# Low-Level Design ( LLD )
# Salary Prediction

Revision Number – 1.2 Last Date of Revision: 26 – 11 -2022

Nitesh Sharma

Document Version Control

| Date | Version | Description | Author |
|---|---|---|---|
| 18– 11 - 2022 | 1.0 | Abstract<br>Introduction<br>Architecture | Nitesh |
| 18– 11 - 2022 | 1.1 | Architectural Design | Nitesh |
| 18 – 11 - 2022 | 1.2 | Deployment<br>Unit Test Cases | Nitesh |

## Contents

Abstract

The prominent inequality of wealth and income isa 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 improve 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.

Classification has been done to predict whether a person's yearly income in 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. The Gradient Boosting Classifier Model was deployed which clocked the highest accuracy of 80 %, eventually breaking the benchmark accuracy of existing works.

# 1. Introduction

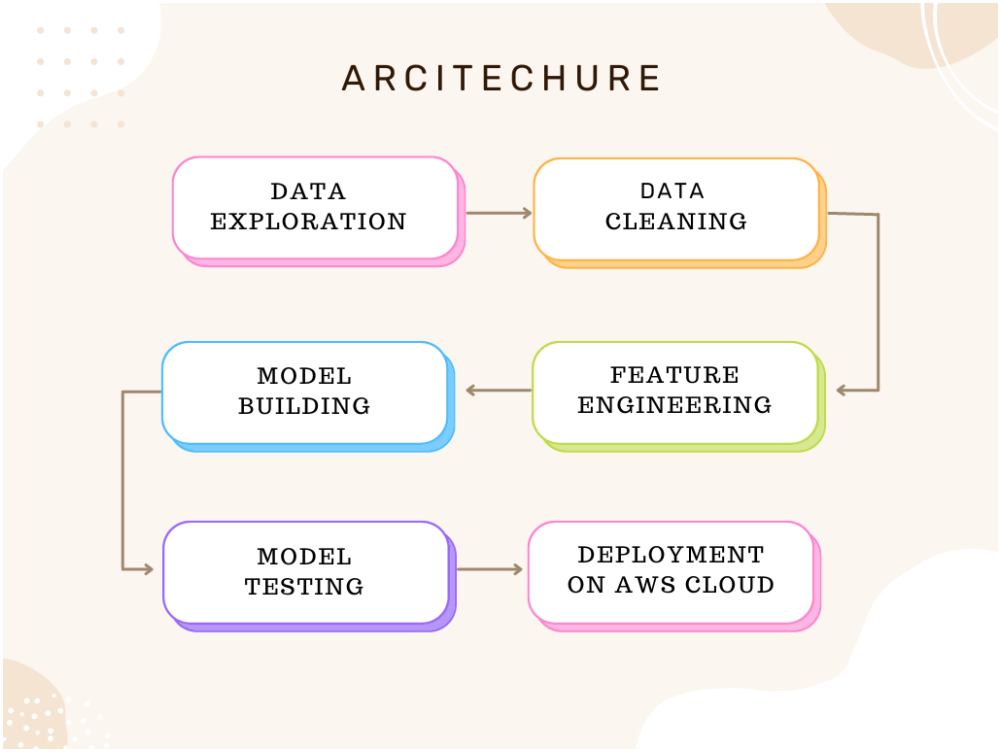
## 1.1 Why is the LLD Document?

The main goal of the LLD document is to give the internal logic design of actual code implementation and supply the outline of the machine learning model and its implementation. Additionally, it provides the description of how our project will be designed end-to-end.

## 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




## 3. Architecture Design

This project is designed to make an interface for the User to predict its Salary .

### 3.1 Data Collection

The data for these project is collected from the Kaggle Dataset, using kaggle API.

### 3.2 Data Description

Adult Census dataset is 30K+ dataset publicly available on kaggle. dataset contain 15 columns named age, workclass, fnlwgt, education, education-num, marital-status, occupation, relationship, race, sex, capital-gain, capital-loss, hours-per-week, country, salary

