iNeuron

High Level Design (HLD)

Insurance Premium Predictor

Revision Number: 1.0

Last date of revision : 26/01/2022

iNeuron

Document Version Control

Date Issued	Version	Description	Author
26/01/2022	1	Initial HLD - V1.0	Naman Kumar

iNeuron

Table of Contents

Abstract

The Insurance market has seemingly increased in the recent years providing more coverage with a variety of Premium Plans. These new and improved Insurance Premium Plans often cover for risks that individuals may not have to worry about. The price of each plan also varies vastly as different individuals are at various risks and don't just share a common threat. So it becomes increasingly important to find a way to understand how much should a person pay for his/her Insurance Premium Plan. Based on this, individuals can then understand the risk coverage each plan has to offer and buy the plan that best suits them.

This work discusses the implementation of Machine Learning to estimate the price an individual should spend on their Insurance Premium based on their health condition.

iNeuron

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 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 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

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

iNeuron

2 General Description

2.1 Product Perspective

The insurance Premium Predictor is a Machine-Learning based regression model which will help us to get an estimate on how much should we spend on our Insurance Premium Plan.

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 proposed here is an IPP (Insurance Premium Predictor) using Machine Learning Regression algorithms can be implemented to perform above mention task. First, we predict values using the following Regression Algorithms :

1. Linear Regression
2. Polynomial Regression
3. Ridge Regression
4. Decision Tree Regression
5. Random Forest Regressor

Then we evaluate the performance of each algorithm using the following evaluation metrics :

- R2 score
- K-fold Cross Validation
- Root Mean Square Error (RMSE) value

The best performing algorithm is used in the final model. The trained Machine Learning model is serialized and wrapped in a static Website which is then deployed to a cloud service.

2.4 Technical Requirements

This document addresses the requirements for Insurance Premium and predicting how much should a person spend on their Premium Plan recommending the amount that the person should spend based on their health condition.
Machine Learning models built using Python programming language are used for this purpose.
Website that wraps the machine learning model is also built using the Python framework 'Flask', 'html, and 'css'.
The website is deployed to Cloud Service 'Heroku'.

2.5 Data Requirements

The dataset used for this project is loaded from here.

The dataset is a comma separated file that contains 1338 observations (rows) and 7 features (columns). The dataset contains 4 numerical features (age, bmi, children and expenses) and 3 nominal features (sex, smoker and region) that were converted into factors with numerical value designated for each level.

Insurance.csv file is obtained from the Machine Learning course website (Spring 2017) from Professor Eric Suess.

2.5 Tools Used

- Jupyter Notebook as IDE for Machine Learning model development
- Visual Studio Code as IDE for Website development
- Heroku is used for deployment of Website
- Front end development is done using HTML/CSS
- Back-end development is done using ‘Flask’ framework in Python
- GitHub is used as Version Control System
- For visualization of plots, ‘matplotlib’ and ‘seaborn’ python packages are used
- For numerical computations and statistical analysis, ‘pandas’, ‘numpy’, ‘statsmodels’ python packages are used
- For Machine Learning, ‘sklearn’ python package is used

2.6 Constraints

The Insurance Premium Predictor must be user friendly, as simple as possible and should not require to know any of the workings.

2.7 Assumptions

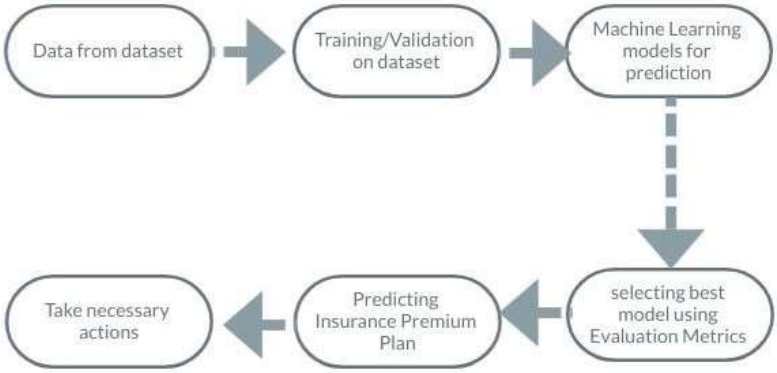
It is assumed that all the aspects of the project have the ability to work together and the user will input the values or select from given values as the model is expecting.

