iNeuron.ai

Insurance Premium Prediction

High Level Design (HLD) Documentation

Shantanu Kukkar

25-09-2024

Document Version Control

Date Issued	Version	Description	Author
25/09/2024	1	Initial HLD – V1.0	Shantanu Kukkar

Table of Contents

1. Introduction

1.1. Why this High-Level Design Document?

This 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.

The HLD will:

- Present all of the design aspect 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

1.2. Scope

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

1.3. Definitions

Term	***Description***
Database	Collection of all the information monitored by this system
IDE	Integrated-Development Environment

2. General Description

2.1. Product Perspective

The Insurance premium prediction is a machine leaning model that helps users to understand their insurance premium price based on some input data.

2.2. Problem Statement

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 han the ineffective part.

2.3. PROPOSED SOLUTION

The solution is a Regression-based machine learning model. This can be implemented by using the different Regression algorithms (such as Random Forest, Decision Tree, Navi Bayes, KNN etc.)

To build this model we will perform Data Preprocessing which contains feature engineering, feature selection, feature transformation etc.

2.4. Technical Requirements

This document addresses the requirements for detecting who is defaulting the bill payments.

Here are some requirements for this project.

- Model should be exposed through API or User interface, so that anyone can test the model
- Model should be deployed on cloud (Azure, AWS, GCP, Heroku).

2.5. Data Requirements

Data requirement completely depends on our problem statement.
1.Age
2.Sex
3.BMI
4.Children
5.Smoker
6.Region
7.Expenses

2.6. Tools used

Python programming language and frameworks such as NumPy, Pandas, Matplotlib, Seaborn, Scikit-learn & Flask are used to build the whole model.

- Jupyter Notebook is used as an IDE.
- For visualization of the plots Matplotlib & Seaborn are used.
- Front end development is done HTML/CSS.
- Flask is used for backend development.
- GitHub is used as a version control system.

2.7. Constraints

The system must be user friendly, as automated as possible and users should not be required to know any of the workings

2.8. Assumptions

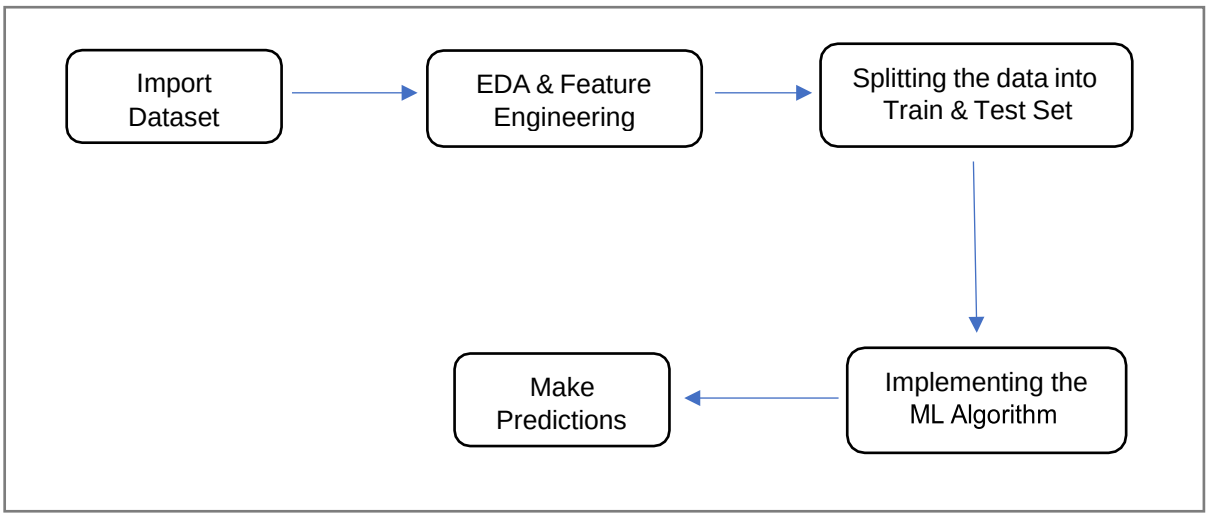
The main objective of the project is to implement the use cases as previously mentioned for new dataset that comes through Hospitals which has this solution install in their campus to capture people reports.

