Low-Level Design (LLD)

Adult Census Income Prediction

Written By	SUDHANSU GOUDA
Version	1.0
Date	25/02/2023

Document Change Control Record

Version	Date	Author	Comments

Reviews:

Version	Date	Reviewer	Comments

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

Index

1. Introduction

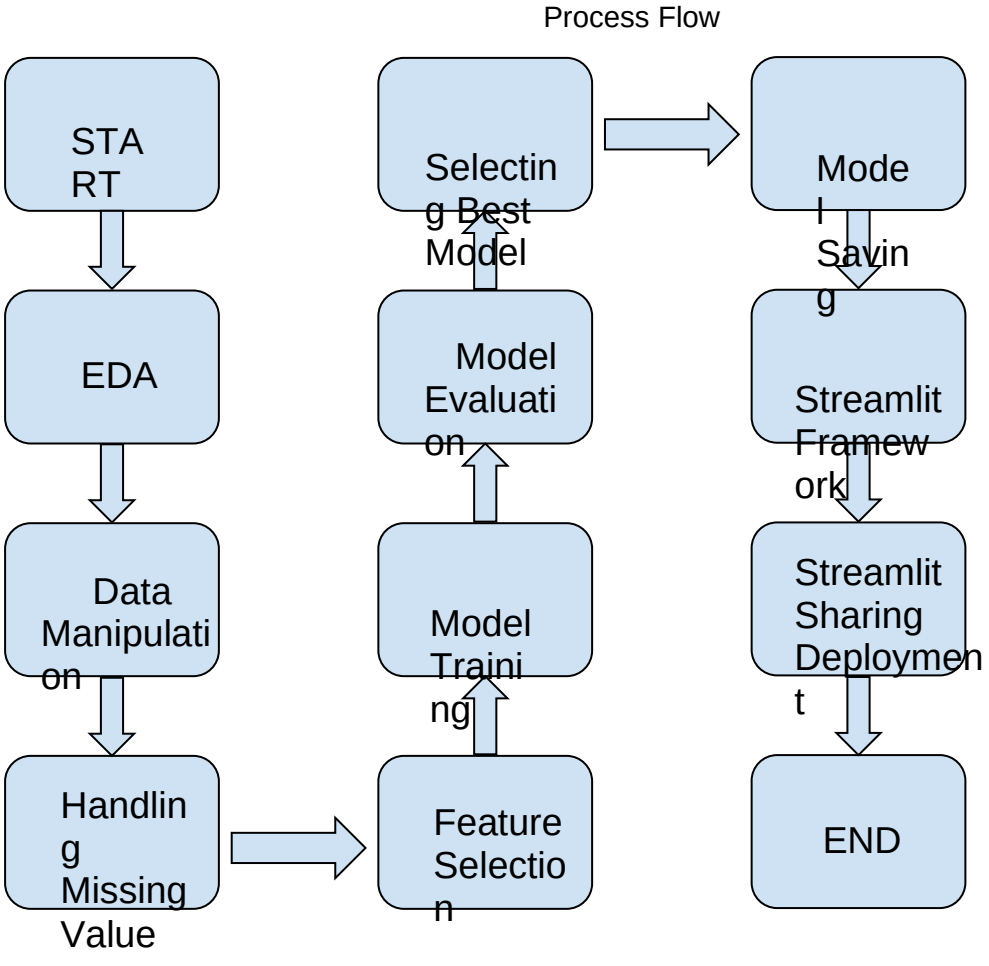
1.1 What is Low-Level Design Document.

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

Architecture



2. Architecture Description

2.1 Data Description

Given is the variable name, variable type, the measurement unit, and a brief description. The concrete compressive strength is the classification problem. The order of this listing corresponds to the order of numerals along the rows of the database.

