Low-Level Design

Insurance Premium Prediction

| Written By | Mohamed Naji Aboo |
| --- | --- |
| Document Version | 0.3 |
| Last Revised Date | 30 July 2022 |

[**1. Introduction**](#_7mforch5z9yx) **3**

[1.1 What is a Low-Level design document?](#_e4xcnzii4yqt) 3

[1.2 Scope](#_bk7uaxgo54hi) 3

[**2. Architecture**](#_k4s04uu8m6hz) **3**

[**3. Architecture Description**](#_fojkcf39g9du) **4**

[3.1 Data Ingestion](#_3foqswltuw6k) 4

[3.2 Data Validation](#_ytfwe052c3do) 4

[3.3 Data Transformation](#_2l07zssc5rob) 4

[3.4 Model Training](#_hnomy28x3tey) 4

[3.4 Model Evaluation](#_f20m027r244u) 4

[3.6 Model Pusher](#_m2mktmksxgs4) 5

[**4. Unit Test Cases**](#_w80h2m54cz3i) **5**

# 1. Introduction

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

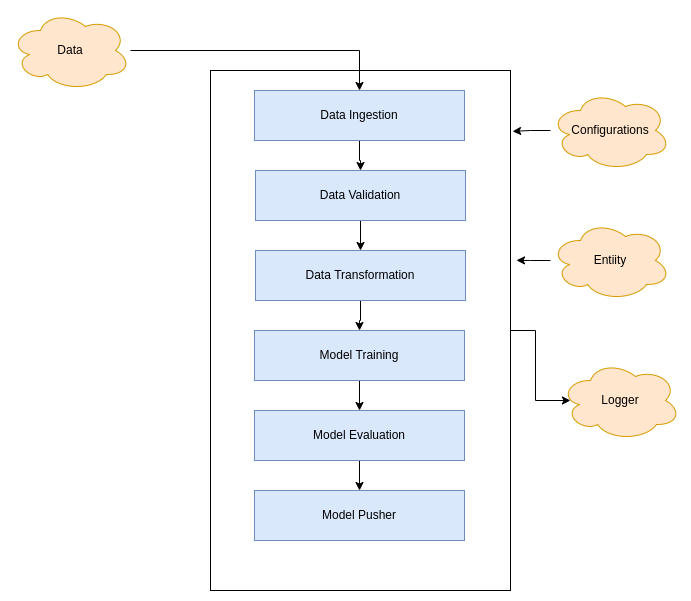
The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Insurance Premium Prediction. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer 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

# 2. Architecture



# 3. Architecture Description

## 3.1 Data Ingestion

* + Here we will download the dataset from the repo
  + We will split it to test and train the data frame

## 3.2 Data Validation

* + Will do the validation
  + We shall ensure that the features are the same
  + We shall ensure the datatypes are matching
  + We shall ensure categories are matching

## 3.3 Data Transformation

* + Here will do the transformation
  + Empty values shall be replaced with the median for integer/float datatypes and for the category it shall be replaced with most frequent

## 3.4 Model Training

* + In model training, We shall run the model based on the configuration
  + We shall be adding multiple regression model configuration
  + Based on the best accuracy, we shall be selecting the model

## 3.4 Model Evaluation

* + Model shall be evaluated for train and test data
  + New model shall be compared with the last model and based on the performance we shall be selecting the model

## 3.6 Model Pusher

* + Once the new model has higher accuracy, it shall be pushed to location

# 4. Unit Test Cases

| **Test Case Description** | **Pre-Requisite** | **Expected Result** |
| --- | --- | --- |
| Verify whether the Application URL is  accessible to 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 see the logs | 1. Application URL accessible 2. Run the model training | Should able to see the logs |
| Verify whether the user is able to see the artifact on the date basis | 1. Application URL accessible 2. Run the model training | Should be able to see the artifact |
| Verify whether the user is able to download the artifact | 1. Application URL accessible 2. Run the model training 3. Browse the artifacts | Should be able to browse and download the artifact |
| Verify whether the user is able to see the latest models | 1. Application URL accessible 2. Train the model | Should be able to see the models |
| Verify whether the user is able to place the information and see the Insurance Premium | 1. Application URL accessible 2. Train the model 3. Place the form | Should be able to see the Insurance Premium |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |