Low Level Design (LLD)

Insurance Premium Prediction

Revision 1.0.1

Date: 07/01/2023

Description: Initial revision HLD

Document Version Control

- Change Information:

Date	Revision	Description	Author
07/01/2023	1.0.1	Initial revision (1.0.1)	Anand Chavan

- Reviewers:

Date	Revision	comments	Reviewer
07/01/2023	1.0.1	Initial revision (1.0.1)	Anand Chavan

- Approval status

Date	Revision	comments	Reviewer

Contents

1. Introduction

1.1 Why this Low-Level Design Document?

The goal of LLD or a Low-level design document 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.

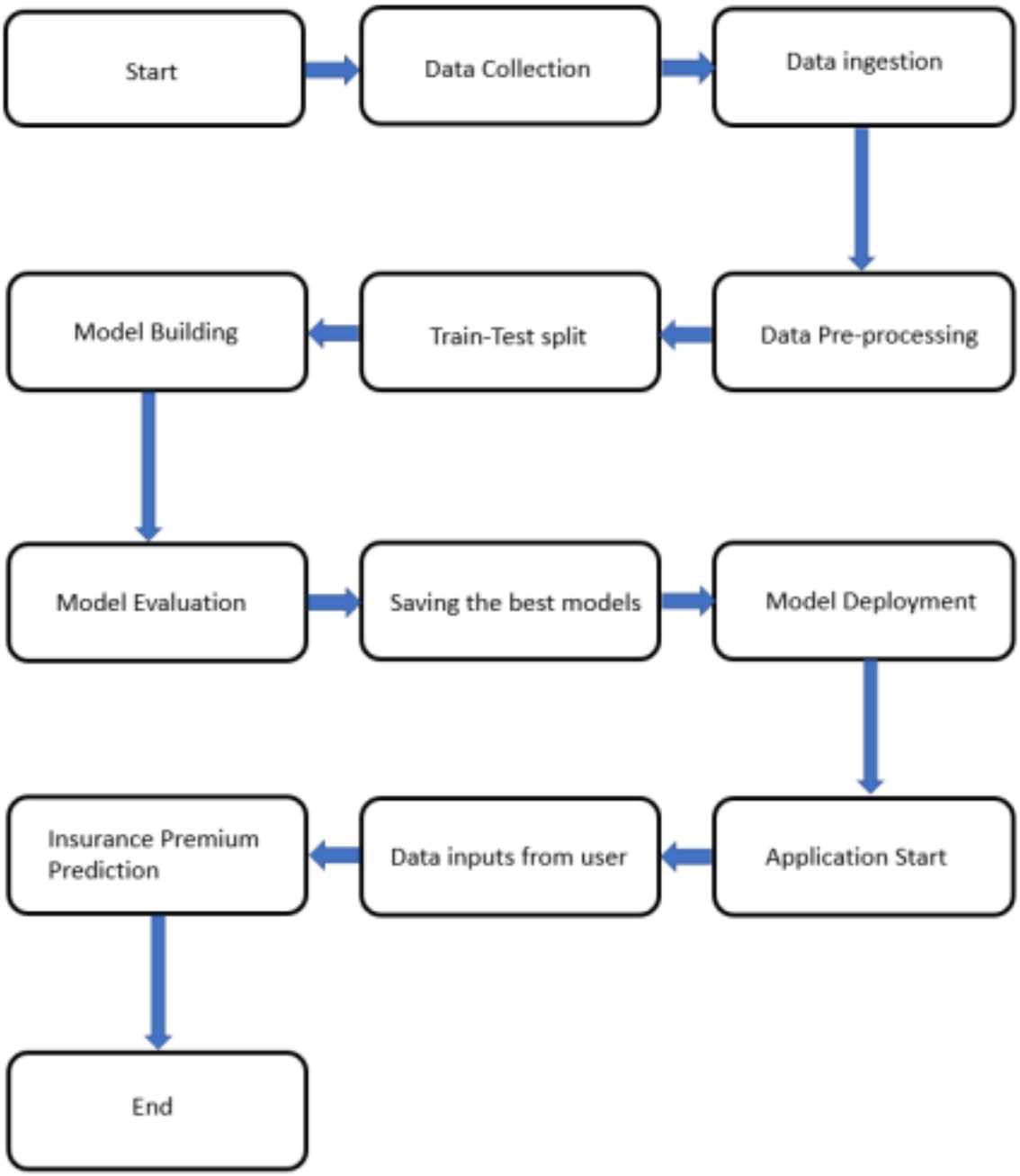
1.2 Scope

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
EDA	Exploratory Data Analysis
IDE	Integrated Development Environment
PaaS	Platform as a Service

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 ingestion

For loading the dataset into the coding environment, created a file containing “Insurance Premium Prediction.ipynb” takes the dataset file path as input.

3.3 Data Pre-processing

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

- Outlier detection: detection of outliers using Interquartile range (IQR) and records logs.
- 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 data frame containing independent and dependent features respectively.

3.4 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.

3.5 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 “Linear regression” model and the “ Linear regression” model into the “models” directory using the joblib library, as these two are the best among all.

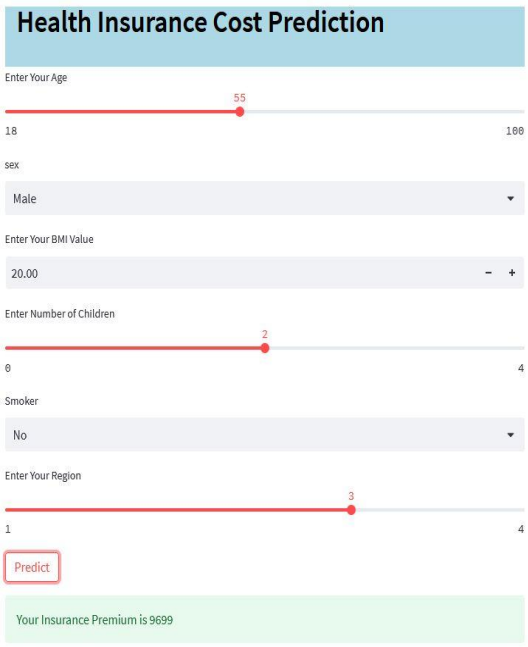
3.7 Model Deployment

Deployed the “Linear Regression” model in the web application using Streamlit a micro web framework in python. The deployment part of the code runs in the “main.py” file, connecting with the web page designed using Streamlit. Then, deployed on the web using the GitHub version control system, and the Render.

Application URL -

https://insurance-premium-prediction-pgzp.onrender.com

Designed user interface using Streamlit . 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,

4. Unit test cases

Test case description	Prerequisite	Expected result
Verify whether the Application URL is accessible to the user	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.	Application URL is accessible Application is deployed	The application should load completely for the user when the URL is accessed
Verify whether user can edit all the input fields	Application loads completely for the user. All the input fields loaded	User should be able to edit all the input fields
Verify whether user gets “Predict” button to make predictions on the given inputs	Application URL is accessible Application loads completely for the user. All the input fields loaded	User should get a “Predict” button to make predictions on the given inputs.
Verify whether user is presented with recommended results on clicking the “Predict” button	Application loads completely for the user All the input fields loaded	Users should be presented with recommended results on clicking the “Predict” button.