**Low Level Design (LLD)**

**Insurance Premium Prediction**

Revision 1.0.1

Date: 17/11/2023

Description: Initial revision HLD

**Document Version Control**

* **Reviewers:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Revision** | **Comments** | **Reviewers** |
| 17/11/2023 | 1.0.1 | Initial revision (1.0.1) | Robin Mitra |
|  |  |  |  |
|  |  |  |  |

* **Change Information:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Revision** | **Description** | **Author** |
| 17/11/2023 | 1.0.1 | Initial revision (1.0.1) | Robin Mitra |
|  |  |  |  |
|  |  |  |  |

**Contents**

Document Version Control…………………………………………………………………………………………………………… 2

Abstract……………………………………………………………………………………………………………………………………….. 4

1 Introduction…………………………………………………………………………………………………………………….. 5

1.1 Why this Low-Level Design Document?.................................................................. 5

1.2 Scope………………………………………………………………………………………………………………….. 5

1.3 Definitions………………………………………………………………………………………………………….. 5

2 Architecture……………………………………………………………………………………………………………………..6

3 Architecture Description…………………………………………………………………………………………………..7

3.1 Data Collection……………………………………………………………………………………………………. 7

3.2 Data Pre-processing……………………………………………………………………………………………. 7

3.3 Model building…………………………………………………………………………………………………… 7

3.4 Model Evaluation……………..………………………………………………………………………………… 8

3.5 Selecting the best models.…………………………………………………………………………………..8

3.7 Model Deployment…………………………………………………………………………………………….. 8

**Abstract**

An insurance premium is the regular payment made by an individual or entity to an insurance company in exchange for coverage. The cost of the premium is calculated based on a variety of factors, including the type of insurance (such as auto, health, life, or property), the level of coverage, and the risk profile of the policyholder (such as age, health status, or driving history). These payments allow the insurer to maintain financial reserves to fulfill its contractual obligations when policyholders make claims. Understanding how premiums are determined is crucial for individuals and businesses to select appropriate coverage and manage their financial protection efficiently.

**1** **Introduction**

**1.1** **Why this Low-Level Design Document?**

The purpose of this Low-Level Design (LLD) Document is to is to give an internal logical design of the actual program code for the Insurance Premium Prediction 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-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 defined during data design work.

**1.3 Definitions**

|  |  |
| --- | --- |
| Term | Description |
| Database | Collection of all the information monitored by this system |
| IDE | Integrated Development Environment |
| EDA | Exploratory Data Analysis |

**2** **Architecture**

**3 Architecture Description**

**3.1 Data Collection**

For training and testing the model, I used the public data set available in Kaggle, “Insurance Premium Prediction” by nursnaaz

URL: https://www.kaggle.com/noordeen/insurance-premium-Prediction

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. Missing Attribute Values: None

**3.2 Data Pre-processing**

For data pre-processing, the dataset in the form of pandas data-frame as an input, follows following procedure:

• A method for data split, which takes the percentage of test data i.e., test size as an input and splits the data-frame into training and testing data-frames respectively using the “train\_test\_split” method from scikit learn. Logs will be updated.

• A method features scaling, which takes the training and testing data-frames as inputs and scales the features in both using the StandardScaler from scikit earn library. Logs will be updated.

• A method for splitting as x & y, which takes the training data-frame, testing data-frame and the target column’s name as inputs, splits both the training and testing data-frames into the dataframe containing independent and dependent features respectively.

**3.3 Model Building**

For model building, used regression models & Tree models using sklearn.

**Linear regression:** Linear regression takes the X\_train, y\_train,X\_test and y\_test data-frames respectively as their inputs. Linear regression model on all the features, eliminates each one with respect to its p value, if it is above 0.05. Then the left-over features are the relevant features, which are used to build a Linear regression model. It returns the linear regression model, its predictions on both the training and testing data-frames and the relevant features in the form of python list.

**Decision Tree:** DecisionTreeRegressor this method builds a model using DecisionTreeRegressor algorithm imported from the scikit learn library, by considering the best hyper parameters, after performing the randomized search cross validation technique. It returns the model and displays the importance of each feature in the console.

**Random Forest:** random forest regressor this method builds a model using RandomForestRegressor algorithm imported from the scikit learn library, by considering the best hyperparameters, after performing the randomized search cross validation technique. It returns the model and displays the importance of each feature in the console.

**Gradient Boosting:** gradient boosting regressor method builds a model using GradientBoostingRegressor algorithm imported from the scikit learn library, by considering the best hyperparameters, after performing the randomized search cross validation technique. It returns the model and displays the importance of each feature in the console.

**3.4 Model Evaluation:**

For model evaluation, the following methods and logs will be updated by each one of them accordingly.

**“r2\_score”: -** This method calculates the r2\_score of a model, by taking both the true and the predicted values of the target variable and returns the result.

**“Adj\_r2\_score”: -** This method calculates the adjusted r2 score of a model, by taking the data-frame containing the predictor features, the true and the predicted values of the target variable and returns the result.

**“RMSE\_score”: -** This method calculates the root mean square error of a model on the given dataset, by taking both the true and the predicted values of the target variable and returns the result.

**3.6 Selecting the best models**

Based on the results, saved the “Xgboost regressor” model and the “Random Forest regressor” model into the “models” directory using the pickle library, as these two are the best among all.

**3.7 Model Deployment**

Deployed the “Xgboost regressor” model in the web application using Flask a micro web framework in python. The deployment part of the code runs in the “app.py” file, connecting with the web page

designed using HTML. Then, deployed on web using the GitHub a version control system.

**Designed user interface using HTML. It looks as per the below image.**

**Users can input the personal details and once they hit “Predict” button, the predicted Insurance premium prediction will be displayed as below**