LOW LEVEL

DESIGN(LLD)

CREDIT CARD DEFAULT PREDICTION

SATYAM SHARMA

DATA SCIENTIST

I.

II. Introduction

In the Low-Level Design (LLD) phase, we delve into the specifics of the Credit Card Default Prediction System, focusing on the finer technical details, system components, and their interactions. This document is tailored to provide an in-depth understanding of the system's architecture, database design, modules, and code structure. The LLD serves as a blueprint for the implementation of the system, ensuring that it aligns with the high-level design and meets the project's objectives

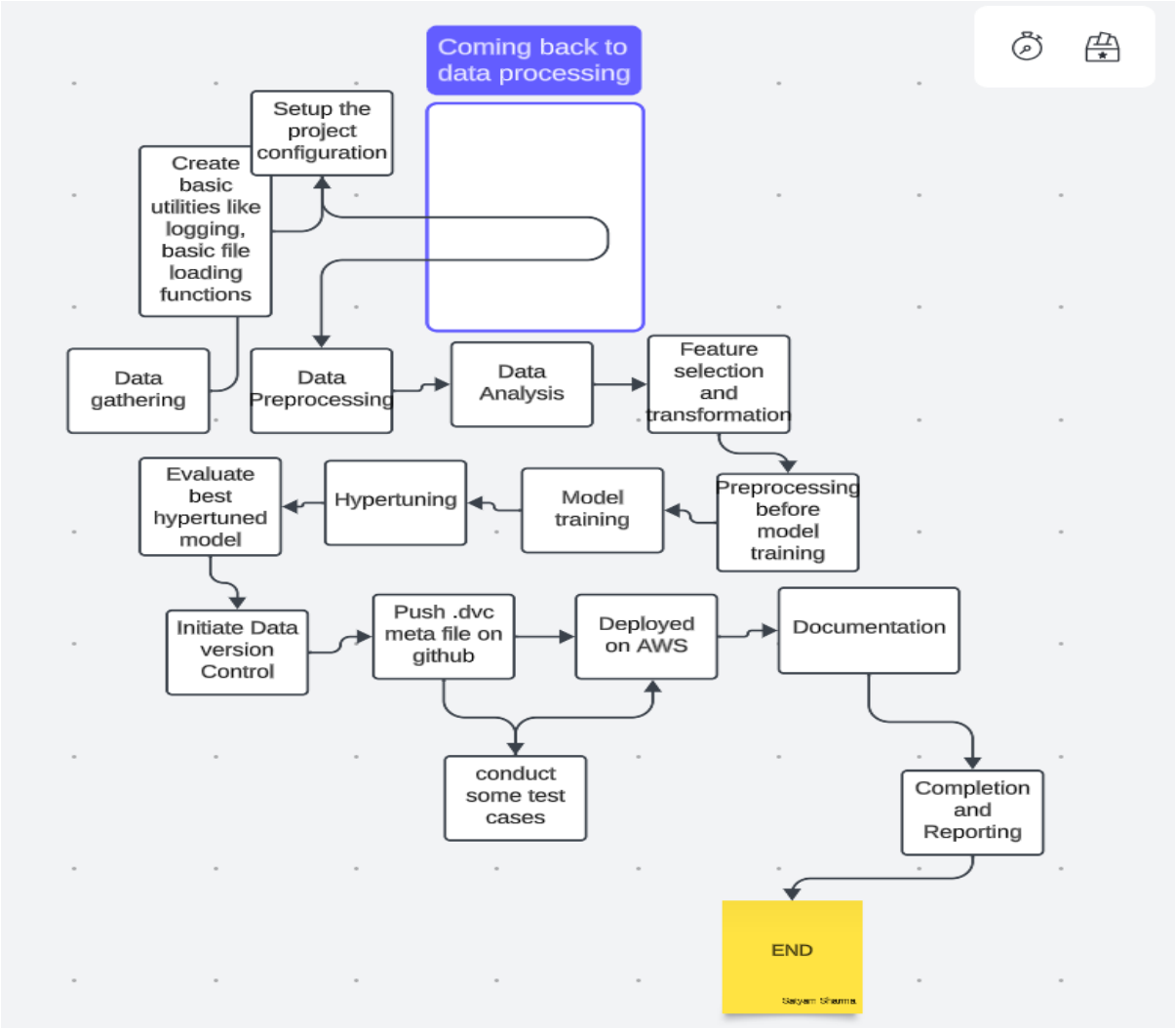
a. What is low-level design document?
   - A Low-Level Design Document (LLD) is a comprehensive technical document that provides a detailed, in-depth description of the architecture, components, modules, data flow, and implementation specifics of a software system or application. The LLD serves as a bridge between the high-level design and the actual coding and development process.
   - Key components of an LLD typically include detailed explanations of individual modules, their functions, and how they interact with other modules in the system. It also comprises data flow diagrams that illustrate the flow of data within the system, showing how data is processed, transformed, and transmitted between different components. The LLD includes information on the database design, including table structures, relationships, and data storage specifications. Additionally, it might feature object-oriented representations of classes and their relationships, if applicable, and detail specific algorithms or methods used within the system. If relevant, user interfaces with detailed descriptions, including screen layouts, form designs, and user interaction workflows are outlined in the document.
   - The LLD also describes security measures and mechanisms incorporated into the system to safeguard data and user privacy. It provides information on how data is stored, accessed, and managed within the system and specifies APIs or external interfaces, including request-response formats, protocols, and endpoints. Error handling and exception handling strategies are elaborated, including error codes, messages, and actions to be taken. Furthermore, the LLD includes techniques and strategies employed to optimize system performance, such as algorithms, caching, and resource utilization.
   - The LLD is a critical document that guides software developers in implementing the system, ensuring that it adheres to the high-level design, meets project objectives, and is technically sound. It acts as a detailed blueprint for the development team, enabling them to efficiently translate design concepts into functional code while addressing technical considerations and requirements.

