Architecture Design

MONEY LAUNDERING PREVENTION

Document Control

Version	Date	Author	Comments
1	31.03.2022	Rohit Prasad	

Index

Abstract

Machine Learning is a category of algorithms that allows software applications to become more accurate in predicting outcomes without being explicitly programmed. The basic premise of machine learning is to build models and employ algorithms that can receive input data and use statistical analysis to predict an output while updating outputs as new data becomes available. These models can be applied in different areas and trained to match the expectations of management so that accurate steps can be taken to achieve the organization's target. In this project, we will predict the fraudulent case of money laundering onthe basis of some transactional data present. We will build a generalised solution which will help prevent future losses

1. Introduction

1.1 What is Architecture Design?

The goal of Architecture Design (AD) is to give the internal design of the actual program code for the `Money Laundering Prevention`. AD describes the class diagrams with the methods and relation between classes and program specification. It describes the modules so that the programmer can directly code the program from the document.

1.2 Scope

Architecture Design (AD) 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. And the complete workflow.

1.3 Constraints

The web app must be user friendly and should give the output in a desirable format to the users

2. Technical Specification

2.1 Dataset

The dataset contains the financial transaction data between some entities (Customer and Merchants). We have more than 6 Million data which comprises of 11 columns in total. The dataset contains some useful columns like type of transactions, amount involved in transaction, Origin id of the customer/merchants and Destination id of the customer/merchants, Balances (Before and after the transactions) and a target column which is the dependent variable which we need to predict.

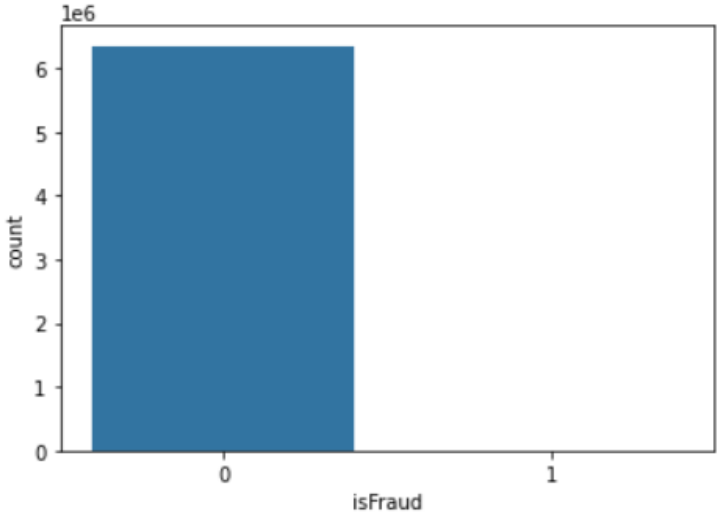
	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.0	0.00	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.0	0.00	0	0
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.0	0.00	1	0
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.0	0.00	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.0	0.00	0	0
5	1	PAYMENT	7817.71	C90045638	53860.00	46042.29	M573487274	0.0	0.00	0	0
6	1	PAYMENT	7107.77	C154988899	183195.00	176087.23	M408069119	0.0	0.00	0	0
7	1	PAYMENT	7861.64	C1912850431	176087.23	168225.59	M633326333	0.0	0.00	0	0
8	1	PAYMENT	4024.36	C1265012928	2671.00	0.00	M1176932104	0.0	0.00	0	0
9	1	DEBIT	5337.77	C712410124	41720.00	36382.23	C195600860	41898.0	40348.79	0	0
10	1	DEBIT	9644.94	C1900366749	4465.00	0.00	C997608398	10845.0	157982.12	0	0
11	1	PAYMENT	3099.97	C249177573	20771.00	17671.03	M2096539129	0.0	0.00	0	0
12	1	PAYMENT	2560.74	C1648232591	5070.00	2509.26	M972865270	0.0	0.00	0	0
13	1	PAYMENT	11633.76	C1716932897	10127.00	0.00	M801569151	0.0	0.00	0	0
14	1	PAYMENT	4098.78	C1026483832	503264.00	499165.22	M1635378213	0.0	0.00	0	0
15	1	CASH_OUT	229133.94	C905080434	15325.00	0.00	C476402209	5083.0	51513.44	0	0
16	1	PAYMENT	1563.82	C761750706	450.00	0.00	M1731217984	0.0	0.00	0	0
17	1	PAYMENT	1157.86	C1237762639	21156.00	19998.14	M1877062907	0.0	0.00	0	0
18	1	PAYMENT	671.64	C2033524545	15123.00	14451.36	M473053293	0.0	0.00	0	0
19	1	TRANSFER	215310.30	C1670993182	705.00	0.00	C1100439041	22425.0	0.00	0	0

