**Fraud Detection ML Model Deployment**

**Overview**

This project demonstrates how to deploy a trained **Fraud Detection Machine Learning model** as a **REST API** using **FastAPI**, and test its **scalability** and **performance** using ApacheBench.

**Step 1: Preparing the Environment**

**1.1. Install Required Tools**

* ***sudo apt update***
* ***sudo apt install python3-pip python3-venv -y***
* ***pip install fastapi uvicorn scikit-learn pandas joblib***

**Step 2: Extract and Save the Model**

***Inside your Jupyter Notebook:***

**2.1. Train and Save the Model**

* ***import joblib***
* ***joblib.dump(model, 'fraud\_model.pkl')***

Make sure the model is saved in the API directory for later use.

**Step 3: Build the FastAPI App**

**3.1. Create the Project Structure**

* ***mkdir fraud\_detection\_api***
* ***cd fraud\_detection\_api***

**3.2. app.py – Main API Code**

***from fastapi import FastAPI, HTTPException***

***from pydantic import BaseModel***

***import joblib***

***import numpy as np***

***app = FastAPI()***

***model = joblib.load("fraud\_model.pkl")***

***class InputData(BaseModel):***

***data: dict***

***@app.post("/predict")***

***async def predict(input\_data: InputData):***

***try:***

***input\_values = list(input\_data.data.values())***

***prediction = model.predict([input\_values])***

***return {"prediction": int(prediction[0])}***

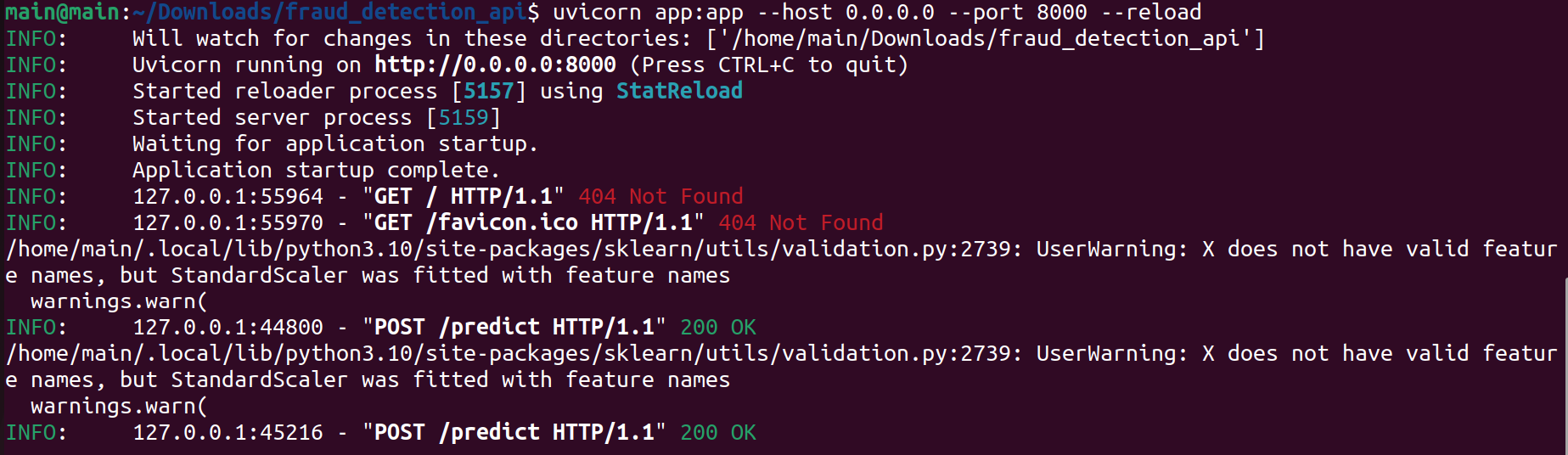
***except Exception as e:***

***raise HTTPException(status\_code=400, detail=str(e))***

**Step 4: Run the API Locally**

***uvicorn app:app --host 0.0.0.0 --port 8000 --reload***

* API will be live at: <http://127.0.0.1:8000>