3. Design Details

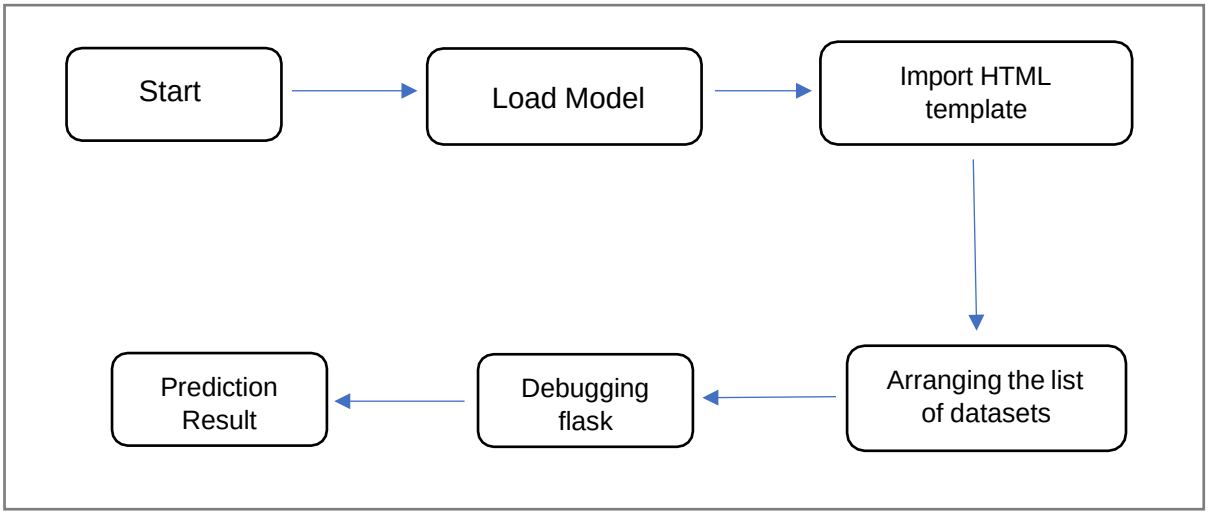
3.1. Process Flow

For identifying the defaulters, we will use a machine learning base model. Below is the process flow diagram as shown below.

Proposed Methodology:



3.1.1. Deployment Process



3.2. Evant log

The system should log every event so that the user will know what process is running internally.

Initial Step-by-Step Description:

1. The system identifies at what step logging required
2. The system should be able to log each and every system flow
3. Developers can choose logging methods. You can choose database logging/ File logging as well.
4. System should not hang even after using so many loggings. Logging just because we can easily debug issues so logging is mandatory to do.

3.3. Error Handling
Should errors be encountered, an explanation will be displayed as to what went wrong? An error will be defined as anything that falls outside the normal and intended usage.

3.4. Performance
We are developing a machine learning-based Thyroid Disease Detection solution for identifying thyroid diseases in patients who exhibit symptoms related to thyroid issues. The main objective is to facilitate early detection and prompt intervention. To ensure the system's effectiveness, regular model retraining will be implemented to continuously improve its performance and accuracy over time. This will enable us to take necessary actions promptly and provide appropriate medical attention to individuals diagnosed positively for thyroid disease.

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

3.6. 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.

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

3.8. Deployment
*Localhost

4. Conclusion
The Insurance Premium Prediction system will predict the price for helping the customers with the trained knowledge with set of rules. The user can use this system to recognize the approximate value of their insurance premium.