**High-Level Design Documentation (HLD)**



**Project On:**

**Title: Insurance Premium Prediction**

**By: - Nidhi Patel**

**Date: - 27/05/2023**

**Content**

1. **Abstract**
2. **Introduction**
   1. **Why this High-Level Document (HLD)?**
   2. **Scope**
3. **Description**
   1. **Problem Perspective**
   2. **Problem Statement**
   3. **Proposed Solution**
   4. **Solution Improvement**
4. **Requirements.**
   1. **Hardware**
   2. **Tools / Software requirements**
5. **Data requirements**
6. **Constraints**
7. **Assumptions**
8. **Design flow**
9. **Logging & Error Handling**
10. **Performance Evaluation**
    1. **Reusability**
    2. **Application compatibility**
    3. **Resource utilization**
11. **Deployment**
12. **Conclusion**
13. **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 RandomForest Regression and determined the relation between price and these features. We trained the system using a 80-20 split and achieved an accuracy of 86%.

1. **Introduction.**
   1. **Why this High level Design Documentation?**

The purpose of High-Level Documentation is to (HLD) 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 the high level.

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
  1. **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.

1. **Description.**
   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.

* 1. **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.

* 1. **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.

* 1. **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.

1. **Requirements**

**4.1Hardware Requirements:-**

A working computer to code with an active internet connection.

**4.2 Tools / Software Requirements:-**

* Python version used for this project 3.8 ( 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.
* Flask has been used to create and deploy web page



1. **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.

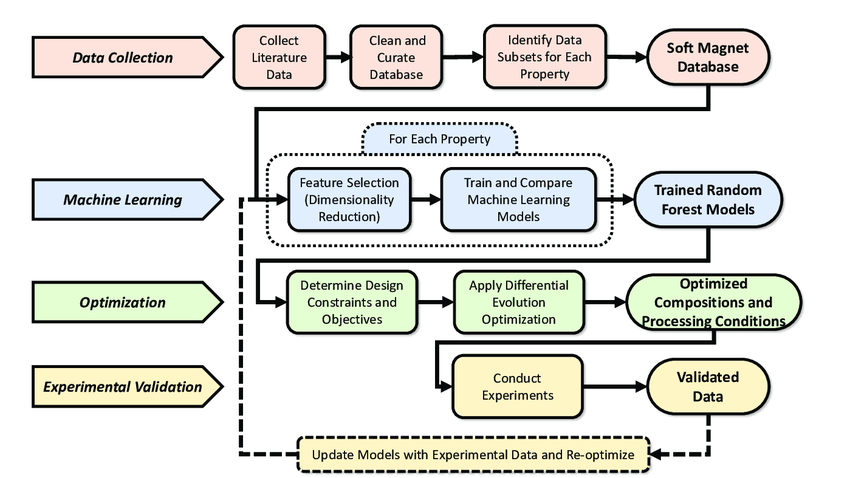
1. **Constraints**

A flask web page has been created and it should be user-friendly.

1. **Assumptions**

The main objective of the project is to utilize the data which is provided by the user and to predict the insurance premium. A flask web page has been created as well and it should be accessible to every user who is connected to the internet.

1. **Design Flow.**



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

1. **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.

1. **Performance Evaluation.** 
   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.

* 1. **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 flask web page can be accessed using any system which is connected to the internet.

* 1. **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.

1. **Deployment.**

This model is deployed on AWS Elasticbean. The following are the steps to deploy the model on the AWS platform:

* **Create Elastic Beanstalk Environment**

Navigate to Elastic Beanstalk.

Click Create application.

On the Configure environment page, set the following values:

* Application name: Insurance\_Premium\_Prediction
* Platform: Python
* Application code: Sample application

Click Next.

Select Use an existing service role and choose the existing service role that contains RootRole.

For EC2 instance profile, select the existing profile that contains InstanceLoggingProfile.

Click Next.

Under Virtual Private Cloud (VPC), select the listed VPC (from the dropdown).

Under Public IP address, check Activated.

