Low Level Design

Adult Census Income Prediction

|  |  |
| --- | --- |
| Written By | Devendra Pitaliya |
| Document Version | 0.1 |
| Last Revised Date | 09 – Aug -2021 |

**Document Control**

**Change Record:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| 0.1 | 09 – Aug -2021 | Devendra  Pitaliya |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Reviews:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Reviewer** | **Comments** |
|  |  |  |  |

**Approval Status:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Review Date** | **Reviewed By** | **Approved By** | **Comments** |
|  |  |  |  |  |

Contents

[**1.** **Introduction** 1](#_Toc15828)

[**1.1.** **What is Low-Level design document?** 1](#_Toc15829)

[**1.2.** **Scope** 1](#_Toc15830)

[**2.** **Architecture** 1](#_Toc15831)

[**3.** **Architecture Description** 1](#_Toc15832)

[**3.1.** **Traing Data** 1](#_Toc15833)

[**3.2.** **Load** **data** 2](#_Toc15834)

[**3.3.** **Data pre-processing** 2](#_Toc15835)

[**3.4.** **Feature** **extraction** 2](#_Toc15836)

[**3.5.** **Model** **training** 2](#_Toc15837)

[**3.6.** **Result** 3](#_Toc15838)

[**3.7.** **Data** **Labelling** 3](#_Toc15839)

[**4.** **Unit Test Cases** 3](#_Toc15848)

# 1. Introduction

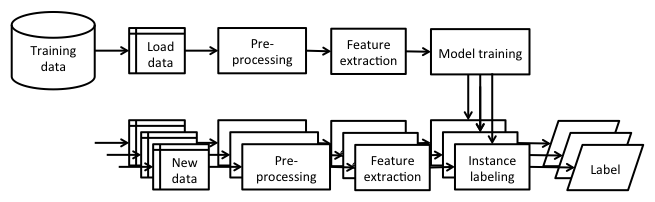
## 1.1.What is Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Food Recommendation 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-bystep [refinement](https://en.wikipedia.org/wiki/Refinement_(computing)) 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 Description

## 3.1. **Traing** Data

Kaggle dataset is the biggest publicly available dataset. The information consist of 8 categorical and 6 continuous attributes containing information on age, education, nationality, marital status, relationship status, occupation, work classification, gender, race, working hours per week, capital loss and capital gain

## 3.2.Load data

In order to process data, we have to load data on Jupyter notebook.

## 3.3.Data pre-processing

## 

Before processing the Adult Dataset, cleaning the data with certain pre-processing techniques becomes a necessity.

This includes: 1) Handling Missing Values: The dataset contains certain set of missing values for categorical features, work class, occupation, native-country which has been dealt with some algorithmic transformations applied to the data. The missing values are flexibly handled for every attribute by setting a default marker called ‘?’ and assigning a unique category for negating information loss.

2) Encoding of Categorical or Non-Numeric features: As all Categorical Features are non-numeric, encoding has been done in

2 stages:

• Label Encoding: All categorical features are label encoded, where alphabetically each category is assigned, numbers starting from 0. This is also done before running the Extra Trees Classifier Algorithm for efficient feature selection

. • One-Hot Encoding: This involves splitting of different categorical features into its own categories where each and every category assumes a binary value i.e., 0 if it does not belong to that category and 1 if it belongs to that category. This is important for those categorical features where there exists no ordinal relationship in between them. One-Hot Encoding has been done for categorical features having more than 2 categories. Here, for all categorical features except sex attribute, all label encoded forms are transformed into One-Hot Encoded Forms. This is because sex attribute has only 2 categories i.e., male and female, which have been already represented in binary form in a single attribute and hence to avoid the curse of dimensionality, no One-Hot Encoding is done for sex attribute.

3) Shuffling: The whole dataset has been shuffled in a consistent way such that all the categories of different attributes remain included in Training Set and Validation Set.

4) Splitting: Now, the dataset is split into training and testing sets. With 80% of the data made available for training purposes and the rest 20% is used for testing.

## 3.4.Feature extraction

Based on the scores of the Extra Tree Classifier for different attributes (as shown from Table 1) the most relevant features have been selected, that are going to be implemented in our model.

## 3.5.Model training

In order to model training, we split the data and then applied many classification algorithms like, algorithms (Logistic Regression, RandomForestClassifier, GaussianNB, DecisionTreeClassifier, SVC) after hyper-parameter-tuning got 83% accuracy with RandomForestClassifier algorithm.

## 

## 3.6. Result

Out of a total of 48,842 instances present in the dataset, 39,074 instances have been used for training while the rest 9,768 instances have been reserved for testing. After complete evaluation, the model performance is evaluated on the following metrics:

• The Training Accuracy describes the accuracy achieved on the Training Set. From the model, a Training Accuracy of 82.00% is achieved.

## 3.7.Data Labelling

This is basically a binary classification problem where a person is classified into the

>50K group or <=50K group.

After prediction 1 Is shows that >50K and 0 is <=50K

## 3.8. Deployment

We will be deploying the model to AWS.

# 4. Unit Test Cases

|  |  |  |
| --- | --- | --- |
| **Test Case Description** | **Pre-Requisite** | **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 the User is able to Enter the data | 1. Application is accessible | The User should be able to enter the data |
| Verify whether user is able to successfully Enter all the data. | 1. Application is accessible 2. User is Enter all the data | User should be able to successfully Enter all the data. |
| Verify whether user is able to see input fields. | 1. Application is accessible 2. User has field all the data | User should be able to see input fields. |
| Verify whether user is able to edit all input fields | 1. Application is accessible 2. User is able edit all input fields | User should be able to edit all input fields |
| Verify whether user gets Submit button to submit the inputs | 1. Application is accessible   2. User is getting Submit button to submit the inputs | User should get Submit button to submit the inputs |
| Verify whether user is presented with recommended results on clicking submit | 1. Application is accessible 2. user is presented with recommended results on clicking submit | User should be presented with recommended results on clicking submit |
| Verify whether the model giving the predicted output. | 1. Application is accessible   2. User is able to see predicted output. | User is able to see predicted output. |