High Level Design (HLD)

Insurance Premium Prediction

By

Siridi Nath Peddina

High Level Design (HLD)

Document Version Control

Date	Version	Description	Author
07-03-2023	1.0	Abstract, Introduction Problem Statement	Siridi Nath
08-03-2023	1.1	Design Flow	Siridi Nath
09-03-2023	1.2	Performance Evaluation	Siridi Nath

High Level Design (HLD)

Content

High Level Design (HLD)

1. Abstract

This project represents a machine learning-based health insurance prediction system. Recently, many attempts have been made to solve this problem, as after the Covid-19 pandemic, health insurance has become one of the most prominent areas of research. We have used the USA's medical cost personal dataset from Kaggle, having 1338 entries. Features in the dataset that are used for the prediction of insurance cost include Age, Gender, BMI, Smoking Habit, number of children etc. We used Xgboost regression and determined the relation between price and these features. We trained the system using a 80-20 split and achieved an accuracy of 85%.

High Level Design (HLD)

2. Introduction

2.1 Why this High-Level Design Document?

The main purpose of this HLD documentation is to feature the required details of the project and supply the outline of the machine learning model and also the written code. This additionally provides a careful description on how the complete project has been designed end-to-end.

The HLD will:

- Present all of 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:
  - Security
  - Reliability
  - Maintainability
  - Portability
  - Reusability
  - Application compatibility
  - Resource utilization
  - Serviceability

2.2 Scope

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

High Level Design (HLD)

3. Description

3.1 Problem Perspective.

The Insurance Premium prediction is a hyper-tuned machine learning Regression model which helps to determine the expenses of insurance on different parameters

Parameters such as: - Age, Gender, Smoking etc.

3.2 Problem Statement.

Insurance Premium Prediction is a model that predicts the premium for various policyholders depending on different parameters. The HealthCare Industry is one of the prime sectors in the Insurance Industry. Also, additionally it helps the Insurance Company to revise their policy plans structure and henceforth help them to revise the premium & serve the policyholders more efficiently.

The basic idea of this system is to predict premiums on various factors & help policyholders choose a plan that is more suitable for them and importantly more economic.

3.3 Proposed Solution.

The solution proposed is to take the required batch file to predict the result. A pipeline has been created to get the prediction for the new dataset.

3.4 Solution Improvements.

The system can be made more futuristic by performing more hyper-tuning methods so that the prediction can be more accurately predictive. The project code has been designed in such a way that whenever new data will come, the model will go under training and if there will be an improvement in the model then the new model will be used for prediction.

4. Requirements

4.1 Hardware Requirements:-

A working computer to code with an active internet connection.

4.2 Tools / Software Requirements:-

- Python version used for this project 3.10 (This may get updated and some features might not be available in new version.)
- Python libraries such as NumPy, Pandas, Matplotlib, Seaborn and scikit-learn (Used for implementation of machine learning algorithms)
- Jupyter Notebook & Visual studio code is used as an IDE for writing the code.
- Github is used as the version control system
- AWS is used for deployment using docker image.
- Apache Airflow has been used to monitor the ML model.
- Streamlit has been used to create and deploy web app.

5. Data Requirements.

Whenever we are working on any project the data is completely dependent on the requirement of the problem statement. For this project, the problem statement was to create a Hyper tuned Regression machine learning model which can predict the insurance premium based on various parameters.