In the Instance subnets section, select all three available subnets.

Click Next.

Under EC2 Security groups, check the non-default security group name.

Click Next.

Click Next.

Review your configurations and click Submit. It will take a few minutes to complete creation.

* **Create an S3 Bucket**

Be sure to download the files listed in the lab instructions for later use in this lab.

Navigate to S3.

Click Create bucket.

In the Bucket name field, type a unique DNS-compliant name.

Click Next.

On the Configure options screen, enable versioning.

Click Next > Next > Create bucket.

Upload the ZIP file that was downloaded earlier in this lab to the new S3 bucket.

Click on the uploaded file name, and copy the key to a text file for use later in this lab.

* **Create an AWS CodePipeline**

Navigate to CodePipeline.

On the Welcome page, click Create pipeline.

If this is your first time using CodePipeline, click Get Started.

On the Choose pipeline settings page, in Pipeline name, enter the name for your pipeline.

In Service role, choose New service role to allow CodePipeline to create a new service role in IAM.

Click Next.

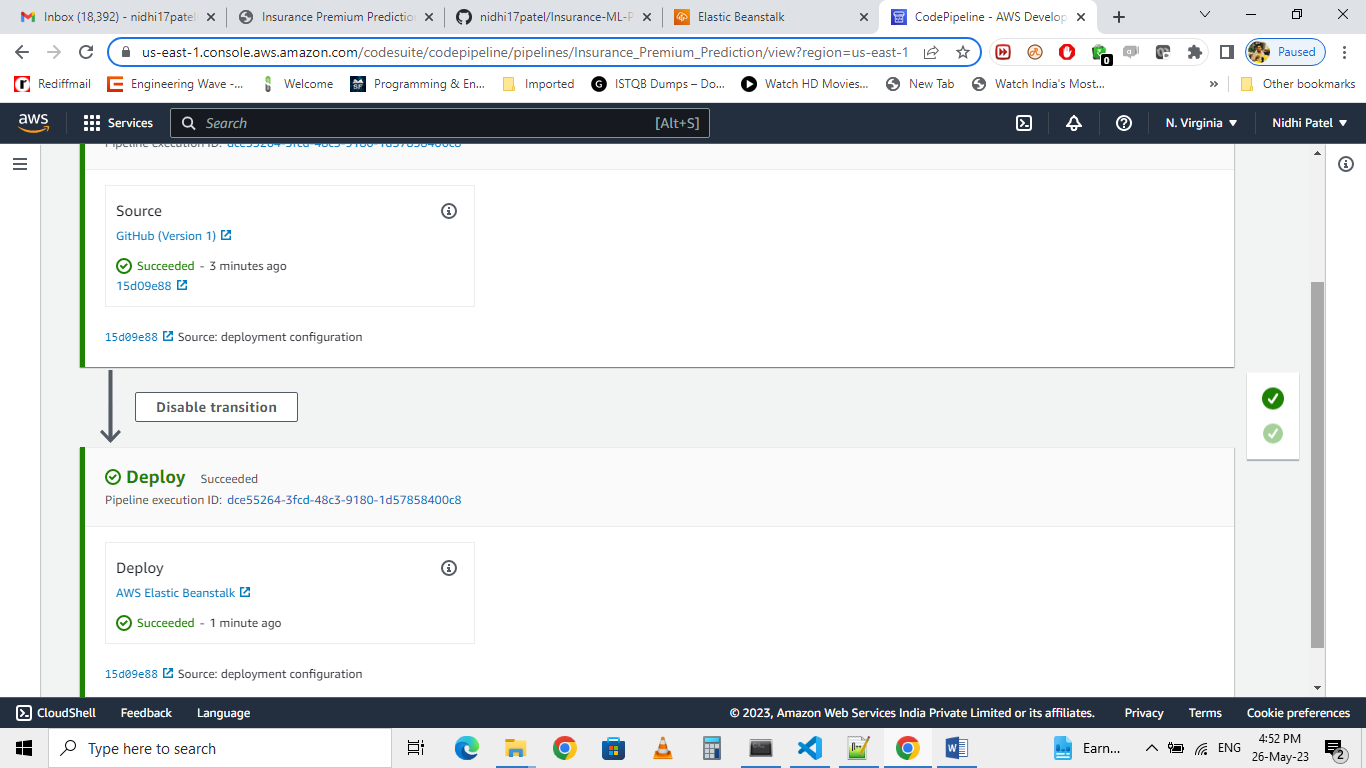
On the Add source stage page, in Source provider, choose S3, specify its required options, and then click Next.