3 Design Details

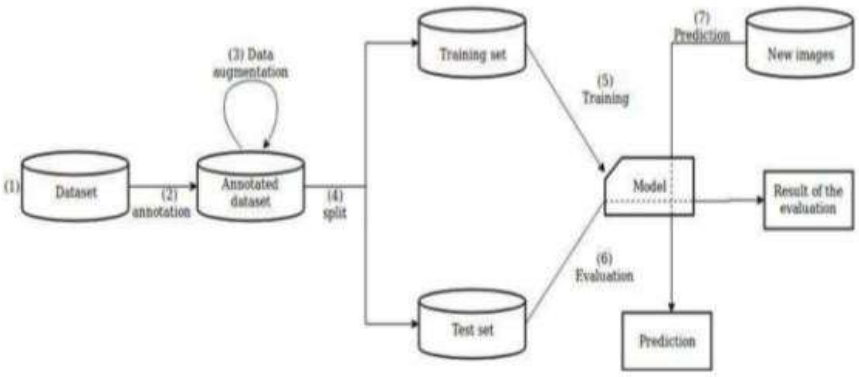
3.1 Process Flow

For estimating Insurance Premium worth, we will use a Machine Learning model. The type of Process Flow Diagram is as shown below

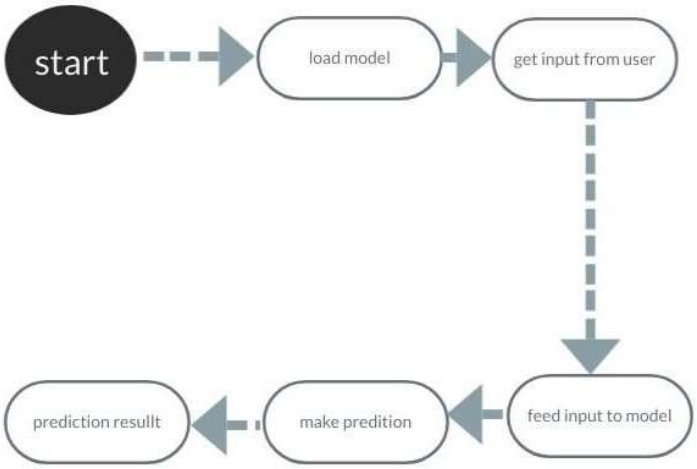
Proposed Methodology



3.1.1 Model Training and Evaluation



3.1.2 Deployment Process



3.2 Event Log

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

Initial Step-By-Step Description:

1. The System should be able to log each and every system flow.
2. Developer can choose logging method. You can choose database logging/ File logging as well.
3. 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 any error be encountered, an explanation will be displayed as to what went wrong and the user would be provided a button to redirect to the homepage to feed the correct inputs and predict the result again.

4 Performance

The IPP is used for predicting how much should a user spend on their Insurance Premium Plan based on their health condition so they can buy the Premium Plan that covers the best benefits in that price range, so it should be as accurate as possible. Also, model retraining is very important to improve the performance.

4.1 Reusability

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

4.2 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.

4.3 Resource Utilization

When any task is performed, it uses all the processing power available until that task is finished

4.4 Deployment

The static website is deployed on Heroku Cloud services

5 Website

The website will act as a User Interface between the user and the Software. It prompts the user for data and to choose an option from a given set of choices for categorical features and finally offers a submit button that redirects to a page displaying the predicted amount.

6 Conclusion

The designed IPP (Insurance Premium Predictor) will predict how much a person should spend on their Insurance Premium Plan so the user can find the best suiting Premium Plan in that price range.

7 References

- Machine Learning notes by Andrew Ng
- Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow by Aurélien Géron
- https://www.python.org/
- https://numpy.org/
- https://pandas.pydata.org/
- https://scikit-learn.org/stable/
- https://flask.palletsprojects.com/en/2.0.x/