

# Architecture Design

## Fraud Transaction Detection

**Document Control**

Version	Date	Author	Comments
1	08/2022	Shubham Chavhan	

## Index

Content
Abstract
1. Introduction
1.1 What is Architecture Design?
1.2 Scope
1.3 Constraints
2. Technical Specification
2.1 Dataset
2.2 Logging
2.3 Deployment
3. Technology Stack
4. Proposed Solution
5. Architecture

## **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 estimate the amount of insurance premium on the basis of personal health information. Taking various aspects of a dataset collected from people, and the methodology followed for building a predictive model.

## **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 Insurance Premium Prediction . 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**

We only predict the expected estimating cost of expenses customers based on some personal health information.

## **2. Technical Specification**

## Dataset

The dataset given by INeuron

**Name : Shubham Chavhan**

These two lines import the Pandas library and the glob module. The glob module is used in the code to read multiple CSV files from a directory that match a certain pattern.

```
In [1]: import pandas as pd
import glob
```

This block of code reads all the CSV files present in the "simulated-data-csv" directory and concatenates them into a single Pandas DataFrame object called "df". It does this by first creating an empty list called "all\_data". Then, for each CSV file in the directory, it reads the file using Pandas' read\_csv() function and appends the resulting DataFrame to the "all\_data" list. Finally, it concatenates all the DataFrames in the "all\_data" list into a single DataFrame using Pandas' concat() function.

```
In [2]: all_data = []
```

```
In [3]: for i in glob.glob("simulated-data-csv/*.csv"):
data = pd.read_csv(i)
all_data.append(data)
```

```
In [4]: df = pd.concat(all_data, ignore_index=True)
```

This line of code displays the first 5 rows of the DataFrame "df" to verify that the data was loaded correctly.

```
In [5]: df.head()
```

```
Out[5]:
```

	TRANSACTION_ID	TX_DATETIME	CUSTOMER_ID	TERMINAL_ID	TX_AMOUNT	TX_TIME_SECONDS	TX_TIME_DAYS	TX_FRAUD	TX_FRAUD_SCENARIO
0	0	2018-04-01 00:00:31	596	3156	37.16	31	0	0	0
1	1	2018-04-01 00:00:10	4961	3412	81.51	130	0	0	0
2	2	2018-04-01 00:07:56	2	1365	146.00	476	0	0	0
3	3	2018-04-01 00:09:29	4128	8737	64.49	569	0	0	0
4	4	2018-04-01 00:10:34	927	9906	50.99	634	0	0	0

The data set consists of various data types from integer to floating to object as shown in Fig.

```
Out[5]:
```

	TRANSACTION_ID	TX_DATETIME	CUSTOMER_ID	TERMINAL_ID	TX_AMOUNT	TX_TIME_SECONDS	TX_TIME_DAYS	TX_FRAUD	TX_FRAUD_SCENARIO
0	0	2018-04-01 00:00:31	596	3156	37.16	31	0	0	0
1	1	2018-04-01 00:00:10	4961	3412	81.51	130	0	0	0
2	2	2018-04-01 00:07:56	2	1365	146.00	476	0	0	0
3	3	2018-04-01 00:09:29	4128	8737	64.49	569	0	0	0
4	4	2018-04-01 00:10:34	927	9906	50.99	634	0	0	0

This line of code displays information about the DataFrame "df", such as the number of rows and columns, the data types of the columns, and the amount of memory used by the DataFrame.

```
In [6]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1754155 entries, 0 to 1754154
Data columns (total 9 columns):
#   Column                Dtype
---  ---
0   TRANSACTION_ID         int64
1   TX_DATETIME            object
2   CUSTOMER_ID            int64
3   TERMINAL_ID            int64
4   TX_AMOUNT              float64
5   TX_TIME_SECONDS        int64
6   TX_TIME_DAYS           int64
7   TX_FRAUD               int64
8   TX_FRAUD_SCENARIO      int64
dtypes: float64(1), int64(7), object(1)
memory usage: 120.4+ MB
```

This line of code checks for missing values in the DataFrame "df" and returns the sum of missing values for each column.

```
In [7]: df.isnull().sum()
```

```
Out[7]:
```

	TRANSACTION_ID	TX_DATETIME	CUSTOMER_ID	TERMINAL_ID	TX_AMOUNT	TX_TIME_SECONDS	TX_TIME_DAYS	TX_FRAUD	TX_FRAUD_SCENARIO
TRANSACTION_ID	0								
TX_DATETIME	0								
CUSTOMER_ID	0								
TERMINAL_ID	0								
TX_AMOUNT	0								
TX_TIME_SECONDS	0								
TX_TIME_DAYS	0								
TX_FRAUD	0								
TX_FRAUD_SCENARIO	0								

This line of code drops some columns from the DataFrame "df" that are not needed for the fraud detection model. axis=1 parameter specifies that the columns should be dropped, and the inplace=True parameter specifies that the changes should be made to the DataFrame "df" itself.

