Low Level Design

Adult Census Income Prediction System

Written By	**Rushikesh Shinde**
Document Version	1.0
Last Revised Data	

Document Version Control

Change Record:

Version	Date	Author	Comments
1.0	07-02-2024	Rushikesh Shinde	Introduction & Architecture defined

Reviews:

Version	Date	Reviewer	Comments

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

Contents

1. Introduction

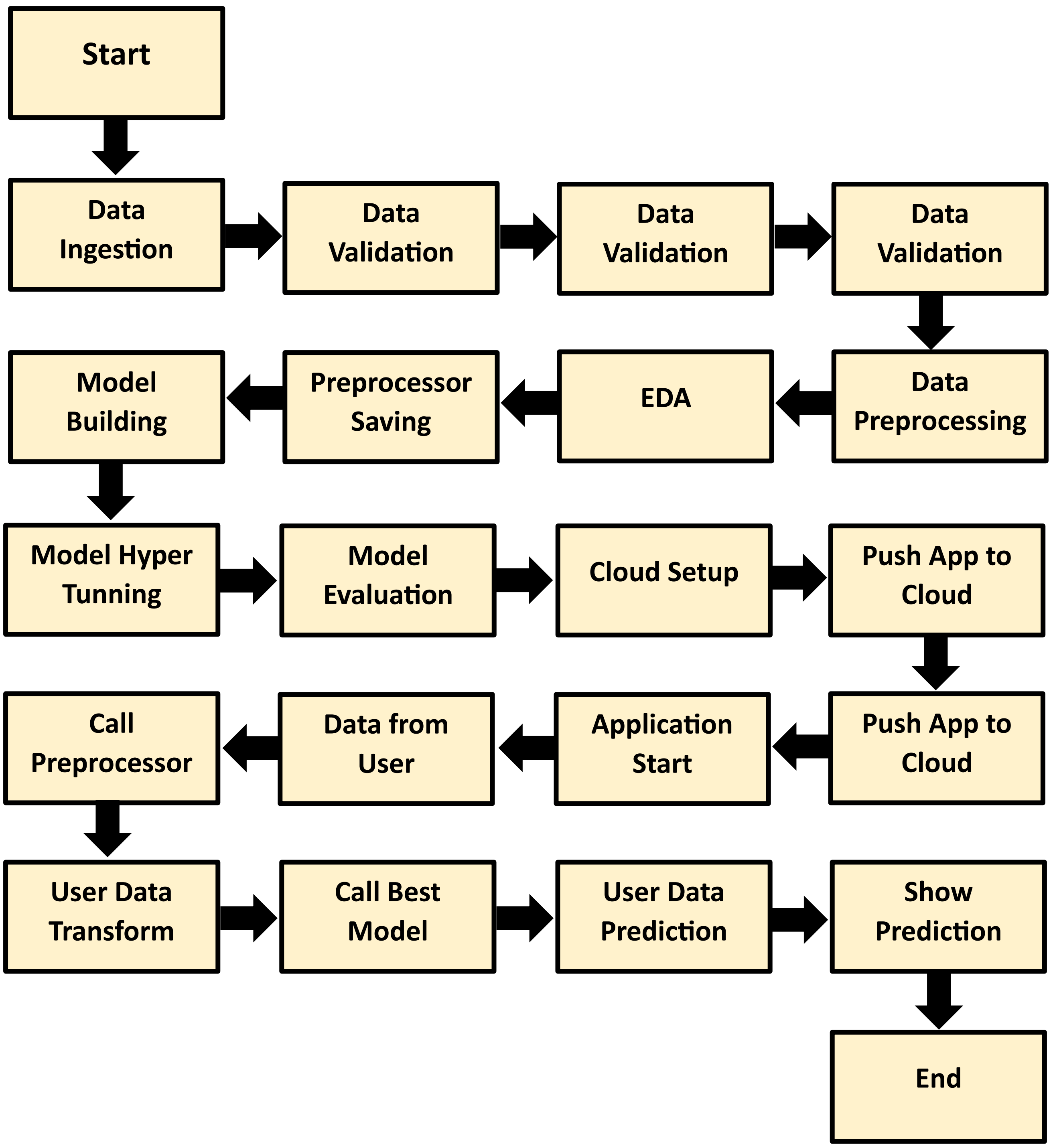
1.1 What is Low-Level design document?

The goal of LLD or a low-level design document (LLD) is to give the internal logical design of the actual program code for Thyroid Disease Detection System. LLD describe 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

We will be using Adult Census Income Prediction Data Set present on Kaggle. This Data set is satisfying our data requirement. Where the data consists of total 32561 records and 15 features including target column.

3.2 Downloading Data from Site to CSV for Training

Here we will be downloading the data from GitHub repo with data ingestion for data preprocessing and training.

3.3 Data Preprocessing

We will be exploring our data set here and perform data preprocessing depending on the data set. We first explore our data set in Jupyter Notebook and decide what pre-processing and Validation we must do such as imputation of null values, dropping some column, etc and then we must write separate modules according to our analysis, so that we can implement that for training as well as prediction data.

3.4 Exploratory Data Analysis

We will be performing inferential and descriptive data analysis to find patterns in data. Like correlation for finding collinearity between feature, univariant analysis and multivariant analysis finding relation between single and multiple features in dataset.

3.5 Train Multiple Models

Here we will train multiple models at the same time with python modular coding and obtain the results for training and test data with respect to each model with custom performance matrix for evaluation.

3.6 Hyperparameter Tuning for Multiple Models

Here we will be doing some hyperparameter tuning with models for increasing the performance and as well as look for any over fitted models, will be using the same python modular code to obtain results for train and test data for evaluation of the hyper tunned models.

3.7 Model Saving

After performing hyperparameter tuning for models, we will save our models so that we can use them for prediction purpose.

3.8 Cloud Setup

Here We will do cloud setup for model deployment. Here we also create our flask app and user interface and integrate our model with flask app and UI.

3.9 Push app to cloud

After doing cloud setup and checking app locally, we will push our app to cloud to start the application.

3.10 Data from client side for prediction purpose

Now our application on cloud is ready for doing prediction. The prediction data which we receive from client side will be exported from form and further will do same data cleansing process as we have done for training data using modules, we will write for training data. Client data will also go along the same process with the preprocessor we build and saved, will be using the same best model for client income prediction.

3.11 Show prediction for client data

Finally, when we get the prediction for client data, then our final task is to show prediction to client with the help of flask API.

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 user can successfully login to the application	1. Application URL is accessible. 2. Application is deployed.	User should be able to successfully login to the application
Verify whether user can see input fields on logging in	1. Application URL is accessible. 2. Application is deployed.	User should be able to see input fields on logging in
Verify whether user can edit all input fields	1. Application URL is accessible. 2. Application is deployed.	User should be able to edit all input fields
Verify whether user gets Submit button to submit the inputs	1. Application URL is accessible. 2. Application is deployed.	User should get Submit button to submit the inputs