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.

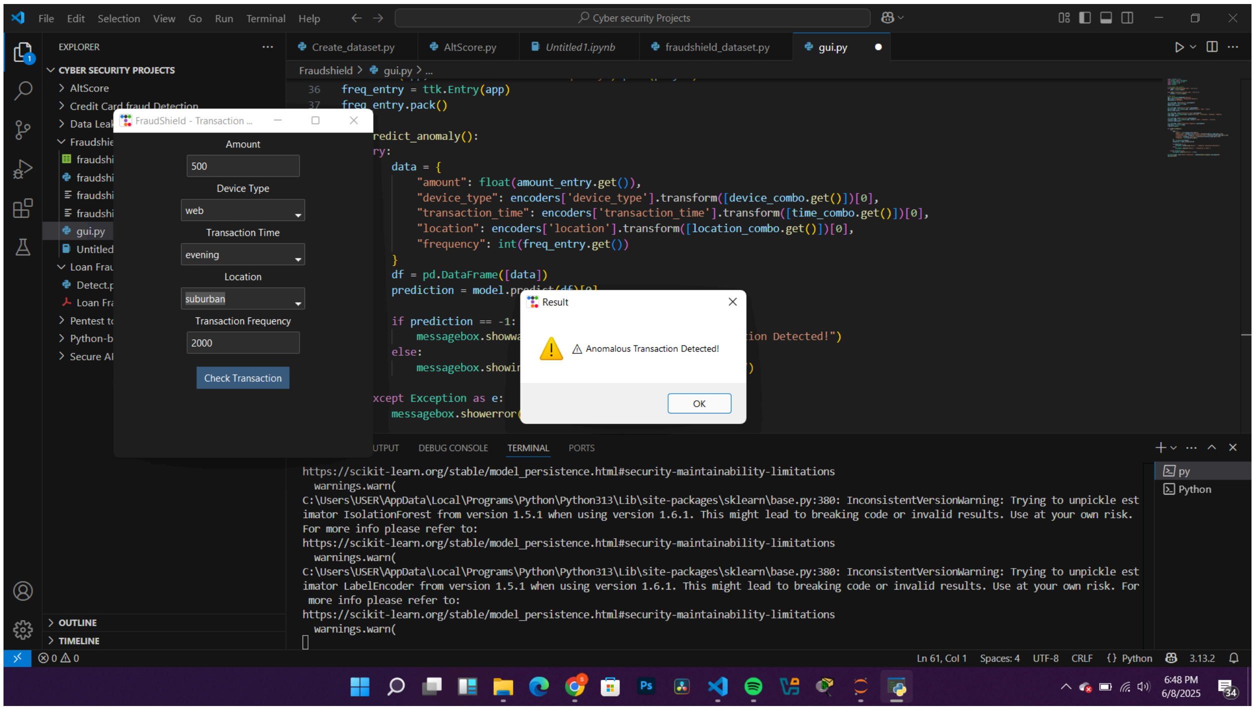
Challenges I Faced

- **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.

