Low Level Design

Credit Card Fraud Detection

Written By	Sandeep Kashyap
Document Version	1.2
Last Revised Date	26th sep 2021

Document Version Control

Date Issued	Version	Description	Author
15th sep 2021	1.0	First Draft	Sandeep Kashyap
25th sep 2021	1.1	Added Test Cases	Sandeep Kashyap
26th sep 2021	1.2	Added Model training/validation workflow and I/O workflow	Sandeep Kashyap

Contents

Abstract

It is vital that credit card companies are able to identify fraudulent credit card transactions so that customers are not charged for items that they did not purchase.
Such problems can be tackled with Machine Learning. This project intends to illustrate the modelling of a data set using machine learning with Credit Card Fraud Detection. The Credit Card Fraud Detection Problem includes modelling past credit card transactions with the data of the ones that turned out to be fraudulent or not. Our objective here is to detect 100% of the fraudulent transactions while minimizing the incorrect fraud classification. In this process, we have focused on analysing and preprocessing data sets as well as the deployment of multiple anomaly detection algorithms such as Local Outlier Factor and Isolation Forest algorithms on the PCA transformed Credit Card Transaction data.

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 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.

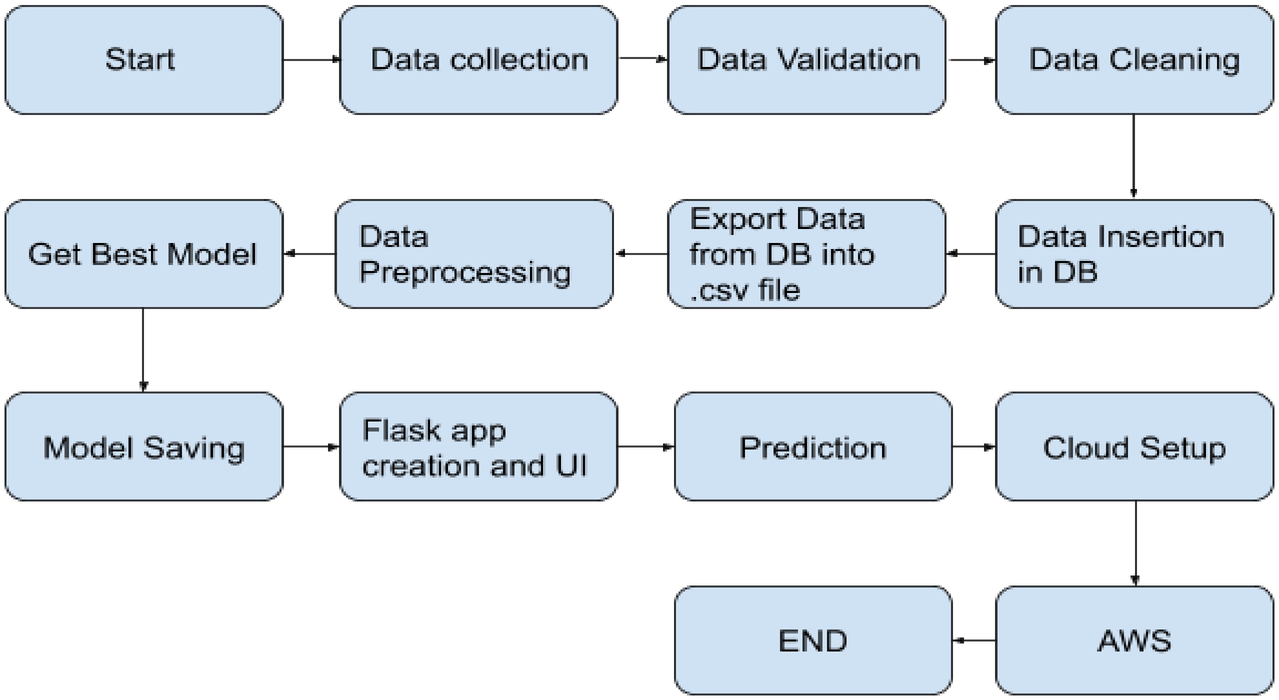
1.2 Scope

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

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



3. Architecture Description

3.1 Data Description

This data is about fraud detection in credit card transactions. The data was made by credit cards 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, I cleaned up all the data having 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

I 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 what process is running internally. Logging is implemented using python's standard logging library.Initial step by step description:-

- The system should be able to log each and every system flow.
- System must be able to handle logging at greater scale because it helps debugging the issue and hence it is mandatory too.

