Low Level Design

Insurance Premium Prediction

Written By	Joji Samuel
Document Version	0.3
Last Revised Date	29 – Nov -2023

Document Control

Change Record:

Version	Date	Author	Comments
0.1	20 – Nov -2023	Joji Samuel	Introduction & Architecture defined
0.2	27 – Nov -2023	Joji Samuel	Architecture & Architecture Description appended and updated
0.3	28 – Nov -2023	Joji Samuel	Unit Test Cases defined and appended

Reviews:

Version	Date	Author	Comments
0.2	30 – Nov -2023	Joji Samuel	Document Content , Version Control and Unit Test Cases to be added

Approval Status:

Version	Date	Author	Comments

Contents

1. Introduction

1.1 Project Overview

The Insurance Premium Predictor is a machine learning-based system designed to estimate insurance premiums for individuals based on various input factors. This project aims to leverage predictive modeling to assist insurance providers in determining appropriate premium rates, enhancing accuracy, and optimizing risk assessment.

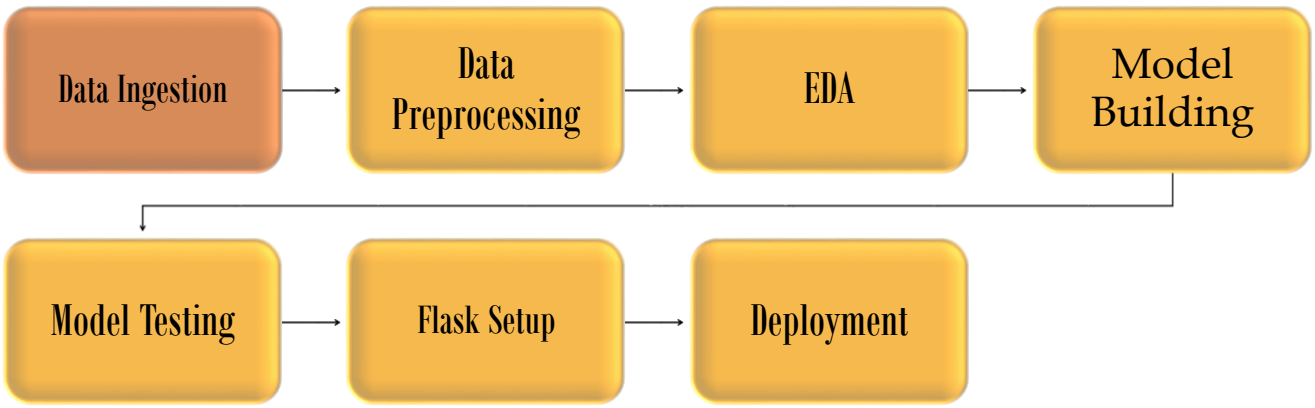
1.2 Purpose of the LLD Document

The purpose of this Low-Level Design (LLD) document is to provide a detailed blueprint for the implementation of the Insurance Premium Predictor. It serves as a guide for developers, outlining the architecture, data processing, model design, and other technical aspects of the system. The LLD ensures a comprehensive understanding of the system's components, fostering consistency and efficiency during the development phase.

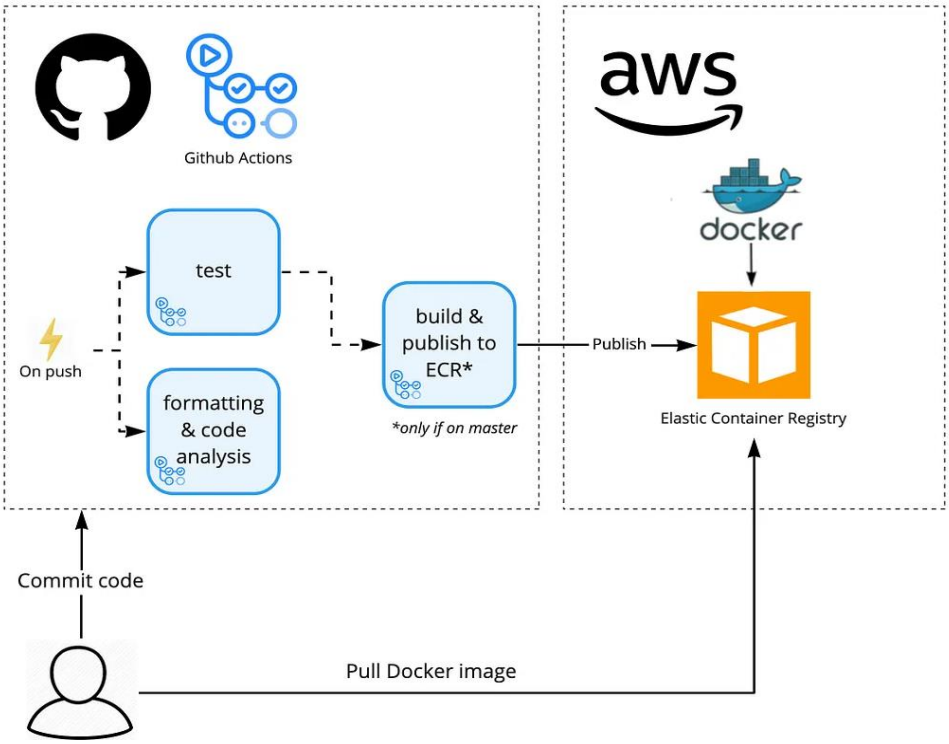
1.3 Scope of the Project

The scope of the Insurance Premium Predictor encompasses the development of a robust machine learning model capable of accurately predicting insurance premiums. The system will process input data, including age, sex, BMI, number of children, smoking status, and region, to generate reliable premium estimates. The project also includes considerations for scalability, maintainability, and potential integration with external systems through APIs.

