Online Payment Fraud Detection using Machine Learning

Project Description

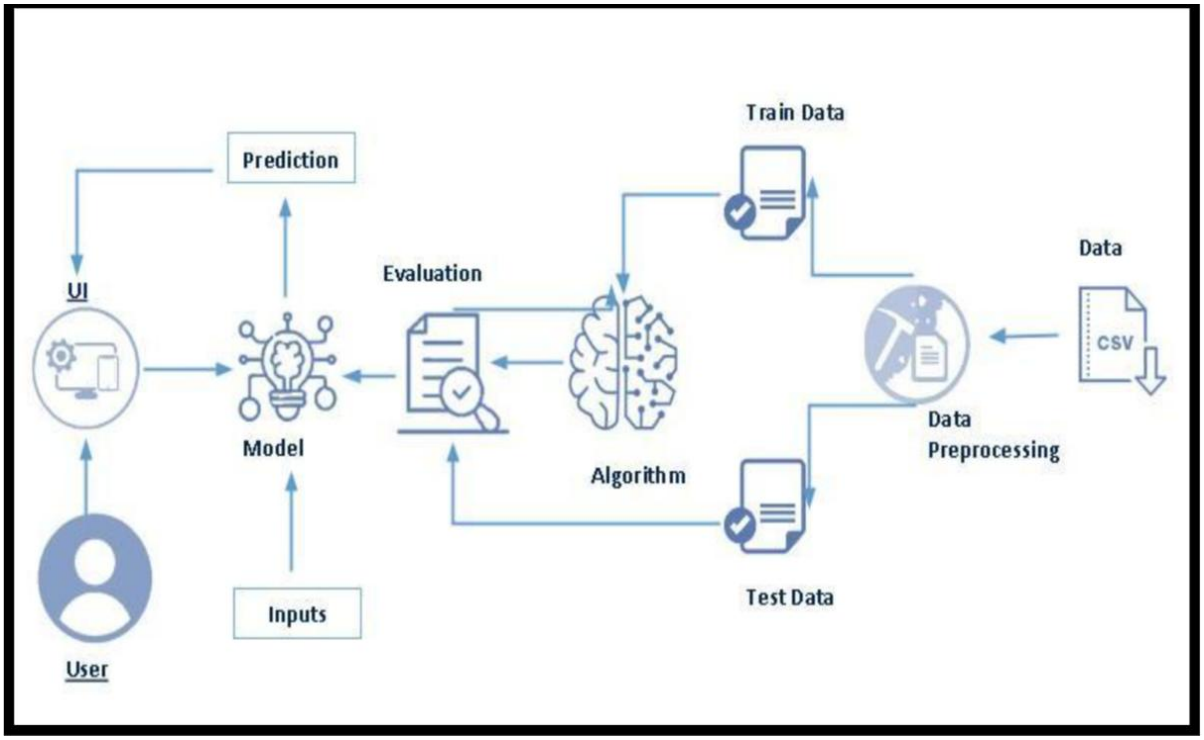
One of the most important factors that affects the digital economy and financial security is the safety of online transactions. With the rapid growth of online payment systems such as credit cards, digital wallets, and internet banking, the risk of fraudulent transactions has also increased significantly. Fraud detection is a crucial task for financial institutions and payment platforms to protect customers and prevent financial losses.

Detecting fraudulent transactions manually is a difficult and time-consuming process because of the large volume of transactions and complex fraud patterns. Therefore, machine learning techniques play an important role in identifying suspicious activities automatically. By analyzing transaction data and identifying unusual patterns, fraud detection systems can help reduce financial risks and improve transaction security.

In this project, we develop a machine learning model to detect whether an online payment transaction is fraudulent or legitimate based on transaction details. Various classification algorithms such as Logistic Regression, Decision Tree, and Random Forest are used to train and test the model. The best performing model is selected and saved in pickle (.pkl) format.

The trained model is integrated with a Flask web application, where users can enter transaction details through a web interface and get real-time fraud prediction results. This system helps financial organizations detect fraud efficiently and improve secure online payment processing.

Technical architecture:



Pre requisites:

To complete this project, you must required following software's, concepts and packages

- **Anaconda navigator and PyCharm:**
- Refer the link below to download anaconda navigator
- Link : https://youtu.be/1ra4zH2G4o0
- **Python packages:**
- Open anaconda prompt as administrator
- Type "pip install numpy" and click enter.
- Type "pip install pandas" and click enter.
- Type "pip install scikit-learn" and click enter.
- Type "pip install matplotlib" and click enter.
- Type "pip install scipy" and click enter.
- Type "pip install pickle-mixin" and click enter.
- Type "pip install seaborn" and click enter.
- Type "pip install Flask" and click enter.

Prior Knowledge:

You must have prior knowledge of following topics to complete this project.

- **ML Concepts**
- Supervised learning: https://www.javatpoint.com/supervised-machine-learning
- Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning
- Regression and classification
- Decision tree: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm
- Random forest: https://www.javatpoint.com/machine-learning-random-forest-algorithm
- KNN: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
- Xgboost: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/

- Evaluation metrics: https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/
- **Flask Basics** : https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Objectives:

By the end of this project you will:

- Know fundamental concepts and techniques used for machine learning.
- Gain a broad understanding about data.
- Have knowledge on pre-processing the data/transformation techniques on outlier and some visualization concepts.

Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

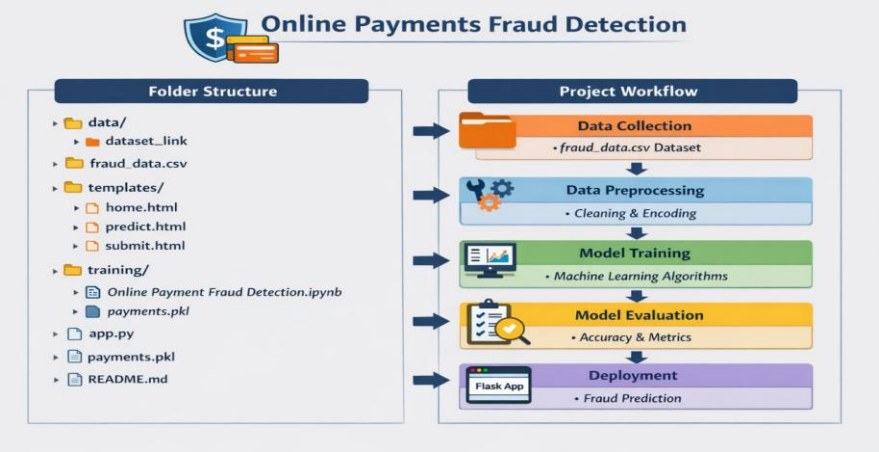
To accomplish this, we have to complete all the activities listed below,

- Data collection
- Collect the dataset or create the dataset
- Visualizing and analyzing data
- Univariate analysis
- Bivariate analysis
- Multivariate analysis
- Descriptive analysis
- Data pre-processing
- Checking for null values
- Handling outlier
- Handling categorical data
- Splitting data into train and test
- Model building
- Import the model building libraries
- Initializing the model

- Training and testing the model
- Evaluating performance of model
- Save the model
- Application Building
- Create an HTML file
- Build python code

Project Structure:

Create the Project folder which contains files as shown below



- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- rdf.pkl is our saved model. Further we will use this model for flask integration.
- Training folder contains model training files and training_ibm folder contains IBM deployment files.

Milestone 1: Data Collection

Machine Learning depends heavily on data. It is the most crucial aspect that makes algorithm training possible. This milestone focuses on downloading the required dataset for the project.

Activity 1: Download the Dataset

There are many popular open sources for collecting data, such as Kaggle, UCI Repository, etc.

For this project, we have used **fraud_data.csv**, which contains transaction data for online payment fraud detection.

Dataset Link:
[Download the dataset]

Milestone 2: Exploratory Data Analysis (EDA)

Exploratory Data Analysis helps us understand the dataset before training the machine learning model. In this project, we analyzed numerical and categorical features to detect patterns related to fraudulent transactions.

Activity 1: Importing the libraries

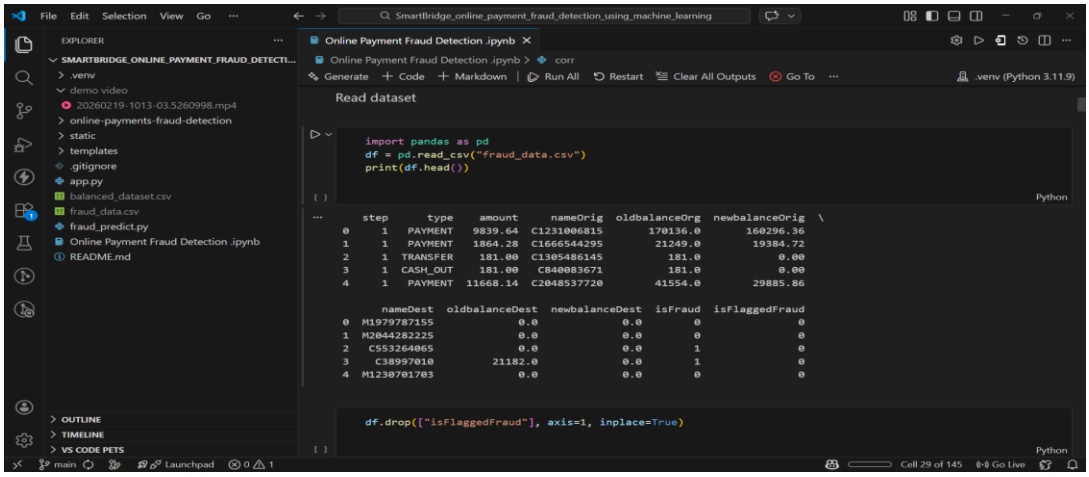
Import the necessary libraries as shown in the image. (optional) Here we have used visualization style as fivethirtyeight.

```
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import sklearn
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import RandomizedSearchCV
import imblearn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_score
```

Activity 2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

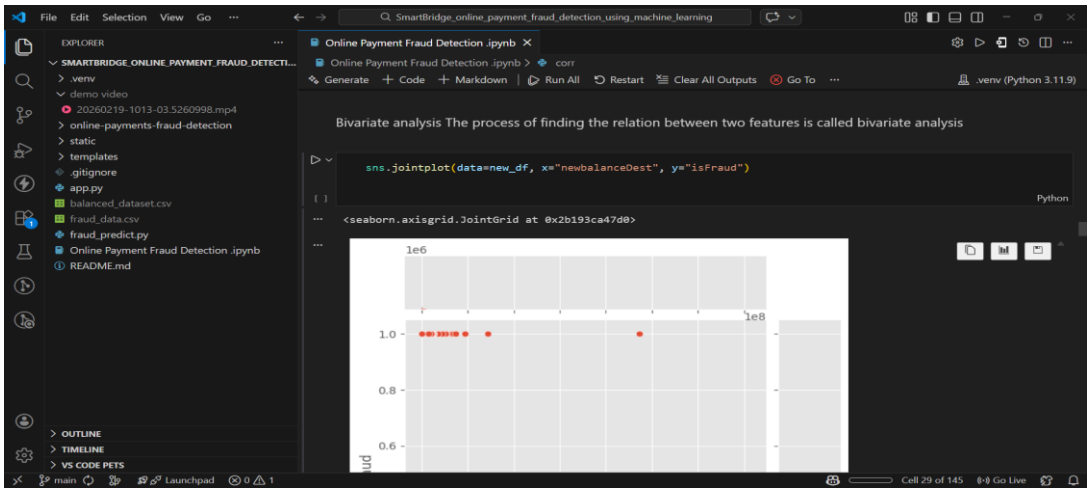
In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of csv file.



Activity 3: Bivariate Analysis

Bivariate analysis studies the **relationship between two features**.

- We used **countplots** and **bar graphs** to compare categorical features against each other.



Examples and Insights:

- Segmenting **Gender vs Fraud Label** to see which gender has more fraudulent transactions.
- Segmenting **Education vs Fraud Label** to observe if education affects fraud probability.
- Segmenting **Credit History vs Transaction Amount** to see how prior credit experience relates to high-value transactions.

From the above graph we can infer the analysis such as

- Segmenting the gender column and married column based on bar graphs
- Segmenting the Education and Self-employed based on bar graphs ,for drawing insights such as educated people are employed.
- Loan amount term based on the property area of a person holding

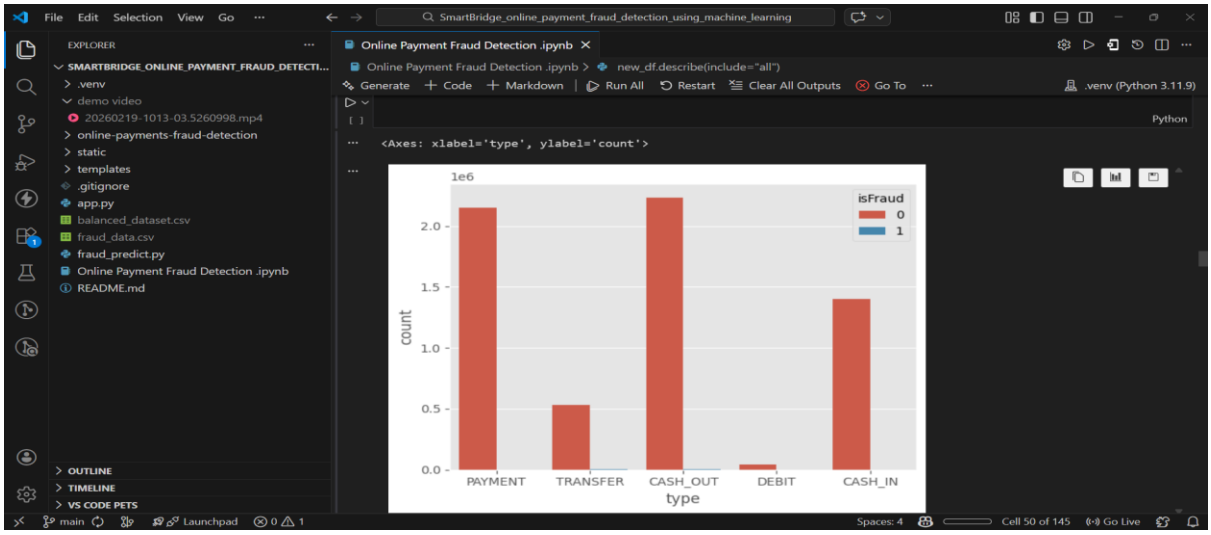
Activity4:Countplot of Transaction Type vs Fraud Label

sns.countplot(data=new_df, x="type", hue="isFraud")

Explanation:

- **x = "type"**: Shows the different types of transactions (e.g., CASH_IN, CASH_OUT, TRANSFER, etc.).
- **hue = "isFraud"**: Separates the transactions into:
  - **0 → Legal / Legitimate transactions**
  - **1 → Fraudulent transactions**

- This plot helps us **identify which transaction types have higher fraud occurrences**, providing insights for fraud detection.



MileStone3: Descriptive Analysis:

1. step

- **Description:** Represents the time step of the transaction (in hours).
- **Type:** Numerical (integer)
- **Observation:** Shows the chronological order of transactions. Useful to analyze **fraud trends over time**.

2. type

- **Description:** Type of transaction (e.g., CASH_IN, CASH_OUT, TRANSFER, PAYMENT, DEBIT).
- **Type:** Categorical
- **Observation:** Some transaction types, such as **TRANSFER** or **CASH_OUT**, are more likely to be fraudulent.

3. oldbalanceOrig

- **Description:** Initial balance of the origin account before the transaction.
- **Type:** Numerical (float)
- **Observation:** Fraudulent transactions often occur in accounts with **low or zero balances**.

4. newbalanceOrig

- **Description:** Balance of the origin account after the transaction.
- **Type:** Numerical (float)

- **Observation:** Shows how the transaction affects the account. Large drops may indicate suspicious activity.

5. oldbalanceDest

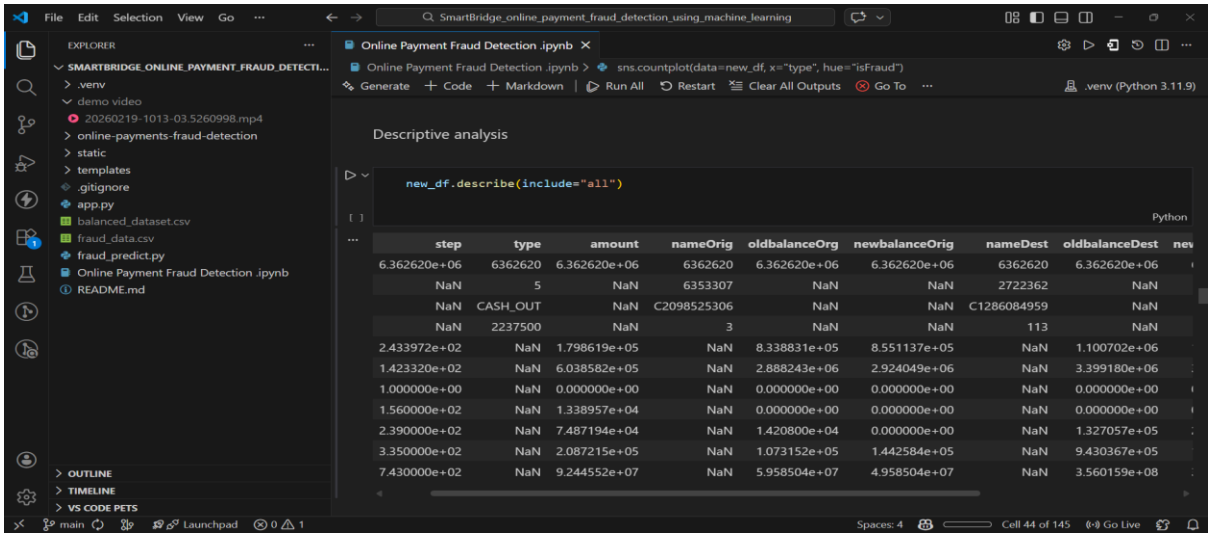
- **Description:** Initial balance of the destination account before the transaction.
- **Type:** Numerical (float)
- **Observation:** Used to detect unusual inflow to accounts (may be linked to fraud).

