LOW LEVEL DESIGN

Insurance Premium Prediction

Created by Vikram Singh

GitHub Profile : VkasRajpurohit (github.com)

Document Version Control :

Date issued	Version	Description	Author
Jan 18th, 2022	1	Initial LLD V1.0	VIKRAM SINGH

TABLE OF CONTENTS

ABSTRACT

The purpose of this LLD (Low Level Design) document is to give the internal logical design of the actual program code for Insurance Premium Prediction. Low-level design is created based on the high-level design. LLD describes the class diagrams with the methods and relations between classes and program specs.

It describes the modules so that the programmer can directly code the program from the document. The code can be developed directly from the low-level design document with minimal debugging and testing. Other advantages include lower cost and easier maintenance.

Overview:

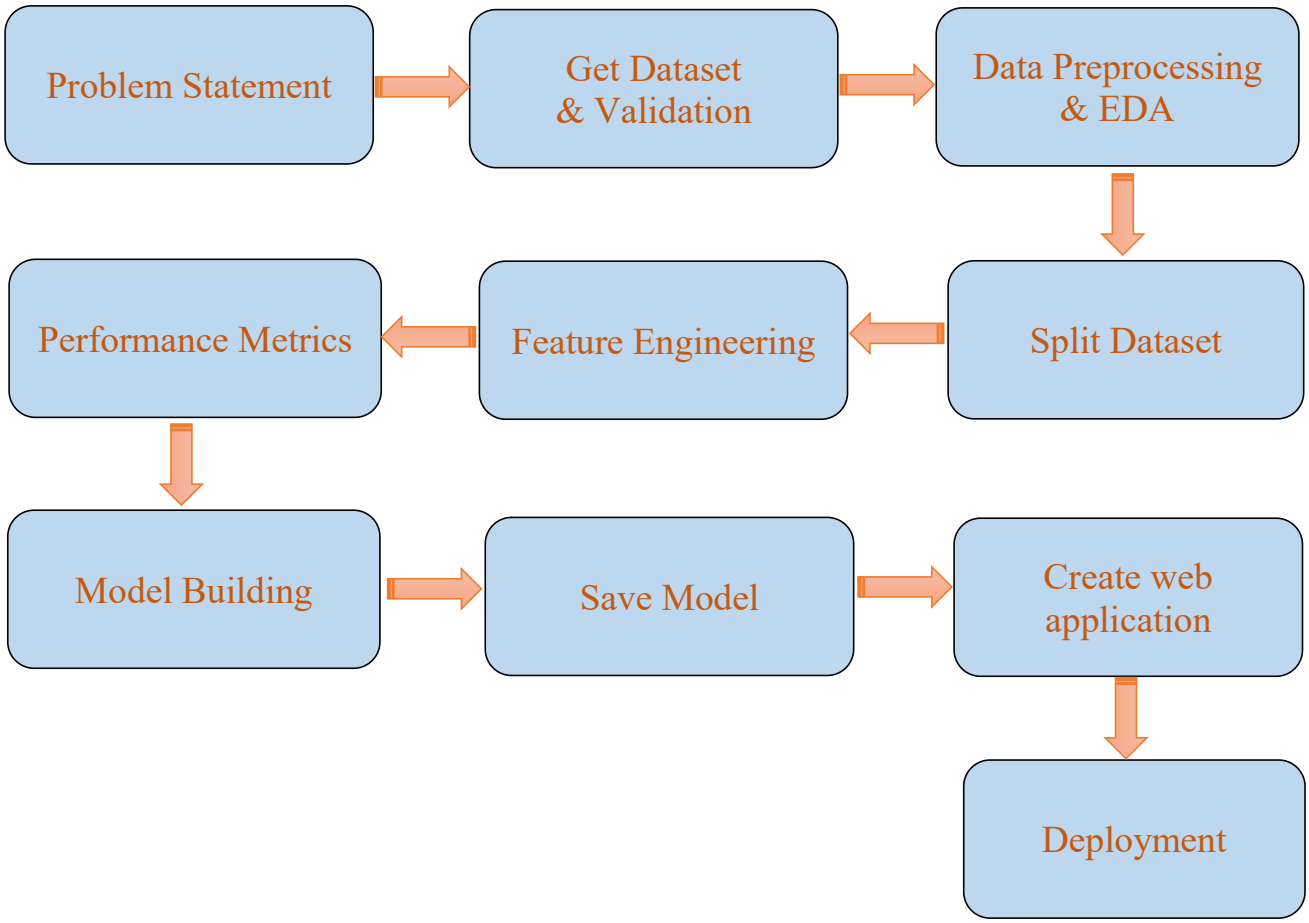
Title	:	Insurance Premium Prediction
Domain	:	Insurance
Tools & Technology	:	Python | Data-Preprocessing | EDA | Feature Engineering | Feature Transformation & Scaling | Machine Learning | Flask-API | HTML | GitHub | Heroku | AWS
ML Algorithms	:	Linear Regressor | SVM Regressor | DT Regressor | RF Regressor | Gradient Boosting Regressor | Stacking Regressor
IDE	:	PyCharm | Google Colab

1. Logging:

Model should be able to log every activity done by the user.

- Logging is mandatory for easy debug issues.
- The system should be able to log each and every system flow.

2. Process Flow Detail:



2.1 Problem Statement:

Build a solution that should be able to predict the premium of the an individual health insurance based on given features in dataset.

2.2 Get Dataset & Validation:

For the Problem statement data collected via Kaggle platform and stored in local system for later use.

2.3 Data Preprocessing & EDA:

In this data preprocessing & EDA will be handled for the entire data. In this data preprocessing & EDA need to check below details -

- Basic information about the data
- Basic statistics information
- Check nominal/categorical features
- Missing values
- Check correlation
- Correlation with dummy variables of categorical features
- Analysis of each features separately
- And others techniques as per the dataset.

