LOW LEVEL DESIGN

Credit Card Defaulter Prediction

WRITTEN BY	RAHUL YADAV
DOCUMENT VERSION	0.1
LAST REVISED DATED	24.JUNE.2024

Contents

1. Introduction

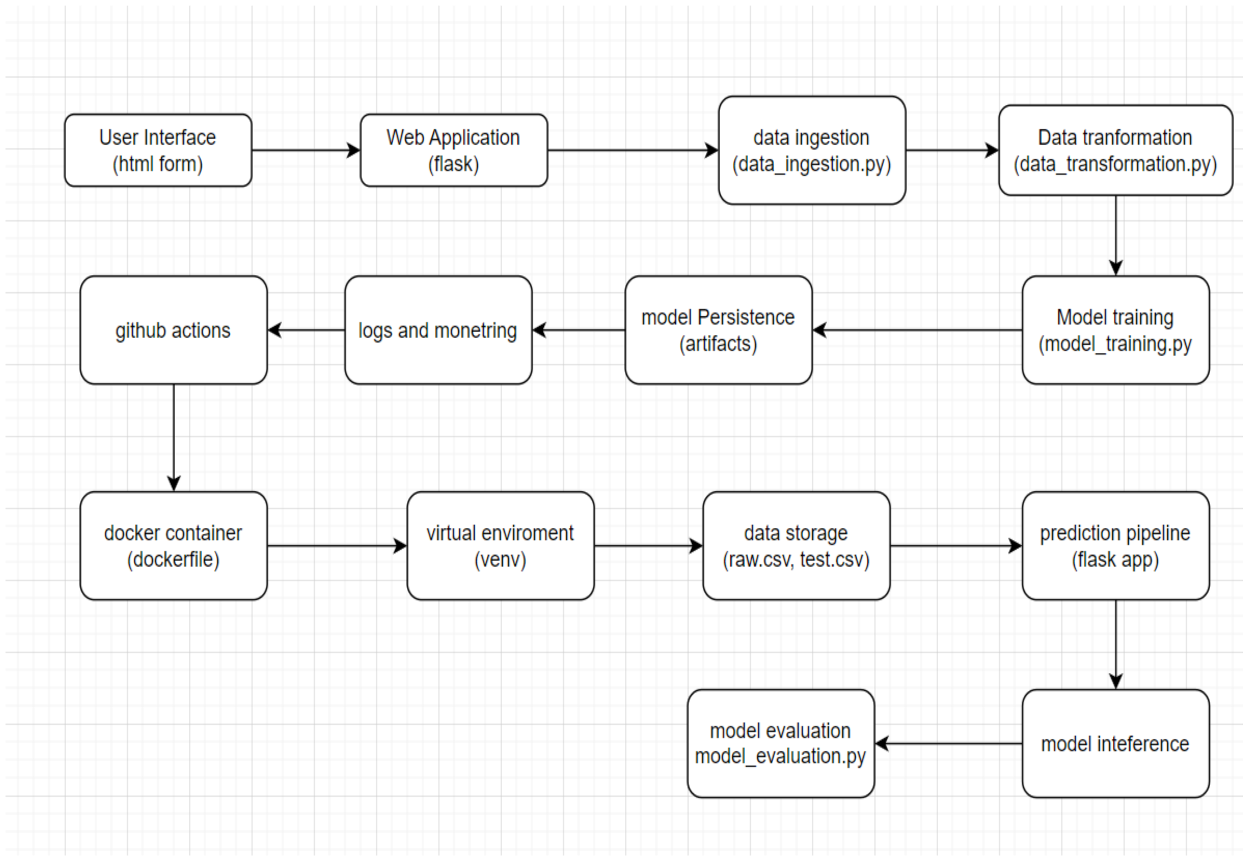
1.1. What is Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Food Recommendation System. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

1.2. Scope

Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work

2. Architecture


User Interface (html form)
Web Application (flask)
data ingestion (data_ingestion.py)
Data tranformation (data_transformation.py)
Model training (model_training.py
model Persistence (artifacts)
logs and monetring
github actions
docker container (dockerfile)
virtual enviroment (venv)
data storage (raw.csv, test.csv)
prediction pipeline (flask app)
model inteference
model evaluation model_evaluation.py

3. Architecture Description

The Credit Card Defaulter Prediction system architecture is designed to predict credit card defaulters based on credit card owner's characteristics and payment history.. It consists of several key components that work together to collect, process, train models, and make predictions. Here's a detailed description of each component:

3.1 User Interface (HTML Forms):

- Provides a user-friendly interface for users to input data and request predictions.
- Sends user inputs to the Web Application for processing.

3.2 Web Application (Flask):

- Receives requests from the User Interface and directs them to the appropriate components.
- Manages the overall flow of the application, including data processing and model inference.

3.3Data Ingestion (data_ingestion.py):

- Collects data from external sources, such as databases or APIs.
- Prepares the data for further processing by cleaning and transforming it into a suitable format.

3.4 Data Transformation (data_transformation.py):

- Processes the raw data to extract relevant features for model training.
- Transforms the data into a format suitable for machine learning model training.

3.5 Model Training (model_trainer.py):

- Utilizes the transformed data to train machine learning models.
- Implements algorithms to build predictive models based on historical credit card data.

3.6 Model Persistence (artifacts):

- Stores the trained machine learning models and associated metadata for future use.
- Allows for easy retrieval and deployment of models for making predictions.

3.7 Logs and Monitoring:

- Captures logs and monitoring data to track the performance and behavior of the system.
- Helps in identifying and resolving issues in the system.

3.8 CI/CD Pipeline (GitHub Actions):

- Automates the testing and deployment processes to ensure the reliability and scalability of the application.
- Integrates with version control systems to manage code changes and updates.

3.9 Docker Container (Dockerfile):

- Packages the application and its dependencies into a container for easy deployment and scalability.
- Ensures consistency in the runtime environment across different deployment environments.

3.10 Virtual Environment (venv):

- Provides an isolated environment for the application to run, ensuring that dependencies are consistent and do not conflict with other applications.

3.11 Data Storage (raw.csv, test.csv):

- Stores the original and processed data used for model training and evaluation.
- Ensures data integrity and availability for future use.

3.12 Prediction Pipeline (Flask app):

- Accepts incoming requests for predictions from the Web Application.
- Utilizes the trained machine learning models to make predictions based on the input data.

3.13 Model Inference:

- Uses the trained machine learning models to make real-time predictions on new data.
- Provides the predicted outcomes, such as whether a credit card user is likely to default or not.

3.14 Model Evaluation (model_evaluation.py):

- Evaluates the performance of the trained machine learning models using test data.
- Calculates metrics such as accuracy, precision, recall, and F1-score to assess the model's effectiveness.

3.15 Deployment

- We will be deploying the model to AWS..