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 Preprocessing

Data pre-processing steps we could use are Null Value handling. 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 and processing the data, completing feature engineering I have done train test splitting using method build in preprocessing file and implemented 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

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.

3.11 Data Validation

Here Data Validation will be done, given by the user.

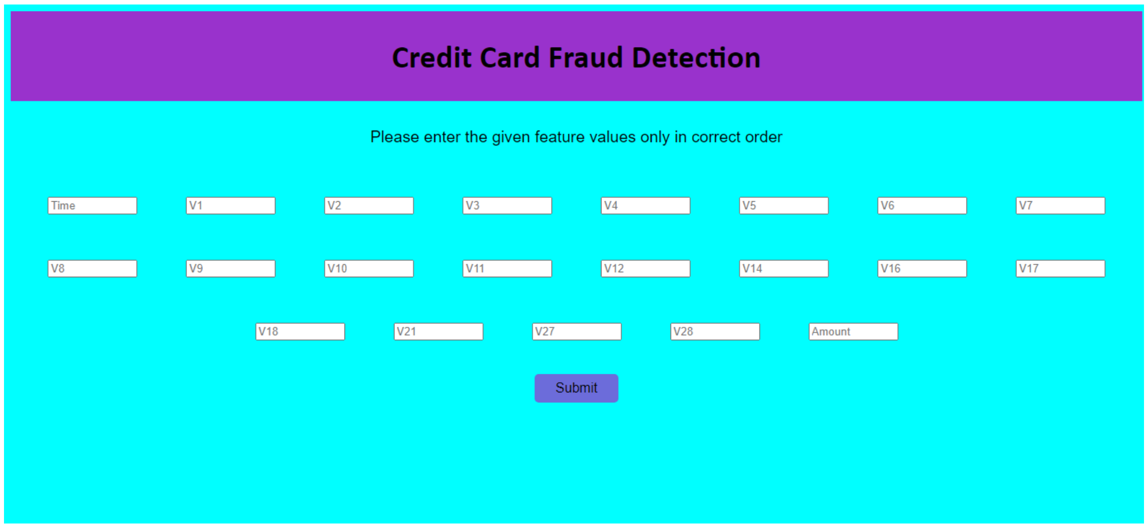
3.12 Model Call for specific input

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

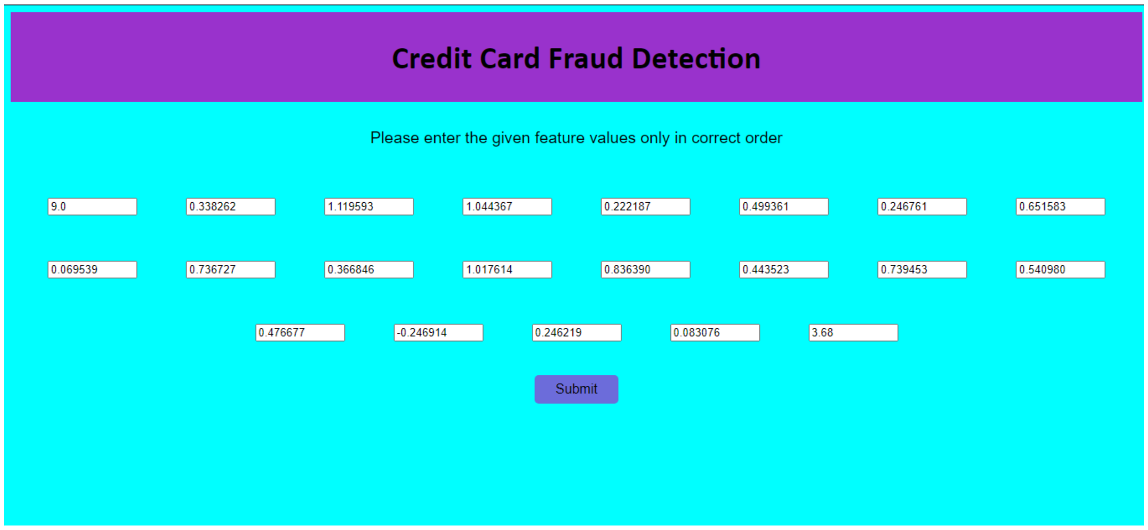
In Frontend creation I have made a user interactive page where users can enter their input values to our application. In their frontend page I have made a form which has beautiful styling with CSS. This HTML user input data is transferred in variable format to the backend. Made these html fully in a decoupled format.

