**HIGH LEVEL DOCUMENT**

**by Himanshu balodi**

**Project Title Insurance Premium Prediction**

**Technologies Machine Learning Technology**

**Domain Insurance**

**Project Difficulties level Intermediate**

**.Document Version Control**

Date 25/4/2023

Version 1.0

Description Initial

Author Himanshu Balodi

**.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**

**1. Abstract.**

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. We used Xgboost classifier and determined the relation between person and these features. Which is have thyroid and how much effective the person health and what kind of treatment doctor’s gives .We trained the system using an 80-20 split and achieved an accuracy of 99%.

**2. Introduction**

**1.1 Why High-Level Design Document?**

As a machine learning practitioner, it is easy to overlook some of the many difficulties associated with building machine learning models while starting any particular project. Could the data be biased? What metric performance would be enough? Am I solving the right problem in the first place? Will the project be ready in time?

Training and building the Model is often the easiest part of the ML project. The main difficulty can be something else, for example,

>> Am I asking the correct questions to the data?

>> Do I’ve enough data to answer the questions?

>> Am I solving the right problem statement altogether?

>> Do I really have the clarity about the problem statement? Etc.

In worst scenario, all this can lead to a creation of a Model which will never see the production server. Luckily, we can prevent this unfortunate scenario by creating a design doc or HLD at the first place.

**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
  + Re usability
  + Application compatibility
  + Resource utilization
  + Serviceability

**2.1 Scope**

Design documents can bring the following benefits to machine learning projects:

1. It forces the author to organize their thoughts and get feedback on their ideas.

2. A design doc greatly improves communication with other teams.

3. It clarifies the scope of the project, highlights expectations and risks, and makes sure all the important questions about the project get answered.

**3. Description.**

1. **Problem Perspective.** The amount of the premium for a health insurance policy depends from person to person, as many factors affect the amount of the premium for a health insurance policy. Let’s say age, a young person is very less likely to have major health problems compared to an older person. Thus, treating an older person will be expensive compared to a young one. That is why an older person is required to pay a high premium compared to a young person.
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. We have to build a solution that should able to predict the premium of the

personal for health insurance

1. **Proposed Solution.**

Approach: The classical machine learning tasks like Data Exploration, Data Cleaning , Feature Engineering, Model Building and Model Testing. Try out different machine learning algorithms that’s best fit for the above case.

Some Famous Algorithms: - Multiple Linear Regression, Decision tree Regression and Gradient Boosting, Decision tree, Regression

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 3.10 ( 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.
* Flask 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. Accuracy of the prediction totally depends on continuous availability of updated data and retraining of the model with updated data. The predicted disease is an approximate amount which may vary from the current disease.

**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 Flask web app has been created as well and it should be accessible to every user who is connected to the internet.

**8. Design Flow.**   




**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.

1. **Performance Evaluation.**
2. **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 app 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 Ec2 instances. The following are the steps to deploy the model on the AWS platform:

* Create an AWS account
* Create an ECR
* Create S3 bucket
* Create an EC2 instance
* Edit security group
* Connect to an EC2 instance
* Install Docker
* Add the runner in the GitHub
* Add all the secret keys in the GitHub
* In the GitHub actions, run the continuous delivery and deployment workflow once after starting the runner in the ec2 instance
* Start the instance & locate the Docker run.sh file for to initiate the “Runner” to pick the job.
* Use Apache airflow to  monitor the model and perform batch prediction
* A web app has been created and deployed using Flask



