Low Level Design (LLD)

Credit Card Default Prediction

Revision Number: 1.1

Gourav Rishi

Sandeep Kashyap

Document Version Control

Table of Contents

1. Introduction

1.1 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 the Credit Card Fraud Detection Model. 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

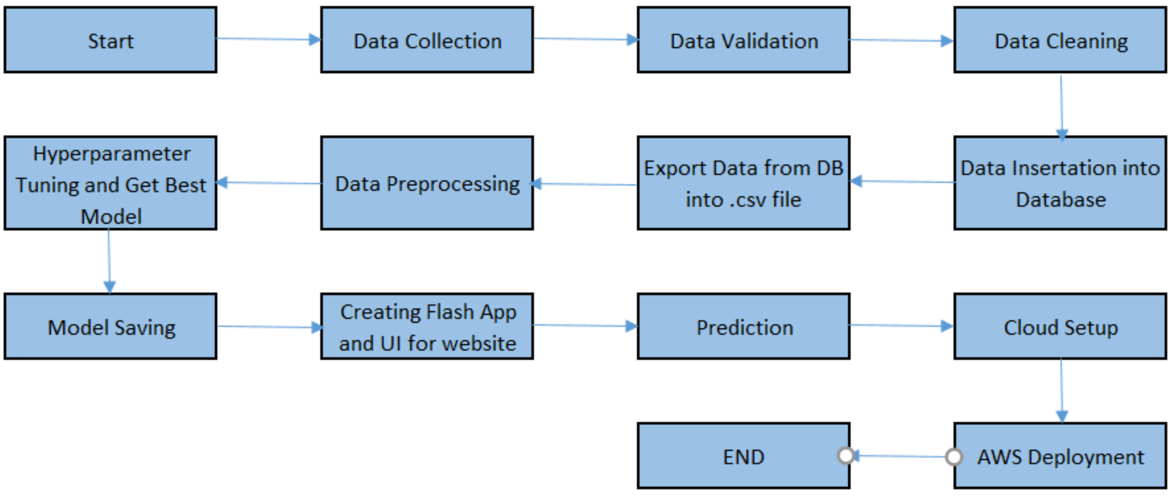
1.2 Scope

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.

1.3 Constraints

Constraints Internet connection is a constraint for the application. Since the application fetched the data from the database, it is crucial that there is an Internet connection for the application to function. Since the model can make multiple requests at same time, it may be forced to queue incoming requests and therefore increase the time it takes to provide the response

2. Architecture


Start
Data Collection
Data Validation
Data Cleaning
Data Insertation into Database
Export Data from DB into .csv file
Data Preprocessing
Hyperparameter Tuning and Get Best Model
Model Saving
Creating Flash App and UI for website
Prediction
Cloud Setup
AWS Deployment
END

3. Architecture Description

3.1 Data Description

This data is about fraud detection in credit card transactions. The data is of the credit cards transaction in September 2013 by European cardholders. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions. It contains only numeric input variables which are the result of a PCA transformation. The original features and more background information about the data is not provided. The dataset contains 284,807 instances, 492 instances are fraudulent, the remaining 284,315 instances are genuine.

3.2 Data Cleaning

In the Cleaning process, we cleaned up all the data which have null values because the percentage of null values in the dataset was very less. So I have dropped all the rows that were containing null values.

3.3 Exploratory Data Analysis

We have done EDA in such a way that every nook and corner of features were clearly justified with the help of correlation, plotting the heat map using seaborn and matplotlib and so on, and found out that the data set is quite good but is highly unbalanced.

3.4 Event Log

The system should log every event so that the user will know which process is running internally. Logging is implemented using python's standard logging library. Step by step description is as follow:

● The system should be able to log each and every system flow.

● System must be able to handle logging at greater scale and ensure debugging the entire issue

3.5 Data Insertion into Database

❖ Database Creation and connection - Create a database with name passed. If the database is already created, open the connection to the database

❖ Table creation in the database

❖ Insertion of files in the table

3.6 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.

3.7 Data Pre-processing

In data pre-processing steps we handled Null Value. Categorical to Numerical Transformation of columns, Undersampling the unbalanced data, splitting the data into train and test sets. Handling columns with standard deviation zero or below a threshold, etc.