Input Page :-



Credit Card Fraud Detection

Please enter the given feature values only in correct order

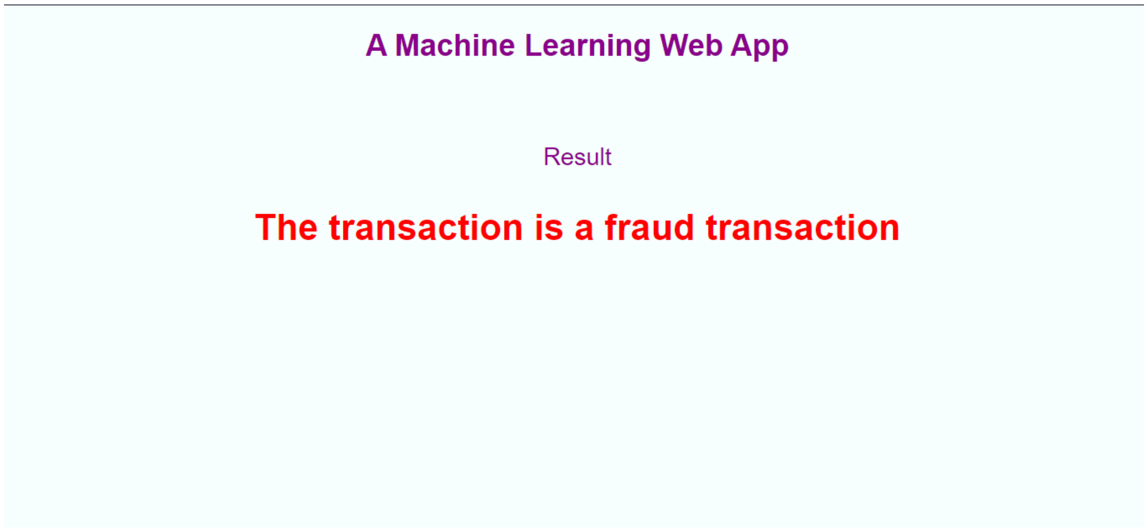


Credit Card Fraud Detection

Please enter the given feature values only in correct order

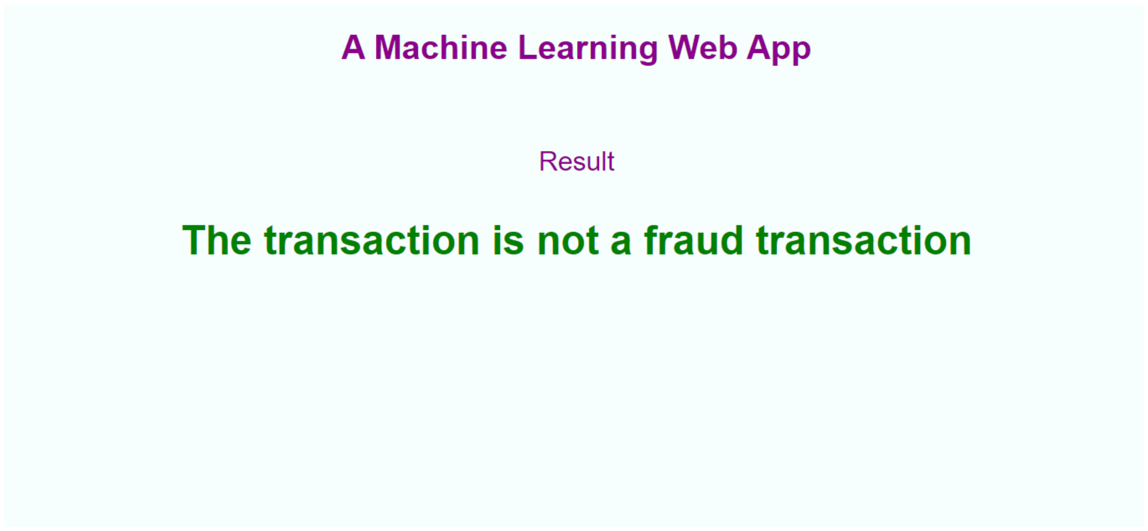
9.0	0.338262	1.119593	1.044367	0.222187	0.499361	0.246761	0.651583
0.069539	0.736727	0.366846	1.017614	0.836390	0.443523	0.739453	0.540980
0.476677	-0.246914	0.246219	0.083076	3.68			