6. newbalanceDest

- **Description:** Balance of the destination account after the transaction.
- **Type:** Numerical (float)
- **Observation:** Large increases in balance in a short time may indicate fraudulent behavior.

7. isFraud

- **Description:** Fraud label
- **Type:** Categorical (0 = Legitimate, 1 = Fraud)
- **Observation:** Target variable for prediction. Shows that fraudulent transactions are rare compared to legitimate ones.



Milestone 4: Data Preprocessing

Data preprocessing is a **crucial step** in any machine learning project. It ensures that the dataset is **clean, consistent, and ready** for model training. In this project, we performed preprocessing on both **numerical** and **categorical** features.

1. Handling Missing Values

- Checked the dataset for missing values in all columns.

- Filled missing values or removed rows if necessary to avoid errors during model training.

2. Encoding Categorical Features

- Machine learning models require **numerical input**, so categorical variables must be encoded.
- Used LabelEncoder from **Scikit-learn** to convert categories into numbers.

Features encoded:

- **type** → Transaction type
- **isFraud** → Target variable (0 = Legitimate, 1 = Fraud)

3. Feature Scaling

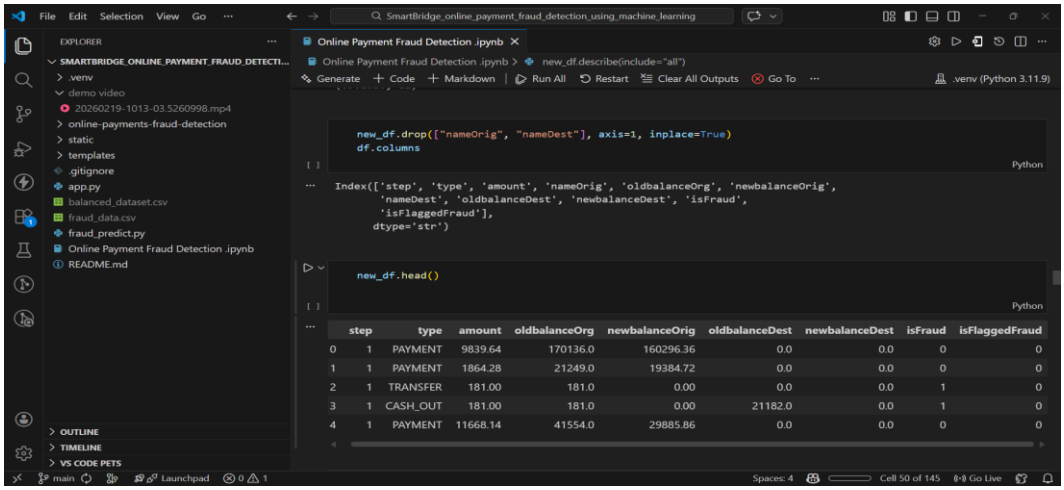
- Scaled numerical features to bring them to a **similar range**.
- Helps algorithms like **SVM** or **Random Forest** perform better.

Features scaled:

- oldbalanceOrig, newbalanceOrig, oldbalanceDest, newbalanceDest, amount

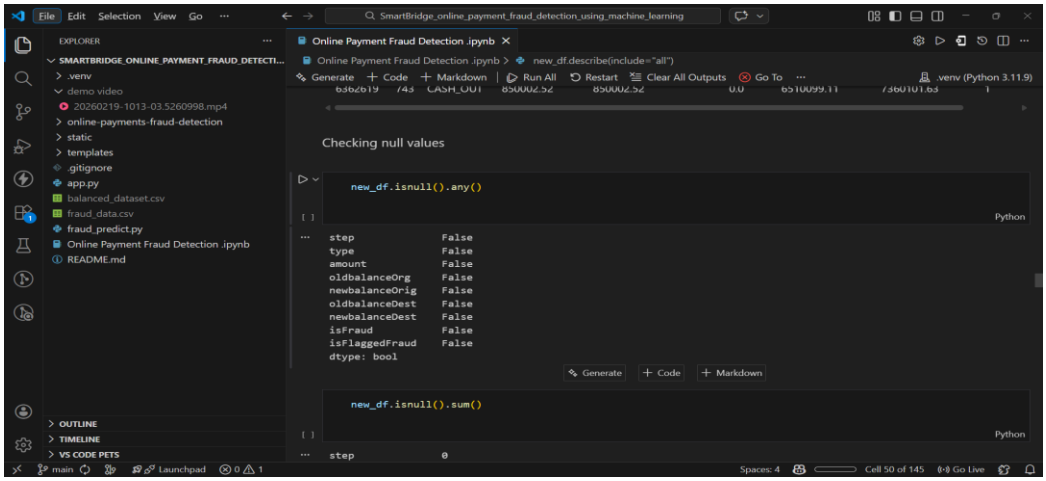
4. Train-Test Split

- Split the dataset into **training** and **testing** sets to evaluate model performance.
- Common ratio: 80% training, 20% testing.