b. Scope
   - The scope of the Low-Level Design (LLD) document is to provide a detailed technical blueprint for the implementation of the Credit Card Default Prediction System. It encompasses all technical aspects, modules, components, data flow, and interfaces that are essential for building the system.

   - The LLD outlines the technical architecture, specifying how different components interact and communicate within the system. It defines the boundaries and

responsibilities of each module, ensuring that developers have a clear understanding of their roles and functions.

III. Architecture



IV. Architecture Description

a. Data Gathering

- Collect raw data from kaggle, including financial records, credit card transactions, and other relevant datasets.
- Ensure the data collected is comprehensive, covering a wide range of attributes and variables related to credit card usage and customer behaviour.
- Utilize data extraction tools and techniques to retrieve structured and unstructured data from multiple sources.
- Implement data cleansing processes to remove duplicates and handle missing values effectively.
- Consider using external data sources and APIs to enrich the dataset with additional information that might enhance prediction accuracy.
- Establish data retrieval and update schedules to keep the dataset current for ongoing model training and prediction.

b. Data Description

- Provide an overview of the data structure, including the number of records and attributes in the dataset.

- Present details about each attribute, such as the data type, whether it's categorical or numerical, and its significance in the context of credit card default prediction.
- Highlight any unique identifiers, such as customer IDs, that can be used for data linkage.
- Discuss the presence of any target variables or labels, which will be crucial for prediction tasks.
- Include a summary of key statistics for numerical attributes, such as mean, median, standard deviation, and quartiles.
- Describe the distribution of categorical attributes, showcasing the frequency of different categories.

c. Data Transformation
   - Aggregate multiple columns related to dues into a single "Dues" column, summing up the values for each month (e.g., April to September).
   - Combine and transform columns related to previous payments into a consolidated "Previous Payments" column, calculating the sum or other relevant aggregations.
   - Apply one-hot encoding to categorical variables, converting them into binary columns to represent categories and make them suitable for machine learning algorithms.
   - Standardize or normalize numerical attributes as necessary to ensure that they have a consistent scale for modeling.
   - Perform feature engineering, creating new variables that capture meaningful information related to credit card usage and customer behavior.
   - Handle any outliers in the data through appropriate techniques, such as clipping or transformation.
   - Address missing values by imputing them using relevant strategies, ensuring that the data is complete and suitable for modeling.