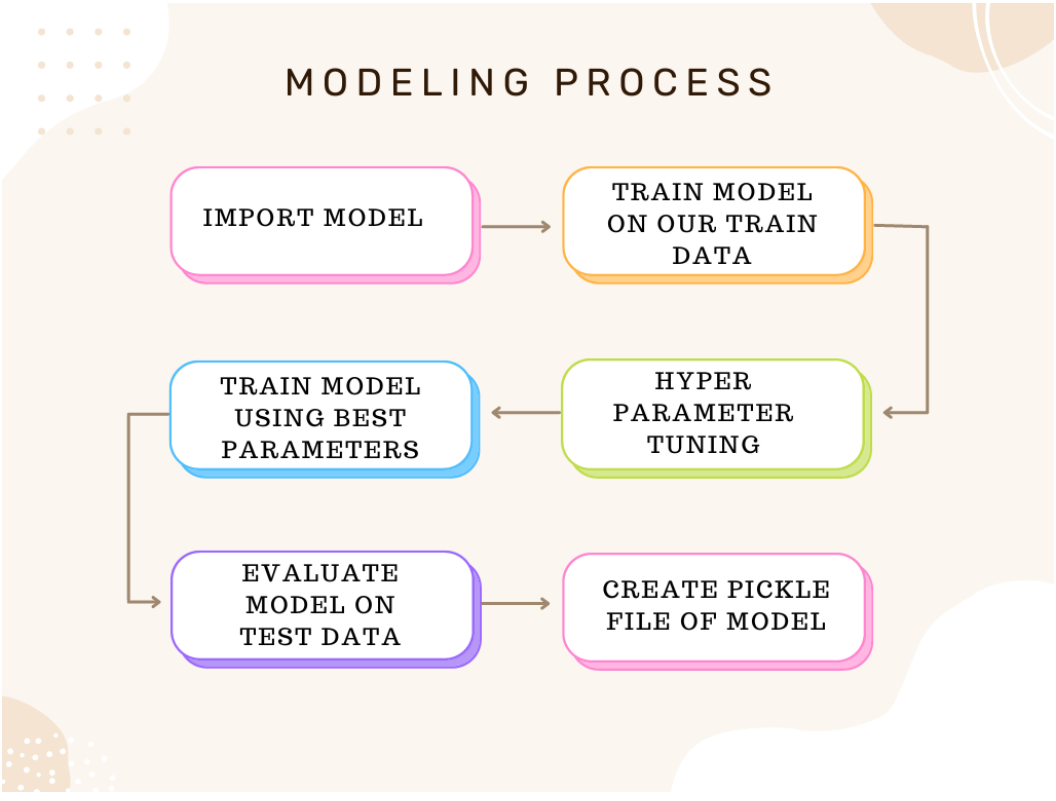
## 3.3 Data Preprocessing

- Checked for info of the Dataset, to verify the correct datatype of the Columns.
- Checked for Null values, because the null values can affect the accuracy of the model.
- Converted all the desired columns into Datetime format.
- Performed One - Hot encoding on the desired columns.
- Checking the distribution of the columns to interpret its importance.

Now, the info is prepared to train a Machine Learning Model.

## 3.4 Model Creation

The Preprocessed info is now envisioned and drawn insights help us to select the feature that improves the accuracy of the model. The info is randomly used for modeling with different machine learning algorithms to create a model to predict the Salary . After performing on different algorithms, we use Logistic Regression to create a model and then also perform Hyperparameter Tuning to improve the accuracy of the model.



### 3.5 Data from User

Take input from user unsing streamlit objects like slider, container, expender etc.

### 3.6 Data Validation

The data provided by the user is then processed by the app.py file and validated. The validated data is then sent to the prepared model for prediction.

### 3.7 Rendering the Results

The data sent for the prediction is then rendered to the web page.

## 4. Deployment

The tested model is then deployed to AWS. So, users can access the project from any internet device .

### 4.1 unit test cases


| Test Case Description | Pre - Requisites | Expected Results |
|---|---|---|
| Verify whether the Webpage is accessible to the User or not. | Webpage URL is accessible. | Webpage should be accessible to the User. |
| Verify whether the webpage is completely loads for the User or not. | 1. Webpage URL is accessible.<br>2. Webpage is deployed. | The Webpage should be completely loads for the User when it is accessed. |
| Verify whether the user is able to enter data in input fields or not | 1. Webpage URL is accessible.<br>2. Webpage is deployed.<br>3. Webpage input fields are editable. | The User is able to enter data in input fields. |
| Verify whether the user is able to enter data in input fields or not | 1. Webpage URL is accessible.<br>2. Webpage is deployed.<br>3. Webpage input fields are editable. | The User is able to submit details to process. |
| Verify whether the user gets recommended results on submitting the details or not. | 1. Webpage URL is accessible.<br>2. Webpage is deployed.<br>3. Webpage input fields are editable. | The User gets recommended results on submitting the details. |