LOW LEVEL DESIGN (LLD)

Low Level Document

Adult Census Income Prediction

Written By Pravin Sharma

Document Version 0.3

Revised Date 02 – 10 -2022

Document Control

Change Record:

Version	Date	Author	Description
1.0	02/10/2022	Pravin Sharma	Initial release
1.1	15/12/2022	Pravin Sharma	Design and Development
1.2	4/01/2023	Pravin Sharma	Implementation
2.0	10/02/2023	Pravin Sharma	Deployment

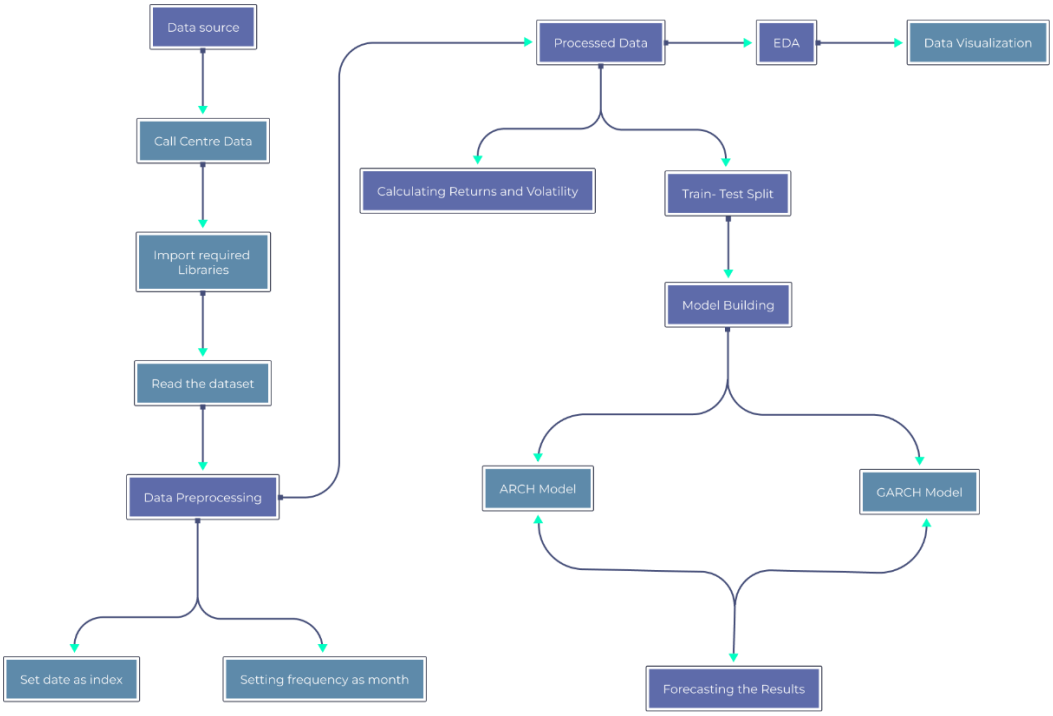
1. INTRODUCTION

Introduction 1.1. What is Low-Level Design Document? The low-level design document is a detailed technical document that describes the system's architecture, components, and their interactions. It outlines the system's functionalities and provides a comprehensive understanding of how the system will be implemented.

1.2. Scope The scope of this document is to provide a detailed low-level design for the Adult Census Income Prediction project. It will describe the architecture, components, and their interactions to enable developers to implement the system efficiently.

2. Architecture

Architecture The Adult Census Income Prediction project's architecture comprises several components, including data description, web scrapping, data transformation, data insertion into the database, data pre-processing, data clustering, model building, data from the user, data validation, user data insertion into the database, model call for specific clusters, recipe recommendation, and saving output in the database, and deployment



3. Architecture Description

3.1. Data Description :

The Adult Census Income Prediction project uses the Adult Census Income dataset obtained from the UCI Machine Learning Repository. The dataset contains demographic and employment information about individuals and their income levels.

