LOW LEVEL DESIGN(LLD)

INSURANCE PREMIUM PREDICTION

Dilip Kumar Panda

Document Version Control

Date Issued	Version	Description	Author
11/6/23	1.0	Introduction And Architecture	Dilip Kumar Panda
13/6/23	2.0	Architecture Description	Dilip Kumar Panda

Contents

Introduction

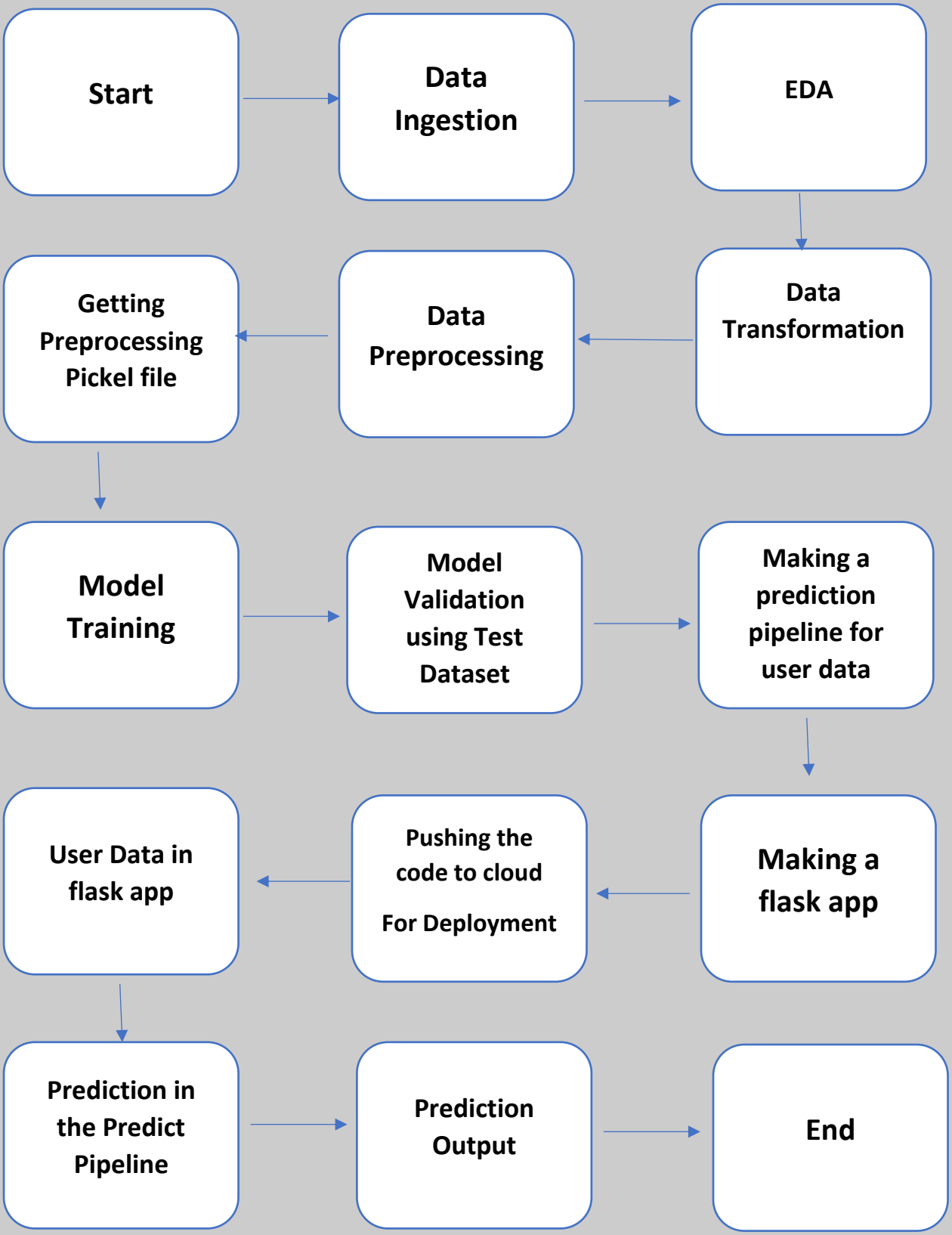
The Low-Level Design (LLD) for Insurance Premium Prediction focuses on the detailed design and implementation aspects of the predictive model. This design document serves as a guide for developers, outlining the specific components, modules, and algorithms involved in building the system. In this LLD, we will delve into the technical details of the system, including the architecture, data processing flow, model selection, and training methodologies. The document will also outline the integration of external data sources, model deployment strategies, and performance metrics. By following this LLD, the development team can proceed with the implementation phase, ensuring a systematic and efficient construction of the Insurance Premium Prediction system. The detailed design considerations and technical specifications provided in this document will enable the team to build a robust and accurate predictive model, contributing to improved pricing strategies and customer satisfaction in the insurance industry.

Scope

It focuses on the internal structure of the system, including component design, algorithms, data processing, interfaces, and error handling .The LLD defines the architecture and interactions between different modules, specifying how data flows and control is handled within the system. It guides developers during the implementation phase, ensuring a systematic and efficient construction of the system.

Additionally, the LLD covers algorithm selection, data processing and storage, interfaces and APIs, error handling, and integration and deployment strategies. It provides a comprehensive scope for developers to understand the technical intricacies and requirements of building a robust Insurance Premium Prediction system.

Architecture



Architecture Description

Data Description

Data was taken from Kaggle having 6 independent columns and one target column. There were about 1300 rows of data for building the model.

EDA

Eda was performed over the data set to know relationship between all the variable and the target column, relationship between independent features, to know about missing value and duplicate value and to visualize the data.

Data Transformation

Standardisation of data development of different pipelines for numerical and categorical column and dividing the data to train and test set was done.

Data Preprocessing

Null value handling, applying column Transform to different pipelines and finally getting a pre-processor pickle file.

Model Training

Different machine learning Regression model was tried over the train data set and r-square value was found out from different model.

Model Validation

The trained model was used to predict the test dataset and similarly r-squared value was found out. Out of all the models which has the highest accuracy value that model was chosen as the best model.

Prediction Pipeline

A prediction pipeline was made to connect the user input data to all the transformation pickle file and model training pickle file.

Flask App

A flask app was made to take user input data for prediction.

Deployment

The whole code was pushed to github and then deployment was done using Azure and github actions.

Prediction

User data was taken using flask app and the data was passed over to prediction pipeline and finally prediction was made and output was shown over the same flask app to the user.

Unit Test Cases

Test case description	Prerequisites	Expected results
Verify whether the application URL is accessible to the user or not	Application URL should be defined	The application URL should be accessible to the user.
Verify whether the Application loads completely for the user when the URL is accessed	1. Application URL is accessible 2. Application is deployed	The Application should load completely for the user when the URL is accessed.
Verify whether user is able to see input fields on accessing	Application is accessible	User should be able to see input fields on logging in
Verify whether user is able to edit all input fields	Application is accessible	User should be able to edit all input fields
Verify whether user gets Submit button to submit the inputs	Application is accessible	User should get Submit button to submit the inputs
Verify whether user get the prediction as the output.	Application is accessible	User should get the desired prediction value.

Verify whether the prediction results are in accordance to the selections user made	Application is accessible	The prediction results should be in accordance to the selections user made.