High Level Design(HLD)

Insurance Premium Prediction

Version 1.0

25/08/2021

Version Control

Version	Description	Responsible Party	Date
1.0	Initial version	Motheeshkumar Jay	24-08-2021

1 Abstract

Buying a health insurance policy for yourself and your family is important because medical care is expensive, especially in the private sector. Hospitalisation can burn a hole in your pocket and derail your finances. All this can be avoided by just paying a small annual premium which would lessen your stress in case of medical emergencies. A good health insurance policy would usually cover expenses made towards doctor consultation fees, costs towards medical tests, ambulance charges, hospitalization costs and even post-hospitalization recovery costs to a certain extent. But people are not sure which health insurance scheme they need to choose and how much they have to pay.

The goal of this project is to give people an estimate of how much they need based on their individual health situation. After that, customers can work with any health insurance carrier and its plans and perks while keeping the projected cost from our study in mind. This can assist a person in concentrating on the health side of an insurance policy rather than the ineffective part.

1. Introduction

1.1. Why this High-Level Design Document?

The purpose of this High-Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding and can be used as a reference manual for how the modules interact at a high level.

- Present all the design aspects and define them in detail
- Describe the user interface being implemented
- Describe the hardware and software interfaces
- Describe the performance requirements
- Include design features and the architecture of the project
- List and describe the non-functional attributes like
    1. Security
    2. Reliability
    3. Maintainability
    4. Portability
    5. Reusability
    6. Application compatibility
    7. Resource utilization
    8. Serviceability

1.2. Scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

1.3. Definition

Term	Description
IPP	Insurance Premium Prediction
Database	Collection of all information monitored by this application
AWS	Amazon Web Services
IDE	Integrated Development Environment

2. General Description

2.1. Product perspective

The IPP is a web application designed to help the users to get an estimate of how much insurance premium they need based on their individual health situation.

2.2. Problem Statement

Create a web application to facilitate the functionality for the users to enter their details and get an estimate of premium amount.

Approach: Implement the below feature in your application.

1. IPP home page (Accept detail of users related to their health and family details)

2. Allow users to get the estimate by clicking on the predict button.

3. Display the estimated amount in the web application.

2.3. Proposed Solution

The proposed solution is to create a web application that allows the user to enter details related to health and family to get the estimate of how much they need based on their individual health situation. After that, customers can work with any health insurance carrier and its plans and perks while keeping the projected cost from our study in mind. This can assist a person in concentrating on the health side of an insurance policy rather than the ineffective part.

2.4. Further Improvements

We can suggest a list of insurance schemes related to the estimate based on the user input. It will help users to get more insights about insurance schemes and they can investigate more on that. Also we can increase the number of features related to user health in order to add more info which will improve the estimation accuracy. We can get feedback from users and based on that we can improve our IPP application.

2.5. Technical Requirements

- Database to store training data
- Web framework for building web applications, including managing HTTP requests and rendering templates
- Proper computing power to process requests and fetch data from the server.

2.6. Tools Used

- VS Code as IDE.
- Python Flask as backend.
- Python Jinja template, HTML, CSS, JS used as frontend.
- Apache Cassandra to retrieve, insert, create and update databases.
- GitHub is used as a version control system.
- AWS is used for deployment.

2.7. Constraints

- The IPP web application must be user friendly, as automated as possible and users should not be required to know any of the workings.
- Need to follow the same structure of training dataset from client source.

2.8. Assumptions

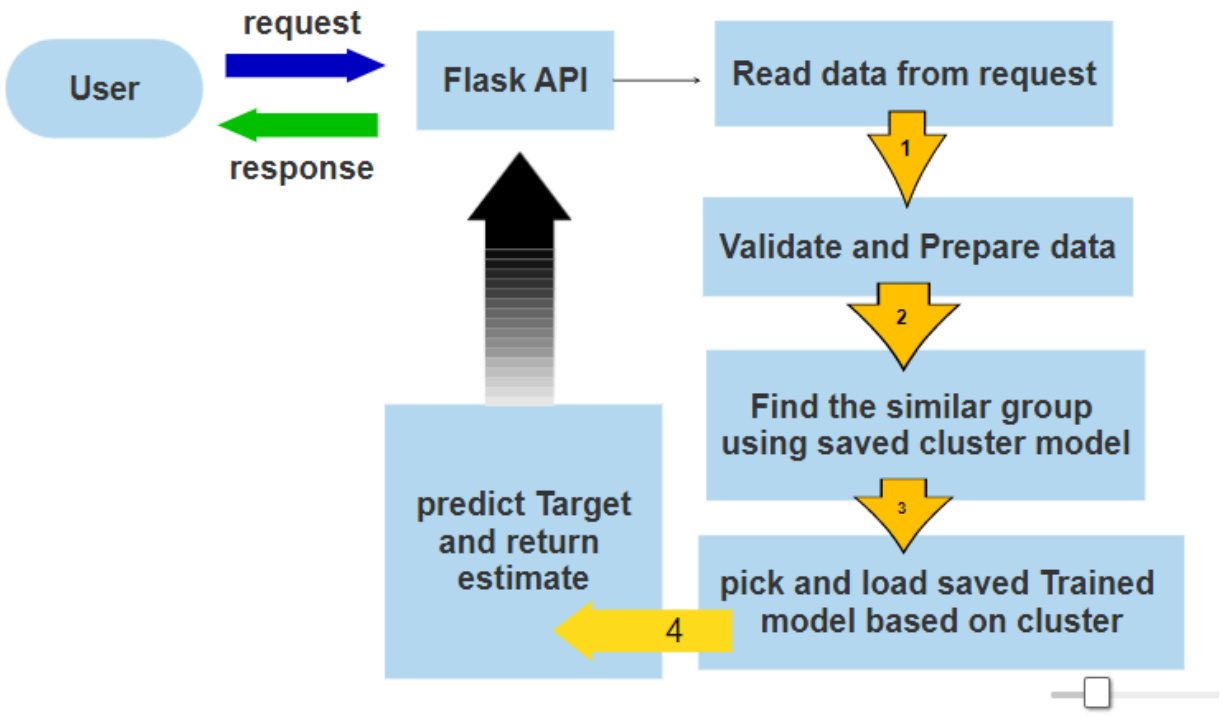
The main objective of this project is to implement the use cases as previously mentioned (2.2 problem statement). Machine learning models are used to find the pattern from a given batch dataset from client source. Everytime new dataset comes, the model needs to validate, pre-process and train the model accordingly so that we can educate the machine on a regular basis based on new updates.