Milestone 5: Checking Null Values

Checking for **missing or null values** is a crucial step before training any machine learning model. Null values can **cause errors** during model training and affect predictions, so they need to be handled properly.



Activity: Identify Null Values

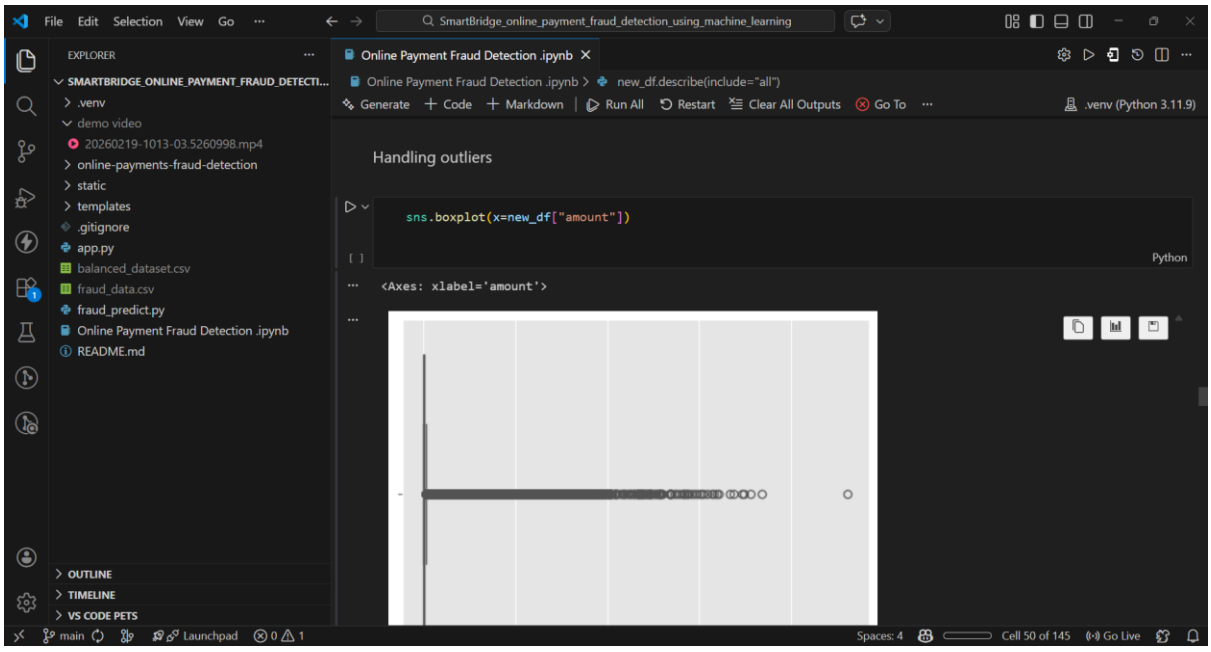
- We used the isnull() and sum() functions in Pandas to identify any missing values in the dataset.

Observation:

- After running the above code, we found that **there were no null values** in the dataset.
- This indicates that the dataset is **clean and complete**, and no additional imputation or removal of rows is required.

Milestone 6: Handling Outliers

Outliers are **extreme values** in the dataset that differ significantly from other observations. They can **negatively affect machine learning models**, especially algorithms sensitive to feature ranges, such as **SVM or KNN**.



Activity: Detect and Handle Outliers

1. Detecting Outliers

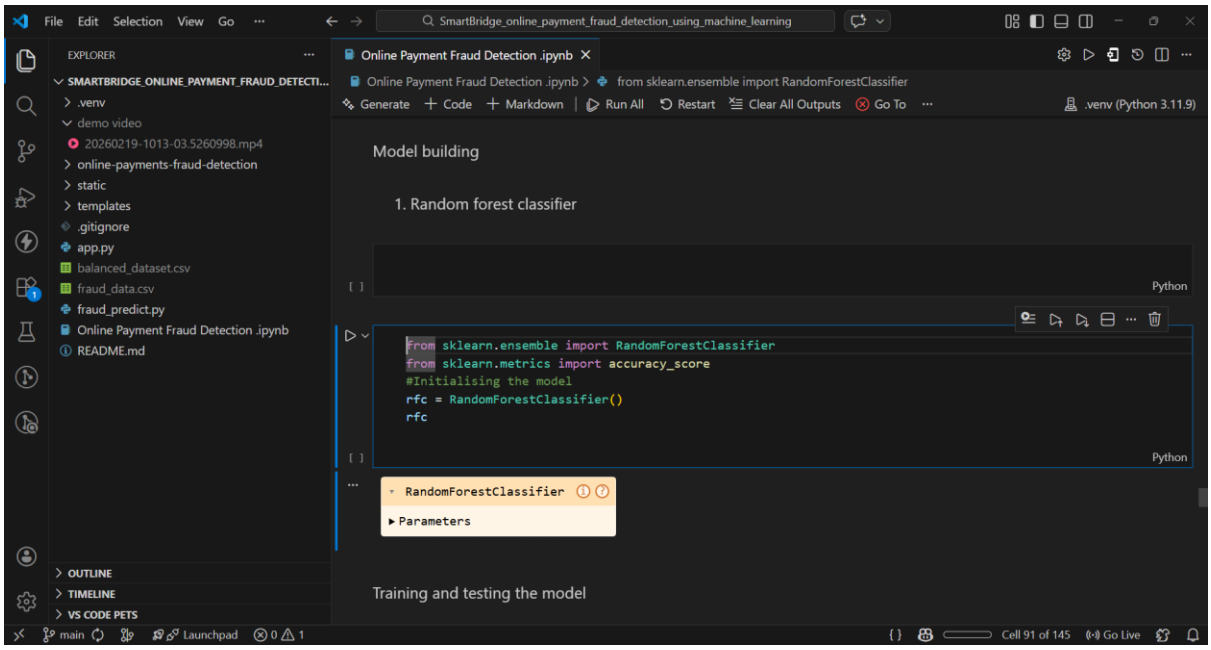
- We used **boxplots** to visualize numerical features and identify outliers.
- Features checked for outliers:
    - amount (Transaction Amount)
    - oldbalanceOrig (Origin Account Balance before transaction)
    - newbalanceOrig (Origin Account Balance after transaction)
    - oldbalanceDest (Destination Account Balance before transaction)
    - newbalanceDest (Destination Account Balance after transaction)

Observation:

- Some transactions have **extremely high amounts** compared to the majority.
- These could represent **potential fraudulent activities**.

