Low Level Design

Money Laundering Prevention

Written By	Rohit Prasad
Document Version	0.3
Last Revised Date	30 -Mar-2022

Document Control

Version	Date	Author	Comments
0.1	15 – Mar - 2022	Rohit Prasad	Introduction & Architecture defined
0.2	20 – Mar - 2022	Rohit Prasad	Architecture & Architecture Description appended and updated
0.3	30 – Mar - 2022	Rohit Prasad	Unit Test Cases defined and appended

Contents

1. Introduction

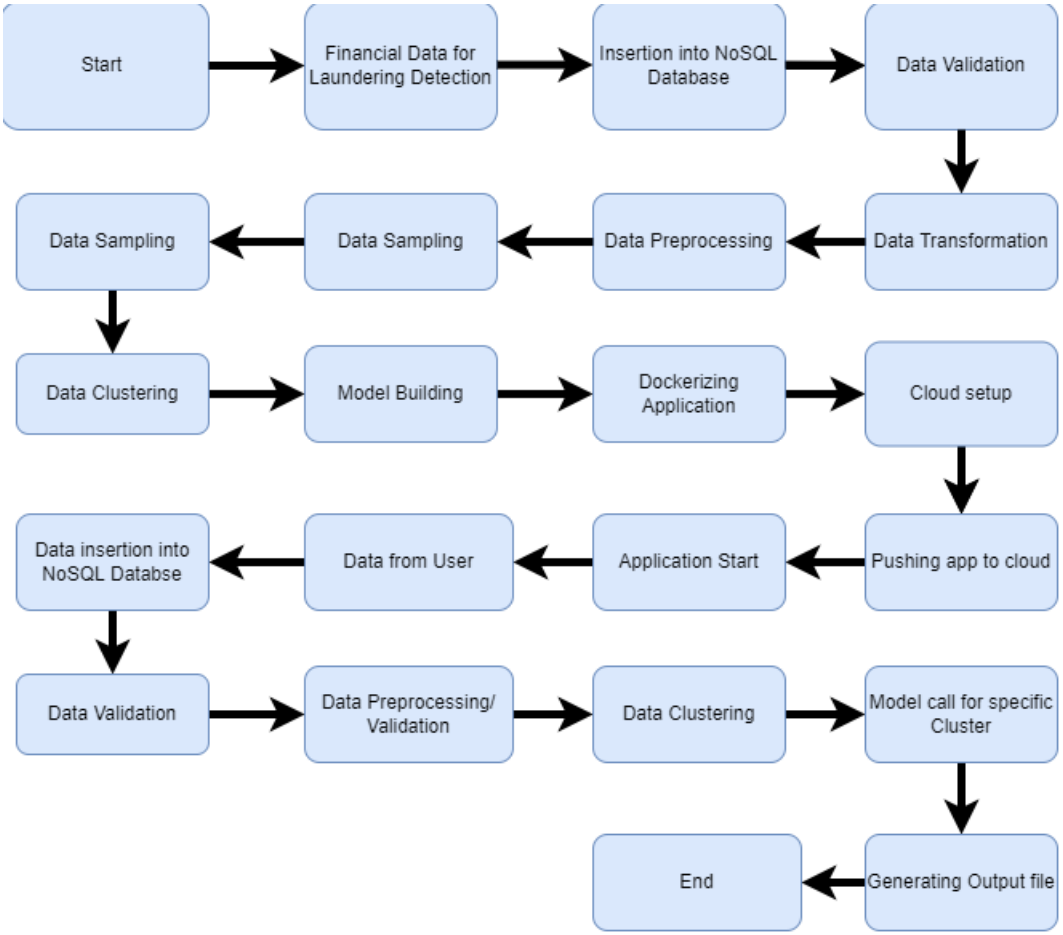
1.1. What is Low-Level design document?

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

Dataset Contains 6M+ financial transaction data containing overall 11 columns.
Each column has a specific information related to the financial transaction. There are no null or missing values in the dataset provided

3.2. data Insertion into database

The data is validated and then fetched from the file to MongoDB database for future analysis and evaluation. All the database operation is created and maintained so that we don't have any issue while loading the data in the database

3.3. Data Transformation

The dataset is already in a well transformed manner so that we can just validate and do some feature engineering just to get useful insights from data which will help us to make a more generalized model for our project

3.4. Data Preprocessing

Data Preprocessing involves some steps like encoding variables, sampling, scaling and splitting of the data. Null and missing value handling is also included to take care of any future missing data

3.5 Data Clustering

K-Means algorithm will be used to create clusters in the pre-processed data. The optimum number of clusters is selected by plotting the elbow plot. The idea behind clustering is to I mplement differentalgorithms to train data in different clusters. The K-means model is trained over preprocessed data and the model is saved for further use in prediction

3.6 Model Building

After clusters are created, we will find the best model for each cluster. For each cluster, algorithms will be passed with the best parameters derived from Grid-Search. We will calculate the F1 scores for models and select the model with the best score. Similarly, the models will be selected for each cluster. All the models for every cluster will be saved for use in the prediction data.

3.7 Data from User

Here we will collect data from user in a csv file format. The file should contain the data which is prerequisite/ Defined in our programming in order to avoid any error while validation. Order does not matter for the data in the csv.

3.8 User Data Inserting into Database

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

3.9 Data Validation

Validation of the data will take place after insertion into database. Validation will check for count of columns, column datatypes etc

3.10 Data Clustering

The model created during training will be loaded, and clusters for the user data will be predicted.

3.11 Model Call for Specific Cluster

Based on the cluster number, the respective model will be loaded and will be used topredict the outcome for that cluster.

3.12 Saving Output in CSV file

After calling model, Outcome will be recommended, this output will be saved in CSV file and it will be downloaded automatically in user's system

3.13 Deployment

We will be deploying the model to AWS EC2 instance and use docker image to initiate the app

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 user are able to upload their prediction file	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 able to upload files	User should be able to successfully upload file
Verify whether user is able to see the progress of the prediction	1. Application is accessible 2. User is able to upload files 3. User is able to see the progress of prediction	User should be able to see progress bar for the prediction
Verify whether user is able to click and generate the prediction outcome	1. Application is accessible 2. User is able to upload files 3. User is able to see the progress of predict 4. User is able to generate the prediction outcome	User should be able to click and generate the prediction outcome
Verify whether the output generated is in a specific/proper format as per user's prerequisite	1. Application isaccessible 2. User is able to upload files 3. User is able to see the progress of predict 4. User is able to generate the prediction outcome 5. Output is in the proper format as expected	User should get output in a csv file in a proper/required format