Low Level Design

Adult Census Income Prediction

Written By	Susan Elizabeth Varghese
Document Version	0.2
Last Revised Date	01 September 2021

Document Control

Version	Date	Author	Comments
0.1	28.08.21	Susan Elizabeth Varghese	Introductions
0.2	28.08.21	Marchetty Pavan Sai	Architecture Description

Approval Status

Version	Review Date	Reviewed by	Approved by	Comments

Contents

1. Introduction

1.1 What is Low Level design document? 1

1.2 Scope 1

2. Architecture

3. Architecture Description

3.1 Data Description 3

3.2 Export Data from Database 3

3.3 Data Preprocessing 3

3.4 Model Building 3

3.5 Creating Flask Web Application 4

3.6.Uploading in the GitHub Repository 4

3.7. Heroku Setup 4

3.8. Pushing the app to Heroku 4

3.9. Application start 5

3.10. Data from the User 5

3.11 Data Validation 5

4. Unit Test Cases

1.Introduction

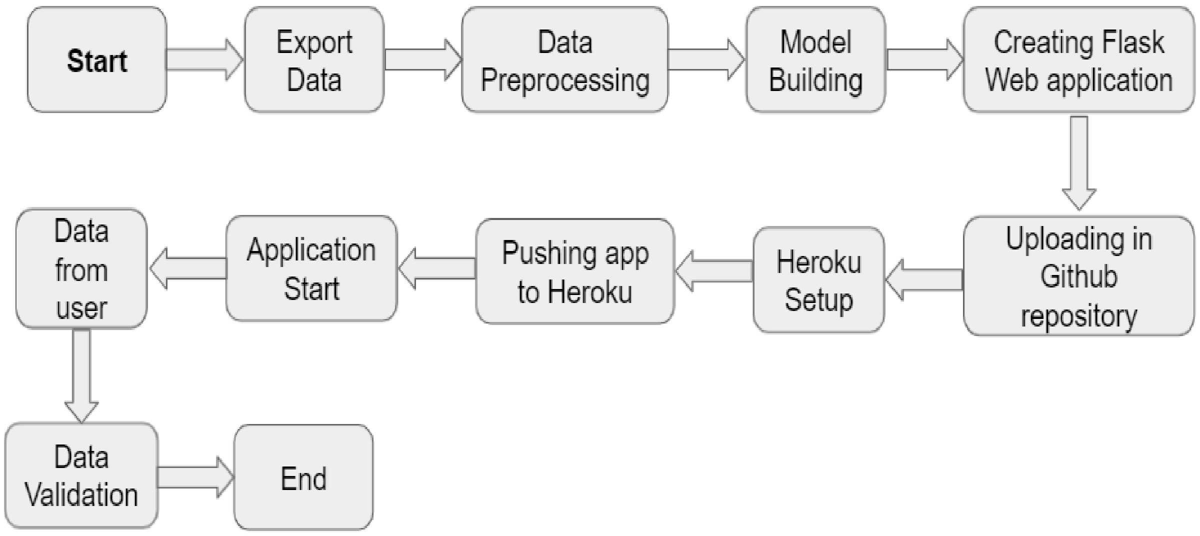
1.1 What is a Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Adult Census Income Prediction. 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

Adult Census Income Prediction dataset is the biggest publicly available dataset. The document is in comma separated values (csv) format. This dataset contains 32586 rows including 15 different columns.

3.2 Export Data from Database

Data Export from Database. The data in a stored database is exported as a CSV file to be used for Data Pre-processing and Model Training.

3.3 Data Preprocessing

Data Pre-processing steps we could use are Null value handling, Imbalanced data set handling, handling columns with standard deviation zero or below a threshold, dropping columns etc.

3.4 Model Building

We built a model using the DecisionTree Classifier. The algorithms will be passed by the help of best parameters derived using RandomizedSearchCV. We used a confusion matrix to know the true positive and the true negative data points.The model is converted into a pickle file at the end.

3.5 Creating Flask Web Application

The pickle file which is converted during the model building is loaded in the flask framework. The flask framework is the back end of the web application. It mainly depends on the conditional statement (IF -ELSE Statement). For the front end development, HTML & CSS are used.

3.6. Uploading in the GitHub Repository

We need to upload the files such as templates, Procfile, requirements.txt, pickle file, app.py, ipynb file related to the project. In templates, it consists of the index and result of the project.

3.7. Heroku Setup

Heroku is a container-based cloud Platform as a Service (PaaS). Developers use Heroku to deploy, manage, and scale modern apps. Our platform is elegant, flexible, and easy to use, offering developers the simplest path to getting their apps to market. In Heroku, we need to connect it to the github repository.

3.8. Pushing the app to Heroku

We need to create a new app assigning a name to the web application and connect to the github repository link. We need to select the manual deploy option. The necessary link will be generated for the web application.

3.9. Application start

We launch the application using link which is provided in the heroku deployment. https://salarycountry.herokuapp.com

3.10. Data from the User

User needs to provide the relevant information required for the prediction of adult census income without leaving any blank fields.

3.11 Data Validation

Data is checked for validation and the output is predicted depending on the input provided by the user.

4. Unit Test Cases

Test Case Description	Prerequisite	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 see input fields on launching the application.	1. Application is accessible	The user should be able to see input fields on launching the application.
Verify whether user is able to edit all input fields	1. Application is accessible	User should be able to edit all input fields
Verify whether user is presented with results on clicking submit	1. Application is accessible	The results should be in accordance to the selections user made