Architecture Design
Insurance Premium Prediction

Revision Number – 1.0

Last Date of Revision: 17/05/2023

Siddharth Tyagi

Document Version Control

Date	Version	Description	Author
17/05/2023	1.0	First Draft	Siddharth Tyagi

Contents

1.Abstract

The Architecture Design Document for Insurance Premium Prediction outlines the technical specification and proposed solution for predicting insurance premiums based on various factors. It covers the dataset overview, predicting insurance premiums, logging, technology stack, workflow, and key performance indicators (KPIs) to ensure successful implementation.

2.Introduction

Why Architecture Design Document?

An Architecture Design Document is essential to ensure the successful development and deployment of a project. It helps to define the technical specification, technology stack, and proposed solution for a project.

Scope

The scope of this Architecture Design Document is to provide technical specifications and a proposed solution for predicting insurance premiums based on various factors.

Definition

Insurance Premium Prediction: The process of using machine learning algorithms to predict insurance premiums based on various factors such as age, driving record, and vehicle type.

Dataset: A collection of data used to train and test the machine learning model.

3.Technical Specification

Dataset Overview

The dataset used for this project contains information on the policyholder's age, driving record, vehicle type, and other relevant factors. It is split into training and testing datasets to train and test the machine learning model.

Predicting the Insurance Premium

The insurance premium is predicted using a machine learning algorithm, which is trained on the training dataset. The model is then evaluated on the testing dataset to ensure that it performs well on unseen data.

Logging

Logs are used to track the performance of the machine learning model during training and testing. The logs include metrics such as accuracy, precision, and recall to evaluate the model's performance.

4.Technology Stack

The technology stack used for this project includes:

Python: The programming language used for data preprocessing and model development.

Scikit-learn: A machine learning library used for model development and evaluation.

Streamlit: Streamlit is an open-source Python library that allows users to create interactive web apps to predict insurance premiums and making it easy to share and visualize data.

MongoDB: A relational database used to store the dataset.

5.Proposed Solution

The proposed solution for predicting insurance premiums involves the following steps:

Data preprocessing: The dataset is cleaned and preprocessed to remove missing values and outliers.

Model development: A machine learning algorithm is trained on the preprocessed dataset to predict insurance premiums.

Model evaluation: The model is evaluated on the testing dataset to ensure that it performs well on unseen data.

Web application: A web application is created using Streamlit to allow users to input their information and receive a predicted insurance premium.

6.Workflow

The workflow for this project is as follows:

Data collection: Collect the dataset containing information on policyholders and their insurance premiums.

Data preprocessing: Clean and preprocess the dataset to remove missing values and outliers.

Model development: Train a machine learning algorithm on the preprocessed dataset to predict insurance premiums.

Model evaluation: Evaluate the model's performance on the testing dataset to ensure that it performs well on unseen data.

Web application development: Develop a web application using Flask to allow users to input their information and receive a predicted insurance premium.

Deployment: Deploy the web application to a server for public use.

7.Key Performance Indicators (KPIs)

The following KPIs will be used to measure the success of the project:

Model accuracy: The accuracy of the machine learning model in predicting insurance premiums.

User engagement: The number of users who interact with the web application.

Response time: The time it takes for the web application to respond to user requests.

Server uptime: The percentage of time the server hosting the web application is operational.