2.4 Split Dataset:

Split the dataset into train data & test data, it is a good practice to split the dataset before Feature Engineering to avoid the data leakage problem.

2.5 Feature Engineering:

In Feature Engineering part below points must be considered -

- Handle missing values.
- Handle **categorical features,** categorical features can be divided into **Nominal & Ordinal categorical features**.
  - Nominal Categorical Features can be handled by One-hot-encoding with dummy variable trap, One-hot-encoding with many categorical variables, Mean-encoding etc.
  - Ordinal Categorical Features can be handled by Label encoding, Target guided ordinal encoding etc.
- Handle feature transformation & scaling (perform QQ plot, Standardization).

2.6 Performance Metrics:

As per the problem statement, need to predict Expenses i.e. labeled data, hence can go with the supervised machine learning techniques. It is a Regression problem. For Regression problem performance metrics are -

- MSE, RMSE, MAE
- R2, Adjusted R2

Use suitable performance metrics.

2.7 Model Building:

Built models for Regression problem - Linear Regressor, SVM Regressor, Decision Tree Regressor and Ensemble techniques models i.e. Random Forest Regressor, Gradient Boosting Regressor & StackingRegressor.

- Check **Cross_Validation_Score** to handle bias-variance problem.
- Select the best model and try to improve performance by **Hyper-parameters tuning.**
- Show Final Result **- Scatter-plot : y_test vs predictions**

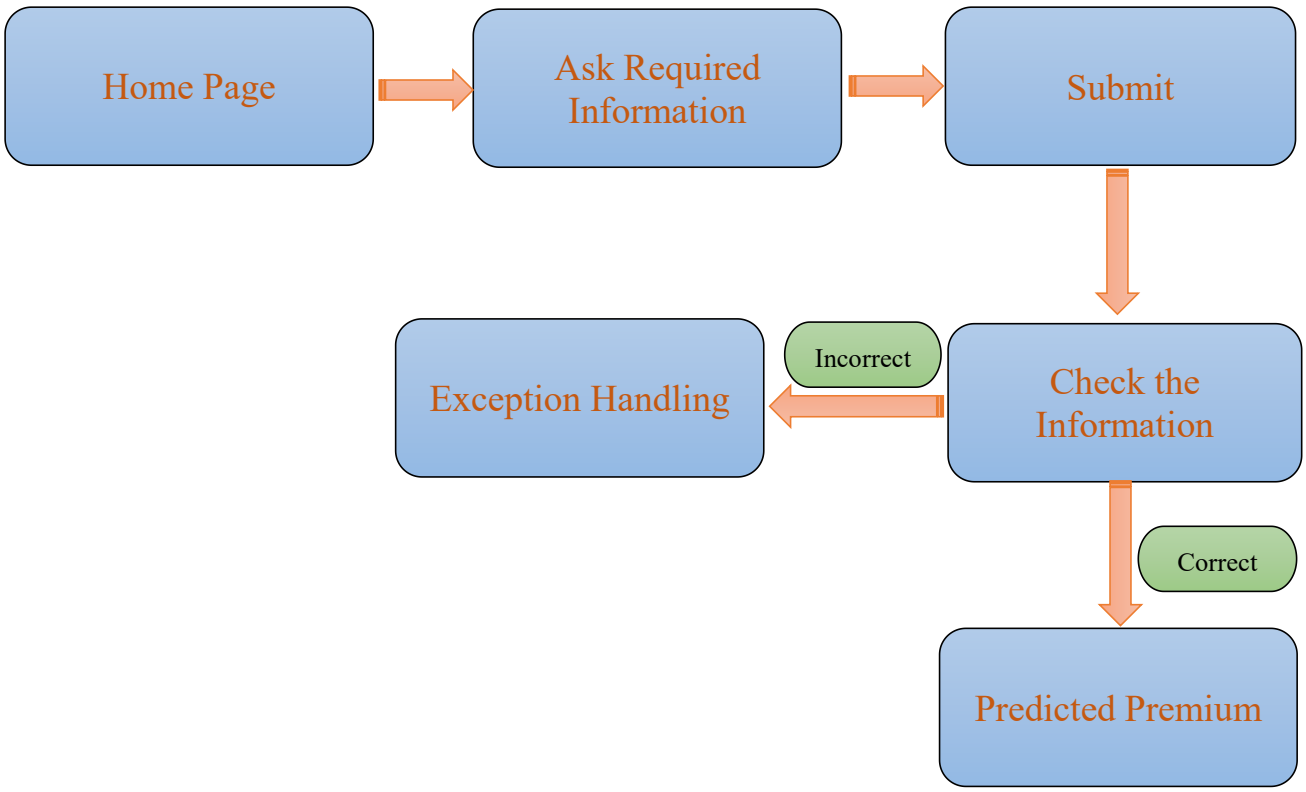
2.8 Save Model:

Save the **best_tuned_model** using the pickle module, it is also called serialization. Saved model can be used later. Now the training part has been completed.

2.9 Create web application:

Create a User Interface web application using Flask-API & HTML. Model prediction part will be done here. Need to do all the steps for prediction part as done for the training part i.e. feature engineering, feature transformation & scaling steps, and use the **best_tuned_model** for prediction of the user input data.

User I/O workflow:



Home page is the landing page, it will ask for the user input. User will enter the required information & hit submit/Get Premium. If user entered the information correctly then show result as predicted premium, however if the entered information is incorrect then handle accordingly.

2.10 Deployment:

Now model is ready for productionisation. For model deployment use **Heroku** or **AWS.**

---------- **End of LLD** ----------