Low Level Design (LLD)

**Insurance premium predicition**

Document Version: 0.2 Last Revised Date: 17-mar-2023

# Kanishka kumar

Document Version Control:

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Description |
| 0.1 | 21-01-2023 | Kanishka kumar | Introduction Architecture, Document Content, Version Control and Unit Test cases to be  added |
| 0.2 | 16-02-2023 | Kanishka kumar | Data Pre-processing,  Model Building |
| 0.3 | 17-03-2023 | Kanishka kumar | Deployment |

**Contents**

1. [Introduction 3](#_TOC_250008)
   1. [What is Low-Level design document ? 3](#_TOC_250007)
   2. [Scope 3](#_TOC_250006)
2. [Architecture 3](#_TOC_250005)
3. [Architecture Description 4](#_TOC_250004)
   1. [Data Gathering 4](#_TOC_250003)
   2. [Data Description 4](#_TOC_250002)
   3. [Tool Used 5](#_TOC_250001)
   4. [Data Pre-processing 5](#_TOC_250000)
   5. Model Building 5
   6. Data from User 6
   7. Data Validation 6
   8. Rendering Result 6
   9. Deployment 6
4. Unit Test Cases 6

# Introduction

## What is Low-Level design document?

The main purpose of this LLD documentation is to feature the required details of the project and supply the outline of the machine learning model and also the written code. This additionally provides the careful description on however the complete project has been designed end-to-end.

* 1. **Scope**

Low-level design (LLD) is a component-level design process that follows a step-by step [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

# Architecture

Import python libraries and read data

Data Processing

EDA

Handling categorical data

Saving Data in Pickle File

Hyper Parameter Tuning

Model Building

Feature Selection

Create a webpage Using Streamlit for Deployment

Deployment on Docker Cloud

Input Value and Predict Final Result

End

# Architecture Description

This project is to make associate interface for the user to grasp their approximate insurance premium prediction. we have a tendency to square measure mercantilism the gathered information into our own information then begin the project from the scratch.

## Data Gathering

The data for the current project is being gathered from Kaggle dataset, the link to the data is: https://www.kaggle.com/datasets/noordeen/insurance-premium-prediction

## Data Description

## Data dictionary as follows:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| Age | Integer | Input variable |
| Sex | String | Input variable |
| BMI | Decimal | Input variable |
| Children | Integer | Input variable |
| Smoker | String | Input variable |
| Region | String | Input variable |
| Expenses | Decimal | Output variable |

Summary Statistics:

Number of instances (observations): 1338 Number of Attributes: 7  Attribute breakdown: 4 numerical features and 3 nominal features input variables, and 1 numerical output variable.

## Tool Used

* Python 3.9 is employed because the programming language and frame works like numpy, pandas, sklearn and alternative modules for building the model.
* PyCharm is employed as IDE.
* For visualizations seaborn and components of matplotlib are getting used
* For information assortment prophetess info is getting used version management.
* Heroku is employed for deployment

## Data Pre-processing

Necessary data preprocessing is performed on the dataset before fetching it in the model building phase.

## Model Building

The pre-processed information is then envisioned and every one the specified insights are being drawn. though from the drawn insights, the info is at random unfold however still modelling is performed with completely different machine learning algorithms to form positive we tend to cowl all the chances. and eventually, for sure random forest regression performed well and any

hyperparameter calibration is finished to extend the model’s accuracy.

## Data from User

The data from the user is retrieved from the created HTML web page.

## Data Validation

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

## Rendering Result

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

## Deployment

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

## Unit Tests