Milestone 7: Model Building

After **data preprocessing, handling missing values, and outliers**, the dataset is ready for **model training**. In this project, we used multiple **supervised machine learning algorithms** to predict fraudulent transactions.



1. Features and Target

- **Features (X):** All columns except isFraud
- **Target (y):** isFraud column (0 = Legitimate, 1 = Fraud)

2. Train-Test Split

- Split the dataset into **training (80%)** and **testing (20%)** sets to evaluate model performance.

3. Machine Learning Models Used

We trained **multiple models** to compare performance:

1. **Random Forest Classifier** – Ensemble method using multiple decision trees.
2. **Decision Tree Classifier** – Simple tree-based classifier for easy interpretation.
3. **Extra Trees Classifier** – Ensemble of randomized decision trees, faster than Random Forest.
4. **Support Vector Classifier (SVC)** – Good for high-dimensional data.

4. Model Evaluation

We evaluated models using:

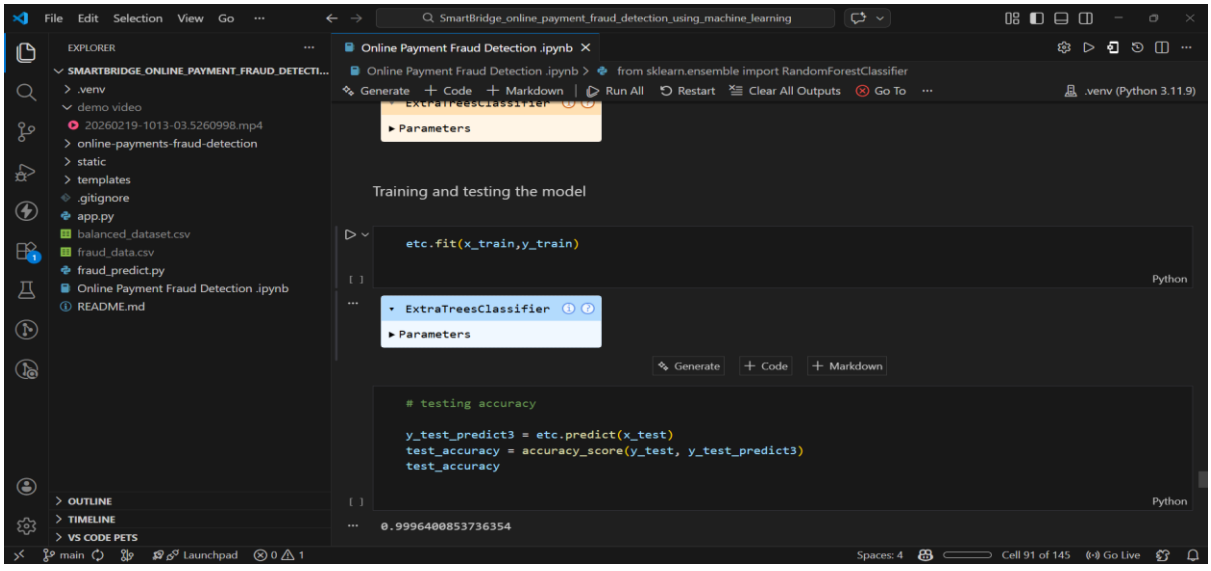
- **Accuracy Score** – Measures overall correctness of predictions.
- **Confusion Matrix** – Shows True Positives, True Negatives, False Positives, and False Negatives.

5. Saving the Model

- Once the model is trained and tested, we save it using **pickle** for deployment in the Flask app.

Milestone 8: Training and Testing the Model

After preprocessing the dataset and handling outliers, the next step is to **train the model** and **test its performance**. This ensures that the machine learning model can **accurately predict fraudulent transactions**.



1. Splitting the Dataset

- The dataset is divided into **features (X)** and **target (y)**.
    - **Features (X):** All columns except isFraud
    - **Target (y):** isFraud column (0 = Legitimate, 1 = Fraud)
- The dataset is then split into **training (80%)** and **testing (20%)** sets using train_test_split from Scikit-learn.

2. Training the Model

- Multiple machine learning models can be trained. Here, **Random Forest Classifier** is used as an example.

Observation:

- The model learns the **patterns and relationships** between transaction features and the isFraud label.

3. Testing the Model

- The trained model is evaluated on the **testing dataset** to measure performance on unseen data.

