Low Level Design(LLD)

Insurance Premium Prediction

Version 1.0

25/08/2021

Version Control

Version	Description	Responsible Party	Date
1.0	Initial version	Motheeshkumar Jay	25-08-2021

1 Introduction

1.1 What is a Low Level Design Document?

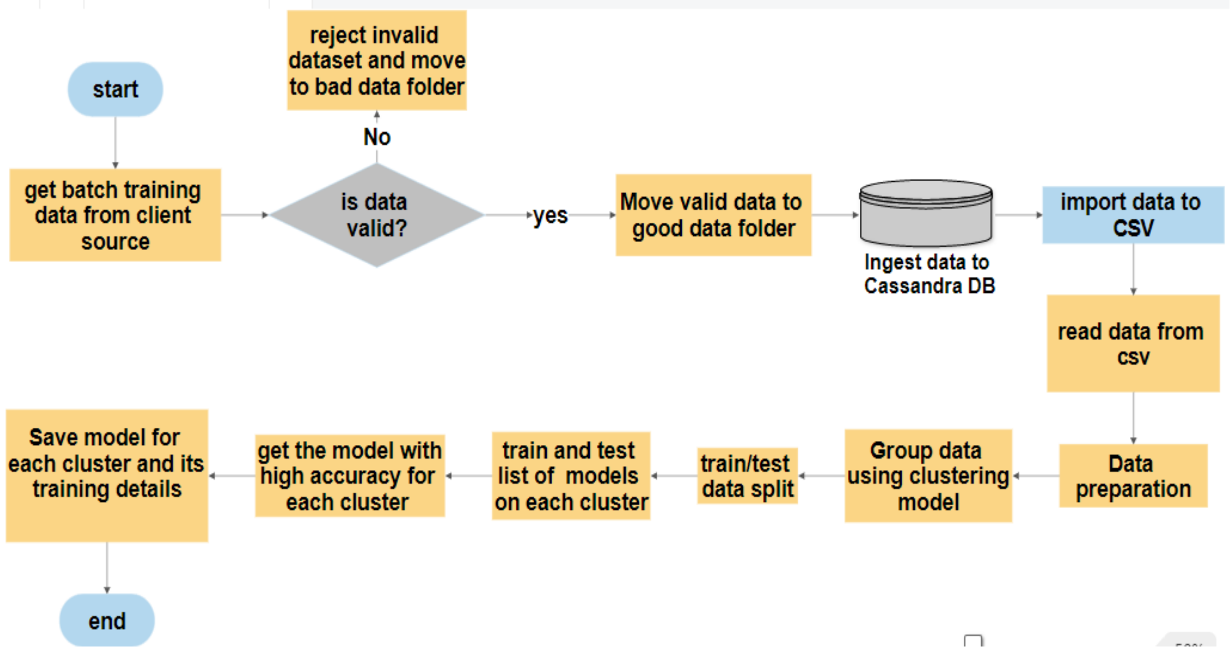
The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Online Doctor Visit Appointment Application. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

1.2 Scope

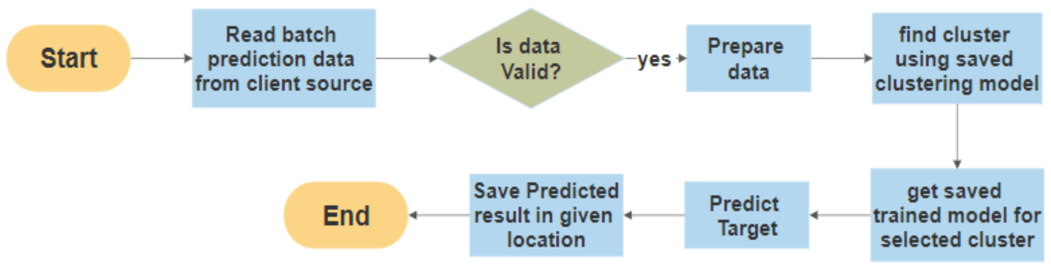
Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement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

