High Level Design (HLD)

Insurance Premium Prediction

Document Version Control

Date	Version	Description	Author
05-02-2023	V1.0	Initial high level design	Satya Nerurkar

Contents

Abstract

In today's age and time, health insurances have become a guarded necessity. There is awareness for buying heath insurances but very often the vision, planning, implementation and capital do not go hand in hand for finding the most suitable health insurance for themselves. Therefore, having a basic idea, of the predicted cost of their health insurance individually will save a lot of money and time.

This system created with the help of machine learning algorithms will give the people this very idea of predicted cost based on personal features of BMI, age, sex, children, smoker, region they belong to & their current expenses catering solely to the individual at large.

Our project does not give the exact amount required for any health insurance company but gives enough idea about the amount associated with an individual for his/her own health insurance.

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 project description to represent a suitable model and coding for application. This document is also intended to help detect contradictions before 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 aspects and define them in detail.
- Describe the user interface is 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

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 stack. The HLD uses non-technical to mildly technical terms which should be understandable to the administrators of the system.

2. General Description

2.1 Problem Statement & Product Perspective

The dataset contains Age, Sex, BMI (Body Mass index), Children, Smoker, Region and Expenses where I have to predict Insurance Premium Expenses with

- To detect BMI value affects the premium.
- To detect smoking affects the premium of the insurance.
- To create API interface to predict the premium

2.2 Proposed Solution

The solution proposed here is an estimating premium of insurance based on people health data and this can be implemented to perform above mention use cases. In first case, analysing how BMI value affect the people health as well as premium of the insurance. In the second case, if model detects the smoking affecting the premium, we will inform that to people. And in the last use case, we will be making an interface to predict the premium.

2.3 Technical Requirements

The solution can be a cloud-based or application hosted on an internal server or even be hosted on a local machine. For accessing this application below are the minimum requirements:

- Good internet connection.
- Web Browser.

For training model, the system requirements are as follows:

1. +4 GB RAM preferred
2. Operation System: Windows, Linux, Mac
3. Visual Studio Code / Jupyter notebook

2.4 Data Requirements

Data requirements completely depends on out problem statement.

- Comma separated values (CSV) file.
- There were six variables out of which sex, smoker and region were categorical variable which had to be encoded to transform them into numeric variables at large.

2.6 Tools Used

Python programming language and frameworks listed below are used to build the whole model.

- VS code is used as IDE.
- Matplotlib , Seaborn, plotly are used for visualization of plots.
- GitHub is used as version control system.
- Github Actions is used for CI/CD pipeline.
- MongoDB used for insert, retrieve and update the database.
- Pandas used for data analysis, NumPy for scientific computation.
- Flask is used to build API.
- Scikit-learn used for machine learning.
- Front end development done using HTML/CSS.
- AWS is used for deployment of the model.

2.6 Constraints

MLOPs on the cloud must be fully automated in consideration of continuous integration, continuous deployment with retraining approach of model, and archiving the data over time. The application should be user friendly as automated as possible. Users can easily use the application and not needed to know any of the workings.

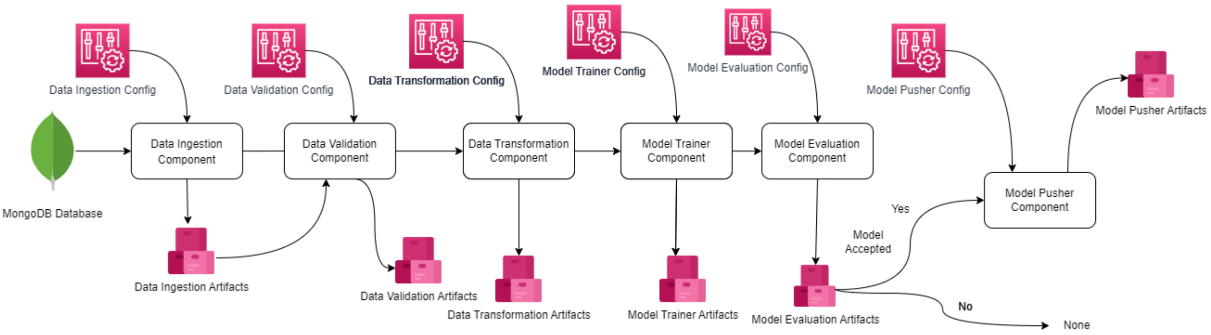
2.7 Assumptions

The main objective of the project is to develop an API to predict the premium for people on the basis of their health information. Machine learning based regression model is used for predicting above mentioned cases on the input data.