**Step 5: Test the API**

**5.1. Using cURL**

***curl -X POST "http://127.0.0.1:8000/predict" \***

***-H "Content-Type: application/json" \***

***-d '{***

***"data": {***

***"GENDER": 1,***

***"CAR": 1,***

***"REALITY": 1,***

***"NO\_OF\_CHILD": 2,***

***"INCOME": 120000,***

***"INCOME\_TYPE": 1,***

***"EDUCATION\_TYPE": 0,***

***"FAMILY\_TYPE": 2,***

***"HOUSE\_TYPE": 0,***

***"FLAG\_MOBIL": 1,***

***"WORK\_PHONE": 0,***

***"PHONE": 1,***

***"E\_MAIL": 1,***

***"FAMILY SIZE": 4,***

***"BEGIN\_MONTH": -6,***

***"AGE": 35,***

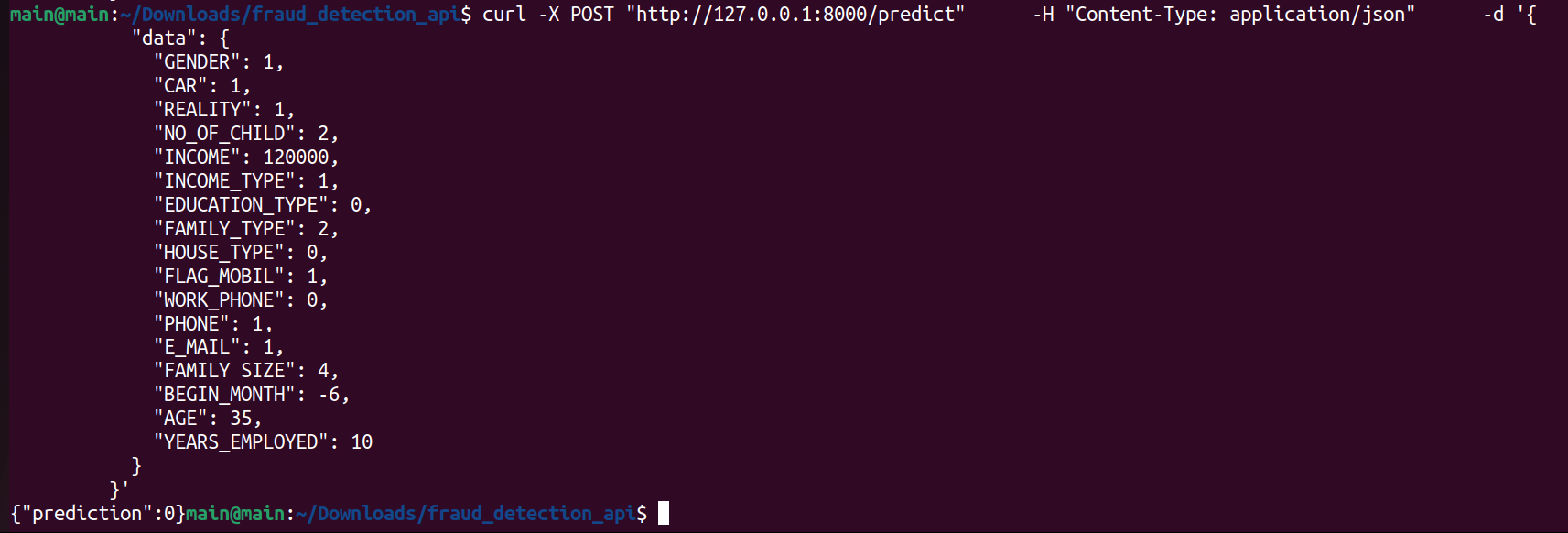
***"YEARS\_EMPLOYED": 10***

***}***

***}'***

Expected Output:

**{"prediction": 0}**



**Step 6: Benchmark API Scalability**

**6.1. Create a Payload File**

* Save the request JSON as payload.json.

***{***

***"data": {***

***"GENDER": 1,***

***"CAR": 1,***

***"REALITY": 1,***

***"NO\_OF\_CHILD": 2,***

***"INCOME": 120000,***

***"INCOME\_TYPE": 1,***

***"EDUCATION\_TYPE": 0,***

***"FAMILY\_TYPE": 2,***

***"HOUSE\_TYPE": 0,***

***"FLAG\_MOBIL": 1,***

***"WORK\_PHONE": 0,***

***"PHONE": 1,***

***"E\_MAIL": 1,***

***"FAMILY SIZE": 4,***

***"BEGIN\_MONTH": -6,***

***"AGE": 35,***

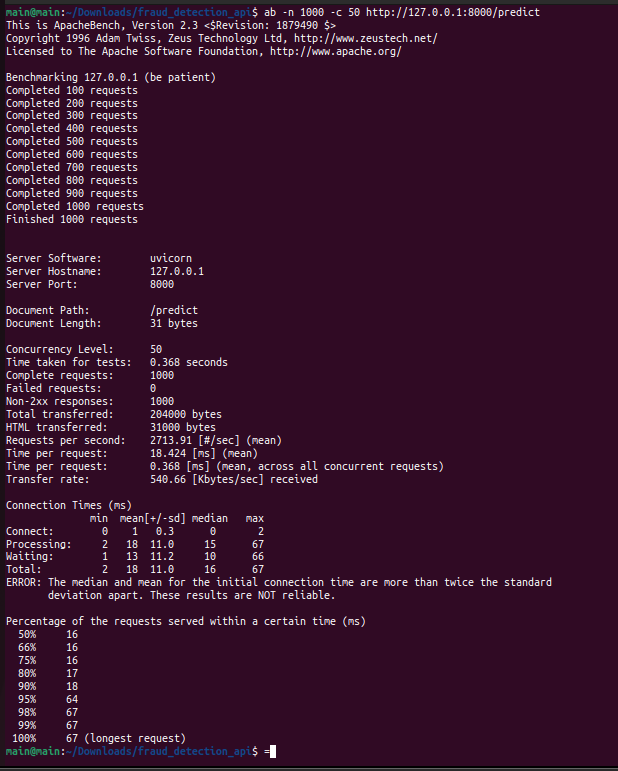
***"YEARS\_EMPLOYED": 10***

***}***

***}***

**6.2. Run ApacheBench**

* ***ab -n 1000 -c 50 -p payload.json -T application/json http://127.0.0.1:8000/predict***



**Conclusion**

This project demonstrated:

* Deploying an ML model with **FastAPI**
* Serving real-time predictions via REST
* Testing performance using **ApacheBench**
* Achieving high throughput with low latency under concurrent load