Output page:


A Machine Learning Web App

Result

The transaction is a fraud transaction


A Machine Learning Web App

Result

The transaction is not a fraud transaction

3.14 Deployment

I will be deploying the model with the help of AWS and Heroku cloud platform.

This is a workflow diagram for the Credit Card Fraud Detection Prediction Application…..

4. Technology Stack

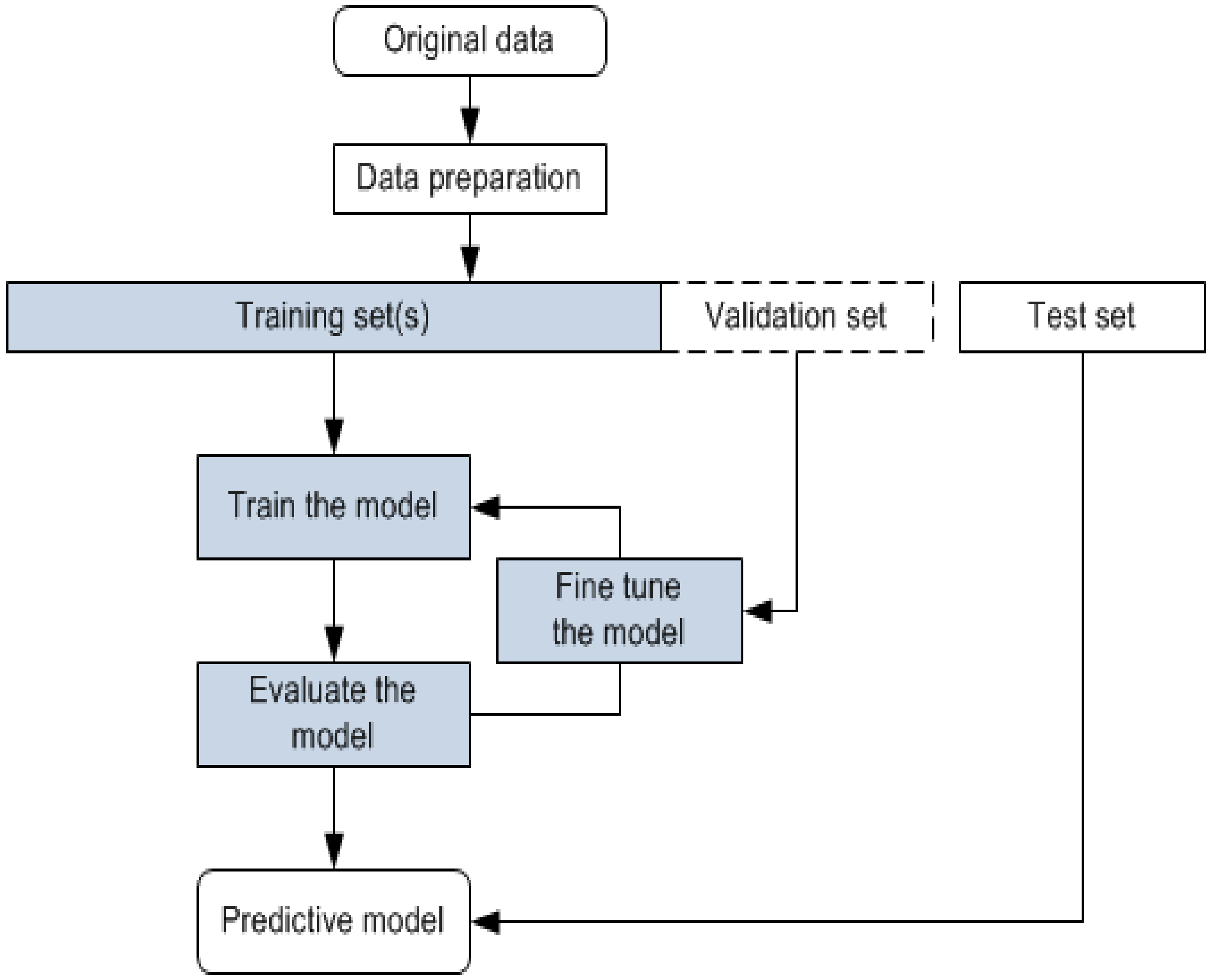
Front End	HTML/CSS
Back End	Flask, Pandas, Numpy, Scikit-learn etc
Database	Mongodb
Deployment	AWS,heroku

