

FraudShield – Personal Project Report

What the Project Does

FraudShield is a machine learning–powered desktop application that simulates a fraud detection system for financial transactions. The system analyzes basic transaction information and uses an anomaly detection model to classify transactions as either **safe** or **anomalous**.

The project focuses on replicating the backend logic of fraud monitoring systems used in banks and digital wallets. The detection process considers multiple transaction factors such as device used, time of transaction, and user frequency patterns. The goal was to build a system that aligns with real-world cybersecurity challenges in the fintech space.

How I Built It

1. Dataset Creation

I didn't use a pre-made dataset. Instead, I generated a synthetic dataset myself using NumPy and Pandas. It contains 500 user records with the following fields:

- **amount** (numerical)
- **device_type** (mobile, web, atm)
- **transaction_time** (morning, afternoon, evening, night)
- **location** (urban, suburban, rural)
- **frequency** (number of transactions over time)

I tried to keep the data distribution realistic. For example, most users transact via mobile and in urban areas, while fewer use ATMs or live in rural areas.

2. Model Training

The detection system is based on the Isolation Forest algorithm, which is an unsupervised machine learning model used for anomaly detection. It doesn't need labeled "fraud" or "not fraud" data — it identifies outliers based on how different they are from the rest of the dataset.

I used `LabelEncoder` to convert the categorical values to numerical values, trained the model on the generated dataset, and saved the trained model and encoders using `pickle`.

3. GUI Development

For the interface, I chose to use **Tkinter** instead of Streamlit because I wanted a true desktop app experience. To make it look more modern and clean, I used the `ttkbootstrap` library, which adds theming support. The app collects user inputs like amount, device type, time of transaction, etc., and then feeds the inputs into the trained model to make a prediction.

When a transaction is predicted to be anomalous, a warning message is shown. If it's safe, the user is informed that the transaction looks normal.