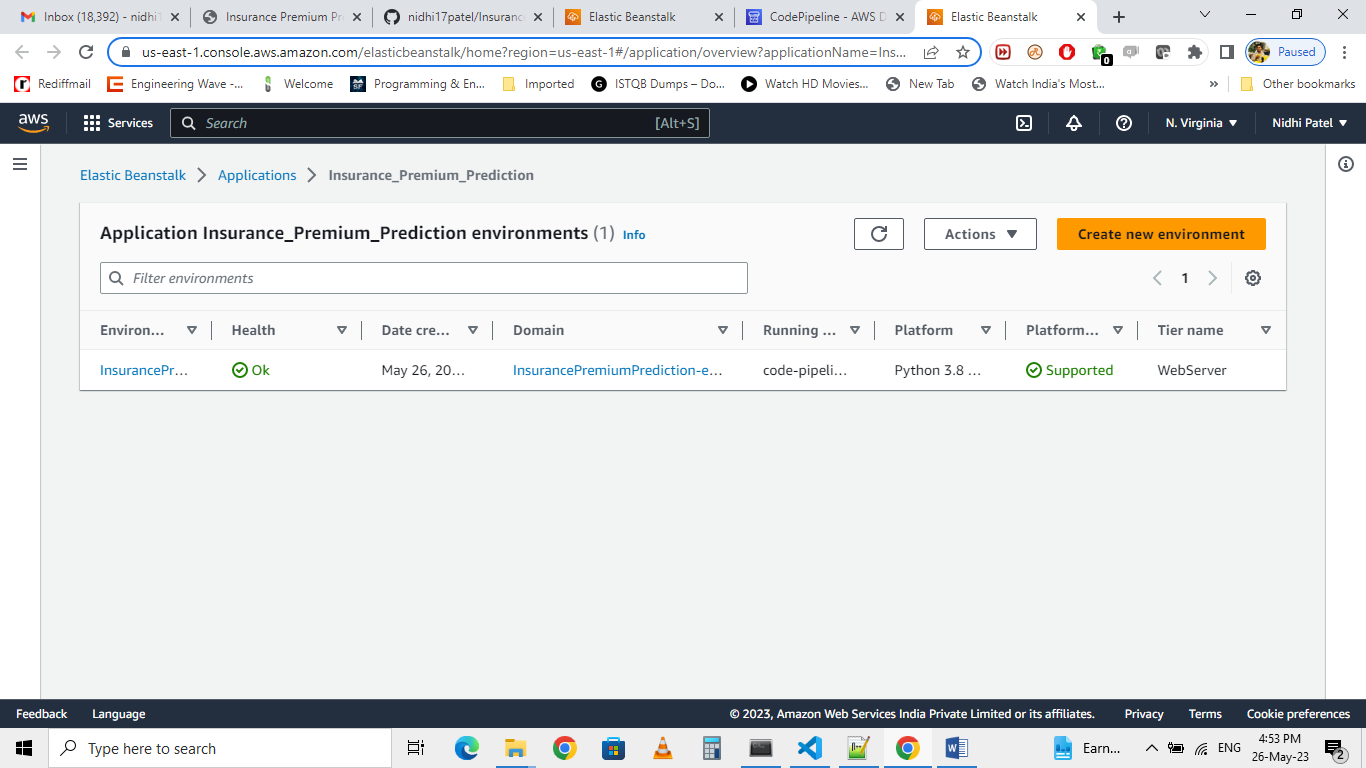
Click Skip build stage.