Name	Data Type	Measurement
age	Integer	Age of the candidate
workclass	String	Work sector of the candidate
fnlwgt	Integer	Final weight
eduction	String	Highest level of education
education-num	Integer	Count of the value of education occurred
marital-status	String	Current marital status
occupation	String	Current job
relationship	String	Relation in married life or else not in family
race	String	Race of the person
sex	String	Gender
capital-gain	Integer	Profit earned on the sale of an asset like stocks, bonds or real estate
capital-loss	Integer	Loss incurred on the sale of an asset like stocks, bonds or real estate

hours-per-week	Integer	Sales of the product in the particular store
country	String	The country where the person is working currently
salary	String	The salary category either more than $50,000 or less than $50,000

2.2 Data Gathering

Data source: https://www.kaggle.com/datasets/overload10/adult-census-dataset

2.3 Raw Data Validation

After data is loaded, various types of validation are required. Validations like checking for zero standard deviation for all the columns, checking for complete missing values in any columns, etc. These are required because The attributes which contain these are of no use. It will not be much of use in determining of the salary category.

Like if any attribute is having zero standard deviation, it means that's all the values are the same, its mean is zero. Missing data in the training data stands as a problem in front of us.

2.4 Data Transformation

Before sending the data into the database, data transformation is required so that data are converted into such a form with which it can easily be inserted into the database. Here, the 'capital-gain' and 'capital-loss' attributes contain a vast amount of missing values. So, they are removed.

2.5 New feature generations

We haven't derived a new category.

2.6 Data Pre-processing

In data pre-processing all the processes required before sending the data for model building are performed. Like, here 'capital-gain' and 'capital-loss' the attributes are having some values equal to 0 no doubt both of these attributes are viable for prediction due to maximum entry being zero in the model they can't contribute much. So they have been removed from the dataset. In 'workclass' there were some fields '?' which ment null so the rows were removed. The column of 'hours-per-week' having numerical values was further categorised into brackets of 10 to 10 values. The column of 'marital-status' having string values with 7 different values were categorised into two categories.

2.7 Feature Engineering

After preprocessing it was found that some of the attributes are not important to the item sales for the particular outlet. So those attributes are removed. There are some columns that needs to be dropped as they don't seem to help in our analysis.

Model Selection:

For model selection we had used evaluation techniques based on that we will select perfect model to train.

2.8 Model Building

After doing all kinds of preprocessing operations mention above and performing model training and testing the accuracy we came to the conclusion that Random Forest model have the highest accuracy with 81.68% accuracy.

2.9 Model Saving

Model is saved using the pickle library in '. pkl' format.

2.10 Web app setup

After saving the model in .pkl file format we then create an app.py streamlit web app framework (Written in python) and then use requests to extract all the form selection selected by the user and then we predict the salary prediction by using the selected records by the user.

2.11 GitHub

The whole project directory will be pushed into the GitHub repository.

GitHub Project link: https://github.com/cursD15/Adult_Census_Income_Prediction

2.12 Deployment

The cloud environment was set up and the project was deployed from GitHub into the Streamlit Sharing cloud platform.

WebApp link - https://cursd15-ineuron-project-app-te9xub.streamlit.app/

3. Unit Test Cases.

Case	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 a user is able to see input fields while opening the application	1.Application is accessible 2.The user is able to see the input fields	Users should be able to see input fields on logging in
Verify whether a user is able to enter the input values.	1.Application is accessible 2. The user is able to see the input fields	The user should be able to fill the input field
Verify whether a user gets predict button to submit the inputs	1.Application is accessible 2.The user is able to see the input fields	Users should get Submit button to submit the inputs

Verify whether a user is presented with recommended results on clicking submit	1. Application is accessible 2. The user is able to see the input fields. 3. The user is able to see the submit button	Users should be presented with recommended results on clicking submit
Verify whether a result is in accordance with the input that the user has entered	1. Application is accessible 2. The user is able to see the input fields. 3. The user is able to see the submit button	The result should be in accordance with the input that the user has entered