Low-Level Design (LLD) Document

1. Introduction

The "Credit Card Default Prediction" project aims to develop a machine learning pipeline for predicting credit card defaults based on historical transaction data. This document provides a detailed overview of the project's design, including architectural components, data flow, and system interactions.

2. Architectural Overview

The project architecture consists of the following components:

- Data Collection Module: Responsible for gathering historical credit card transaction data from reliable sources.
- Data Preprocessing Module: Handles data cleaning, missing value handling, and numerical feature scaling.
- Feature Engineering Module: Selects relevant features and engineers new features for model training.
- Model Training Module: Trains a machine learning model (e.g., SVM) on the preprocessed data.
- Web Application Module: Develops a user-friendly web application for user input and predictions.
- Model Deployment Module: Deploys the trained model as a service accessible through the web application.

3. Detailed Component Design

3.1 Data Collection Module

- Interfaces with external data sources (e.g., CSV files, databases) to retrieve historical credit card transaction data.
- Cleans the raw data by handling missing values and removing irrelevant features.
- Stores the cleaned data in a suitable format for further processing.

3.2 Data Preprocessing Module

- Receives the cleaned data from the Data Collection Module.
- Scales numerical features using techniques like Min-Max scaling or Standardization.
- Performs one-hot encoding for categorical variables and handles any remaining missing values.

3.3 Feature Engineering Module

- Selects relevant features based on domain knowledge and correlation analysis.
- Engineers new features using techniques like binning, log transformations, or interaction terms.

3.4 Model Training Module

- Receives the preprocessed data from the Data Preprocessing Module.
- Splits the data into training and validation sets for model evaluation.
- Trains a Support Vector Machine (SVM) classifier using the training data.
- Evaluate the SVM model's performance using metrics like accuracy, precision, recall, and F1-score.

3.5 Web Application Module

- Develop a user-friendly web application using the Flask web framework.
- Provides a user interface for users to input credit card details (e.g., credit limit, age, payment history).
- Collects user inputs and sends them to the Model Deployment Module for prediction.

3.6 Model Deployment Module

- Deploys the trained SVM model as a RESTful API using Flask.
- Accepts user input from the Web Application Module and passes it to the SVM model.
- Returns the credit card default prediction (0 or 1) to the web application.

4. Database Design

The project uses a database to store historical credit card transaction data. The database schema includes tables for storing customer information, payment history, and default labels. The database design ensures data integrity and efficient retrieval for model training and prediction.

5. Error Handling

The project implements robust error-handling mechanisms at various data processing and prediction stages. In case of any issues with data input or model prediction, detailed error messages are provided to users.

6. Performance Optimization

Efforts are made to optimize the performance of the model training and prediction processes. Techniques like parallel processing and caching are used to enhance the system's speed and resource utilization.

7. Security Considerations

To ensure data security and privacy, the project implements authentication and authorization mechanisms for user access to the web application. Data encryption and secure communication protocols are used for data transmission.

8. Testing and Validation

The project includes unit tests and integration tests to validate the correctness of each component. Model evaluation using cross-validation ensures the SVM model's accuracy and generalization to new data.

9. Deployment Strategy

The web application and model are deployed on a cloud server using Amazon Web Services (AWS) or a similar cloud platform. Continuous integration and continuous deployment (CI/CD) pipelines are set up for seamless deployment and updates.

10. Documentation

The project includes comprehensive code comments, user documentation, and developer documentation to facilitate easy understanding, maintenance, and collaboration.

11. References

- Scikit-learn: Machine Learning in Python. Pedregosa, F., et al. (2011). Journal of Machine Learning Research, 12, 2825-2830.
- Flask: Web Development with Python. Ronacher, A. (2018). O'Reilly Media.
- Python. https://www.python.org
- HTML, CSS, and JavaScript. MDN Web Docs. https://developer.mozilla.org/
- Credit Card Default Dataset. https://www.kaggle.com/
- XGBoost: A Scalable Tree Boosting System. Chen, T., & Guestrin, C. (2016). Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785-794.
- Amazon Web Services (AWS). https://aws.amazon.com/

This revision aims to present the document in a clear and concise manner, maintaining a professional tone and structure.