The screenshot displays a Jupyter Notebook interface with the following content:

```
In [9]: df.head()
```

	TX_AMOUNT	TX_TIME_SECONDS	TX_FRAUD	TX_FRAUD_SCENARIO
0	57.16	31	0	0
1	81.51	130	0	0
2	146.00	476	0	0
3	64.49	569	0	0
4	50.99	634	0	0

This block of code prints out the unique values for each column in the DataFrame "df". This is useful to see what the range of values is for each feature.

```
In [10]: for i in df:
print(i, "-", df[i].unique())
```

TX\_AMOUNT - [ 57.16 81.51 146. ... 569.4 109.19 358.2 ]  
TX\_TIME\_SECONDS - [ 31 130 476 ... 15811181 15811192 15811197 ]  
TX\_FRAUD - [ 0 1 ]  
TX\_FRAUD\_SCENARIO - [ 0 1 3 2 ]

These two lines of code split the DataFrame "df" into two parts: the features (stored in "X") and the target variable (stored in "y"). The target variable is the "TX\_FRAUD\_SCENARIO" column, and it is dropped from the features because it is the variable we want to predict.

```
In [11]: X = df.drop("TX_FRAUD_SCENARIO", axis = 1)
y = df.TX_FRAUD_SCENARIO
```

This line of code splits the DataFrame ("X") and target variable ("y") into training and testing sets. The training set contains 80% of the data, and the testing set contains 20% of the data. The random\_state=42 parameter ensures that the data is split in a reproducible way.

```
In [12]: from sklearn.model_selection import train_test_split
```

```
In [13]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
```

These lines of code import the logistic regression algorithm from scikit-learn and use

```
In [14]: from sklearn.linear_model import LogisticRegression
```

```
In [15]: lr = LogisticRegression(max_iter=100000)
lr.fit(X_train, y_train)
lr.score(X_test, y_test)
```

Out[15]: 0.9918450764043886

```
In [16]: from sklearn.naive_bayes import BernoulliNB
```

```
In [17]: berno = BernoulliNB()
berno.fit(X_train, y_train)
berno.score(X_test, y_test)
```

Out[17]: 0.9969073428516865

```
In [18]: from sklearn.naive_bayes import MultinomialNB
```

```
In [19]: multi = MultinomialNB()
multi.fit(X_train, y_train)
multi.score(X_test, y_test)
```

Out[19]: 0.802828142324937

```
In [20]: from sklearn.model_selection import GridSearchCV
```

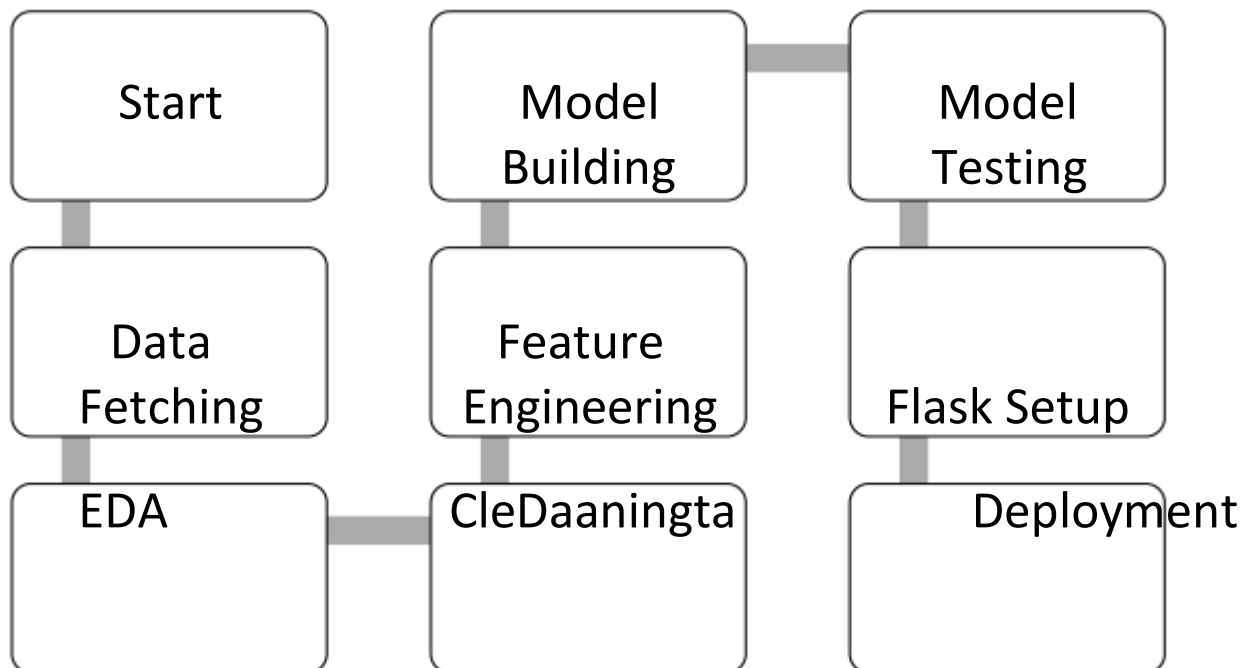
```
In [21]: param_grid = {
'alpha': [0.1, 0.5, 1.0, 2.0],
'binarize': [0.0, 0.5, 1.0]
}
```

```
In [22]: grid_search = GridSearchCV(berno, param_grid, cv=5)
```

## 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 client will be filled the required feature as input and will get results through the web application. 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 to predict the final outcome.

## 5. Architecture



### 5.1 Data Gathering

Data source: [https://github.com/shubhamchavhan271/Fraud-transaction-detection\\_shubham/blob/main/Fraud%20detection%20INeuron-shubham.ipynb](https://github.com/shubhamchavhan271/Fraud-transaction-detection_shubham/blob/main/Fraud%20detection%20INeuron-shubham.ipynb)

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 checking for zero standard deviation for all the columns, checking for complete missing values in any columns, etc. These are required because the attributes which contain these are of no use. It will not play role in contributing to the estimating cost of the premium.

## 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.

## 5.4 Feature Engineering

After pre-processing standard scalar is performed to scale down all the numeric features. Even one hot encoding is also performed to convert the categorical features into numerical features. For this process, pipeline is created to scale numerical features and encoding the categorical features.

.

## 5.8 GitHub

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

## 5.9 Deployment

The project was deployed from GitHub into the Heroku platform.

# 6. User Input / Output Workflow.