2 Architecture

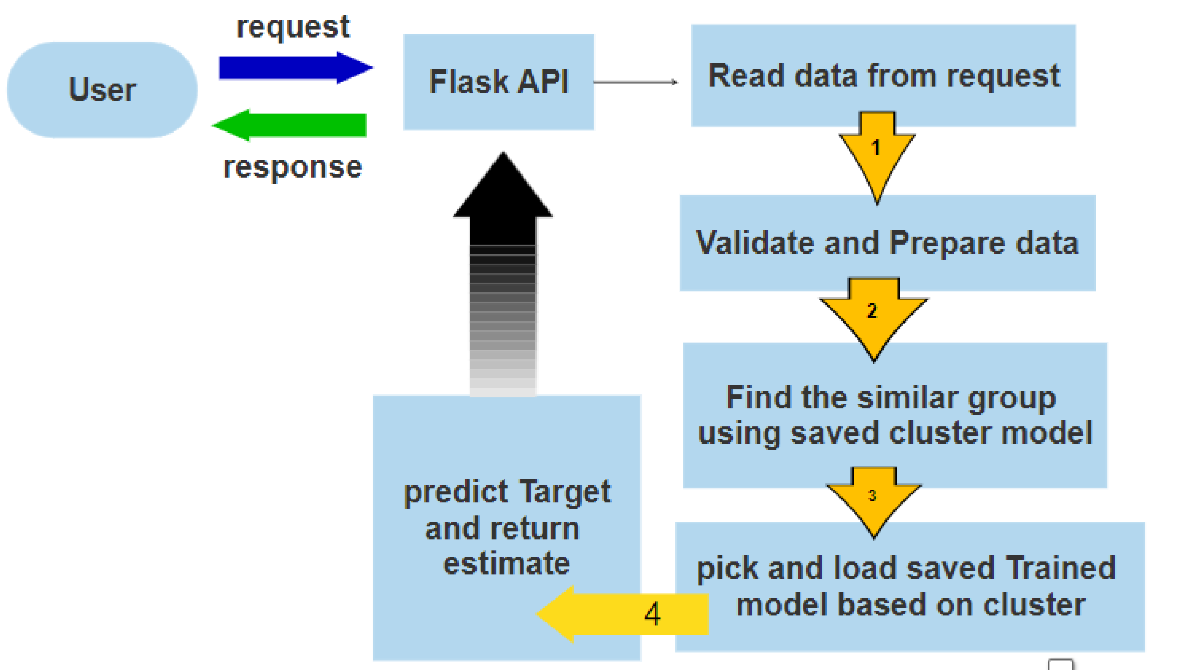
Model training:


start
get batch training data from client source
is data valid?
No
reject invalid dataset and move to bad data folder
yes
Move valid data to good data folder
Ingest data to Cassandra DB
import data to CSV
read data from csv
Data preparation
Group data using clustering model
train/test data split
train and test list of models on each cluster
get the model with high accuracy for each cluster
Save model for each cluster and its training details
end

Batch Data Prediction:


Start
Read batch prediction data from client source
Is data Valid?
yes
Prepare data
find cluster using saved clustering model
get saved trained model for selected cluster
Predict Target
Save Predicted result in given location
End

IPP Web Application:


User
request
response
Flask API
Read data from request
1
Validate and Prepare data
2
Find the similar group using saved cluster model
3
pick and load saved Trained model based on cluster
4
predict Target and return estimate

3 Architecture Description

3.1 Data Validation

3.1.1 Data source and format

In this project, we are going to read a set of datasets from a client source where the client will be updating the training dataset whenever they want.

3.1.2 Data format:

CSV from client source

3.1.3 Data schema

Define structure in JSON format of data set which allows us to validate data structure and create table structure based on that. Training batch data should be read from client source.

3.1.4 Feature validation

Check whether all features are present or not. Check all the names and data types are matching with predefined data schema.Check if any of the features have missing values >95 percent. if yes reject dataset and move to bad dataset folder else move to good dataset folder.once data validation done prepare data for database by replacing missing values with NULL values.If Data validation is successful move data set to good data folder from client dataset source.If Data validation is failed move data set to bad data folder from client dataset source and specify the reason for rejection of data.

