Nitin Udmale
Akshay Nikam

Low Level Design (HLD)

Credit Card Default Prediction

Document Version Control

Date Issued	Version	Description	Author
23 July 2023	1	Initial LLD – V1.0	Nitin Udmale
			Akshay Nikam

Contents

1. Introduction……………………………………………………..…………………………………...4

1.1 Why this Low-Level Design Document ? ………………………………….……...………4

1.2 Scope……………………………………………………………………………………….4

2. Architecture……………………………………………………………………………………….5

3. Architecture Description………………………………………………………………………….6

3.1 Data Description…..……………………………………………………...……….6

3.2 Data Ingestion……………………………………………………………...………...6

3.3 Data Validation……..……………………………………………………………………6

3.4 Data Transformation..……………………………………………………………………6

3.5 Model Trainer…………..…………………………………………………………………..6

3.6 Model Evaluation……………………………………………………………………………..7

3.7 Model Pusher..……………………………………………………………………………….7

3.8 Data From User..…………………………………………………………………………….7

4. Unit Test Cases……………………………………………………………………………...8

1. Introduction

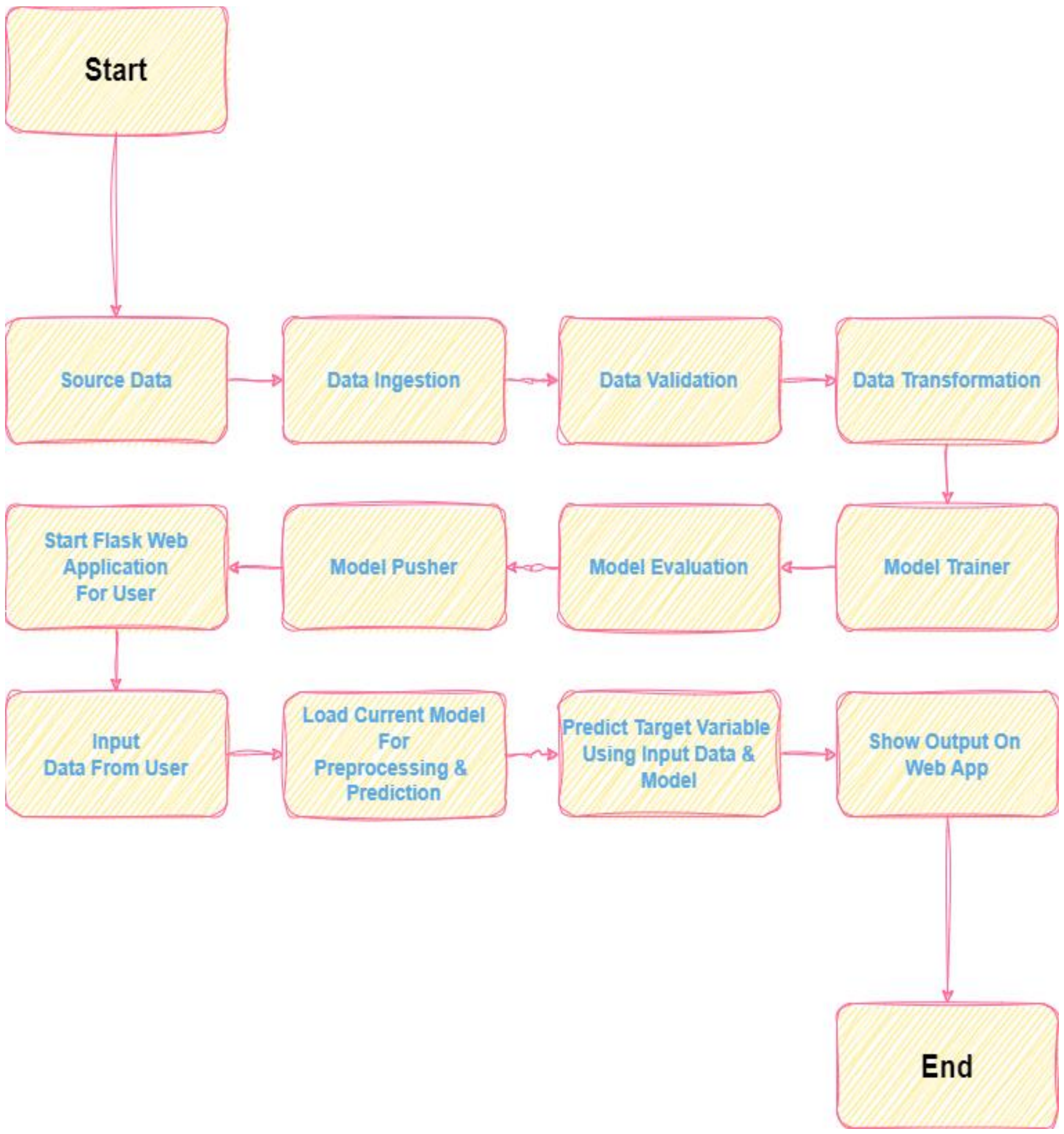
1.1. What is Low-Level design document?

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



3. Architecture Description

3.1 Data Description

This dataset contains information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005. Data is having 30000 records and 24 attributes.

3.2 Data Ingestion

In this process, we will download data from, https://archive.ics.uci.edu/static/public/350/default+of+credit+card+clients.zip , Which is zip file containing excel (XLS) file. We will download the data and split data into training and test set, and export it in artifacts.

3.3 Data Validation

In this process, we will perform schema validation. Further we will check if any data drift is found between train and test set, as well as with old datasets.

3.4 Data Transformation

In data pre-processing or data transformation steps we will, handle missing, duplicate values. We will also standardize numerical values using Standard Scaler from sklearn. We will create object of preprocessing pipeline here.

3.5 Model Trainer

In this step, we will train our models using custom built functions and GridSearchCV. We will combine preprocessing object with best trained (best performing) model object and pass best performing model object for evaluation process.

3.6 Model Evaluation

Trained model from previous step will be compared with model already in production, whichever is the best model, it will forwarded for model pusher component.

3.7 Model Pusher

Here, the model received from model evaluation will be exported and pushed for deployment for real time prediction.

3.8 Data from User

Here we will create web page application, using flask framework. User will give input to this page, data from page will be used for prediction purpose. Prediction will be done by model in production which will handle preprocessing of data and prediction.

4.Unit Test Cases

Test Case Description	Pre-Requisite	Expected Result
Verify whether the Application URL is accessible to the user	1. Application URL should be defined	Application URL should be accessible to the user
Verify whether the Application loads completely for the user when the URL is accessed	1. Application URL is accessible 2. Application is deployed	The Application should load completely for the user when the URL is accessed
Verify whether the User is able to sign up in the application	1. Application is accessible	The User should be able to sign up in the application
Verify whether user is able to successfully login to the application	1. Application is accessible 2. User is signed up to the application	User should be able to successfully login to the application
Verify whether user is able to see input fields on logging in	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	User should be able to see input fields on logging in
Verify whether user is able to edit all input fields	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	User should be able to edit all input fields
Verify whether user gets Predict button to submit the inputs	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	User should get Submit button to submit the inputs

Verify whether user is presented with recommended results on clicking submit	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	User should be presented with recommended results on clicking submit
Verify whether the recommended results are in accordance to the selections user made	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	The recommended results should be in accordance to the selections user made