3. Design Details

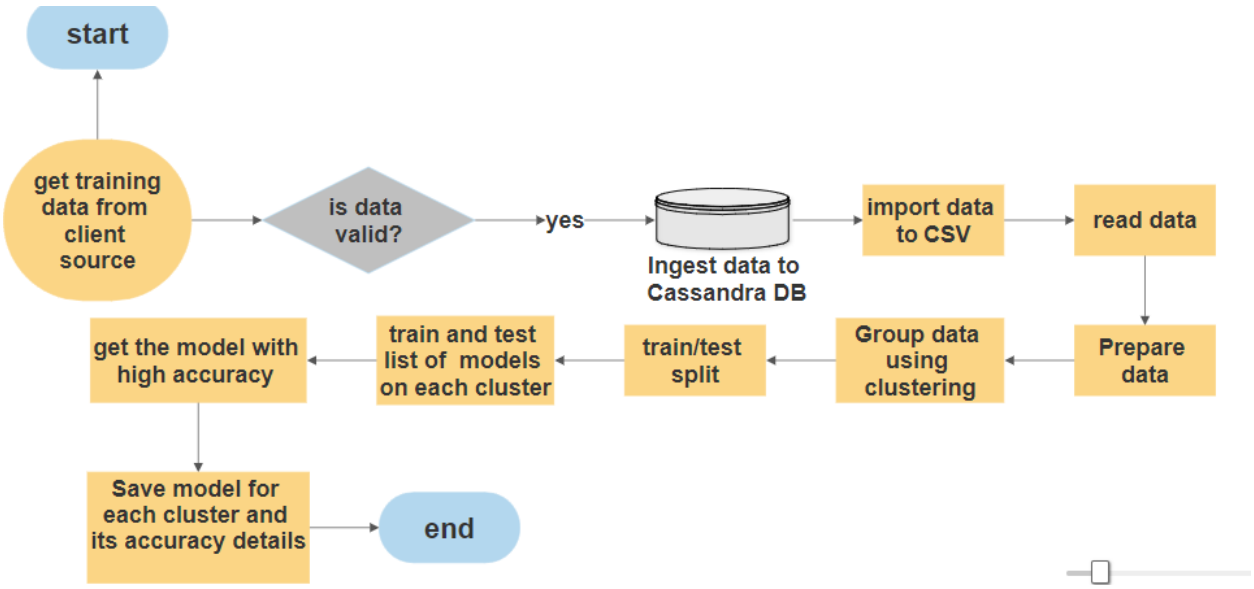
3.1. Process Flow

For identifying the pattern from the training data set we use different Machine learning models and then we use the best model to predict premium for user request. Below is the process flow diagram as shown below.

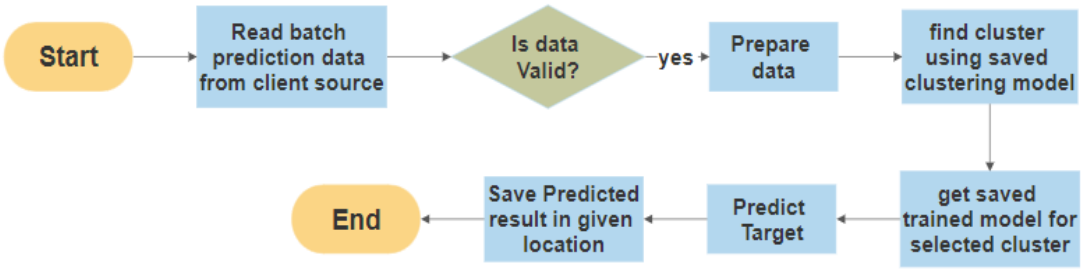
IPP Web Application workflow:



Model Training workflow:



Batch Prediction workflow:



3.2. Event Handling

Application should log each and every request so that it will be easy to track and debug.

1. Each and every process of training should be logged
2. Each and every request from the user and its status should be logged.
3. Each and every log should be maintained in separate folders.

3.3. Error Handling

- In case of any error while processing request, it should be handled and logged for debugging
- Error logging must contain below information
    1. at what step error occurred
    2. Reason for error
    3. When the error occurred
    4. In which module
    5. At what time
    6. While handling which request

4. Performance

4.1. Reusability

The code written, and the components used should have the ability to be reused with no problems

4.2. Application Compatibility

The different components for this project will be using Python as an interface between them. Each component will have its own task to perform, and it is the job of the Python to ensure proper transfer of information.

4.3. Resource Utilization

When any task is performed, it will likely use all the processing power available until that function is finished

4.4. Deployment

Deploy the model in Heroku cloud.