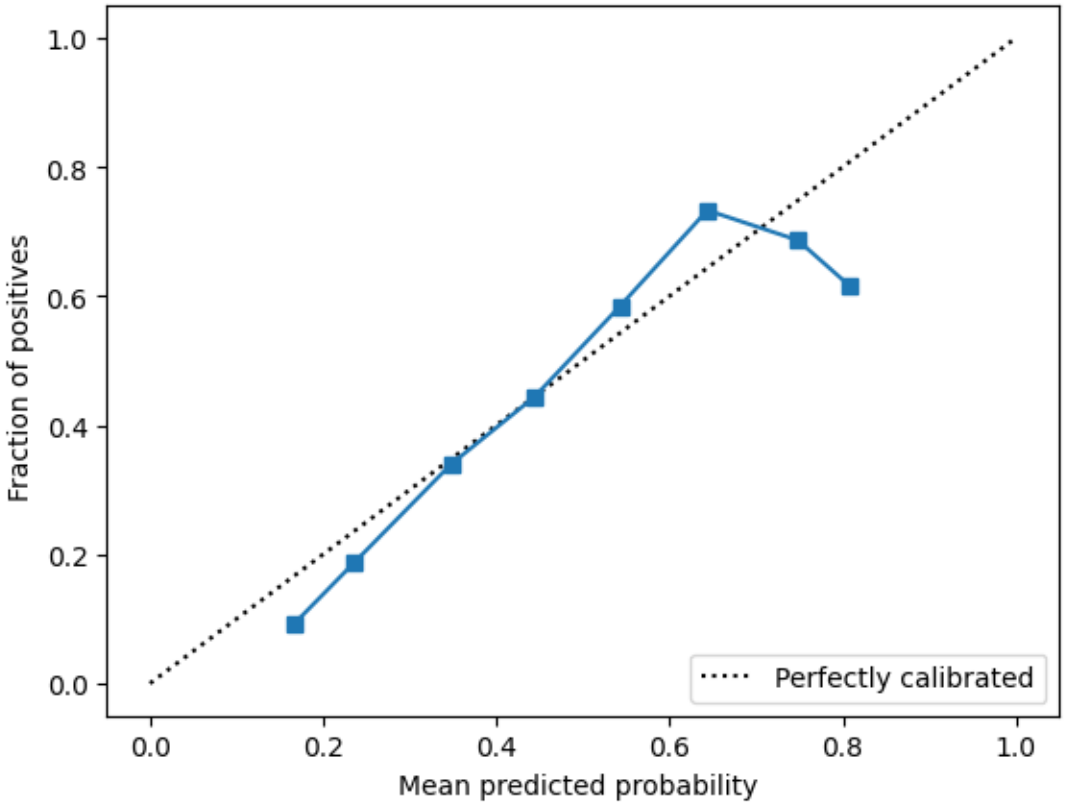
d. Data Preprocessing before model training
   - Perform feature selection to identify and retain only relevant attributes, removing unnecessary or redundant features that do not significantly contribute to the prediction task.
   - Address class imbalance, if present, by using techniques like oversampling, undersampling, or the Synthetic Minority Over-sampling Technique (SMOTE).
   - Split the dataset into training and testing sets to facilitate model evaluation and validation.
   - Standardize or normalize the data to ensure that all features have the same scale and distribution, preventing any attribute from dominating model training.
   - Address any remaining missing values through imputation, ensuring that the dataset is complete for model training.
   - Encode the target variable or labels into a format suitable for machine learning algorithms, such as converting categorical labels into numerical values.
   - Ensure that the dataset is well-prepared for machine learning model training, addressing issues related to imbalance, scaling, and feature selection.