3.2 Data Ingestion

3.2.1 Database details

Database - Cassandra

Version - cqlsh 6.8.0

3.2.2 Read data

Read all the valid dataset from a good data folder and merge it in one data frame.

3.2.3 create table for dataset

Create a table based on data schema dynamically. drop if it already exists and create a new table with a data schema.

3.2.4 insert data into table

Once the table creation is successful, add all the valid data to the newly created table.

3.2.5 Read data and convert to CSV file.

Once the data is added, read the data from DB and convert it to a single CSV file for training the model.

3.3 Data preprocessing

This step involves data cleaning, data transformation and feature selection. We will be preparing data for machine learning models.

Data cleaning - imputing null values with KNN weighted average

Data Transformation - feature encoding and feature scaling process.

Feature selection - removing constant features, removing duplicate features, removing irrelevant features

Preserve Data Structure - Once we complete data preprocessing, feature column names will get changed after feature encoding. so we will save the feature names as pickle files. In future, we will use this data to check the column name match so that we won't get any column mismatch error while predicting using the saved model.

Data Clustering

In order to get the good accuray , we are using K-means clustering algorithm to cluster the similar data and then we will try to train a different ML algorithm on each cluster. we will choose the model with high accuracy and without overfitting.

clustering involves below steps:

1. Finding the K number of clusters using elbow method(KneeLocator)
2. Then we will try to cluster data using k-means clustering algorithm
3. Once we train we will save the clustering model for future prediction.

3.4 Model building

Once the data preparation is ready, model building involves the below step.

Test/ train split:

We will split each cluster into train and test data.

Training model:

We will try Random forest and Gradient boosting regressor algorithms on each cluster.

Testing:

Once we train the model, we will test each model with test data and pick the model with high accuracy without overfitting

Overfitting Threshold:

In this use case, we are using 5% as an overfitting threshold. We will reject models if test accuracy is less than 5% than training accuracy.

Saving Model:

Once we choose the model with high accuracy, we will save the model for prediction in pickle format and also save the model details.

3.5 Data Prediction

Batch data prediction:

Batch data prediction involves the steps below.

1. Reading data from batch prediction dataset location.
2. validating the data.
3. prepare data by imputing null values, encoding categorical values.
4. Once we are done with data preparation, we will use a saved clustering model to find which cluster it belongs to.
5. Then, we will load a saved model based on the cluster and we will predict the estimation.
6. Once we predict the data, we will save the result in the required destination.

UI based prediction from users:

In this process, we will use the same steps as batch data prediction but for a single record. We will follow the steps below.

1. Receive request from UI.
2. Read data from requests.
3. We will follow the same steps as batch prediction.
4. Respond back to users with estimated value and response time.

4 Page details

4.1 Home

First thing anyone will see is the insurance premium prediction form, which will ask the user to enter details related to health and family details to predict the insurance premium. By clicking on the predict button, the user will be able to get the estimation.

4.2 Batch Prediction result screen

By clicking on the batch prediction result, the client will be able to get the batch prediction result.

4.3 Trained model details screen

By clicking on the model details page, the client will be able to see the trained model details like accuracy, number of groups from training dataset, parameters, and model details.

5 Unit Test Cases

Test Case Description	Prerequisite	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 can fill details about their health and family	1. Application is accessible 2. Application is deployed	The User should be able to fill the form.
Verify whether the user can successfully get the insurance premium estimate.	1. Application is accessible 2. Application is deployed 3. User is able to get the premium estimation successfully.	Users should be able to successfully get the estimate after clicking on the predict button.
Verify whether the user can see the response time required to get the prediction.	1. Application is accessible. 2. Application is deployed 3. User is able to get the premium estimation successfully. 4. Users are able to see the response time.	Users should be able to see the response time.
Verify whether the client is able to get the trained model details.	1. Application is accessible 2. Application is deployed 3. Clients are able to see trained model details.	Clients should be able to see trained model details.
Verify whether the client is able to get the batch prediction results.	1. Application is accessible. 2. Application is deployed 3. Clients are able to get the batch prediction results.	Clients should be able to get batch prediction results by clicking on batch prediction results.