3.8 Model Creation/ Model Building

After cleaning, processing the data, and feature engineering. We have done train test split using method build in pre-processing file and implement various Classification Algorithm and found out that Logistic Regression suits best for the model with an excellent accuracy.

3.9 Model Dump

After comparing all accuracies and finding the best model for the dataset I have created a model and dumped the model in a pickle file format with the help of pickle module.

3.10 Data from User

The user will 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 further predict whether the feature set represents a fraudulent transaction or not.

3.11 Data Validation

Data Validation is preformed when data is given by the user.

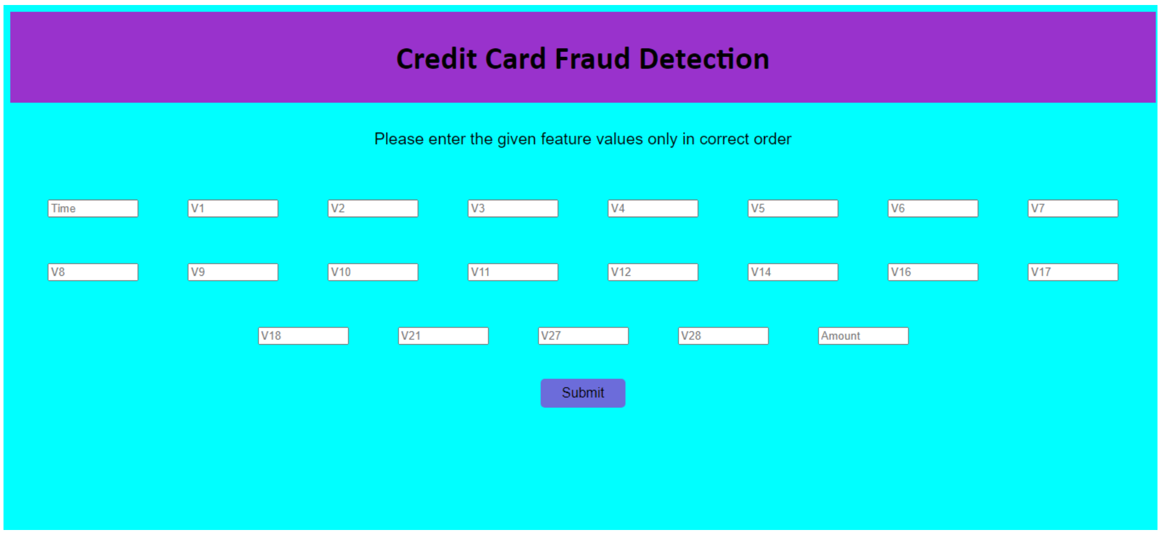
3.12 Model Call for input

Based on the User Input will be thrown to the backend in the variable format then it will be converted into a numpy array which will be fed to our model. The loading of the pickle file will be done and then the model will predict whether the Input is fraudulent or not by sending the result to our html page.