5. Unit Test Cases

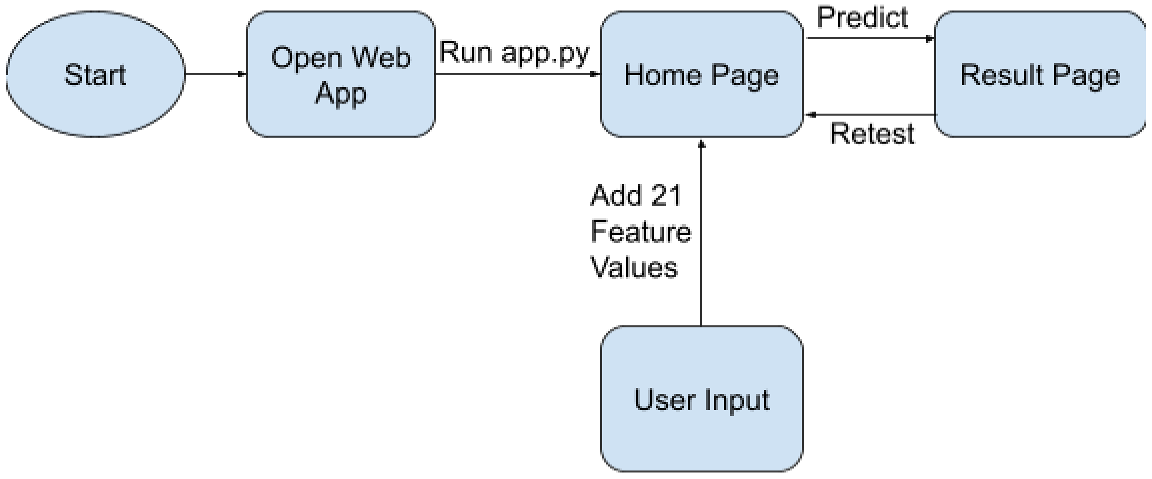
Test Case Description	Prerequisite	Expected Result
Verify whether the Application URL is accessible to the user.	1.Application URL should be defined	Application URL should be accessible to the user
Verify whether the Application loads completely for the user when the URL is accessed	1.Application URL is accessible 2.Application is deployed	The Application should load completely for the user when the URL is accessed.
Verify whether user is able to see input fields in logging in	1.Application is accessible 2.User is signed up 3.User successfully logged in to the app.	Users should be able to see the input field 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 recommended 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 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 signed up to the application 3. User is logged in to the application	The recommended results should be in accordance to the selections user made
Verify whether KPIs modify as per the user inputs for the particular transaction details.	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 particular transaction details.

6. Model training/validation workflow



7. I/O workflow



Credit Card Fraud Detection I/O work flow