Adult Census Income Prediction

Low-Level Document

Written By	Gurjeet Singh
Document Version	0.1
Description	Initial LLD V0.1
Date Issued	14 November, 2022

Contents

1. Introduction

1.1 What is a Low-Level Design Document?

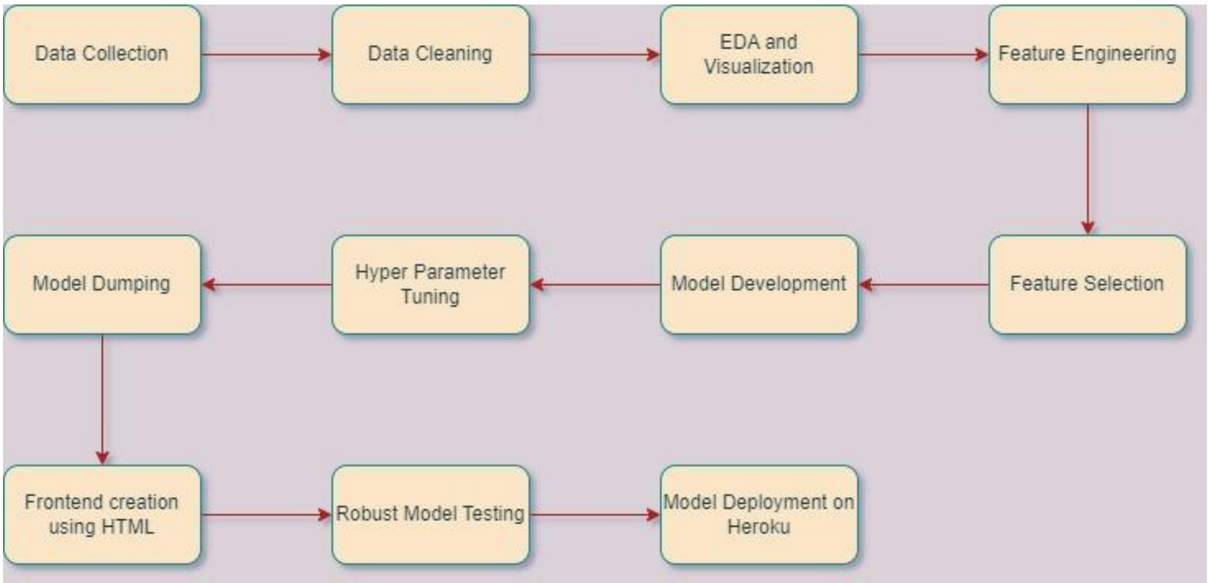
The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for the Adult Income Prediction estimation system. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that 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

Proposed methodology



3. Architecture Description

3.1 Data Collection

The dataset named Adult Census Income is available in Kaggle and UCIrepository. This data was extracted from the 1994 census bureau dataset by Ronny Kohavi and Barry Becker (Data Mining and Visualization, Silicon Graphics). The dataset contains 15 columns and 32k+ rows. The dataset includes age, workclass, fnlwgt, education, education-num, marital-status, occupation, relationship, race, sex, capital-gain, capital-loss, hours-per-week, country, salary. This is given in the comma separated value format (.csv). The prediction task is to determine whether a person makes over $50K a year or not.

3.2 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an approach to analyse the data using visual techniques. It is used to discover trends, patterns, or to check assumptions with the help of statistical summary and graphical representations.
In this step, we find relation between our dependent and independent features and also treat any outliers for our data to get prepared to be

fed to our ML model.

3.3 Data pre-processing

In this step object data types are converted to int or float according to need and also correlation between different features are checked.

3.4 Feature Engineering

In this step categorical features are encoded using one hot encoder or label encoder and new features are created to be passed onto our model.

3.5 Feature Selection

Feature Selection is the method of reducing the input variable to our model by using only relevant data and getting rid of noise in data. It is the process of automatically choosing relevant features for our machine learning model based on the type of problem you we trying to solve.

3.6 Model Development

After cleaning the data and completing the feature engineering and selection, we split data into training and testing sets implement various classification algorithms like Logistic Regression, Random Forest, Gradient Boosting, Decision Tree, SVM, XGBoost, Catboost, and KNN to make the best possible model that can be made for accurate and correct prediction.

3.7 Hyperparameter Tuning

Hyperparameter tuning consists of finding a set of optimal hyperparameter values for a learning algorithm while applying this optimized algorithm to any data set. That combination of hyperparameters maximizes the model's performance, minimizing a predefined loss function to produce better results with fewer errors.

3.8 Model Dumping

After the training is completed a best model is selected among all the models by choosing an accuracy parameter according to our need and dumped the model in a pickle file format.

3.9 Frontend using HTML

In this step, a frontend is created using HTML to receive input from the user and connected to a flask backend server to do the predictions.

3.10 Robust Model Testing

The model is tested by giving an input data to check for its accuracy and response time.

3.11 Model Deployment on Heroku

The model is deployed on Heroku using a corn job on the 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 beable 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 result on clicking submit	Application is accessible	User should be presented with result on clicking submit
Verify whether the result is in accordance to the selections user made	Application is accessible	The result should be in accordance to the selections user made