Adult Census Income Prediction

LOW LEVEL DOCUMENT DESIGN

05.10.2021

Project By :

Rauhan Ahmed Siddiqui

Document Version Control

DATE ISSUED	VERSION	DESCRIPTION	AUTHOR
October 5, 2021	1	Initial LLD V1	Rauhan Ahmed Siddiqui

1. Introduction

1.1 Why this Low Level Document ?

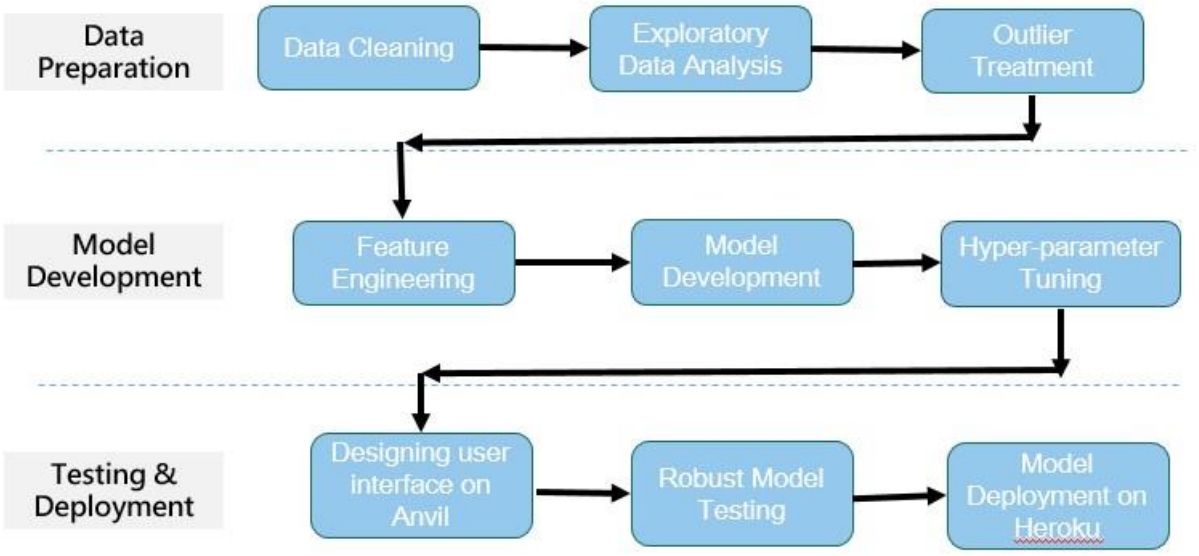
The goal of LLD or a low-level design document is to give the internal logical design of the actual program code for Adult Census Income Prediction. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly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

1.2 Scope

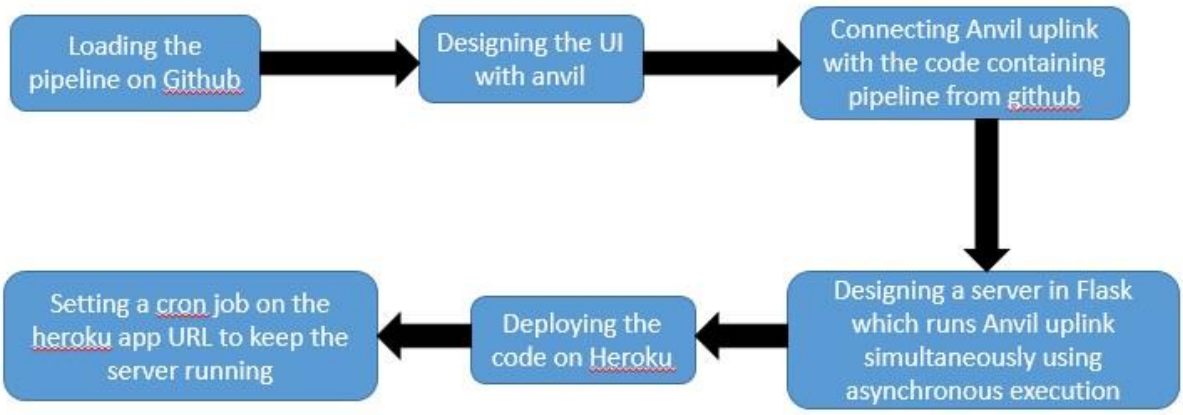
Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement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

2. Architecture

Proposed Methodology



Deployment Process



3. Architecture Description

3.1 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.

3.2 Data Preparation

This step includes all the necessary steps that take place in the life cycle of a data science project namely, Data cleaning, Exploratory Data Analysis (EDA), and outlier treatment. In this step, our data gets prepared to be fed to our ML model.

3.3 Model Development

This step contains all other necessary steps such as Feature Engineering, Feature Selection, Model Selection and Hyperparameter tuning to make the best possible model that can be made for accurate and correct prediction.

3.4 Deployment Process

In this step, we first develop the UI using Anvil and connect with our code in which our model is running with the help of an uplink and create a server using Flask which runs the uplink code (server code) using parallel execution or asynchronous execution and we will then upload the hole code in Heroku cloud using git and github. We will then set a cron job on that server to keep the server and server code running forever.

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

-----x---X---x----