Low Level Design

Insurance Premium Prediction

Written By	Subhankar Roy
Document Version	0.1
Last Revised Date	07 – March -2022

Document Control

Change Record:

Version	Date	Author	Comments
0.1	19 – May - 2021	Ankur Das	Introduction & Architecture defined

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

Contents

1. Introduction

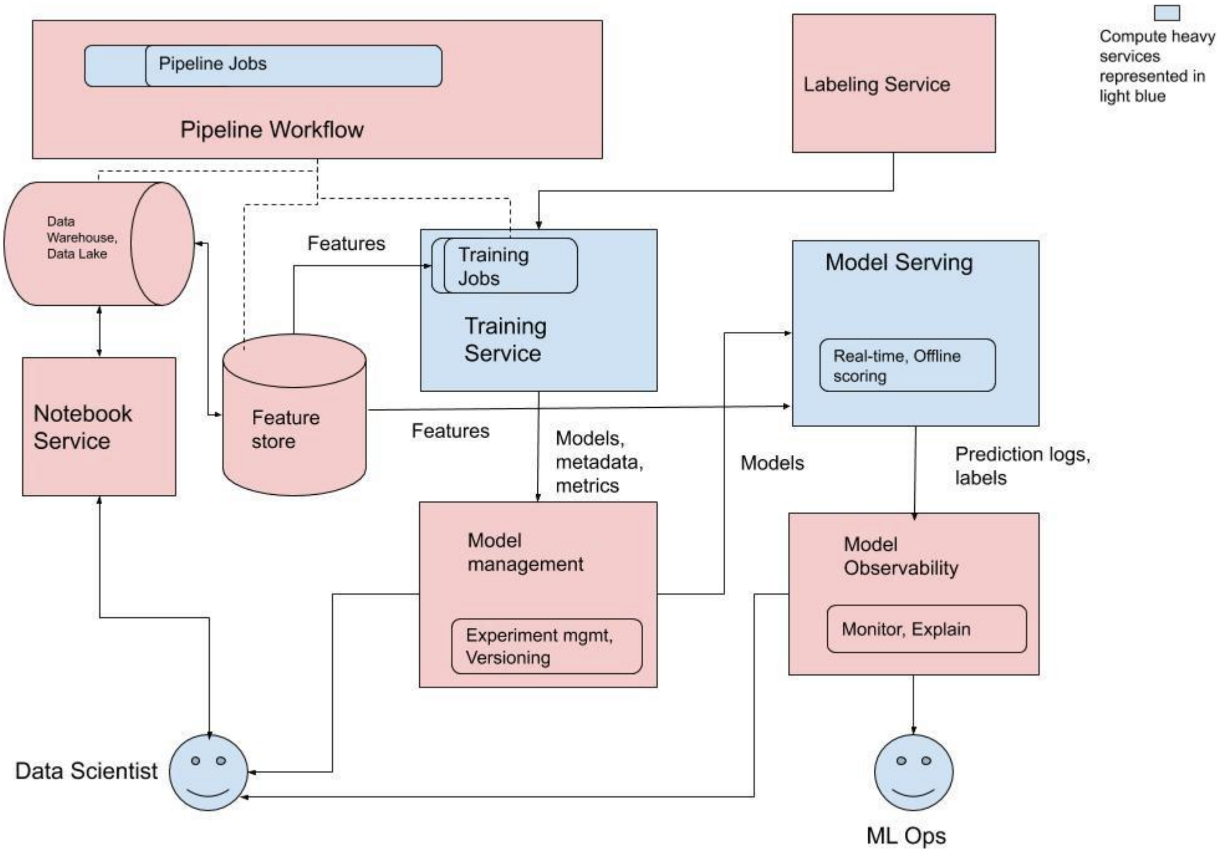
1.1. What is Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Food Recommendation 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. Architecture



3. Architecture Description

3.1. Data Description

Insurance Premium Prediction dataset is the Kaggle dataset. The dataset contains 1338 observations (rows) and 7 features (columns). The dataset contains 4 numerical features (age, bmi, children and expenses) and 3 nominal features (sex, smoker and region) that were converted into factors with numerical value designated for each level.

3.2. Data Insertion into Database

a. Database Creation and connection - Create a database with name passed. If the database is already created, open the connection to the database.
b. Table creation in the database.
c. Insertion of files in the table

3.3. Export Data from Database

Data Export from Database - The data in a stored database is exported as a CSV file to be used for Data Pre-processing and Model Training.

3.4. Data Pre-processing

Data Pre-processing steps we could use are Null value handling, mapping categorical values, Imbalanced data set handling, etc.

3.10. Model Building

We will find the best model for the dataset. We checked four algorithms out of which Gradient Boosting was selected. We will calculate the r2 scores and mean absolute error for models and select the model with the best score.

3.11. Data from User

Here we will collect physiological data from user such as age, sex, bmi, smoker, children and region.

3.12. Data Validation

Here Data Validation will be done, given by the user

3.13. User Data Inserting into Database

Collecting the data from the user and storing it into the database. The database can be either MySQL or Mongo DB.

3.14. Premium Prediction & Saving Output in Database

After calling model Output will be predicted, this output will be saved in Database and it will be used to show the same Output if other users provide the same data.

3.15. Deployment

We will be deploying the model to GCP.
This is a workflow diagram for the Premium Prediction.

4. Unit Test Cases

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 the User is able to sign up in the application	1. Application is accessible	The User should be able to sign up in the application
Verify whether user is able to successfully login to the application	1. Application is accessible 2. User is signed up to the application	User should be able to successfully login to the application
Verify whether user is able to see input fields on logging in	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 see input fields on logging in
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
Verify whether the predicted results are in accordance to the selections user made	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	The predicted results should be in accordance to the selections user made
Verify whether user has options to filter the predicted results as well	1. Application is accessible 2. User is signed up	User should have options to filter the predicted results as well

	to the application 3. User is logged in to the application	
Verify whether KPIs modify as per the user inputs	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	KPIs should modify as per the user inputs
Verify whether the KPIs indicate details of the predicted premium	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	The KPIs should indicate details of the predicted premium