LLD (Low Level Design)

Adult Census Income Prediction

(With complete CI/CD pipelines)

Revision Number: 2.0
Last date of revision: 21/06/2023

Document Version Control

Date Issued	Version	Description	Author
15/06/2023	1	Initial LLD — V1.0	Suraj M Mali
21/06/2023	2	Final LLD — V1.1	Suraj M Mali

Contents

1. Introduction

1.1 What is Low-Level design document?

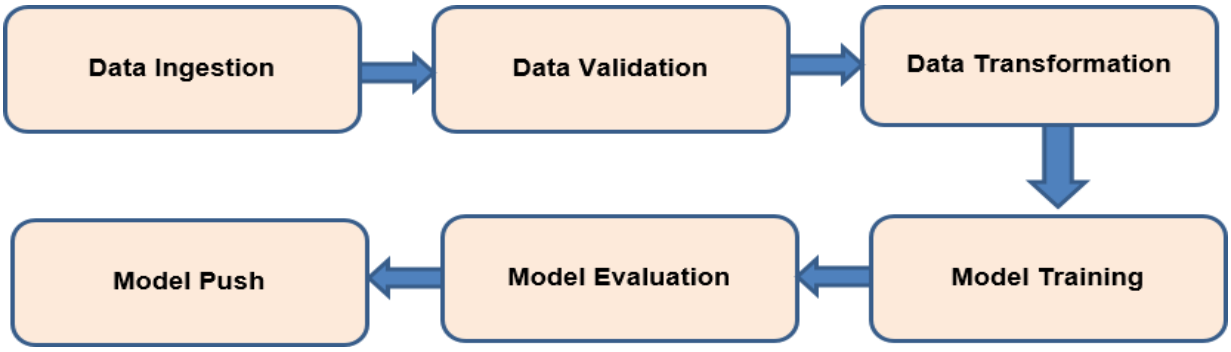
The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Food Recommendation System. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can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

1.2. Scope

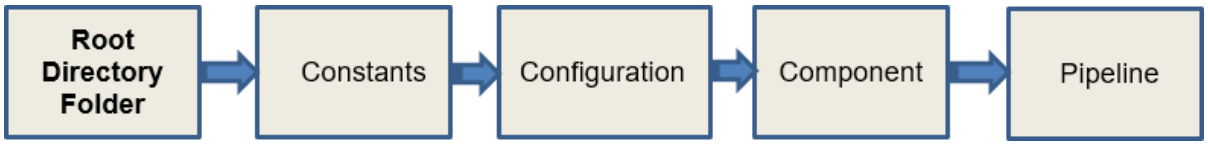
Low-level design (LLD) is a component-level design process that follows a step-bystep refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement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