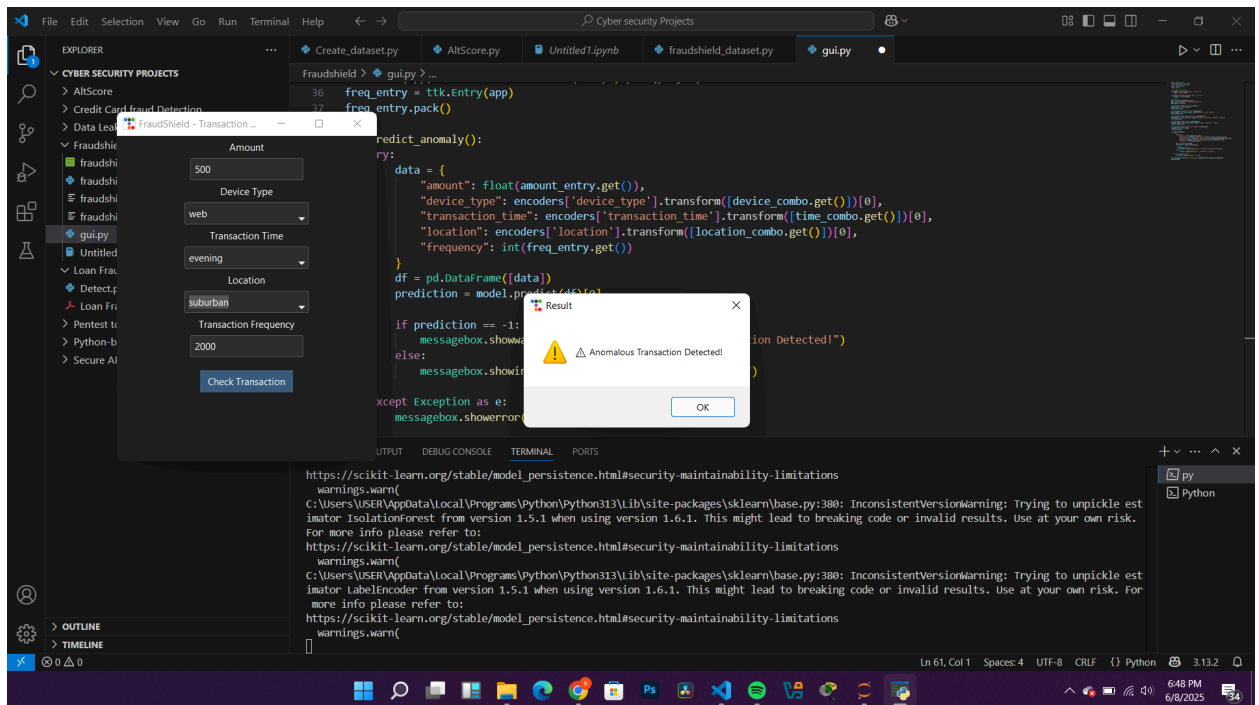
Challenges I Faced

- **Designing the dataset:** Without real data, I had to think through how fraud actually behaves and simulate realistic variations without making the data too random.
- **Unsupervised learning:** I initially tried using logistic regression, but it didn't make sense without true fraud labels. Switching to an Isolation Forest model required understanding how unsupervised anomaly detection works.
- **GUI responsiveness:** Making the Tkinter interface look decent was more work than expected. `ttkbootstrap` helped a lot, but I still had to test different layouts to make it feel smooth and not clunky.

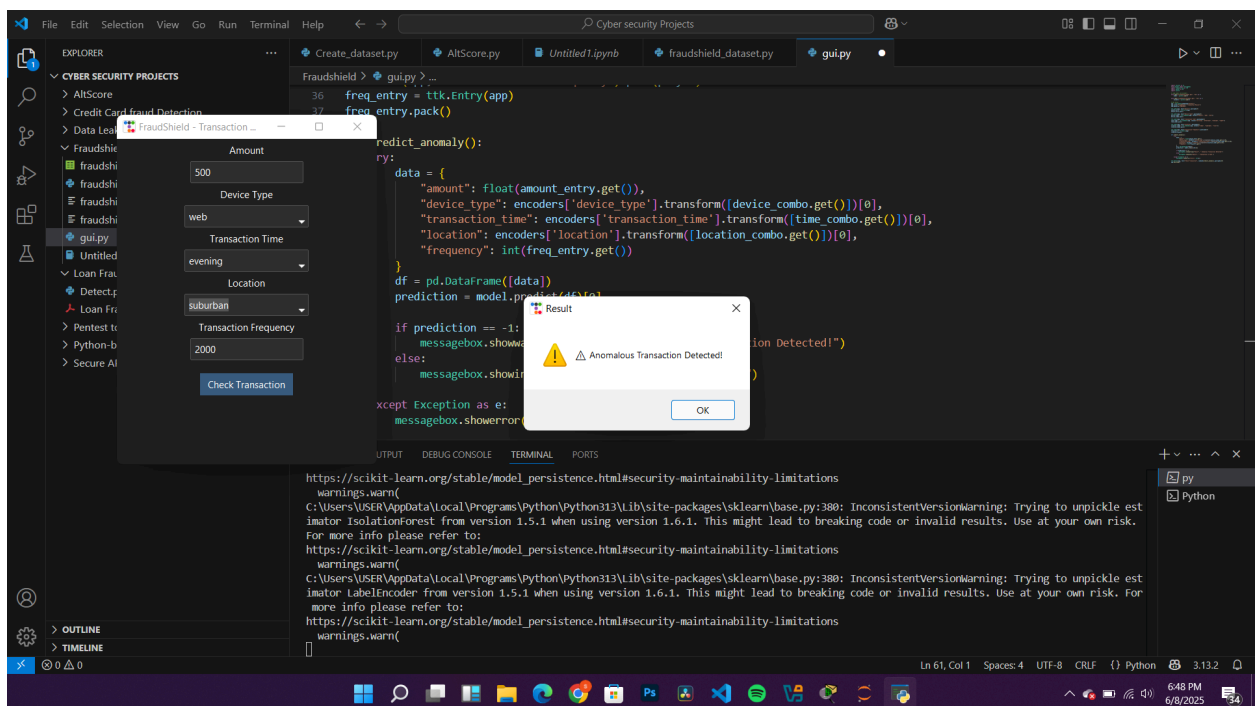
Photographic Evidence

I've included screenshots of the following:

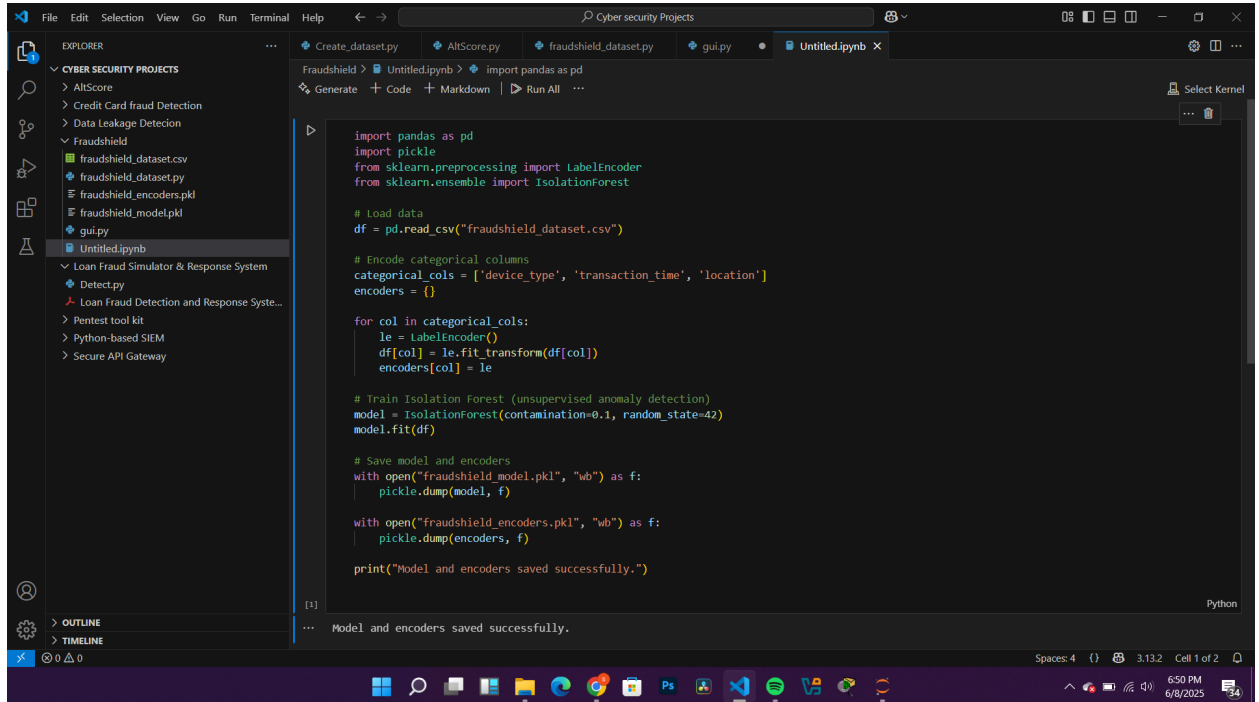
- The app interface



- A sample transaction being tested



- Training of the model



The screenshot shows a Jupyter Notebook titled 'Untitled.ipynb' in the VS Code editor. The notebook contains a Python script for training a fraud detection model. The script imports pandas and pickle, loads data from 'fraudshield_dataset.csv', encodes categorical columns, and trains an Isolation Forest model. The model and encoders are saved to 'fraudshield_model.pkl' and 'fraudshield_encoders.pkl' respectively. The output of the script is 'Model and encoders saved successfully.'

```
import pandas as pd
import pickle
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import IsolationForest

# Load data
df = pd.read_csv("fraudshield_dataset.csv")

# Encode categorical columns
categorical_cols = ['device_type', 'transaction_time', 'location']
encoders = {}

for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    encoders[col] = le

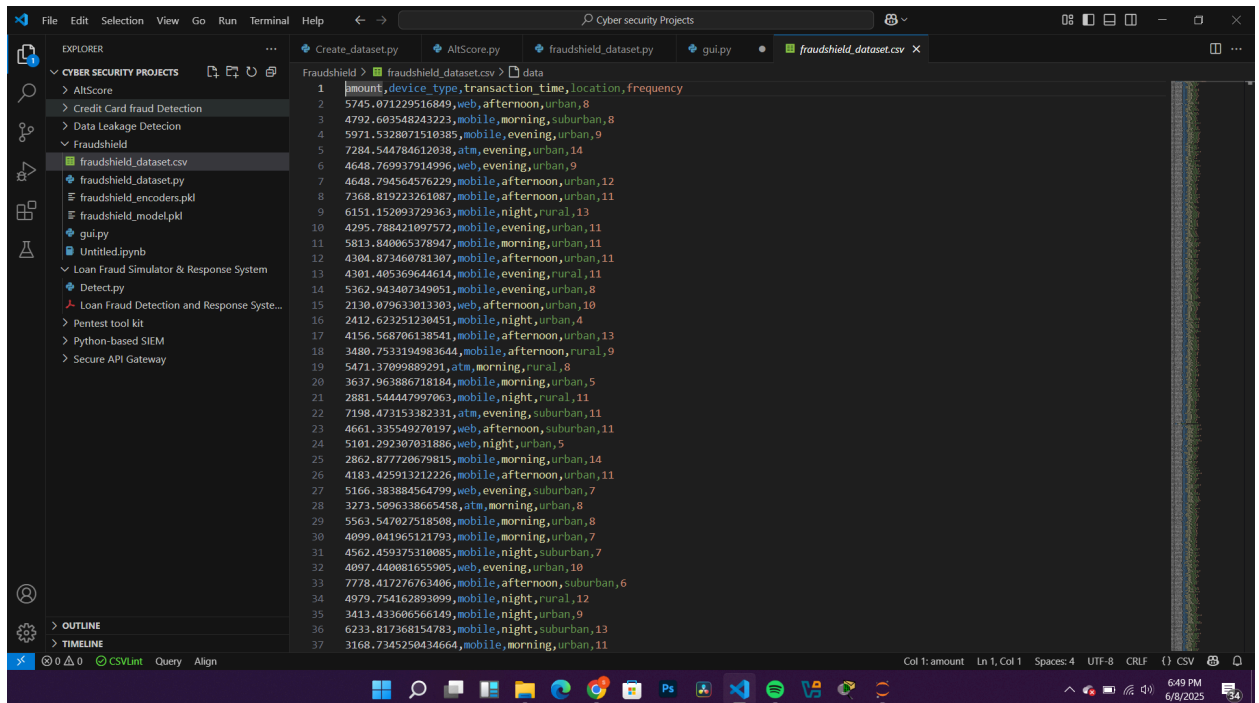
# Train Isolation Forest (unsupervised anomaly detection)
model = IsolationForest(contamination=0.1, random_state=42)
model.fit(df)

# Save model and encoders
with open("fraudshield_model.pkl", "wb") as f:
    pickle.dump(model, f)

with open("fraudshield_encoders.pkl", "wb") as f:
    pickle.dump(encoders, f)

print("Model and encoders saved successfully.")
```