2. Architecture



Deployment



3.0 Architecture Description

3.1 Data Description

Insurance.csv file was obtained from the Machine Learning course website (Spring 2017) from Professor Eric Suess - CSU East Bay College of Science.

The insurance.csv dataset 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.

3.2 Data Pre-processing

Data pre-processing for the insurance premium prediction project involves several key steps. The process begins with identifying and addressing missing values through imputation or removal. Categorical variables like sex, smoker, and region are encoded into numerical formats, ensuring consistency across datasets. These pre-processing steps collectively contribute to enhancing the dataset's quality and preparing it for effective machine learning model training.

3.3 Exploratory Data Analysis

The Exploratory Data Analysis (EDA) for the insurance premium prediction project involves a systematic exploration of the dataset. Starting with data loading and basic summary statistics, the process includes univariate analysis to understand feature distributions and identify outliers, bivariate analysis to investigate relationships with the target variable (insurance premiums), and categorical variable analysis. Outlier detection and handling missing values are crucial steps, followed by data transformation and visualizations to extract insights and patterns. Interactive visualizations may be used for a dynamic exploration experience. This EDA process ensures a thorough understanding of the dataset, providing a solid foundation for subsequent stages in the machine learning project.

3.4 Model Building

The model building process for the insurance premium predictor ML project involves several key steps. Begin by selecting a suitable machine learning algorithm for regression, such as linear regression or decision trees. Split the pre-processed dataset into training and testing sets, then train the model on the training data. Fine-tune hyperparameters through techniques like grid search and implement cross-validation for generalizability assessment. Evaluate the model's performance on the testing set using regression metrics like MAE and MSE. Interpret the model's predictions and assess feature importance. Conduct error analysis and deploy the model for production if performance criteria are met. Establish monitoring and maintenance procedures, and document the entire model building process, including algorithm selection, hyperparameter tuning, and evaluation results. Continuous monitoring and potential adjustments ensure the model's accuracy and reliability over time.

3.5 Model Evaluation

Model evaluation is a critical phase in the insurance premium predictor ML project. Use the trained model to make predictions on the testing set and calculate the chosen metrics to assess performance. Analyze results for insights, conduct error analysis, and visualize predicted versus actual values. Optionally, implement cross-validation for more robust estimates. Compare models if applicable and document the evaluation findings, including any necessary iterative refinements. This process informs the selection of the final model based on its performance, interpretability, and suitability for predicting insurance premiums accurately. Continuous monitoring and documentation are crucial for maintaining the model's effectiveness over time.

3.6 UserInterface

A web UI was created using Flask, a lightweight Python web framework. The Flask application, defined in the app.py file, includes a route for the home page that renders an HTML template (index.html). The HTML file, located in the templates folder, provides a basic structure. The Flask app is run with the python app.py command, enabling access to the initial page in a web browser. A form for user input was added to the HTML file, including fields like age, sex, and a "Predict" button. The app.py file was updated to handle form submissions by creating a new route (/predict) and extracting form data using request.form.

3.7 Deployment

3.7.1 Continuous Integration/Continuous Deployment (CI/CD):

To deploy a machine learning model using a CI/CD pipeline, the process begins by serializing the trained model with Joblib and creating a Dockerfile to encapsulate the model and its dependencies. This setup is then stored in a GitHub repository, and a GitHub Actions workflow is established for CI/CD automation. AWS ECR is employed to securely manage Docker images. GitHub Actions, utilizing secrets for secure credentials storage, builds the Docker image, tags it, and pushes it to the ECR repository. An EC2 instance is configured to serve as the deployment environment, equipped with Docker. Deployment scripts are crafted to pull the Docker image from ECR and run it on the EC2 instance. The CI/CD workflow ensures automated deployment upon repository changes, including testing and validation steps. Monitoring and logging mechanisms are implemented for performance tracking, and a rollback plan is devised for handling deployment issues. Security measures, such as IAM roles and secure credential storage, are integrated, and scalability considerations are factored into the EC2 and Docker setup. This end-to-end process streamlines deployment, promotes consistency, and facilitates continuous improvement in the machine learning model's deployment lifecycle.

4.0 Unit Test Cases

Test Case ID	Test Description	Input	Expected Output
TC_UI_001	Age input validation: valid numeric value	Age = 30	Age input is accepted without errors.
TC_UI_002	Age input validation: invalid input	Age = "abc"	Error message is displayed for invalid age.
TC_UI_003	BMI input validation: valid numeric value	BMI = 25.5	BMI input is accepted without errors.
TC_UI_004	BMI input validation: invalid input	BMI = "xyz"	Error message is displayed for invalid BMI.
TC_UI_005	"Smoker" checkbox selection	Select "Smoker" checkbox	Checkbox is selected.
TC_UI_006	Children input validation: valid numeric value	Children = 2	Children input is accepted without errors.

Test Case ID	Test Description	Input	Expected Output
TC_UI_007	Children input validation: invalid input	Children = -1	Error message is displayed for invalid input.
TC_UI_008	Region dropdown selection	Select "Southwest" from the Region dropdown	"Southwest" is selected without errors.
TC_UI_009	"Predict" button functionality	Complete all required fields and click "Predict"	Prediction is initiated, and the result is displayed.
TC_UI_010	Missing information validation	Leave one or more required fields empty and click "Predict"	Error message indicating missing information is displayed.