Low Level Design

Credit Card Fraud Detection

|  |  |
| --- | --- |
| Written By | Babita Pant |
| Document Version | 0.3 |
| Last Revised Date | 23 – Oct -2021 |

**Document Control**

### Change Record:

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| 0.1 | 19 – Oct -  2021 | Babita Pant | Introduction & Architecture defined |
| 0.2 | 20 – Oct -  2021 | Babita Pant | Architecture & Architecture Description appended and updated |
| 0.3 | 21 – Oct -  2021 | Babita Pant | Unit Test Cases defined and appended |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### Approval Status:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Review**  **Date** | **Reviewed By** | **Approved By** | **Comments** |
|  |  |  |  |  |

Contents

1. [Introduction 1](#_bookmark0)
   1. [What is Low-Level design document? 1](#_bookmark1)
   2. [Scope 1](#_bookmark2)
2. [Architecture 2](#_bookmark3)
3. [Architecture Description 3](#_bookmark4)
   1. [Data Description 3](#_bookmark5)
   2. [Web Scrapping 3](#_bookmark6)
   3. [Data Transformation 3](#_bookmark7)
   4. [Data Insertion into Database 3](#_bookmark8)
   5. [Export Data from Database 3](#_bookmark9)
   6. [Data Pre-processing 3](#_bookmark10)
   7. [Data Clustering 3](#_bookmark11)
   8. [Model Building 4](#_bookmark12)
   9. [Data from User 4](#_bookmark13)
   10. [Data Validation 4](#_bookmark14)
   11. [User Data Inserting into Database 4](#_bookmark15)
   12. [Data Clustering 4](#_bookmark16)
   13. [Model Call for Specific Cluster 4](#_bookmark17)
   14. [Recipe Recommendation & Saving Output in Database 4](#_bookmark18)
   15. [Deployment 4](#_bookmark19)
4. [Unit Test Cases 5](#_bookmark20)

# Introduction

## What is Low-Level design document?

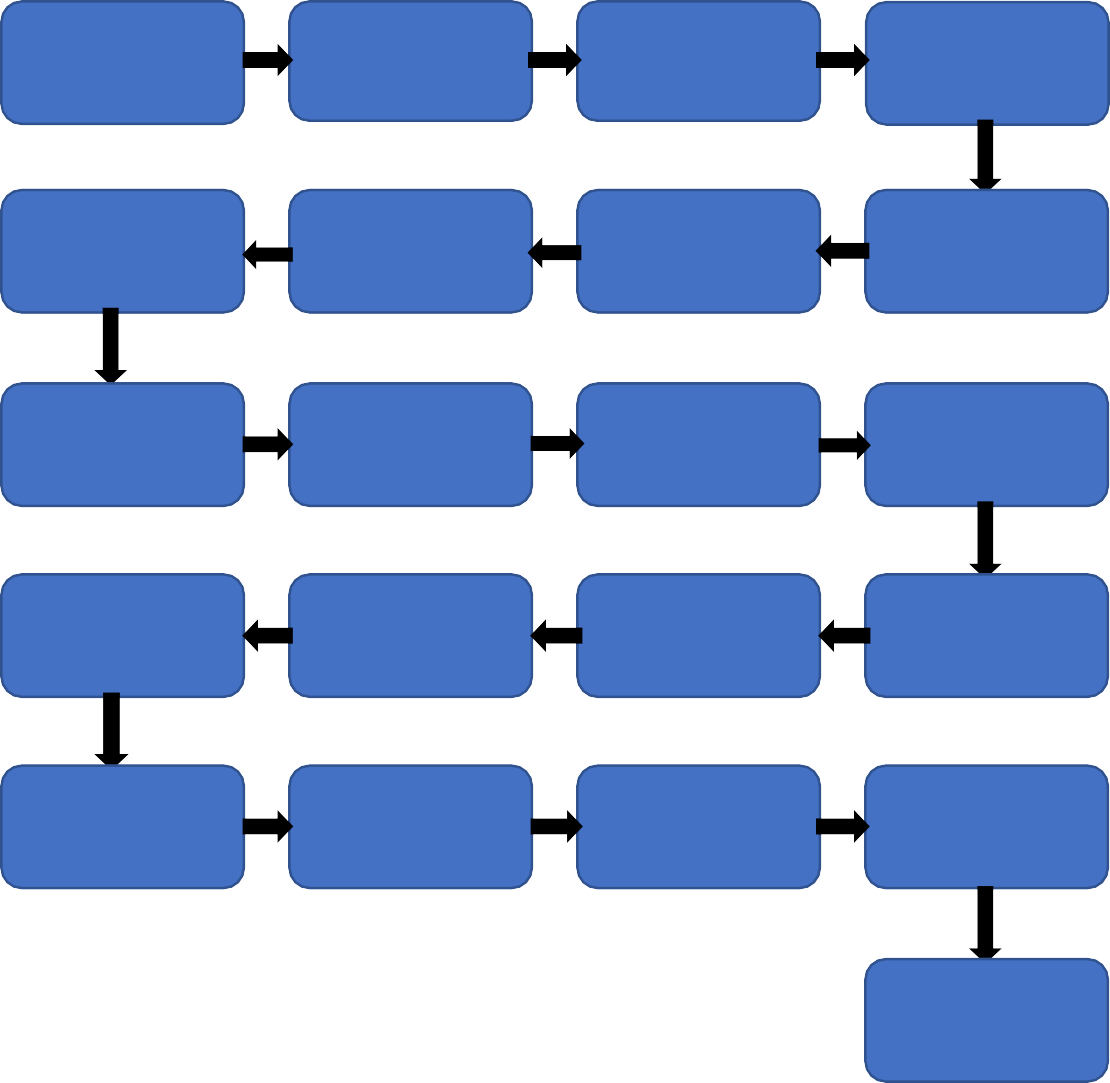
The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for credit card fraud detection. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can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

