Architecture Design

Insurance Premium Prediction

Document Version Control

Date	Version	Description	Author
10 – 06 - 2023	0.1	Abstract Introduction Architecture	Vijay shinde
12 – 06 - 2023	0.1	Architecture Design	Vijay shinde

Contents

Abstract

The primary objective is to provide individuals with an estimate of the required coverage based on their specific health situation. This allows customers to collaborate with any health insurance carrier and explore their plans and benefits, all while keeping the projected cost from our study in mind. The prediction of premiums takes into account variables such as age, sex, BMI, number of children, smoking habits, and living region.

By offering this estimation, individuals can focus more on the health-related aspects of an insurance policy, rather than getting caught up in less impactful elements. This approach helps them make informed decisions and prioritize their health needs when selecting an insurance policy.

Introduction

Why this Architecture Design Document?

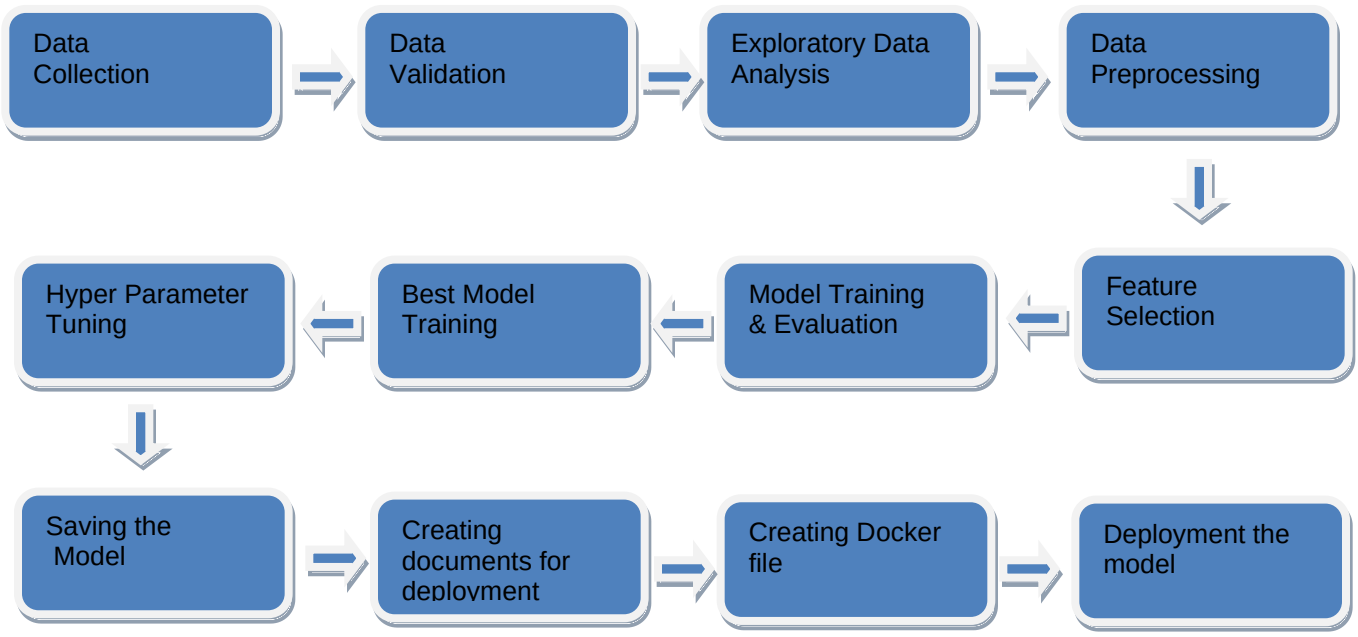
The main objective of the Architecture design documentation is to provide the internal logic understanding of the Insurance Premium Prediction code. The Architecture design documentation is designed in such a way that the programmer can directly code after reading each module description in the documentation.

The Architecture Design Document (ADD) serves as a detailed blueprint for the overall architecture of a system or project. It outlines the structure, components, interactions, and behavior of the system, providing a comprehensive understanding of how different elements of the system work together. The purpose of the ADD is to ensure that the system architecture is well-designed, scalable, maintainable, and meets the specified requirements.

The ADD document serves several important functions

System Understanding
Design Guidelines
Scalability and Maintainability
Risk Mitigation
Collaboration and Communication

Architecture



Architecture Design

Data Collection

The data for this project is sourced from the Kaggle Dataset available at the following URL: https://www.kaggle.com/datasets/noordeen/insurance-prediction.

Data Description

The Insurance Premium dataset is publicly available on Kaggle. The dataset consists of one CSV file named "insurance.csv." It contains 1338 rows of data, including information such as age, BMI, number of children, and expenses

Data Pre-processing

- The dataset was examined to obtain information about the column data types, ensuring the correct data types are assigned to each column.
- Null values were checked for, as they can impact the accuracy of the model.
- One-Hot encoding was performed on specific columns to convert categorical variables into numerical representations.
- The distribution of the columns was analyzed to understand their importance and influence in the dataset.
- With these preparations completed, the dataset is now ready for training a machine learning model.

Modeling Process

After pre-processing the data, the next step is to visualize the data to gain insights and identify patterns. This can be done using various visualization techniques such as histograms, scatter plots, or box plots.

Following that, the dataset is split into two parts: training data and test data. The training data is used to train the machine learning models, while the test data is used to evaluate their performance and generalization.

Different machine learning models, including Linear Regression, Random Forest Regressor, and Decision Tree Regressor, are employed to predict the Insurance Premium Price. Each model has its own algorithms and techniques for making predictions based on the training data.

The models are trained using the training data, and their performance is evaluated using metrics such as mean squared error, mean absolute error, or R-squared. This evaluation helps in comparing the models and selecting the one that performs the best in predicting the Insurance Premium Price.

Once a model is selected, it can be used to make predictions on new, unseen data to estimate the Insurance Premium Price for individuals based on their relevant features.

UI Integration

Once the Streamlit files are prepared, they are integrated with the created machine learning model. This integration involves importing the model and incorporating it into the Streamlit application, allowing users to interact with the model through the user interface.

To ensure the application functions correctly, all the required files, including the machine learning model file, pre-processing scripts, and any other dependencies, are integrated into the app.py file. This consolidation ensures that the application has access to all the necessary components.

After integrating the files, the application is tested locally to verify its functionality and to address any potential issues or bugs. Local testing allows developers to validate that the application is running smoothly before deploying it to a production environment.

Data from User

The data provided by the user is retrieved from the Streamlit web page that has been created. Users interact with the web page by entering their input or selecting options, and the Streamlit application captures and processes this user input to perform further operations such as data validation, preprocessing, or model prediction

Data Validation

The user-provided data is processed and validated by the app.py file. This validation step ensures that the data meets the required criteria and is suitable for further processing. Once the data is validated, it is sent to the prepared model for prediction. The model utilizes the validated data as input to make predictions based on its trained knowledge and set of rules

Rendering the Results

After the data is sent for prediction to the model, the predicted results are obtained. These predicted results are then rendered or displayed on the web page for the user to view.

Deployment

The tested model is deployed to Streamlit Cloud, enabling users to access the project from any internet-connected device. Streamlit Cloud provides a platform for hosting and serving Streamlit applications, making it accessible to users without requiring them to install any dependencies or run the application locally.