Low Level Design (LLD)

Insurance Premium Prediction

Project By:

Siridi Nath Peddina

Low Level Design (LLD)

Document Version Control

Date	Version	Description	Author
07-03-2023	1.0	Abstract, Introduction Architecture	Siridi Nath
08-03-2023	1.1	Data Preprocessing	Siridi Nath
09-03-2023	1.2	Deployment Unit Test	Siridi Nath

Low Level Design (LLD)

Contents

Low Level Design (LLD)

Abstract

This project represents a machine learning-based health insurance prediction system. Recently, many attempts have been made to solve this problem, as after the Covid-19 pandemic, health insurance has become one of the most prominent areas of research. We have used the USA's medical cost personal dataset from Kaggle, having 1338 entries. Features in the dataset that are used for the prediction of insurance cost include Age, Gender, BMI, Smoking Habit, number of children etc. We used Xgboost regression and determined the relation between price and these features. We trained the system using a 80-20 split and achieved an accuracy of 85%.

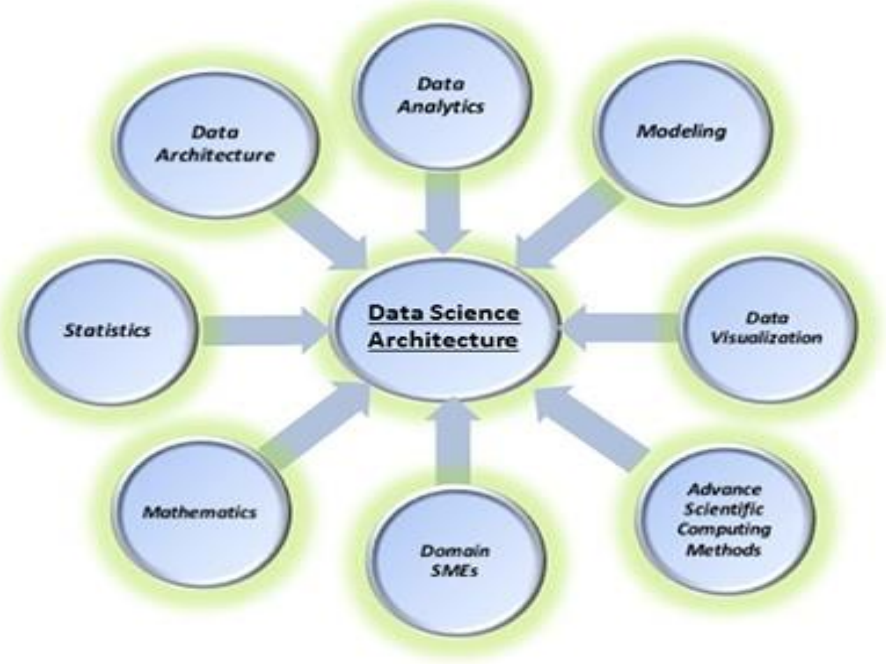
Low Level Design (LLD)

1 Introduction

1.1 Why this Low-Level Design Document?

The main purpose of this LLD documentation is to feature the required details of the project and supply the outline of the machine learning model and also the written code. This additionally provides the careful description on however the complete project has been designed end-to-end.

1.2 Architecture



2. Architecture Design

For this project, we have implemented the machine learning life cycle to create a basic web application which will predict the flight prices by applying machine learning algorithms to historical flight data using python libraries like Pandas, NumPy, Matplotlib, seaborn and sklearn.

Low Level Design (LLD)

2.1 Data Gathering

The data for the current project is being gathered from Kaggle dataset, the link to the data is: *https://www.kaggle.com/datasets/noordeen/insurance-premium-prediction*

2.2 Tool Used

- Python 3.9 is employed because of the programming language and frameworks like numpy, pandas, sklearn and alternative modules for building the model.
- Visual studio is employed as an IDE.
- For visualizations seaborn and components of matplotlib are getting used
- For information assortment prophetess info is getting used version management.
- Streamlit is employed for deployment.

2.3 Data Description

File Contain.

- Readme.txt

Insurance_main_dataset.csv - contains 1338 rows and 7 columns

Following is the description of features available in the dataset :-

- **age**: contains the age of the policyholders from various types of policies.
- **sex** : determines the Gender of the population taking a particular policy
- **bmi**: BMI being the most important factor in the domain of Health Insurance, depending upon which the insurance company decides the premium to be paid by the policyholder.
- **children:** It is taken as no. of beneficiaries added for a particular insurance of a policyholder.

- **smoker**: Like BMI, the smoking parameter is an important factor for calculating the premium of a particular policy for a policyholder.
- **region**: Various insurance companies divide their serving in the form of regions, which in turn decides the copay or any extra premium to be given by the policyholder.
- **expenses**: this is the total premium paid by the policyholder for a given given policy schedule

2.4 Import Data into Database

MongoDB was used for loading the dataset using Pandas Library was used for training and making the machine-learning model.

2.5 Export Data into Database

The data has been dumped to the MongoDB database.

2.6 Data Preprocessing

Steps performed in pre-processing are:

- All the necessary libraries were imported first such as Numpy, Pandas, Matplotlib, and Seaborn.
- Checking the basic profile of the dataset. To get a better understanding of the dataset.
    - Using Info method
    - Using Describe method
    - Checking for unique values of each column.
- Checking for null values, There are no null values present in our dataset.
- The categorical variable has been encoded with the help of a label encoder.
- After performing all the above steps, the dataset is ready and can be processed into the stage of modelling.

Low Level Design (LLD)

2.7 Modeling

- After this the data was split into 2 sets X and y. X contains all the columns except the target column in our case (expenses), and y contains only the Target column.
- Using train test split we first split the dataset into X_train, X_test, y_train, and y_test.
- Standard scaling has been used to bring the data on the same scale
- The following libraries were imported to create Regression models.
- Lazy predict has been used to choose the best model
- After Lazy to predict, XgBoost regressor has been chosen as a final algorithm to create the mode.

2.8 UI Integration

Apache Airflow can be used to monitor the model and predict the new batch dataset. A Streamlit webapp has been created to get the insurance premium amount based on certain inputs like age, BMI, sex, smoker etc.

2.9 Data from User

User can give the required input and get the insurance premium amount as a result in Streamlit webapp. Data from the user is retrieved using the batch file and using Apache Airflow our machine learning model to give the predicted result

Low Level Design (LLD)

2.10 Data Validation

The data provided by the user is then being processed by app.py file and validated. The validated data is then sent for the prediction.

2.11 Rendering Result

The data sent for the prediction is then rendered to the streamlit.

Low Level Design (LLD)

3. Deployment

After getting the model with the best accuracy we store that model in a file using the pickle module. The back-end of the application will be created using Streamlit Framework where API end-points such as GET and POST willbe created to perform operations related to fetching and displaying data on the front-end of the application.

Low Level Design (LLD)