6. Constraints

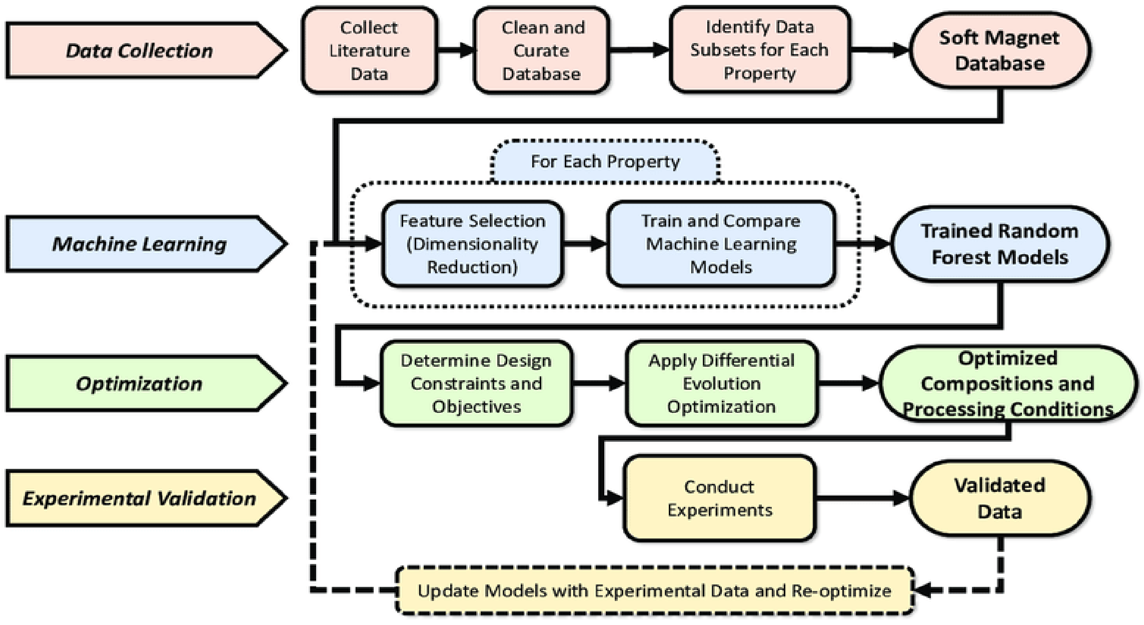
The Apache Airflow application should be user-friendly so that without knowing any technical information he should be able to use our predictive system. A Streamlit web app has been created and it should be user-friendly.

7. Assumptions

The main objective of the project is to utilize the data which is provided by the user and to predict the insurance premium. The Apache Airflow application should be accessible from every system which is connected to the internet to predict the result on new dataset. A Streamlit web app has been created as well and it should be accessible to every user who is connected to the internet.

High Level Design (HLD)

8. Design Flow.



Data Collection
Collect Literature Data
Clean and Curate Database
Identify Data Subsets for Each Property
Soft Magnet Database
For Each Property
Machine Learning
Feature Selection (Dimensionality Reduction)
Train and Compare Machine Learning Models
Trained Random Forest Models
Optimization
Determine Design Constraints and Objectives
Apply Differential Evolution Optimization
Optimized Compositions and Processing Conditions
Experimental Validation
Conduct Experiments
Validated Data
Update Models with Experimental Data and Re-optimize

The above flow chart represents the flow of any machine learning model which needs to be created .

9. Logging & Error Handling.

Each step is logged within the system that runs internally; it basically shows us the data time of each process which is done with our system. It provides us with logging information for end to end web applications.

The logging which we have done in the above process helps us to handle the error because the error is being logged in several log files so that the developer can rectify it.

10. Performance Evaluation.

10.1 Reusability

The elements of the code is written in such a way that it can be changed and easily written again without changing or creating an entirely different code from scratch. Just the slight changed in the code structure need to be adjusted.

10.2 Application Compatibility.

The elements of the project are written in python, it acts as the interface between the machine-learning model and the user. The Apache Airflow application can run on any system with a network connection. Also, the Streamlit web app can be accessed using any system which is connected to the internet.

10.3 Resource Utilization.

Once the task is assigned to the model doubtlessly it will use all the resources which are allocated until the task is finished.

11. Deployment.

This model is deployed on Streamlit for instances.

- Add the runner in the GitHub.
- A web app has been created and deployed using Streamlit.

12. Conclusion.

We have successfully built end-to-end ML projects using machine learning that can help predict the medical expenses of the users based on various conditions. This type of system can help users to get a better understanding of their medical expenses and based on it they can buy their insurance plan. Along with end to end projects, a Stream lit webapp has been created as well to get the result based on certain inputs like age, sex, bmi etc.