability and security for handling real-time transactions.

Project Implementation Steps

1. **Data Preprocessing:**

- Cleaned and imputed missing values.
- Scaled numerical features and encoded categorical variables.

2. **Feature Engineering:**
    - Aggregated customer-level features and extracted temporal details from transaction data.
    - Applied Weight of Evidence (WoE) for better classification.
3. **Model Training:**
    - Trained two models (Logistic Regression and GBM) and selected the best-performing model based on evaluation metrics.
    - Saved the trained model using `joblib`.
4. **REST API Development:**
    - Developed a production-ready REST API to integrate the trained model with real-time systems.
    - Defined preprocessing logic within the API to handle incoming requests.
5. **Deployment:**
    - Prepared the API for deployment on a web server or cloud platform.

Technologies Used

- **Languages and Frameworks:**
    - Python (pandas, NumPy, scikit-learn, FastAPI)
- **Data Visualization:**
    - Matplotlib, Seaborn
- **Model Development:**
    - scikit-learn (Logistic Regression, Gradient Boosting)
- **API Development:**
    - FastAPI
- **Deployment:**
    - Cloud-ready architecture for scalability.

Challenges Addressed

- **Handling Imbalanced Data:**
    - Applied techniques like scaling and WoE binning to improve the model's sensitivity to high-risk groups.
- **Feature Engineering Complexity:**
    - Transformed raw transaction data into meaningful customer-level aggregates and time-based features.
- **Real-Time Prediction:**
    - Built a lightweight REST API for seamless integration with external systems.

Outputs and Deliverables

1. **Credit Scoring Model:**
    - Logistic Regression and Gradient Boosting models with tuned hyperparameters.
2. **Feature-Engineered Dataset:**
    - Dataset with aggregated, encoded, and normalized features.
3. **REST API:**
    - Fully functional API endpoints for real-time credit scoring predictions.
4. **Documentation:**
    - Clear project documentation, including implementation steps, challenges, and key insights.

Future Enhancements

- **Advanced Modeling:**
    - Explore deep learning techniques for complex patterns in transaction data.
- **Fraud Detection Integration:**
    - Incorporate fraud detection capabilities into the credit scoring model.
- **Dashboard Integration:**
    - Develop an interactive dashboard to visualize customer credit scores and risk profiles.
