Architecture

**Contents**Document Version Control  
1 Introduction  
1.1 Why this Low-Level Design Document?  
1.2 Scope  
2 Architecture  
3. Description  
3.1 Data Collection  
3.2. Data Insertion into Database  
3.3. Export Data from the Database  
3.4. Data Cleaning  
3.5. Feature engineering  
3.6. Sampling  
3.8 Model Building  
3.11. Hyperparameter tuning  
3.12. Model Dump with all file  
3.13. Creation of front end  
3.14. Flask Web App  
3.15. Test case check  
3.16. Dockerize  
3.17. Deployment  
4. Unit Test Cases

**Abstract:**

The prominent inequality of wealth and income is a huge concern, especially in the United States. The likelihood of diminishing poverty is one valid reason to reduce the world's surging level of economic inequality. The principle of universal moral equality ensures sustainable development and improves the economic stability of a nation. Governments in different countries have been trying their best to address this problem and provide an optimal solution. This study aims to show the usage of machine learning and data mining techniques in providing a solution to the income equality problem. The UCI Adult Dataset has been used for this purpose. Classification has been done to predict whether a person's yearly income in the US falls in the income category of either greater than 50K Dollars or less equal to 50K Dollars category based on a certain set of attributes

**1 Introduction:**

**1.1 Why this Architecture?**

The goal of architecture is to design the whole process of the model or project, In architecture, we will define the approach to complete our project. Making architecture decisions based on quality attributes makes it easier to fulfill requirements. The architecture allows you to predict a software system’s qualities and avoid expenses.

**1.2 Scope:**

Architecture 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:**

1. Data Collection  
2. Data Cleaning  
3. Feature engineering  
4. Sampling  
5. Model Building  
6. Hyperparameter tuning  
7. Model dump with Pkl file  
8. Driver file  
9. Database file  
10. Take User Input  
11. Insert into Database  
12. Docker  
13. Deploy

**3. Description:**

**3.1 Data Collection:**I have a dataset of 32562 rows which includes all the parameters required to predict the income. The data is available in a CSV file format and it’s collected from the link as per provided in the project description.

**3.2 Data Insertion into the Database:**a. Database Creation and connection - Create a database with Cassandra’s cloud version and connect to the database.  
b. Insert the available data in the Cassandra cloud.

**3.3 Export Data from Database:**Export or read the data from Cassandra’s cloud for further processing.

**3.4 Data Cleaning:**In order to fit the data to the model I need to identify incomplete, incorrect, inaccurate, or irrelevant parts of the data and then replace, modify, or delete the dirty or coarse data.

**3.5 Feature engineering:**In this section, I try to take care of validating or replacing values with other values in the dataset. Additionally, I also take care of converting the categorical columns to numerical columns by one hot encoding and apply label encoding on label data

**3.6 Sampling:**Here dataset is unbalanced so I try random oversampling to balance the data.

**3.8 Model Building:**After the completion of the above process, I split the dataset into test and train. I use various classification algorithms like Decision trees, Random Forest, K-NN, Bagging, XgBoost, and ExtraTreeclassifier and validate the accuracy. Finally, I select the best algorithm and check the accuracy of the training and test data. I used the metrics which validates the variance from model prediction to ground truth.

**3.9 Hyperparameter tuning:**Here I have used a Randomized Search CV for the selection of the best parameters to reduce overfitting criteria. Along with this I used k fold cross-validation technique for the training of the model and finally got 89% accuracy.

**3.10 Model Dump with pkl file:**I have saved the model using pickle.

**3.11 Creation of front end:**As per the required parameters for the user to predict the income I have created the front end in order to link with the Flask App. Here the styling and formatting of the HTML page are taken care of with the CSS file.

**3.14 Flask Web App:**I have created a Flask Web App where I read the contents from the pickle file and made sure it linked to the HTML file to predict the income

**3.15 Test case check:**Here I check for the cases where if a customer inputs wrong data the model should not give wrong results as predicted income. I have taken care of all the conditions that might be possible.

**3.16 Dockerize:**I have used dockerize technique here which helps the application to run within a Docker container. It is useful to take care of the environmental variables and helps the app to run on all the platforms.

**3.17 Deployment:**I have deployed the app on the Heroku platform but I have prepared a deck to help the user in order to deploy it on other platforms as well

**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 | The 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 sign Up for the application | Application is accessible | The User should be able to sign up for the application |
| Verify whether the user is able to successfully use the application | Made sure to check for the test cases from backend. | The user should be able to see successfully valid results |
| Verify whether the user is able to see input fields on logging | 1. Application is accessible 2. User is able to log into the application | The user should be able to edit input fields on logging |