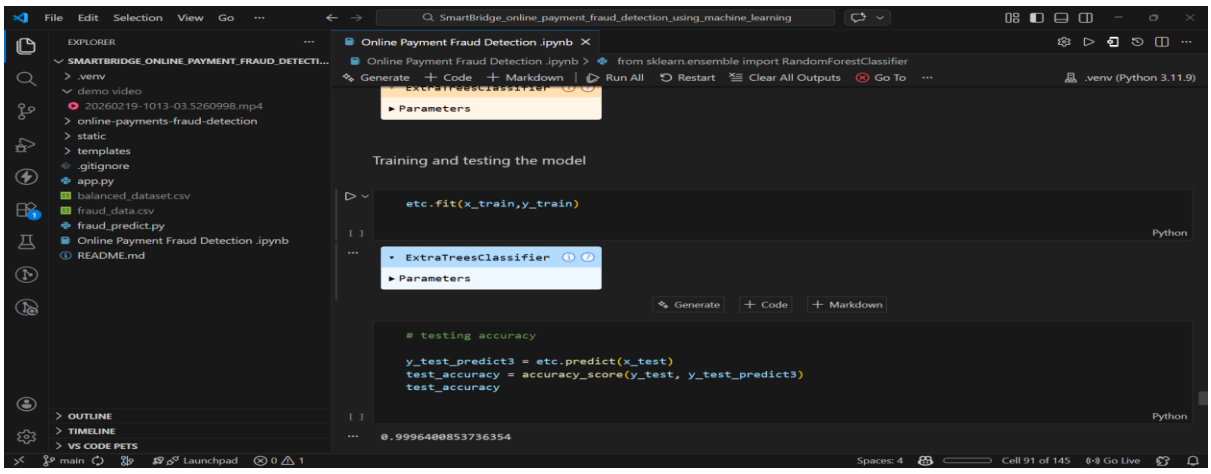
Evaluation Metrics:

1. **Accuracy Score** – Overall correctness of predictions.
2. **Confusion Matrix** – Counts of True Positives, True Negatives, False Positives, and False Negatives.

Observation:

- The accuracy shows how well the model predicts fraud.
- The confusion matrix highlights **fraud detection vs. legitimate transaction predictions**, which is crucial because fraud cases are **rare**.

Milestone 9: Support Vector Machine (SVM) Classifier



The **Support Vector Machine (SVM)** is a **supervised machine learning algorithm** used for classification tasks. In this project, it is applied to detect **fraudulent transactions**.

1. Overview of SVM

- SVM works by finding a **hyperplane** that best separates the classes (fraud vs. legitimate transactions) in the feature space.
- It is particularly effective for **high-dimensional data** and can handle **non-linear relationships** using kernel functions.

2. Training the SVM Model

- The SVM model is trained on the **training dataset**.
- Features are scaled beforehand because SVM is **sensitive to feature ranges**.

3. Testing and Evaluation

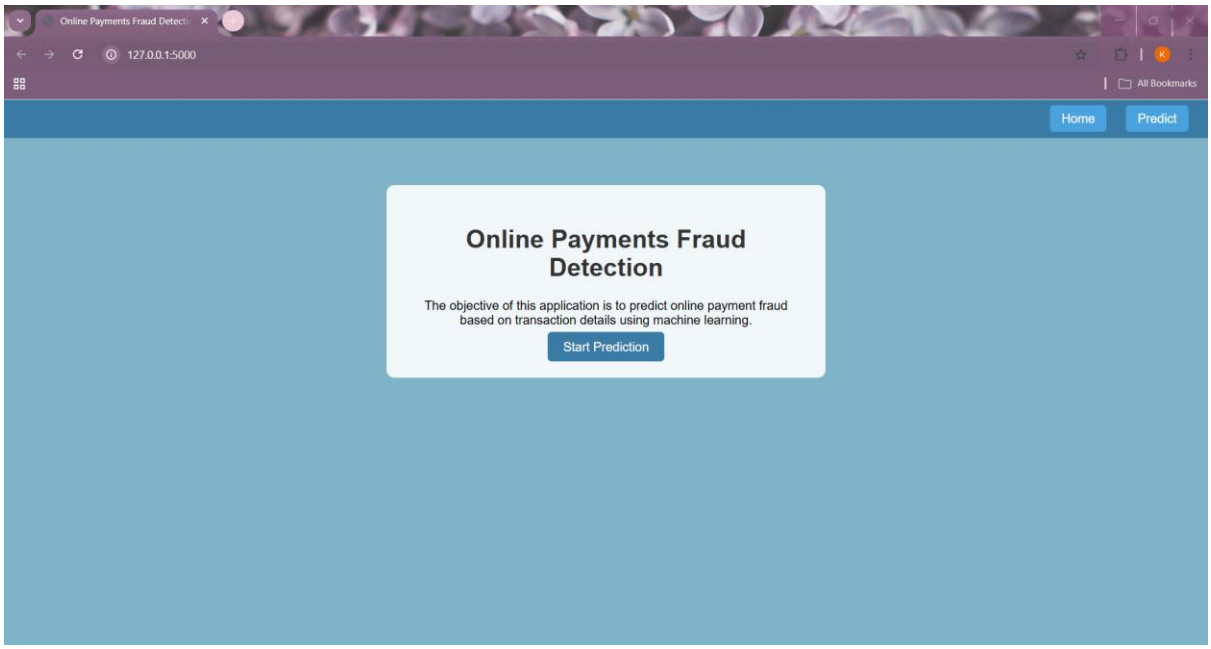
- Predictions are made on the **test dataset** and evaluated using **accuracy and confusion matrix**.

