Low Level Design (LLD)

Insurance Premium Prediction

Revision Number – 1.0

Last Date of Revision – 03/09/2023

Anuj Dhyani

Document Version Control

Date	Version	Description	Author
02-09-2023	1.0	Abstract, Introduction Architecture	Anuj Dhyani
03-09-2023	1.1	Data Preprocessing	Anuj Dhyani
03-09-2023	1.2	Deployment	Anuj Dhyani

Contents

Abstract

This Low-Level Design Document encapsulates .Our primary objective is to empower individuals to approach health insurance providers with confidence, armed with a clear understanding of the expected financial commitments tied to their selected coverage. By eliminating the uncertainty surrounding insurance premiums, we enable people to prioritize their health and well being over pricing complexities. This innovative approach shifts the focus away from the intricacies of policy pricing and toward a user-centric perspective that promotes informed decision-making. Ultimately, our goal is to enhance the overall insurance experience, making it more personalized and user-friendly. We aim to facilitate insurance choices that reflect individual needs and financial considerations, empowering individuals to invest in their health with clarity and assurance

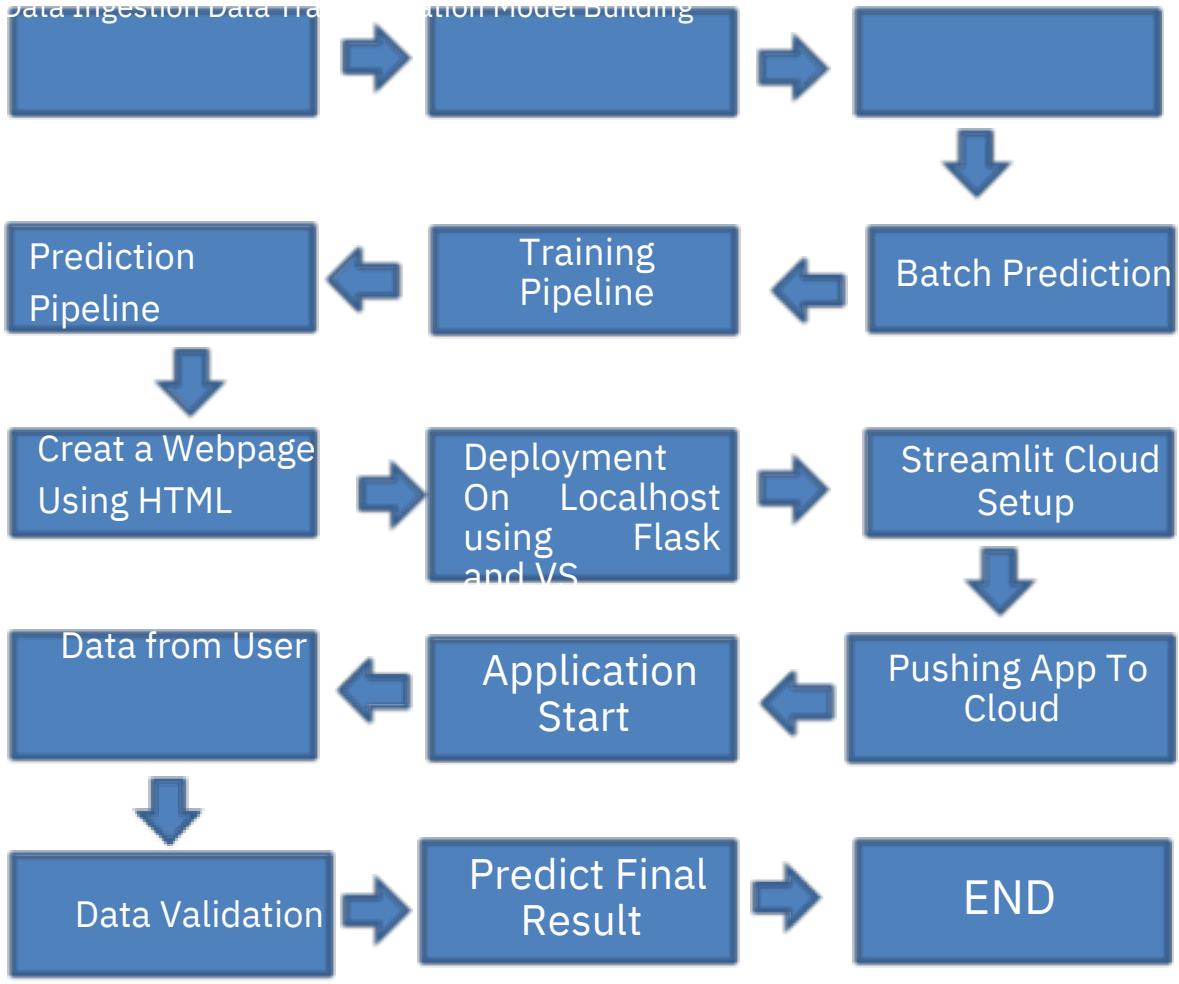
1Introduction

1.1Why 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

T

1.2 Architecture



2.Architecture Description

2.1. Data Gathering

The data for the current project is being gathered from the Kaggle dataset, which is available at the following link: Insurance Premium Prediction | Kaggle. This dataset serves as the primary source for our project's data analysis and premium prediction tasks.

2.2. Tool Used

• Python 3.8 is employed because the programming language and frame works like numpy, pandas, sklearn, Flask, Streamlit and alternative modules for building the model.

• Visual Studio Code is employed as IDE.