- Snippets of the dataset and the model training script



The screenshot shows a CSV file named 'fraudshield_dataset.csv' in the VS Code editor. The file contains a table with columns: amount, device_type, transaction_time, location, frequency. The data is organized into rows, with the first row being the header and the subsequent rows containing numerical values for amount, categorical values for device_type and transaction_time, and numerical values for location and frequency.

amount	device_type	transaction_time	location	frequency
5745.071229516849	web	afternoon	urban	8
4792.603548243223	mobile	morning	suburban	8
5971.5328071510385	mobile	evening	urban	9
7284.544784612038	atm	evening	urban	14
4648.769937914996	web	evening	urban	9
4648.794564576229	mobile	afternoon	urban	12
7368.819223261087	mobile	afternoon	urban	11
6151.152093729363	mobile	night	rural	13
4295.788421097572	mobile	evening	urban	11
5813.840065378947	mobile	morning	urban	11
4304.873460781307	mobile	afternoon	urban	11
4301.405369644614	mobile	evening	rural	11
5362.943407349051	mobile	evening	urban	8
2130.079633013303	web	afternoon	urban	10
2412.623251230451	mobile	night	urban	4
4156.568706138541	mobile	afternoon	urban	13
3480.7533194983644	mobile	afternoon	rural	9
5471.37099889291	atm	morning	rural	8
3637.963886718184	mobile	morning	urban	5
2881.544447997063	mobile	night	rural	11
7198.473153382331	atm	evening	suburban	11
4661.335549270197	web	afternoon	suburban	11
5101.292307031886	web	night	urban	5
2862.077700679815	mobile	morning	urban	14
4183.425913212226	mobile	afternoon	urban	11
5166.383884564799	web	evening	suburban	7
3273.5096338665458	atm	morning	urban	8
5563.547027518508	mobile	morning	urban	8
4099.041965121793	mobile	morning	urban	7
4562.459375310085	mobile	night	suburban	7
4097.440081655905	web	evening	urban	10
7778.417276763406	mobile	afternoon	suburban	6
4979.754162893099	mobile	night	rural	12
3413.433606566149	mobile	night	urban	9
6233.817368154783	mobile	night	suburban	13
3168.7345250434664	mobile	morning	urban	11

Final Notes

I built this project to explore how cybersecurity techniques, especially anomaly detection, can be used in fintech systems. I wrote all the code myself, but I used AI to assist with ideas and bug fixes. It helped me understand how fraud detection can work without relying on traditional rule-based systems.

This is one of the few projects where I combined what I know in machine learning, basic cybersecurity, and software interface development. It's something I can definitely improve later by integrating login systems, logging features, or even connecting it to APIs.