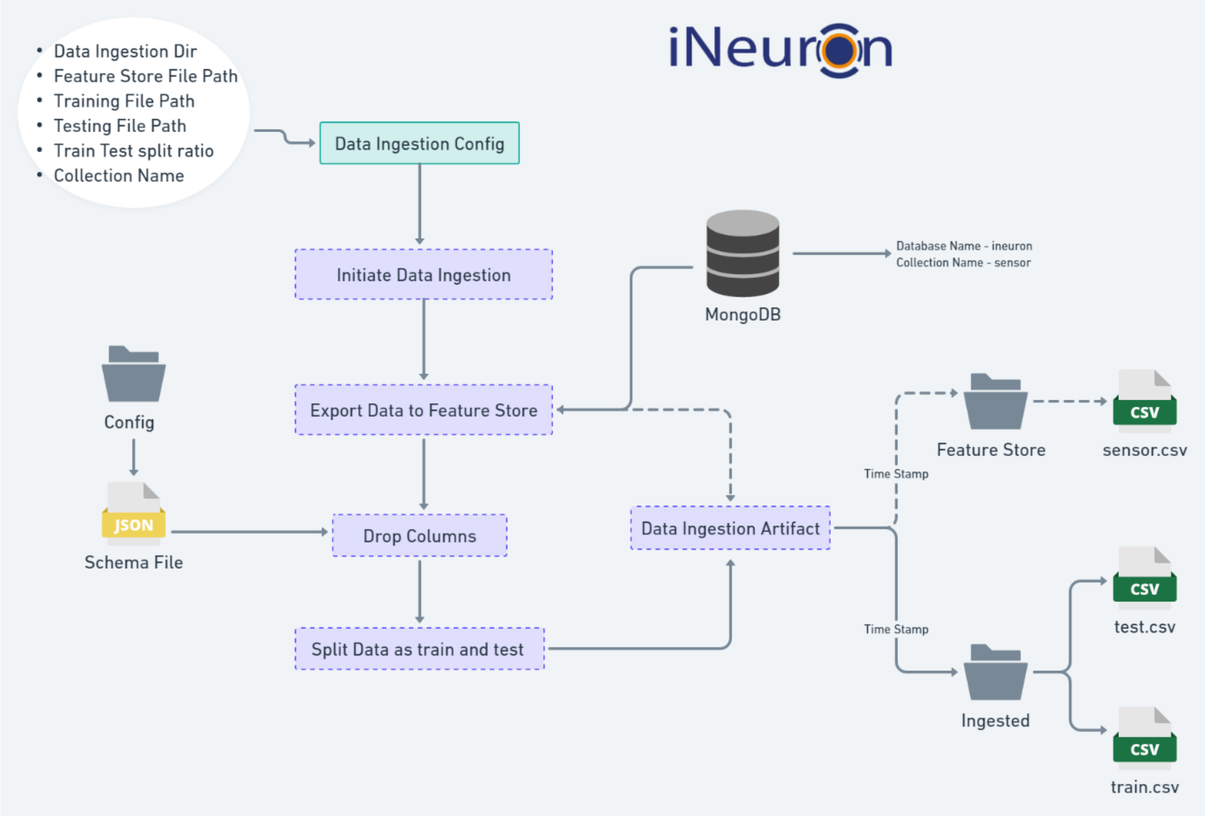
3 Design Details

3.1 Process flow

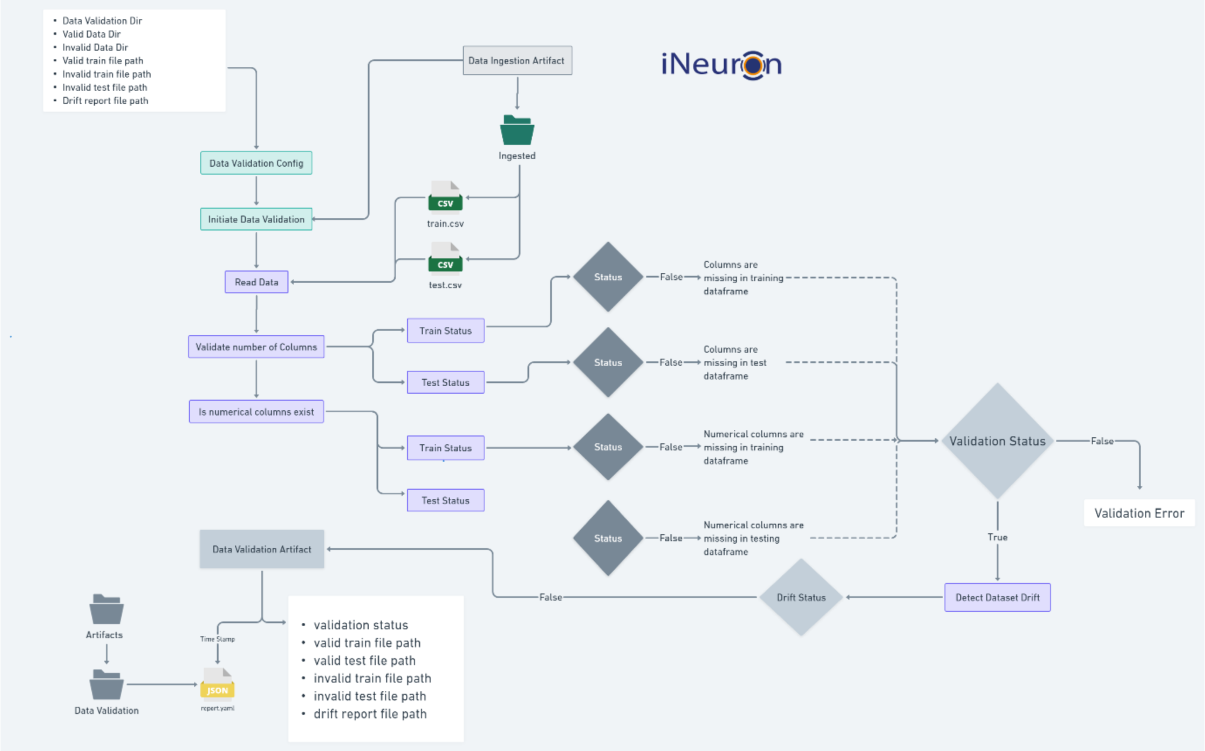
For finding the insurance premium we will be using machine learning model.
Below is the process flow diagram.