• For code versioning GitHub is used

• Localhost and Streamlit Cloud is employed for deployment

2.3 Data Description

We have train (1070) and test (268) data set, train data set has both input and output
Columns Are :

1.Age: Age of the insured individuals, a key factor in premium calculation due to its influence on health risk.

2.Sex: Gender of the insured individuals, impacting premiums based on gender-specific health risks.

3.BMI (Body Mass Index): Measure of body weight relative to height, influencing premiums based on health implications.

4.Children: Number of dependents covered, affecting policy costs due to family size.

5.Smoker: Smoking status (yes/no), a significant factor in premium pricing due to health risks associated with smoking.

6.Region: Geographic location of the insured, which can affect healthcare costs and insurance pricing.

7.Expenses: Actual medical expenses incurred, providing insight into healthcare costs and utilization for premium calculation.

variable to be predicted.

age	sex	bmi	children	smoker	region	expenses
19	female	27.9	0	yes	southwest	16884.92
18	male	33.8	1	no	southeast	1725.55
28	male	33	3	no	southeast	4449.46
33	male	22.7	0	no	northwest	21984.47
32	male	28.9	0	no	northwest	3866.86
31	female	25.7	0	no	southeast	3756.62
46	female	33.4	1	no	southeast	8240.59
37	female	27.7	3	no	northwest	7281.51
37	male	29.8	2	no	northeast	6406.41
60	female	25.8	0	no	northwest	28923.14
25	male	26.2	0	no	northeast	2721.32
62	female	26.3	0	yes	southeast	27808.73
23	male	34.4	0	no	southwest	1826.84
56	female	39.8	0	no	southeast	11090.72
27	male	42.1	0	yes	southeast	39611.76
19	male	24.6	1	no	southwest	1837.24
52	female	30.8	1	no	northeast	10797.34
23	male	23.8	0	no	northeast	2395.17
56	male	40.3	0	no	southwest	10602.39
30	male	35.3	0	yes	southwest	36837.47
60	female	36	0	no	northeast	13228.85
30	female	32.4	1	no	southwest	4149.74

2.4 Data Ingestion

The cornerstone of our data-driven project was established through a systematic process of data acquisition and ingestion. Utilizing Kaggle, a reputable platform renowned for its high-quality datasets, we identified and acquired the crucial data required for our Insurance price prediction project. This dataset, integral to our goal of accurate price forecasting, was meticulously downloaded and securely stored within our local system infrastructure. Subsequently, we initiated the data ingestion phase, where the dataset seamlessly integrated into our project's data pipeline. This meticulous approach ensures that our project is built upon a solid foundation, setting the stage for robust and precise price prediction models and analysis.

2.5 Data Transformation

Steps performed in pre-processing are:

- First read data from Artifact folder
- Checking unnecessary columns
- One column has product id which is unique for every product so I deleted that column.
- Checked for null values
- there are too many null values are present in two columns that’s why I deleted them
- Performed one-hot encoder on categorical columns.
- Scaling is performed for needed information.
- And, the info is prepared for passing to the machine learning formula

2.6 Modelling

The pre-processed information is then envisioned and every one the specified insights are being drawn. though from the drawn insights, the info is at random unfold however still modelling is performed with completely different machine learning algorithms to form positive we tend to cowl all the chances. and eventually, Gradient Boosting performed well .

2.7 Batch Prediction

In the pursuit of creating a comprehensive and efficient system, we have successfully executed batch prediction as a pivotal component of our project. Leveraging a meticulously designed data transformation pipeline, we have harnessed the power of our predictive model to generate accurate and timely batch predictions. This milestone signifies the culmination of our efforts in seamlessly processing and analyzing data, resulting in actionable insights that drive informed decision-making. As we prepare our Low-Level Design Document, this achievement underscores the significance of our data transformation pipeline and predictive model, which will be elaborately detailed to ensure clarity and scalability in our system architecture.

2.8 Training And Prediction Pipeline

In our endeavor to create a robust and end-to-end data-driven solution, we have meticulously crafted both a training pipeline and a prediction pipeline. The training pipeline serves as the backbone for developing our predictive models, allowing us to iteratively train and fine-tune them with the highest precision possible. Meanwhile, the prediction pipeline enables us to seamlessly apply these trained models to new data, ensuring that our insights and forecasts remain consistently accurate and adaptable to real-

world scenarios. This dual pipeline approach embodies our commitment to providing a comprehensive, data-driven solution that empowers decision-makers with the most reliable and up-to-date information. As we delve into the creation of our Low-Level Design Document, we will intricately detail these pipelines, showcasing their sophistication and efficiency in our system architecture.

2.9 UI Integration

Both CSS and HTML files are being created and are being integrated with the created machine learning model. All the required files are then integrated to the app.py file and tested locally

2.3 Data from User

The data from the user is retrieved from the created HTML web page and Streamlit application .

2.4 Data Validation

The data provided by the user is then being processed by app.py and application.py(streamlit application) 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 web page.

3. Deployment

The tested model is then deployed on local machine and Streamlit Cloud. So, users can access the project from any internet devices.

3.1 Unit Test

Test Case	Description Pre-Requisite	Expected Result
Verify whether the Application URL is accessible to the user	1. Application URL should be defined	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 edit all input fields	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	User should be able to edit all input fields
Verify whether user gets Submit button to submit the inputs	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	User should get Submit button to submit the inputs
Verify whether user is presented with Predicted results on clicking submit	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	User should be presented with Predicted results on clicking submit