LOW-LEVEL DESIGN

INSURANCE PREMIUM PREDICTION

Barnalikka Pradhan, Punith B C
iNeuron

Document Version Control

Date Issued	Version	Description	Author
29.09.2022	V1.0	Initial LLD-V1.0	Barnalikka Pradhan, Punith B C

Contents

1. Introduction

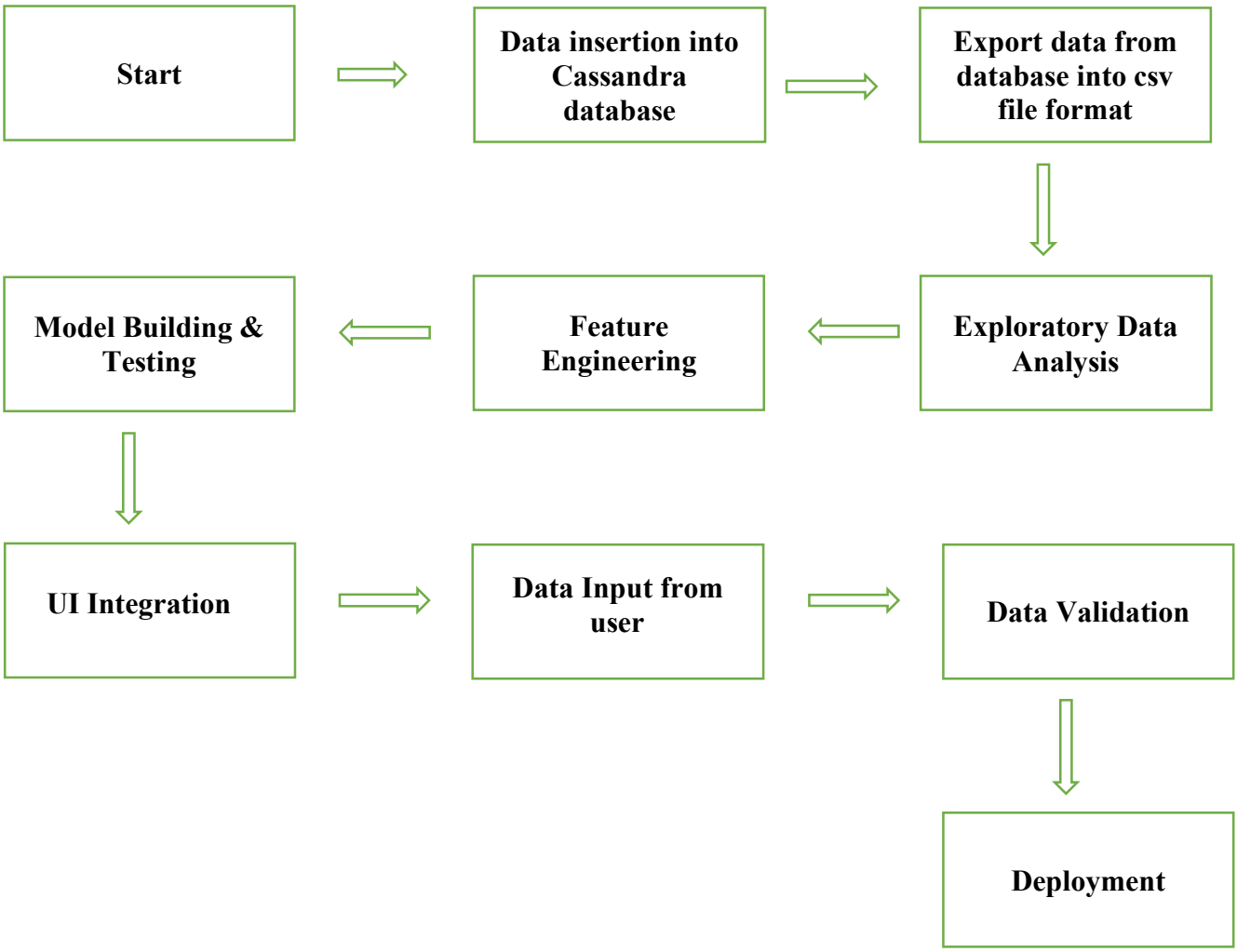
1.1 What is Low-Level Design Document?

The goal of LLD or Low-Level design document (LLD) is to give the internal logical design of the actual program code so that the programmer can directly can directly code the program from the document.
Low-Level design is created based on the High-Level design and is aptly suited for describing the outline of the machine learning model and the written code and how the project has been designed end to end.

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 Description

The dataset has been taken from kaggle consisting of 1338 individuals structured data taking attributes of age, sex, body mass index, no of children, smoker or non smoker, region they belong to and their expenses stored in a csv file format.

3.2 Data Insertion into Database

a. In Cassandra, create database with name experiment, if dtabase already created, open the connection to the database.
b. Next step would be to create a table in that database.
c. Insert all the files in that table.

3.3 Export Data from Database

The data that is stored in cassandra database is exported as a CSV file to be used for Data Pre-processing and Model Training.

3.4 Exploratory Data Analysis

One of the best ways to see the relation among the variables is by visualization.

- In the univariate analysis we have analyzed each of the variables using distplot to see whether expenses has a normal distribution, countplot to see age of insurers, no of children, region to which they belong ,category of BMI and piecharts to check the no of smokers to non-smoker and male to female.

- For bi-variate analysis we have tried to see the relationship between expenses to no of children these individulas have and to region they belong to.

- For multi-variate analysis, we try to explore the relation of whether the person is a smoker or not to his/her expenses related to age for one scatterplot and the other to the bmi.

3.5 Feature Engineering

This part includes encoding to convert categorical variables like sex, smoker to non-smoker and the region they belong to numeric variables.
This is followed by train test split of the dataset accompanied by Standardization which is a scaling technique used which normally rescales the distribution of independent variables such that the mean of observed values in 0 and standard deviation is 1.

3.6 Model building & Testing

Once the data has been standardized one can move forward to do the model building. Here we have taken three machine learning algorithms namely Decision Tree Regressor, Random Forest Regressor, Gradient Boost Regressor along with grid search CV for hyperparmaterized tuning, to train and test the model so that once we get the model with the highest accuracy,

we can choose that model for prediction of the insurance premium which in this case happens to be the Random forest regressor.

3.7 UI Integration

Front end development will be created using HTML/CSS framework where the functionality and design of the app will be created in such a manner to ensure smooth entry of data by the user.

3.8 Data Input from User

The physiological data from the user basically the age, sex, body mass index, no of children, smoker or non smoker, region they belong to is collected.

3.9 Data Validation

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

3.10 Deployment

The data entered by the user will be sent to back end application created using Flask Framework, where the choosen Random Forest Regressor will predict the output from the data provided by the user and API endpoints such as POST and GET will be fetching and displaying data on front end of the application.
The deployement is done on AWS a cloud based plateform.

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 URL is deployed.	Application URL should load completely for the user when URL is accessed.
Verify whether user can see input field after opening URL.	1. Application is accessible.	User should be able to see input fields after opening URL.
Verify whether user can edit all the input fields.	1. Application is accessible.	User should be able to edit all the input fields.
Verify whether user has options to filter the inputs fields.	1. Application is accessible.	User should filter the options of input fields.
Verify whether user gets submit button to submit the inputs.	1. Application is accessible.	User should get submit button to submit the inputs.
Verify whether user can see the output after submitting the inputs.	1. Application is accessible.	User should get outputs after submitting the inputs.