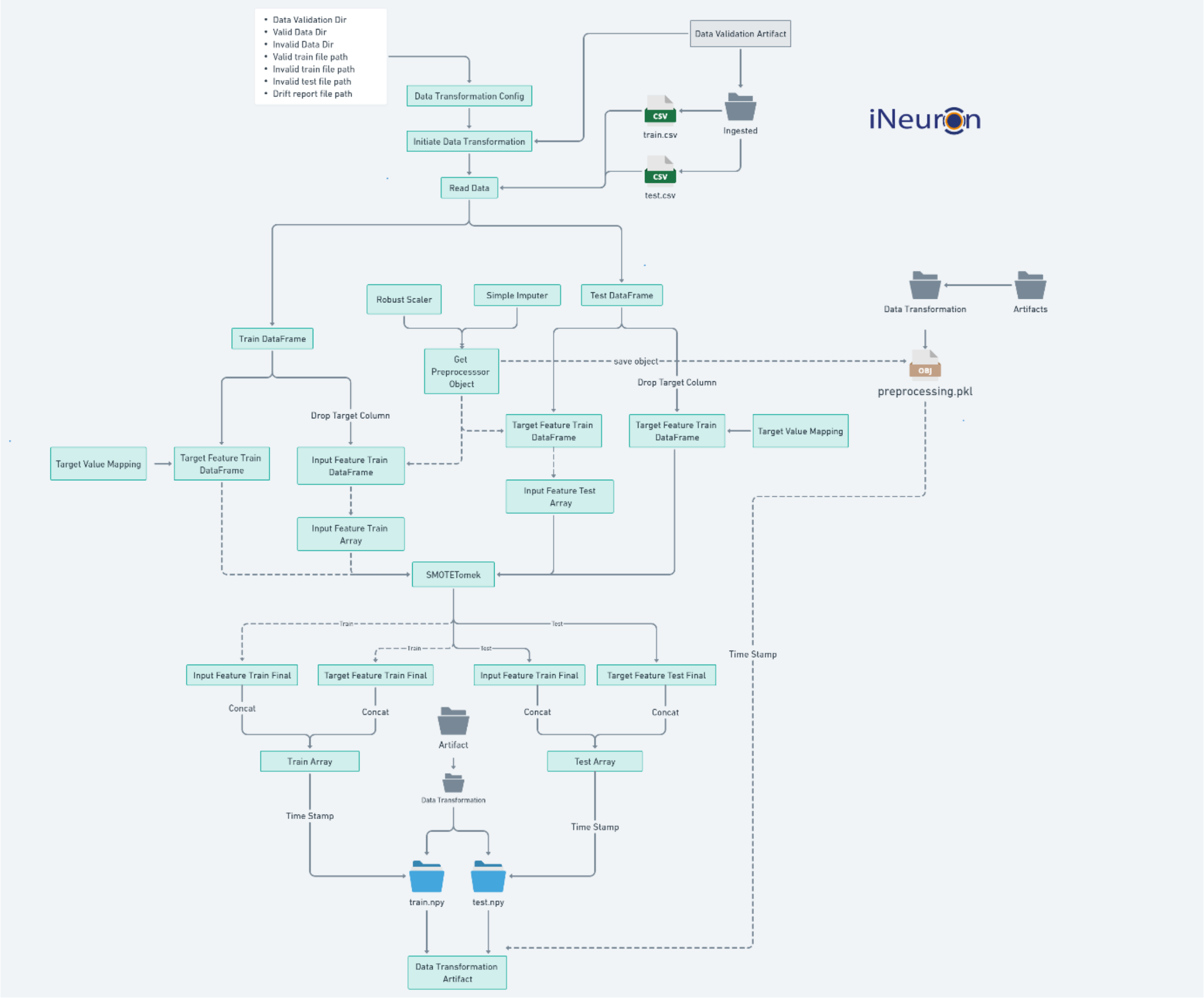
3.1.1 Data Ingestion



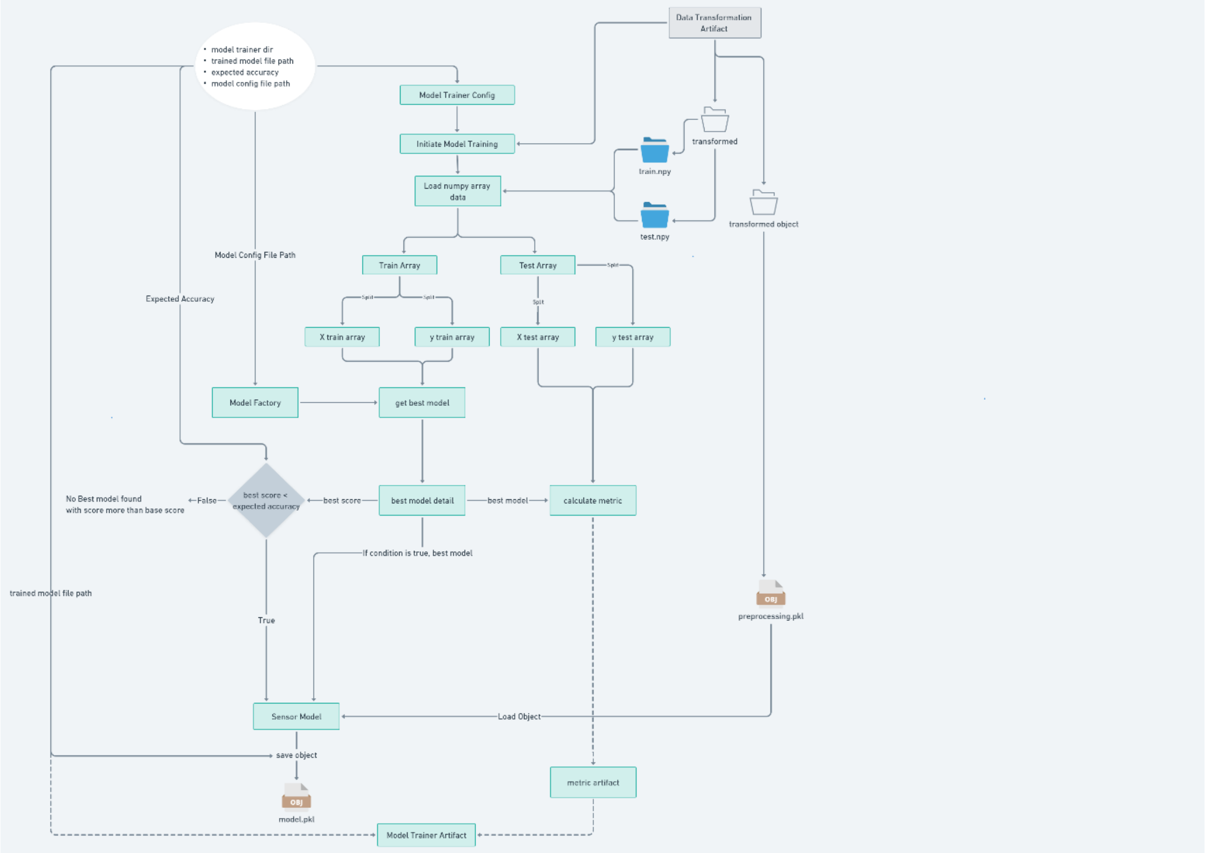
3.1.2 Data Validation



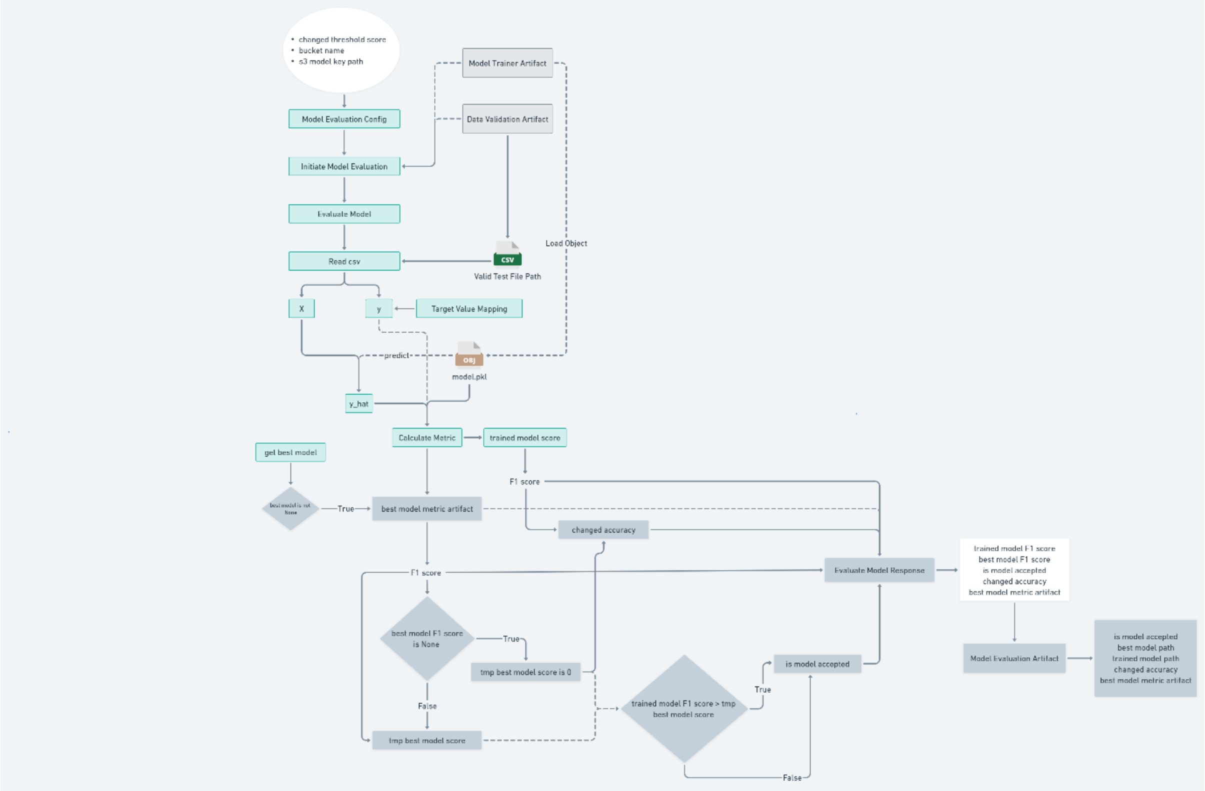
3.1.3 Data Transformation



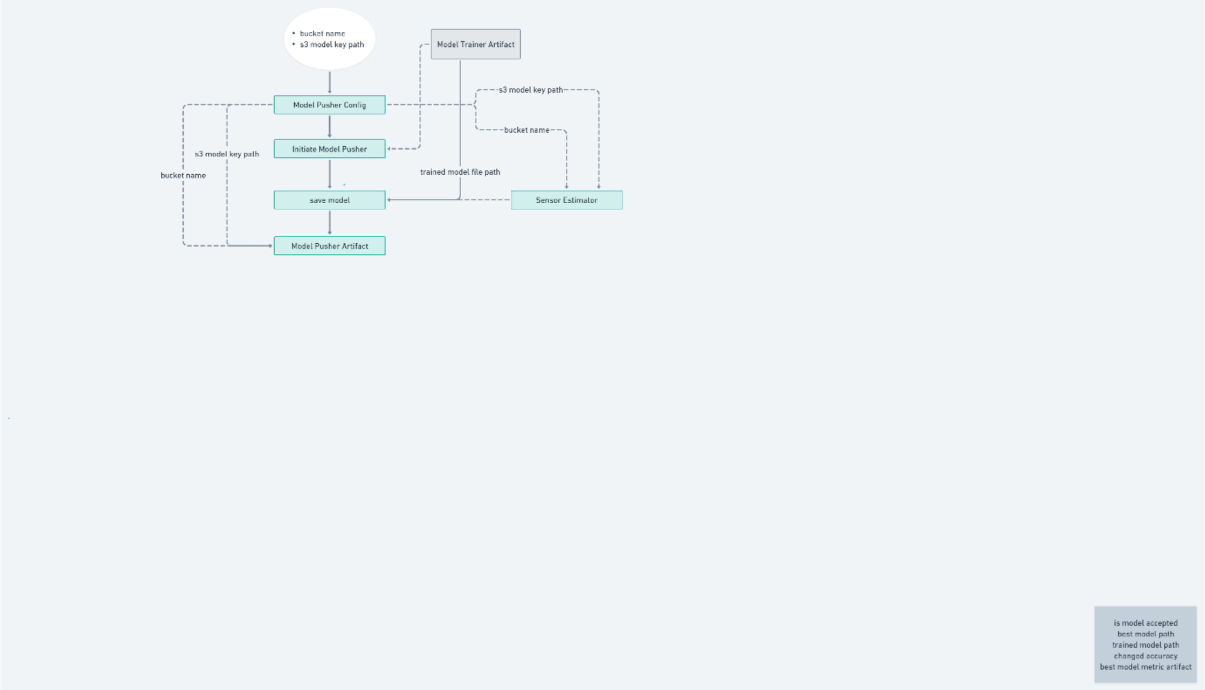
3.1.4 Model Trainer



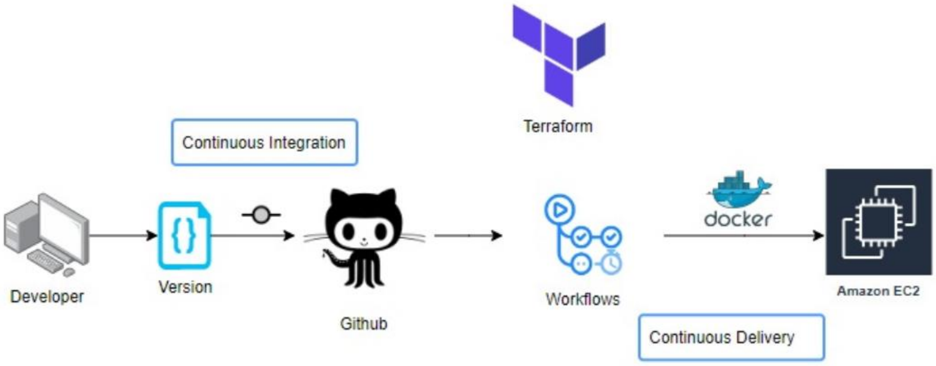
3.1.5 Model Evaluation



3.1.6 Model Pusher



3.1.7 Deployment Process



3.2 Event log

The system should log every event so that the track of every detail will be known and what process is running currently could be seen.

Initial Step-By-Step Description:

1. The System identifies at what step logging is required.
2. The System should be able to log each and every system flow.
3. Developer can choose logging method. 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

The system should identify the errors 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 Optimization

Data strategy derives performance:

- Filling missing values.
- Replacing outliers.
- Creating new features from cut expenses feature.
- Hyper parameter tuning
- Validating score again
- Reusability

The entire solution will be done in modular fashion and will be API oriented. So, in the case of the scaling the application, the components are completely reusable.

3.6 Application compatibility

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

3.7 Deployment

4 Conclusion

In this projects, The system shows us that the different techniques that are used in order to estimate the how much amount of premium required on the basis of individual health situation. After analyzing it shows how a smoker and non-smokers affecting the amount of estimate. Also, significant difference between male and female expenses. Accuracy, which plays a key role in prediction-based system. From the results we could see that Gradient Boosting turned out to be best working model for this problem in terms of the accuracy. Our predictions help user to know how much amount premium they need on the basis of their current health situation.