Low Level Design

Adult census Income Prediction

Written By	L.Sai prakash
Document Version	0.1
Last Revised Date	12 – october -2021

Document Control

Change Record:

Version	Date	Author	Comments
0.1	19 – Sep - 2021	L.Sai prakash	Introduction & Architecture defined
0.2	20 – Sep - 2021	L.Sai prakash	Architecture & Architecture Description appended and updated

Reviews:

Version	Date	Reviewer	Comments
0.1	21 – Sep - 2021	L.Sai prakash	Document Content , Version Control and Unit Test Cases to be added

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

Contents

1. Introduction.....1
1.1. What is Low-Level design document? 1
1.2. Scope.....1
2. Architecture.....2
3. Architecture Description.....3
3.1. Data Collection.....3
3.2. Data Validation.....3
3.3. Data Cleaning.....3
3.4. EDA.....3
3.5. Feature Engineering.....3
3.6. Model Creation 3
3.7. Hyper-parameter tunning.....3
3.8. Model Dump.....4
3.9. FrontEnd creation.....4
3.10. Flask-app creation.....4
3.11. Data Inserting into Database.....4
3.12. Model Call 4
3.13. Deployment.....4
4. Unit Test Cases 5

1. Introduction

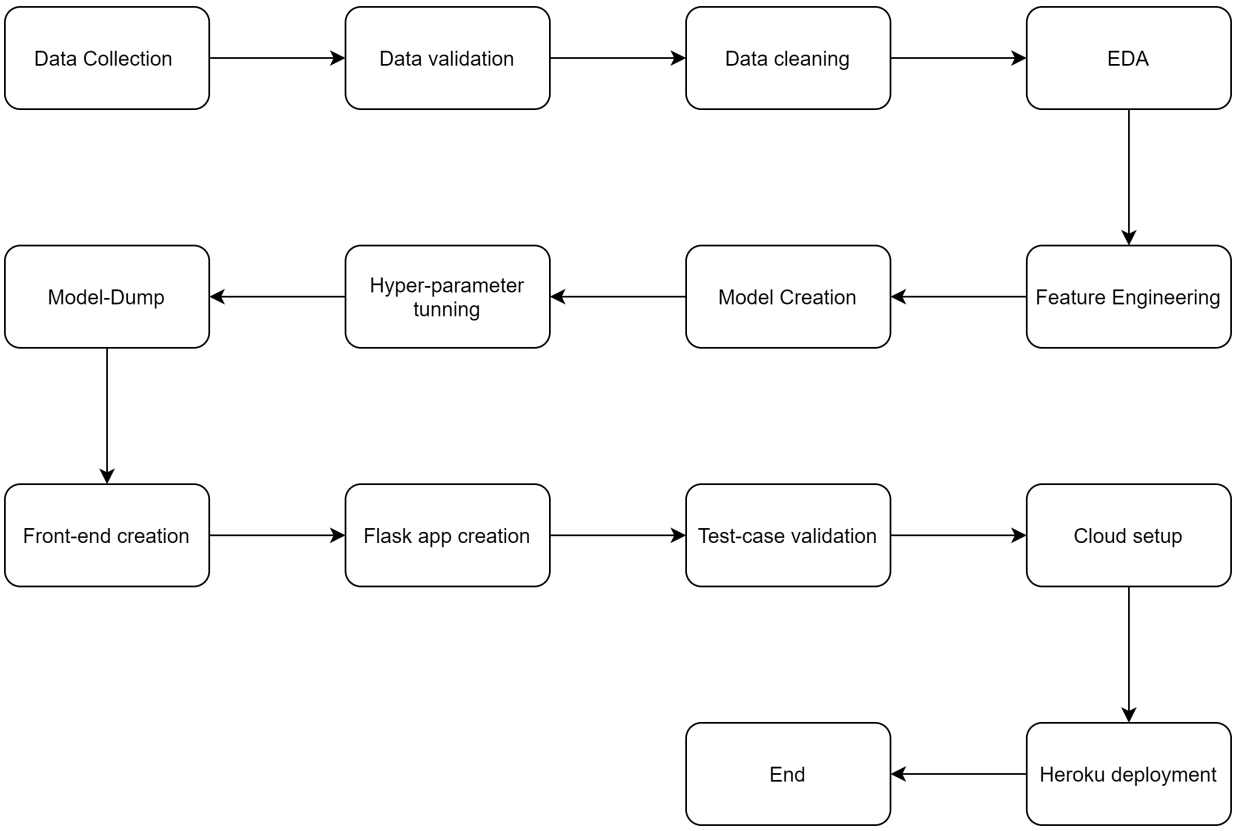
1.1. What is Low-Level design document?

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 Collection

We have Dataset of row columnar which includes various columns like age,occupation,education,etc . The information is given in csv format. These data is collected from kaggle site.

3.2. Data Validation

Data collected from Data source is validated on ceratin criteria and data after getting validated is send to next stage for Data cleaning step .

3.3. Data Cleaning

In the Cleaning process, We have cleaned up all the data because data is present in very bad format which was cannot reconigzed by machine.So data engineering is done very first.

3.4. EDA

In eda we try to perfrom necessary Exploratory data analysis steps like finding missing values, replacing missing values ,removing duplicate values,classifying features into categorical and numerical.

3.5. Feature Engineering

In Feature Engineering we try to encode variable, perform standardization by using standard scalar, split data into dependent and Independent variables for futher model building purpose

3.6. Model Creation

After cleaning the data and completing the feature engineering. we have done splitted data in the train data and test data and implemented classifiaction algorithms like Random Forest classifier and XGboost classifier also calculated their accuracies on test data .

3.7. Hyperparameter Tuning:

In hyperparameter tuning we have implemented various ensemble techniques like random forest regressor, bagging and boosting we also done randomized search cv or grid search cv and from that we also implemented cross validation techniques for that. From that we have choosen best parameters according to hyperparameter tunning and best score from their accuracies so we got 80% accuracy in our XGboost Classifier after hyper parameter tuning.

3.8. Model Dump

After comparing all accuracies and checked all roc, auc curve we have choosen hyperparameterized XG boost classifier as our best model by their results so we have dumped these model in a pickle file format with the help of python pickle module.

3.9. Front-end creation

In Frontend creation we have made a user interactive page where user can enter their input values to our application. In these frontend page we have made a form which is of HTML. These html user input data is transferred in json format to backend. Made these html fully in a decoupled format.

3.10. Flask-app creation

Here we try create Flask app which acts as an api between user and and our backend ,by hitting our flask our application gets starts and gets running.

3.11. Data Inserting into Database

Collecting the data and storing it into the database. The database can be either MySQL or Mongo DB. Here we are using MangoDB database

3.12. Model Call/.pkl file loaded

Based on the User input will be throwing to the backend in the dictionary format so our we are loading our pickle file in the backend and predicting whether income is less than or equal to 50k or it is more than 50k as a output and sending to our html page.

3.13. Deployment

We will be deploying the model to Heroku.

This is a workflow diagram for the Recipe Recommendation..

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 Response time of url from backend model.	1. Application is accessible	Th Latency and accessibilty of url is faster
Verify Response time of url from backend model.	1. Handeled test cases at backends.	User should be able to see successfully valid results.
Verify whether user is able to see input fields on logging in	1. Application is accessible 2. 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 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 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 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 logged in to the application	The recommended results should be in accordance to the selections user made