3.2. Web Scrapping

The data is obtained by web scraping the UCI Machine Learning Repository website using Python libraries such as Beautiful Soup and requests. The data is then stored in a suitable format for further processing.

3.3. Data Transformation

The data is transformed by converting categorical features to numerical features, encoding them using one-hot encoding, and scaling numerical features using the standard scaler.

3.4. Data Insertion into Database

The transformed data is inserted into a database for orage and retrieval. The database used in this project is MySQL.

3.5. Export Data from Database

The data is exported from the database for further processing, such as data pre-processing, data clustering, and model building.

3.6. Data Pre-processing

The data is pre-processed by imputing missing values, removing outliers, and performing feature selection to reduce the feature space's dimensionality.

3.7. Data Clustering

The pre-processed data is clustered using the K-means clustering algorithm. The number of clusters is determined using the elbow method.

3.10. Model Building

Classification algorithms such as Logistic Regression, Random Forest, and Support Vector Machines (SVM) are used to build a predictive model that can predict an individual's income level based on demographic and employment data.

3.11. Data from User

The user can input demographic and employment data into the system to get a prediction of their income level.

3.12. Data Validation

The input data is validated to ensure that it meets the required format and contains all the necessary fields.

3.13. User Data Inserting into Database

The validated user data is inserted into the database for storage and retrieval.

3.14. Data Clustering

The user data is clustered using the K-means clustering algorithm to identify the cluster to which the user belongs.

3.15. Model Call for Specific Cluster

The model is called to make predictions for the specific cluster to which the user belongs.

3.16. Recipe Recommendation & Saving Output in Database

Based on the user's predicted income level, a recipe recommendation is provided, and the output is saved in the database for future reference.

3.17. Deployment

The trained model is deployed as a web service or an API that can be accessed by other applications.

4. UNIT TEST CASES

Unit Test Cases Unit test cases will be developed to ensure that each component and functionality of the system works as expected. These test cases will be designed to validate input data, test data pre-processing, data clustering, and model building, among other things. The test cases will be executed to ensure that the system is

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Pass/Fail
TC001	Verify if the web scrapping function works properly	1. Input URL for scrapping 2. Call web scrapping function 3. Verify the data obtained from web scrapping	Data is obtained without any errors and the data matches the expected data		
TC002	Verify if the data transformation function works properly	1. Input raw data 2. Call data transformation function 3. Verify the transformed data	Data is transformed without any errors and the transformed data matches the expected data		
TC003	Verify if the data insertion function works properly	1. Input transformed data 2. Call data insertion function 3. Verify the data is inserted into the database	Data is inserted into the database without any errors and the data matches the expected data		
TC004	Verify if the data pre-processing function works properly	1. Input raw data 2. Call data pre-processing function 3. Verify the pre-processed data	Data is pre-processed without any errors and the pre-processed data matches the expected data		
TC005	Verify if the data clustering function works properly	1. Input pre-processed data 2. Call data clustering function 3. Verify the data is clustered properly	Data is clustered properly without any errors and the clustered data matches the expected data		
TC006	Verify if the model building function works properly	1. Input clustered data 2. Call model building function 3. Verify the model is built properly	Model is built properly without any errors and the built model matches the expected model		

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Pass/Fail
TC007	Verify if the user data insertion function works properly	1. Input user data 2. Call user data insertion function 3. Verify the data is inserted into the database	User data is inserted into the database without any errors and the data matches the expected data		
TC008	Verify if the model call for specific cluster function works properly	1. Input user data 2. Call model call function 3. Verify the model is called properly	Model is called properly without any errors and the output matches the expected output		
TC009	Verify if the recipe recommendation function works properly	1. Input model output 2. Call recipe recommendation function 3. Verify the recipe recommendation is correct	Recipe recommendation is correct without any errors and the recommended recipe matches the expected recipe		
TC010	Verify if the output is saved in the database properly and can be retrieved successfully using export function	1. Input recommended recipe 2. Call output saving function 3. Call export function to retrieve output	Output is saved in the database and can be retrieved successfully without any errors and the output matches the expected output		