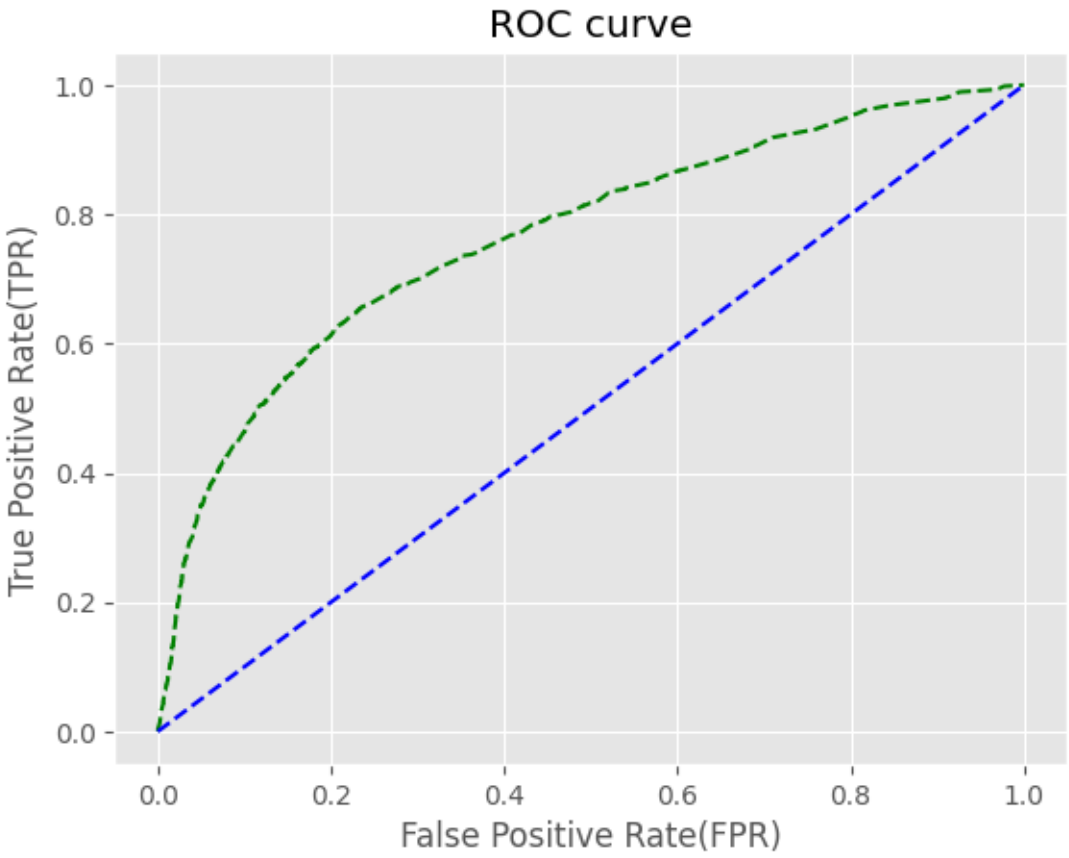
e. Model training

- Begin the model training phase by selecting appropriate machine learning algorithms for credit card default prediction based on the dataset's characteristics and problem requirements.
- Preprocess the training data, which includes feature scaling, encoding, and handling missing values, ensuring that it aligns with the chosen algorithms' prerequisites.
- Divide the data into training and validation sets, facilitating model performance assessment and validation.
- Train multiple machine learning models, including Logistic Regression, AdaBoostClassifier, DecisionTreeClassifier, and Support Vector Classifier (SVC).
- Apply different hyperparameters and configurations to each algorithm, aiming to fine-tune and optimize their performance.

f. Model Evaluation
   - Assess the predictive performance of the trained models using various evaluation metrics, including:
   - *Accuracy Score:* Measure the overall correctness of predictions.
   - *Mean Squared Error (MSE):* Evaluate the model's predictive accuracy by quantifying the squared differences between predicted and actual values.
   - *Receiver Operating Characteristic (ROC) Curve:* Examine the trade-off between true positive rate and false positive rate.
   - *Area Under the ROC Curve (AUC-ROC):* Quantify the model's ability to distinguish between positive and negative classes.
   - *Calibration Curve:* Analyse the agreement between predicted probabilities and actual outcomes to assess the model's calibration.
   - Generate visualizations, such as ROC curves and calibration plots, to gain insights into the models' performance and calibration.
   - Compare the models' results using the specified metrics to identify the model with the best predictive capabilities.

   - Document the evaluation process, recording the metric scores for each model and providing a rationale for selecting the best-performing model for deployment.





g. Data from user(Via UI)

- Create a user-friendly web-based interface that allows users to input credit card transaction details, which include features relevant for default prediction, such as payment history, credit limit, and more.
- Implement an intuitive form or user interface that guides users through the process of providing the necessary information.Ensure the UI validates user input to prevent errors and inconsistencies in the data submitted.
- Design the UI to transmit user-provided data securely to the prediction system for real-time analysis.
- Define the data format and structure expected from the user input, aligning it with the model's requirements.

