iNeuron	Low Level Design(LLD)	Doc no:02
		Rev no:01
		Rev Date:10-11-22

Project: Insurance Premium Predictor

By-Mr.Aniket Dumbre

`

Prepared By	Approved By
Mr.Aniket Dumbre	Mr.Sunny Savita

iNeuron	Low Level Design(LLD)	Doc no:02
		Rev no:01
		Rev Date:10-11-22

Content

`

Prepared By	Approved By
Mr.Aniket Dumbre	Mr.Sunny Savita

1. Introduction

1.1 What is Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Insurance Premium Predictor system. 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.

1.2 Scope

Low-level design (LLD) is a component-level design process that follows a step by step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work.

2. Data Source

Dataset used for this project was taken from kaggle.

https://www.kaggle.com/datasets/noordeen/insurance-premium-prediction

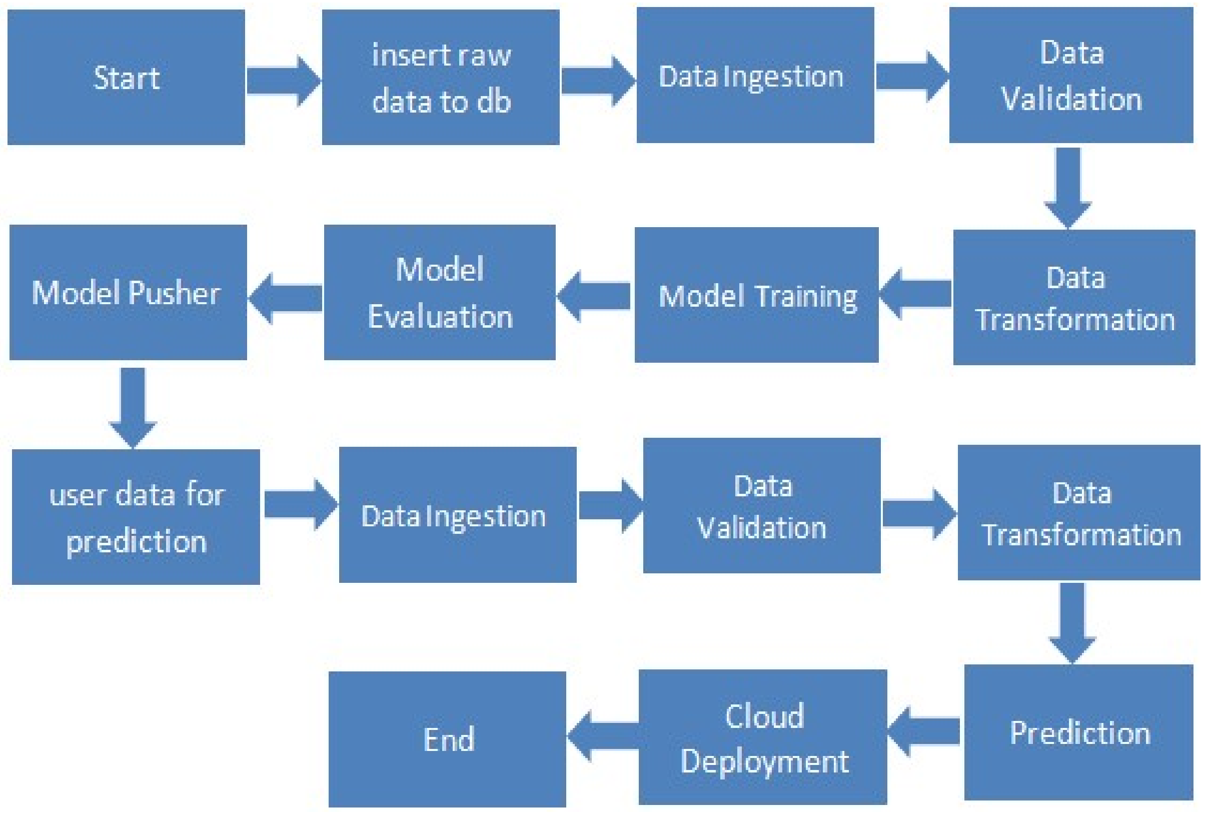
3. Data Description

Dataset have 3 numerical features (age, bmi, children etc), 3 categorical features (sex, smoker, and region) and target feature ‘expenses’ as numerical feature.