onsists of various data types from integer to floating to object as shown in Fig.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 11 columns):
 #   Column          Dtype
---  ------          -----
 0   step            int64
 1   type            object
 2   amount          float64
 3   nameOrig        object
 4   oldbalanceOrg   float64
 5   newbalanceOrig  float64
 6   nameDest        object
 7   oldbalanceDest  float64
 8   newbalanceDest  float64
 9   isFraud         int64
 10  isFlaggedFraud  int64
dtypes: float64(5), int64(3), object(3)
memory usage: 534.0+ MB
```

The dataset provided have a huge imbalance nature (non fraud cases are more than the fraud ones). So some pre-processing steps and some sampling is done in order to gain more

insight from the data to predict the correct outcome. Dataset has been explored for various patterns checks. (Please Refer EDA section of the code in Github for more info)



(0 means non fraudulent and 1 means fraudulent)

A lot of pre-processing is done on the dataset such as splitting of columns, categorical encoding, one hot encoding, scaling of the numeric column values so we can get the data in some kind of similar ranges. Some not so required columns have been dropped. Suitable sampling method has been implemented on the data to finalize it and get it ready for the model training

2.2 Logging

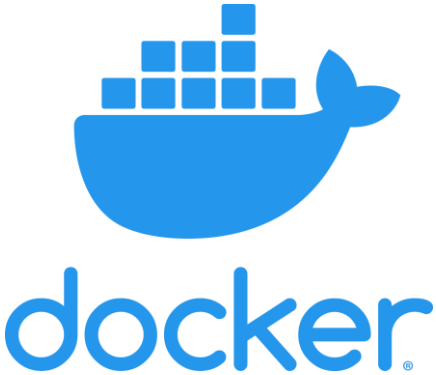
We should be able to log every activity done by the user

- The system identifies at which step logging require.
- The system should be able to log each and every system flow.

- The system should be not be hung even after using so much logging. Logging just because we can easily debug issuing so logging is mandatory to do.

2.3 Deployment

- For the hosting of the project, we will use AWS EC2 system.
- Also, we have made a docker image and pushed it to docker hub (the same will be called inside EC2 instance)



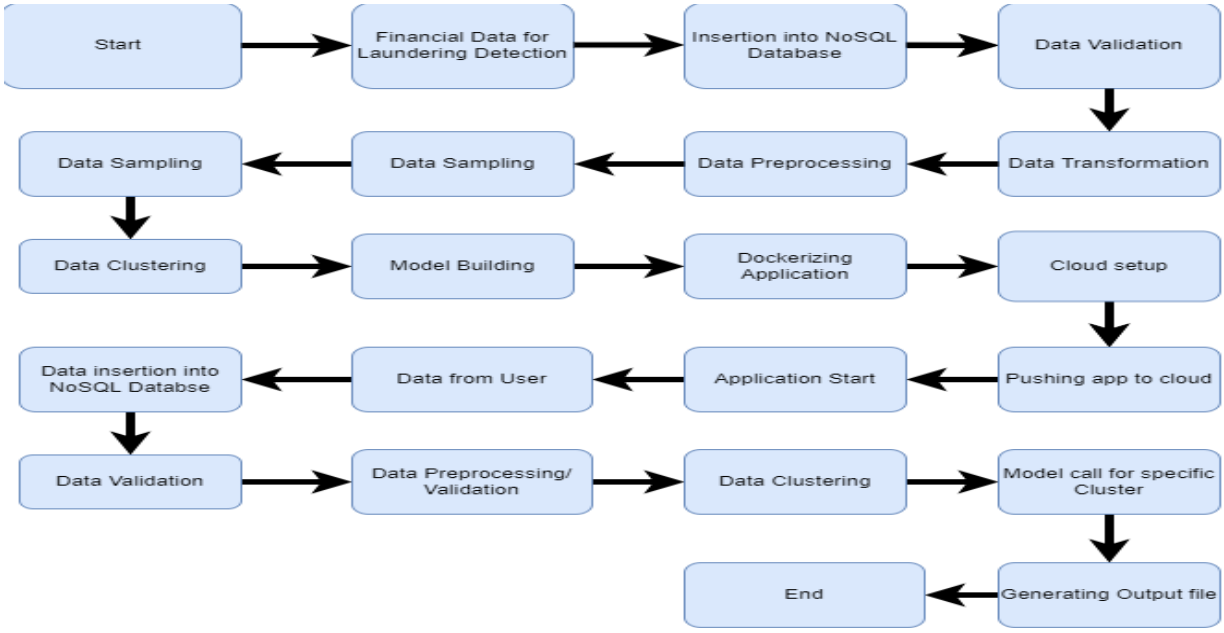
3. Technology Stack

Front End	Python/streamlit
Backend	Python/streamlit
Deployment	AWS EC2

4. Proposed Solution

We will use performed EDA to find the important relation between different attributes and will use a machine-learning algorithm to estimate the cost of expenses. The user will be required to upload a csv file as an input and will get results through the web application (through a csv file). The system will get features and it will be passed into the backend where the features will be validated and preprocessed and then it will be passed to a hyperparameter tuned machine learning model topredict the final outcome.