## Scope

Low-level design (LLD) is a component-level design process that follows a step-by-

step [refinement](https://en.wikipedia.org/wiki/Refinement_(computing)) 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

# Architecture



Start

Data for

Recommendation

Web

Scrapping

Data

Transformation

NLP Techniques

Data

Preprocessing

Export Data

from Database

Data Insertion

Into Databases

Data Clustering

Model Building

Cloud Setup

Pushing app to

cloud

Data Insertion

into Database

Data Validation

Data from User

Application

Start

Data Clustering

Model Call for

Specific Cluster

Recipe

Recommendation

Saving Output

at Database

End

# Architecture Description

## Data Description

The dataset contains transactions made by credit cards in September 2013 by European cardholders.  
 This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependent cost-sensitive learning

* 1. Web Scrapping

In order to create a more complete recipe collection we will need some more datasets which will contain Nutritional value of recipes along with Ratings and total Calories.

## Data Transformation

In the Transformation Process, we will convert our original dataset which is in pickle format to CSV format and will merge it with the Scrapped dataset.

## Data Insertion into Database

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

## 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.

## Data Pre-processing

Data pre-processing steps we could use are Null Value handling. Deal with outliers, Under-sampling the unbalanced data, splitting the data into train and test sets. Handling columns with standard deviation zero or below a threshold, etc.

## Model Building

After doing data preprocessing and feature engineering, we will find the best model for each pipeline. For each pipeline, algorithms will be passed with the best parameters derived from Grid-Search. We will calculate the AUC scores for models and select the model with the best score. Similarly, the models will be selected for each pipeline. All the models for every pipeline will be saved for use in Recommendation.

## Data from User

Here the user will have to enter all the features values in correct order and have to submit it to the model with the help of UI interface. The data will be fed to the model which will predict whether the feature set represents a fraudulent transaction or not.

## Data Validation

Here Data Validation will be done, given by the user

## User Data Inserting into Database

Collecting the data from the user and storing it into the database. The database Cassandra.

## Recipe Recommendation & Saving Output in Database

After calling model Recipe/Output will be recommended, this output will be saved in Database and it will show the same Output if other users provide the same data.

## Deployment

We will be deploying the model to Heroku.

This is a workflow diagram for the Credit Card Fraud Transaction Detection.

# 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 detection 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 detection results on clicking  submit |
| Verify whether the recommended 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 detection results should be in accordance to the selections user made |
| Verify whether user has options to filter the detection results as well | 1. Application is accessible 2. User is signed up | User should have options to filter the detection results as well |

|  |  |  |
| --- | --- | --- |
|  | to the application  3. User is logged in to the application |  |
| Verify whether KPIs modify as per the user inputs for the user's credit card transaction | 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 for the user's credit card transaction |
| Verify whether the KPIs indicate details of the suggested credit card transaction | 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 suggested credit card transaction |