2. Architecture:

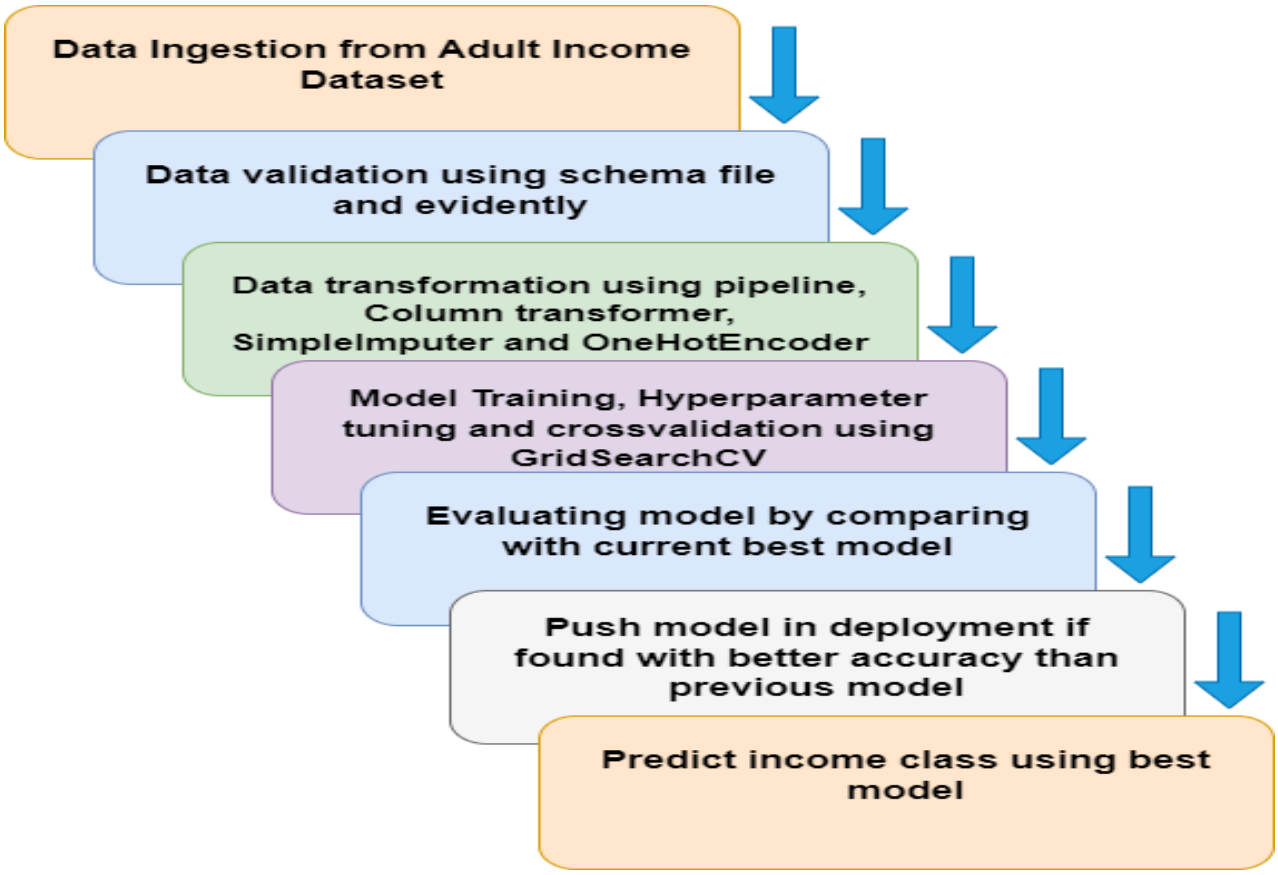
2.1 Components of Machine Learning Pipeline



2.2 Coding flow for building project structure



2.3 Complete pipeline flow



3. Architecture Description:

1. **Data Description**:

   The dataset named Adult Census Income is available in kaggle and UCI repository. This data was extracted from the 1994 census bureau dataset by Ronny Kohavi and Barry Becker (Data Mining and Visualization, Silicon Graphics). The prediction task is to determine whether a person makes over $50K a year or not.

2. **Data Ingestion:**

   Loading data form the apache Cassandra database by establishing connection with database cluster. Ingested data will be in the form of csv file. Splitting file into train and test file using stratified split.

3. **Data Validation:**

   Validating data using schema.yaml file and generating report using evidently package. Checking for data drift.

4. **Data transformation:**

   Removing unnecessary columns from dataframe.Transforming data using various pipeline, columntransformer, simpleimputer and onehotencoder. Saving preprocessing object in pickle file.

5. **Model Training:**

   Training different model using transformed with crossvalidation and gridsearchCV.

6. **Model Evaluation:**

   Evaluating model performance using various performance metrics. Finding the best model by compairing different models.

7. **Model Push:**

   Deploying the model in production only if better model found. Prediction will be done using this model.

8. **Deployment on railway:**

   Deployment using github repository on railway cloud. Automating deployment using github actions. Then, app will automatically get updated when we commit any changes to main branch of github repository.

4. Unit Test Cases:

Test Case Description	Pre-Requisite	Expected Result
Verify whether the Application URL is accessible to the user	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	Application URL should be defined
Verify whether user is able to see input fields.	Application is accessible	User should be able to see input fields
Verify whether user is able to edit all input fields	Application is accessible	User should be able to edit all input fields
Verify whether user gets Submit button to submit the inputs	Application is accessible	User should get Submit button to submit the inputs
Verify whether user is presented with results on clicking submit	Application is accessible	User should be presented with results on clicking submit
Verify whether the results are in accordance to the selections user made	Application is accessible	The results should be in accordance to the selections user made

5. Application Interface