3.13 User Interface

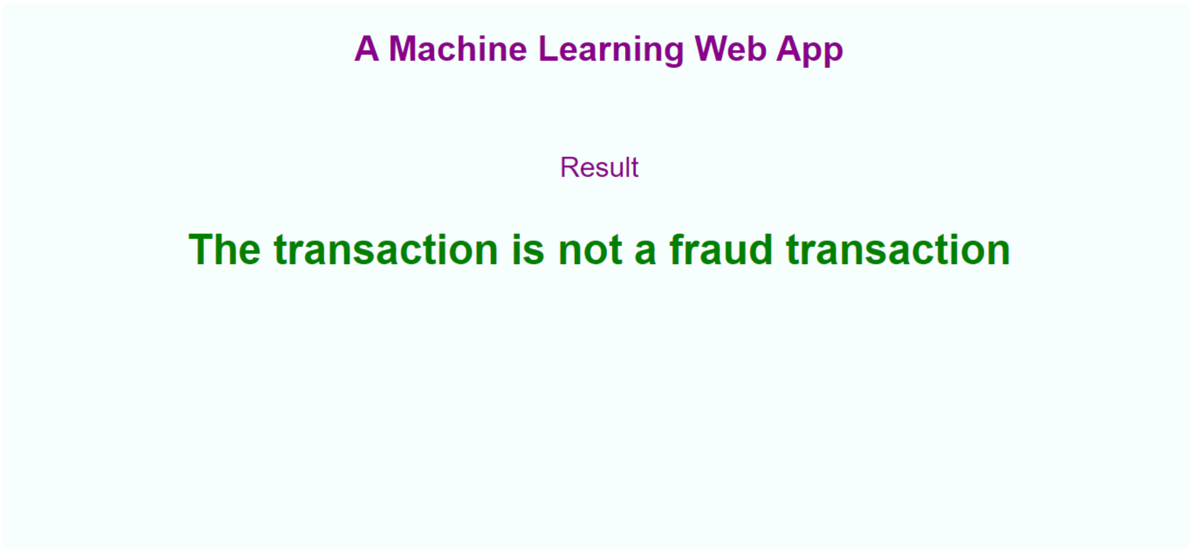
For frontend, we have made a user interactive page where users can enter their input values to our application. In their frontend page we have made a form which is beautified with CSS. This HTML user input data is transferred in variable format to the backend. We have these html fully in a decoupled format.

- Home Page



Credit Card Fraud Detection
Please enter the given feature values only in correct order
Time
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12
V14
V16
V17
V18
V21
V27
V28
Amount
Submit

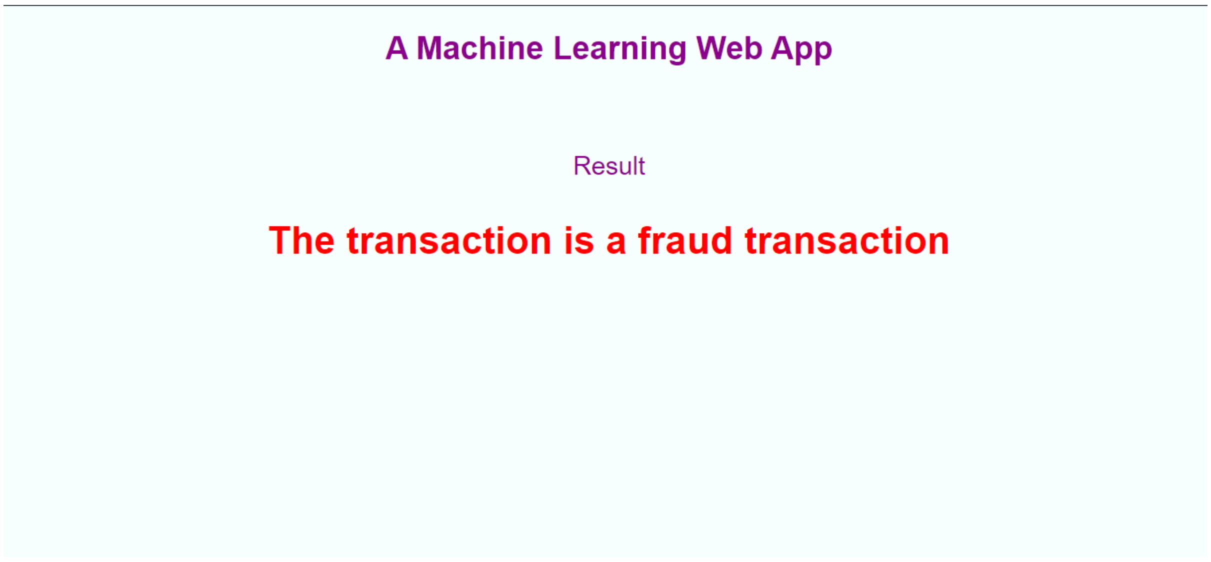
- Prediction page



A Machine Learning Web App

Result

The transaction is not a fraud transaction



A Machine Learning Web App

Result

The transaction is a fraud transaction

3.14 Deployment

We have deployed our model in heroku and aws cloud platform.

4.Technology Stack

Front End	HTML/ CSS
Back End	Flask, Pandas, Numpy, scikit- learn etc
Database	MongoDB
Deployment	AWS, Heroku