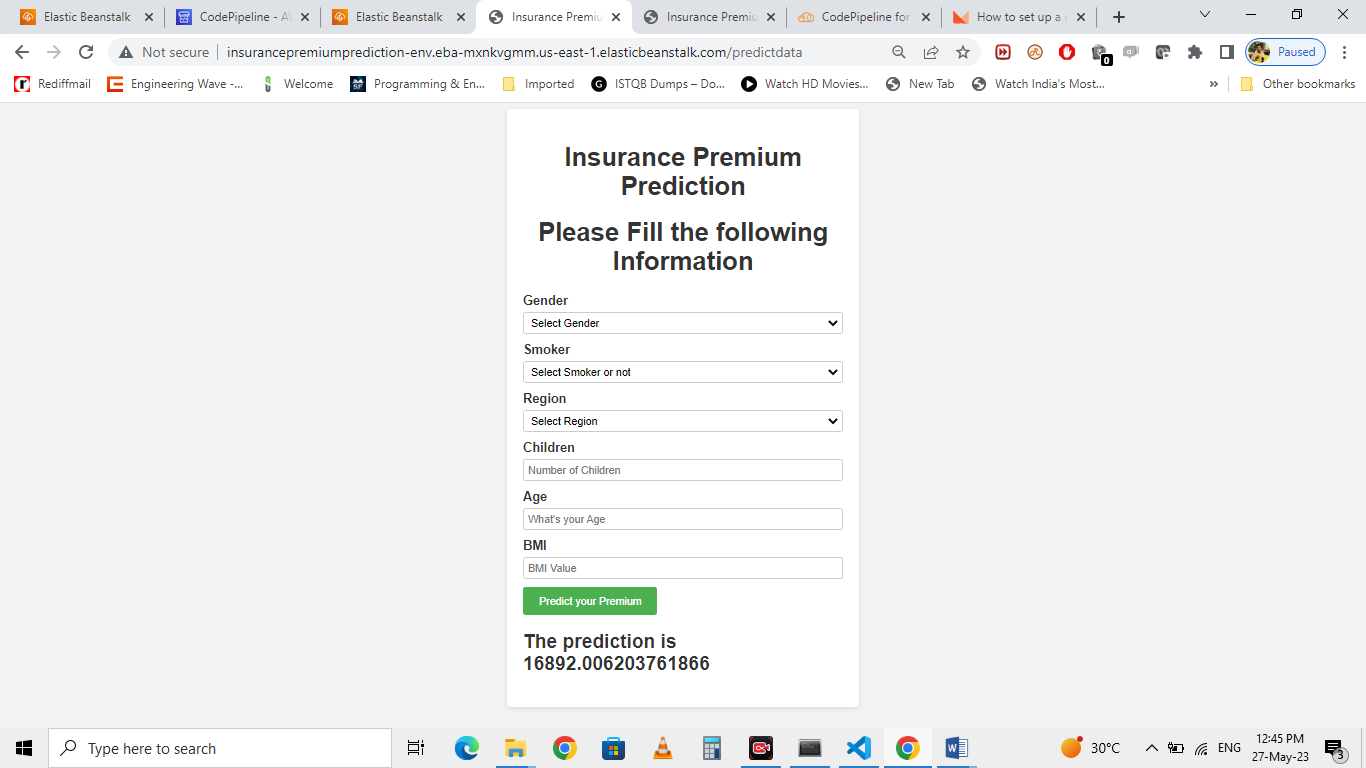
On the Add deploy stage page, set AWS Elastic Beanstalk as the Deploy provider. In Application name, enter or choose the name of an existing Elastic Beanstalk application. In Environment name, enter an environment for the application. Click Next.

On the Review page, review your pipeline configuration, and then click Create pipeline to create the pipeline.

* A web page has been created and deployed using flask







Application is deployed at URL:

**http://insurancepremiumprediction-env.eba-mxnkvgmm.us-east-1.elasticbeanstalk.com/predictdata**

1. **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.