5. Architecture



5.1 Data Gathering

Data source: https://www.kaggle.com/datasets/ealaxi/paysim1

Dataset is stored in .csv format.

5.2 Raw Data Validation

After data is loaded, various types of validation are required before we proceed further with any operation. Validations like if the file is in the csv format, the columns have proper datatypes, the number of columns are exact as mentioned in the code etc. These data need to be validated before processing further with transformation and pre processing

5.3 Exploratory Data Analysis

Visualized the relationship between the dependent and independent features. Also checked relationship between independent features to get more insights about the data. For more information on EDA, check the code in the EDA section of github repository.

5.4 Feature Engineering/ Preprocessing

After the loading of the data successfully, preprocessing and feature engineering step has been done and the final data has been saved in dataframes for further apply it to the model to get some outcomes. Feature engineering involves the step of creating new feature by considering some patterns from the existing features. Pre-processing the data will get your data ready to be put in a specific model call.

5.5 Model Building

After doing all kinds of pre-processing/feature engineering operations mention above and performing scaling and encoding, the data set is passed through a pipeline to all the models, XGBClassifier, RandomForestClassifier and evaluation of all the model has been done with some some metric score (f1- score). The dataset was divided into n cluster by help of Kmeans clustering and each cluster had been passed through the models defined above to check which performed better wrt each clusters. (it was seen that randomforest performed better for each cluster in our project)

5.6 Model Saving

Model is saved using pickle library in pickle` format.

5.7 Web application with streamlit

After saving the model, the API building process started using Streamlit library. Web application creation was created in python. We provided the flexibility of loading csv files and through that users will be able to get the output in a csv format.

5.8 GitHub

The whole project directory will be pushed into the GitHub repository.

5.9 Deployment

The project was deployed into AWS EC2 instance with the help of docker image

6. User Input / Output Workflow.