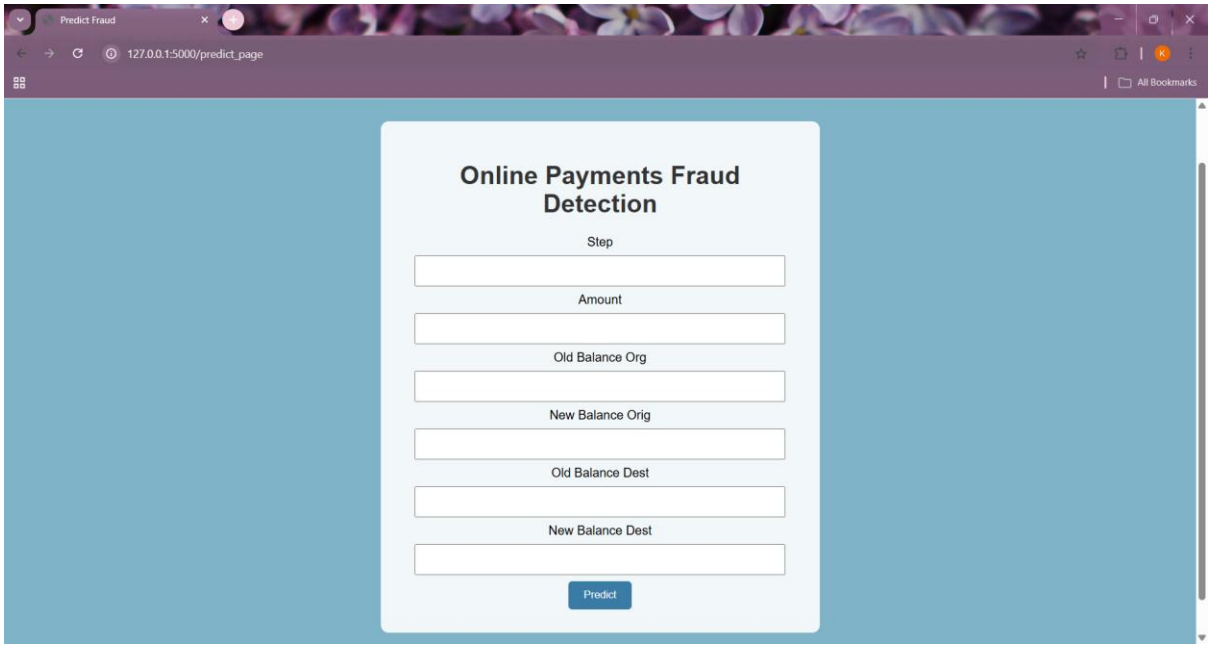
Observation:

- SVM provides **good separation** between fraudulent and legitimate transactions.
- The confusion matrix helps identify **false negatives**, which are critical in fraud detection.

Milestone 10: Web Application

After training and testing the machine learning model, the next step is **deployment** through a **web application**. This allows users to **enter transaction details** and get **real-time fraud predictions**.



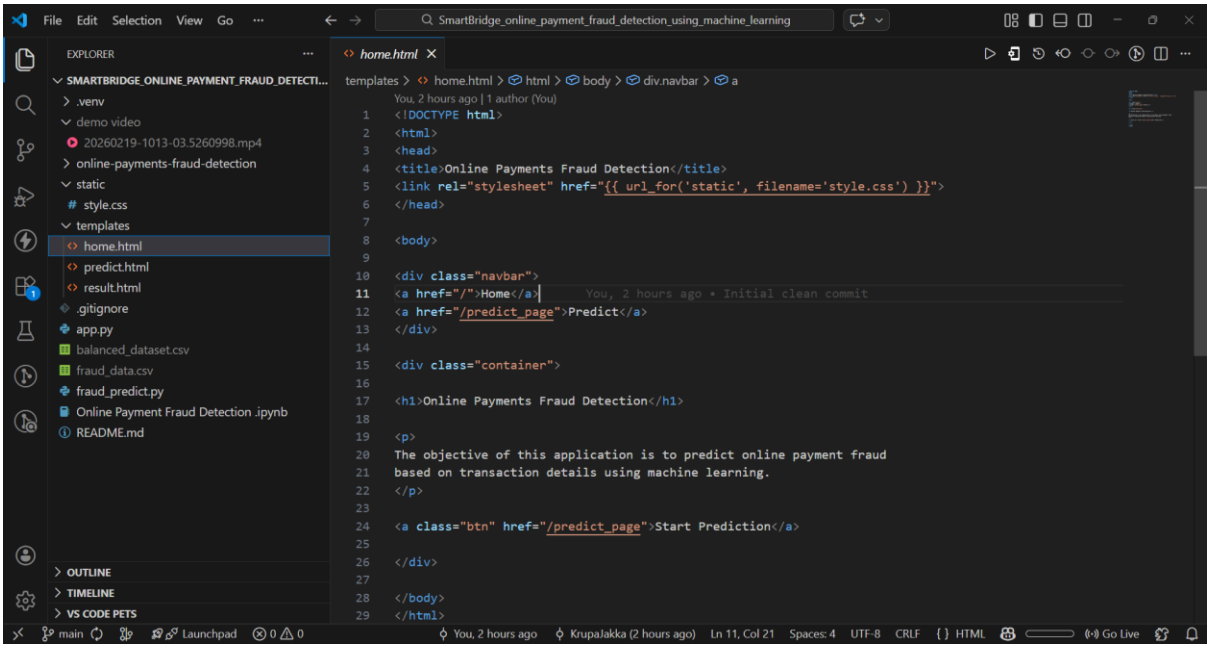


1. Overview

- The project uses **Flask**, a lightweight Python web framework, to create the web interface.
- Users interact with a **simple HTML form** to submit transaction details.
- The trained model (**Random Forest / SVM**) predicts whether the transaction is **fraudulent (1)** or **legitimate (0)**.

2. Workflow of the Web Application

1. **User Input:**
    - User fills in transaction details on the **home.html** page (e.g., type, amount, oldbalanceOrig, newbalanceOrig).
2. **Preprocessing:**
    - Flask receives the input, applies **feature scaling and encoding** to match the model input.
3. **Prediction:**
    - The trained model (**payments.pkl**) predicts the **fraud status** (0 = Legitimate, 1 = Fraud).
4. **Result Display:**
    - The prediction is displayed on **predict.html**, showing whether the transaction is **legal or fraudulent**.
5. **Example Flask Code**

6. 

3. Conclusion

- The web application provides a **user-friendly interface** for fraud detection.
- Users can **enter transaction details** and get **instant predictions**.
- This completes the **end-to-end machine learning pipeline**:
  Data Collection → EDA → Preprocessing → Model Training → Evaluation → Deployment