age – age of individuals in years	**sex** – Male/Female
no. of children – no. of children person have	**bmi** – body mass index
smoker – whether person smokes or not	**region** – 4 region
expenses – expenses of person on health insurance	

`

4. Architecture



`

Prepared By	Approved By
Mr.Aniket Dumbre	Mr.Sunny Savita

5. Architecture Description

5.1 Data Insertion into Database

- Use Mongo atlas database for storing our data which is NoSQL database.
- Read the dataframe file and convert it to json format and then dumb it to Mongodb.
- Database Creation and connection –

  Create a database with the name "ipp" and collection name "insurance". If the database and collection is already present, then the new database and collection are not created and new files are inserted in the already present collection.

5.2 Training Pipeline

5.2.1 Data Ingestion

- Import collection data as pandas dataframe.
- Replace na values with NAN if present in dataset.
- Save updated dataset to 'feature_store ' folder.
- Split dataset into train and test data and save it into 'dataset' folder.
- Save these folders in 'artifact' folder.

5.2.2 Data Validation

- **Null values-** Drop the columns which have missing values more than specified threshold.
- **Data-type –** If required change data type of columns.
- **No of columns** -Validate the number of columns present in the files.
- **Data Drift-**Check whether new dataset is having major data drift or not. Save the report.yml in artifact folder.

`

5.2.3 Data Transformation

Pipeline used for data pre processing. Create two pipelines for transforming input and target features.

- **One hot encoding –** Encode categorical columns using one hot encoder.
- **Handling missing values-** Use imputation techniques to impute missing values in columns eg. Simple imputer, KNN imputer etc.
- **Feature Scaling-** Try Standard scaler, min max scalar or robust scaler for scaling.
- **Outlier handling-**Use inter-quantile range method to handle outlier/robust scaler.
- **Imbalanced data-** Use SMOTETomek technique to handle imbalanced data.
- Save input feature transformation as transformer.pkl, target feature as transformer_target.pkl, train data as train.npz and test data as test.npz in artifact folder.

5.2.4 Model Training

- There are two approaches for model training
  1. Using single algorithm
  2. Multimodal system
- We adapted multimodal approach where model is being trained on multiple algorithms and selected one which gives high accuracy after hyper parameter tuning.
- **Hyper parameter tuning:**

  Hyper parameter tuning was done by using Grid search algorithm.

- Test model for under fitting or over fitting.
- Save best trained model in pickle format artifact folder as model.pkl.

5.2.5 Model Evaluation

While re-training check whether current model performance with previous model.

5.2.6 Model Pusher

Save trained model and transformed feature files in saved_models directory which will used during model evaluation.

5.3 Data for prediction

Dump user data in db for prediction using AWS S3 bucket for batch prediction or use can predict insurance premium using flask app.

5.4 Prediction Pipeline

Apache airflow is used to monitor ML model performance.
Once data is received from user, prediction pipeline get triggered using github actions and model get retrained and best model is selected for prediction.

5.4.1 Data Ingestion

- Import data frame file from db storage.
- Replace na values with NAN if present in dataset.

5.4.2 Data Validation

- **Null values-** Drop the columns which have missing values more than specified threshold.
- **Data-type –** If required change data type of columns.
- **No of columns** -Validate the number of columns present in the files.
- **Data Drift-**Check whether new dataset is having major data drift or not. Save the report.yml in artifact folder.

`

5.4.4 Prediction

Predict the target feature and saved it in S3 bucket in .csv format.Also user can predict insurance premium using flask app.

5.5 Deployment

Deploy model to the AWS. Batch prediction file stored in S3 bucket.

6. Document change log

Sr. No.	Rev. No.	Rev. Date	Comment	Author
1	00	14-11-22	Initial release	Aniket Dumbre
2	01	10-12-22	Architecture structure updated	Aniket Dumbre

`