V. Test Cases

Data Leakage Prevention (data_leak):

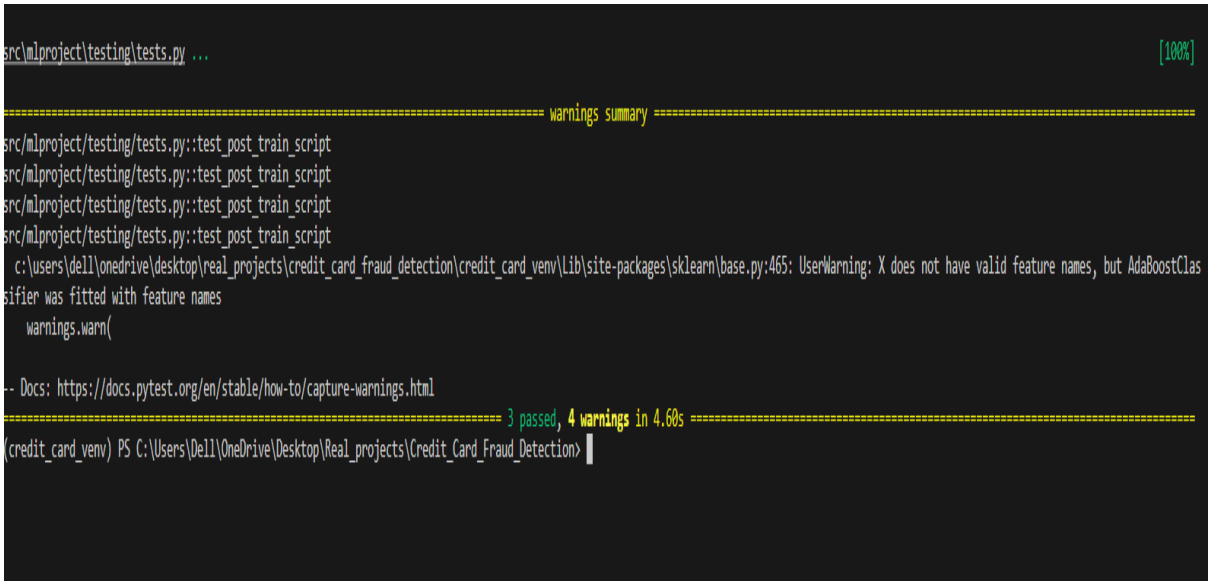
- *Objective:* Ensure there is no data leakage between training and testing datasets.
- *Procedure:* Validate that the combined dataset, created by merging the training and testing datasets, has the expected number of records.
- *Validation:* The test asserts that the combined dataset's row count equals the sum of the training and testing dataset row counts.

Model Output Shape Validation (prediction_output_shape):

- *Objective:* Verify that the shape of model predictions matches the shape of the true labels for testing data.
- *Procedure:* Load the trained model, make predictions on the testing dataset, and compare the shape of the predictions to the shape of the true labels.
- *Validation:* Ensure that the number of predictions aligns with the number of true labels, validating the model's output shape.

Post-Training Predictions (post_train_scrip):

- Objective: Confirm the consistency of predictions for both male and female sample data after model training.
- Procedure: Load the trained model and use it to predict credit card default for both male and female sample input data.
- Validation: Assert that the predictions for both male and female samples are consistent, indicating the model's correctness and uniformity.
- These test cases play a crucial role in the quality assurance and validation process of the system, ensuring that it functions as intended and maintains data integrity and prediction accuracy.


src\mlproject\testing\tests.py ...
[100%]
warnings summary
src/mlproject/testing/tests.py::test_post_train_script
src/mlproject/testing/tests.py::test_post_train_script
src/mlproject/testing/tests.py::test_post_train_script
src/mlproject/testing/tests.py::test_post_train_script
c:\users\dell\onedrive\desktop\real_projects\credit_card_fraud_detection\credit_card_venv\Lib\site-packages\sklearn\base.py:465: UserWarning: X does not have valid feature names, but AdaBoostClassifier was fitted with feature names
warnings.warn(
-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
3 passed, 4 warnings in 4.60s
(credit_card_venv) PS C:\Users\Dell\OneDrive\Desktop\Real_projects\Credit_Card_Fraud_Detection>