Low Level Design

Credit Card Default Prediction

Written By	**Prajjwal Sule**
Document Version	**2.0**
Date	**12-June-2023**

Document Control

Version	Date	Author	Comments
1.0	11-06-2023	Prajjwal Sule	Introduction & Architecture defined.
2.0	12-06-2023	Prajjwal Sule	Architecture & Architecture Description appended and updated

Reviews:

Version	Date	Reviewer	Comments
1.0	12-06-2023	Prajjwal Sule	Document Content, Version Control and Unit Test Cases to be added.

Approval Status:

Version	Review Date	Review By	Approved By	Comments

Content

1. Introduction

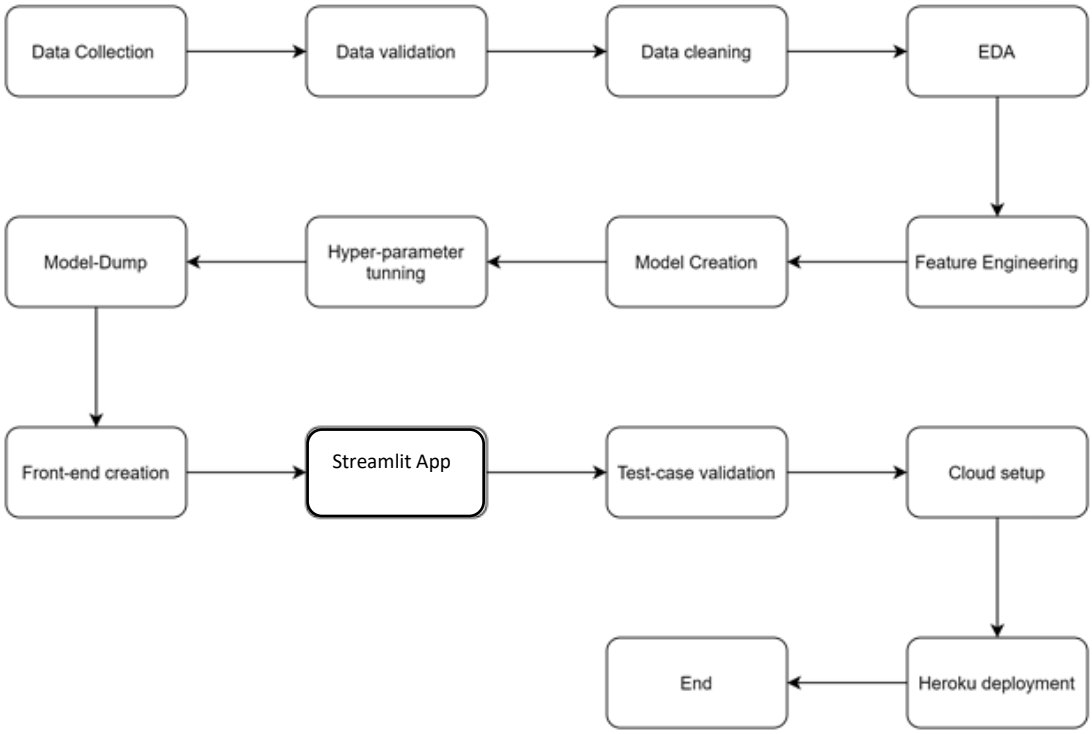
1.1 What is Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Credit Card Prediction 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 Collection

We have Dataset of row columnar which includes various columns like customer limit, Their Gender, Education, Maratial Status, Bill Payments, Monthly Dues and etc . The information is given in csv format. These data is collected from UCI Repository.

3.2 Data Validation

Data collected from Data source is validated on ceratin criteria and data after getting validated is send to next stage which is Data cleaning**.**

3.3 Data Cleaning

In the Cleaning process, We have cleaned up all the data because data is present in very bad format which was cannot reconigzed by machine. So in this process we make data understable so that machine can work easily on top of that.

3.4 EDA

In EDA we try to perfrom necessary data analysis steps to finding the patterns in data and get some observation about the data, like how data bahaves and this step is also help us to find the missing values, finding outliers, get to know more about the features with respect to label.

3.5 Feature Engineering

In Feature Engineering we try to impute Missing values, handle the outliers, encode variable, perform standardization by using standard scalar, split data into dependent and Independent variables for futher model building purpose.

3.6 Model Creation

After cleaning the data and completing the feature engineering. we have done splitted data in the train data and test data and implemented classifiaction algorithms like Random Forest classifier and XGboost classifier and also calculated their accuracies on test data. So, that we can pick and choose final model for the depolyment.

3.7 Hyperparameter Tunning

In hyperparameter tuning we have implemented various ensemble techniques like random forest regressor, bagging and boosting we also done randomized search cv or grid search cv and we also implemented cross validation techniques for that. So that we have choose best parameters according to hyperparameter tunning and best score from their accuracies so we got 78% accuracy in our RandomForest Classifier after hyper parameter tuning.

3.8 Model Dumping

After comparing all accuracies and checked all roc, auc curve we have choosen hyperparameterized RandonForest classifier as our best model by their results so we have dumped these model in a pickle file format with the help of python pickle module.

3.9 Front-end Creation

In Frontend creation we have made a user interactive page where user can enter their input values to our application. In these frontend page we have made a form which is of Streamlit.

3.10 Stramlit-App Creation

Here we try create Streamlit Web app which acts as an user interface between user and and our backend ,by hitting stramlit application, user will get the result of their input data.

3.11 Data Inserting into Database

Collecting the data and storing it into the database. The database can be either MySQL or Cassandra DB. Here we are using Cassandra database.

3.12 Model Call

We are loading our pickle file in the backend and predicting whether the custome is a credible customer (0) or not a credible customer (1) as a output and sending to our Stramlit page.

3.13 Deployment

We will be deploying the model to Heroku.

This is a workflow diagram for the Credit Card Default Prediction System..

4. Unit TestCases

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 URLis accessible 2. Application is deployed	The Application should load completely for the user when the URL is accessed
Verify Response time of URL frombackend model.	1. Application is accessible	The Latency and accessibility of URL is faster.
Verify Response time of URL frombackend model.	1. Handled test cases atbackends.	User should be able to see successfully valid results.
Verify whether user is able to see input fields on logging in	1. Application is accessible . User is logged into the application	User should be able to see inputfields on logging in
Verify whether user is able to edit all input fields	1. Application is accessible 2. User is logged in to the application	User should be able to edit all input fields
Verify whether user gets predict button to submit the inputs	1. Application is accessible 2. User is logged into the application	User should get Submit button tosubmit the inputs
Verify whether user is presented with recommended results on clicking submit	1. Application is accessible 2. User is logged into the application	User should be presented with recommended results on clicking submit
Verify whether the recommended results are in accordance to the selections user made	1. Application is accessible 2. User is logged into the application	The recommended results should be in accordance to the selectionsuser made

LOW LEVEL DESIGN (LLD)