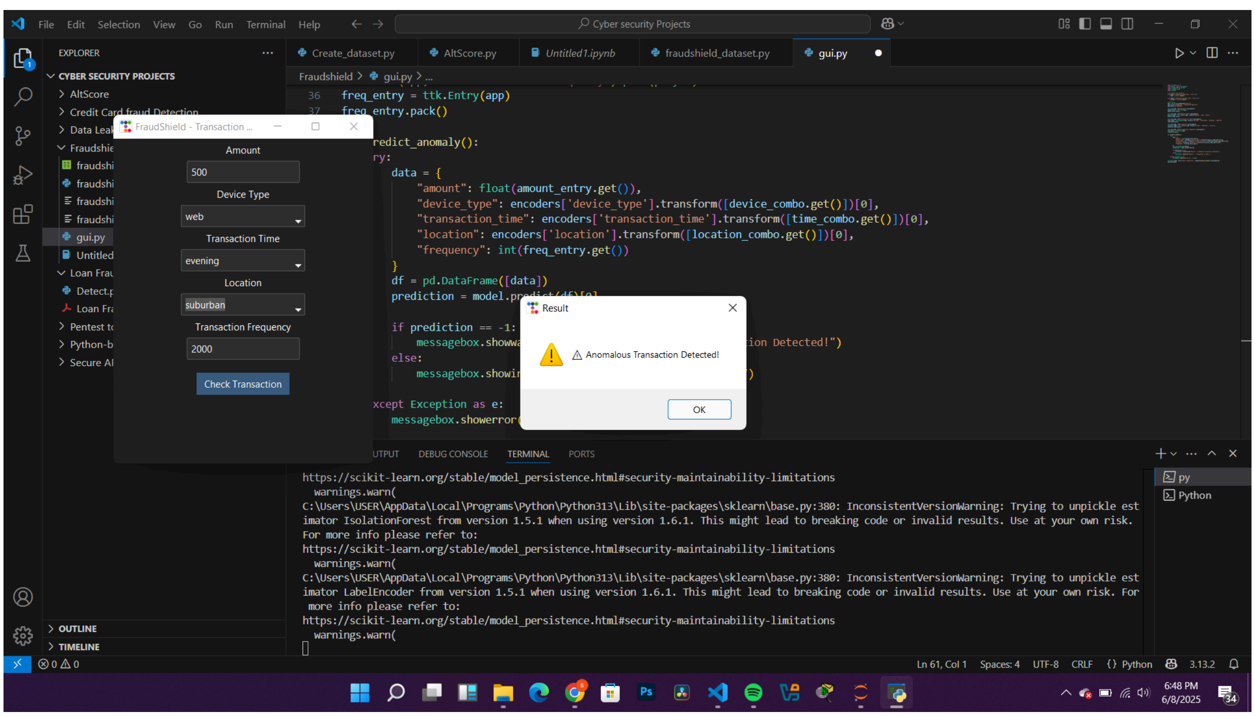
Photographic Evidence

I've included screenshots of the following:

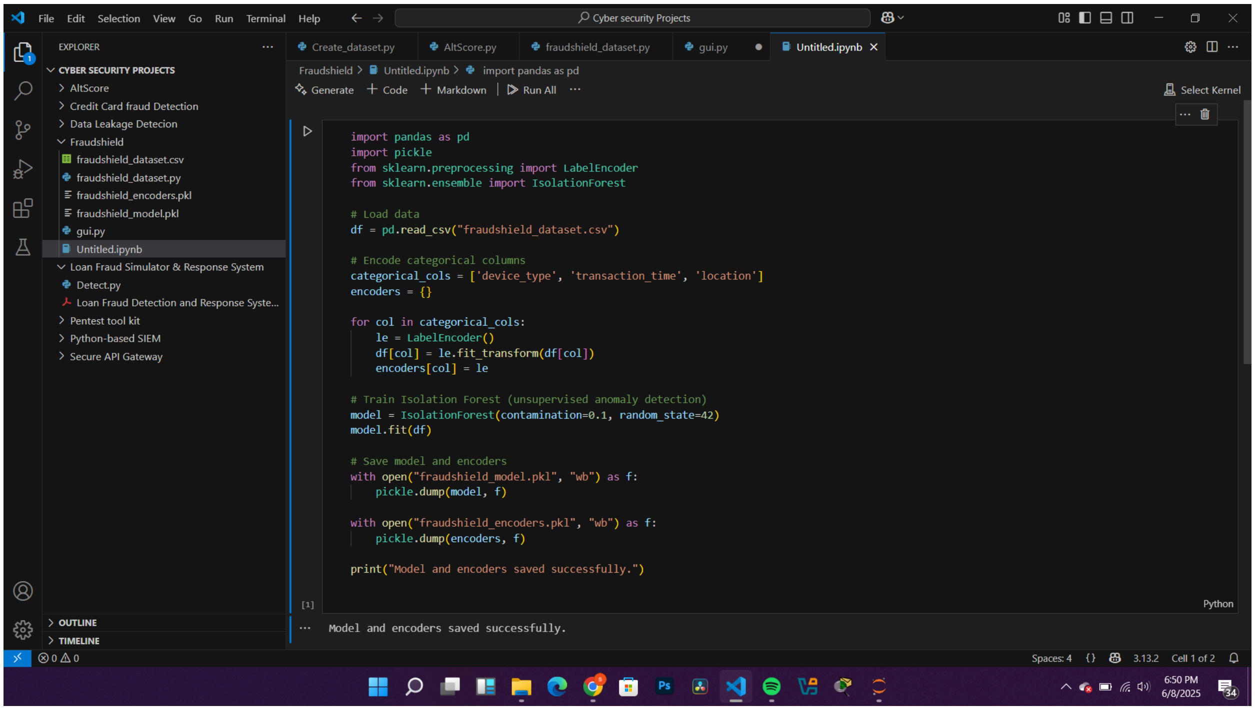
- The app interface



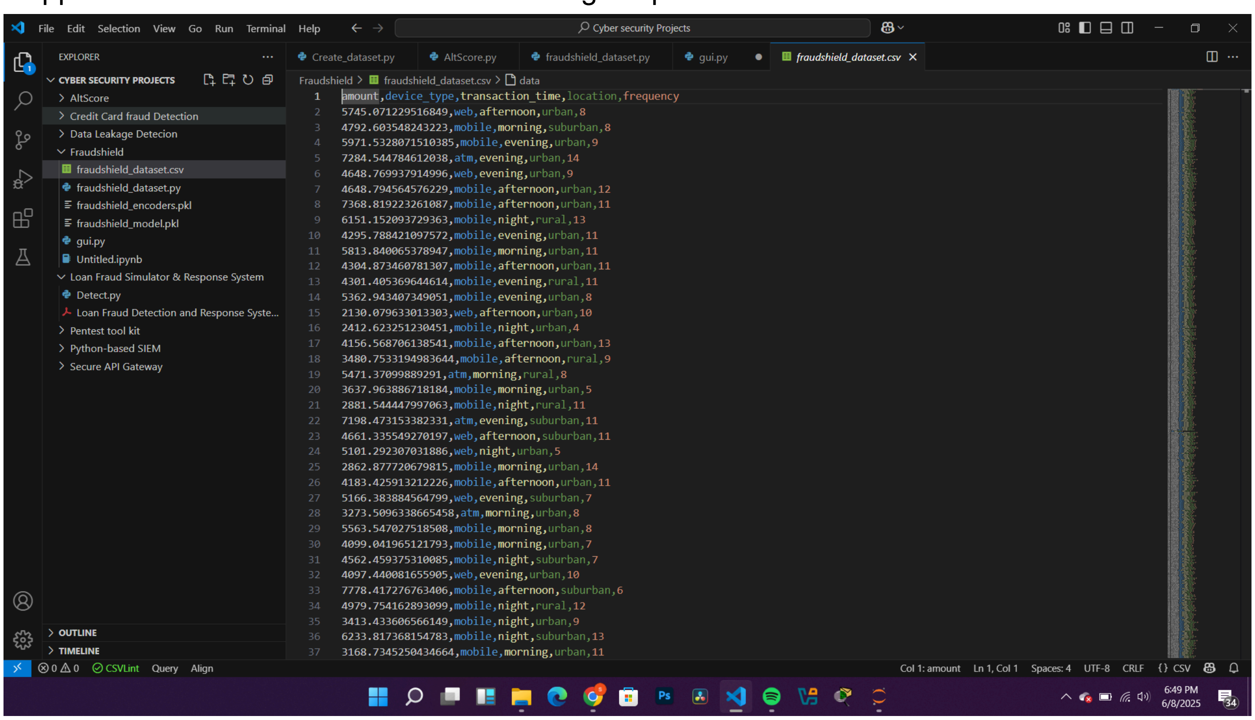
- A sample transaction being tested



- Training of the model



- Snippets of the dataset and the model training script



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.