GUARDIAN TRANSACTION WITH AI POWERED

CREDIT CARD FRAUD DETECTION & PREVENTION

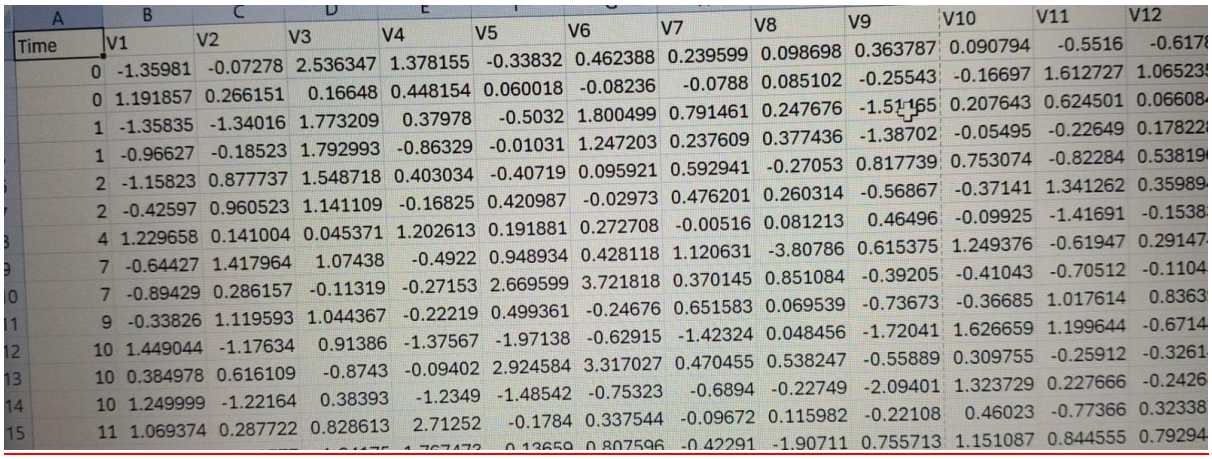
program

```
import os
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
import joblib
# === Ensure required directories exist ===
os.makedirs("model", exist_ok=True)
os.makedirs("data", exist_ok=True)
# === Load dataset ===
DATA_PATH = 'data/creditcard.csv'
if not os.path.exists(DATA_PATH):
    raise FileNotFoundError(
        f"\n❌ Dataset not found at '{DATA_PATH}'.\n"
        "📌 Please download 'creditcard.csv' from:\n"
        "🔗 https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud\n"
        "and place it inside the 'data/' folder."
    )
# Load dataset
df = pd.read_csv(DATA_PATH)
```

```
# === Preprocessing ===
X = df.drop(['Class'], axis=1)
y = df['Class']
# === Split the data ===
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
# === Train the model ===
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
# === Evaluate the model ===
y_pred = model.predict(X_test)
print("\n=== Model Evaluation ===")
print(classification_report(y_test, y_pred, zero_division=0))
# === Save the model ===
MODEL_PATH = 'model/fraud_model.pkl'
joblib.dump(model, MODEL_PATH)
# === Prediction function ===
def predict_transaction(data):
    """
    Predict if a transaction is fraudulent.
    :param data: List of 30 numerical features
    :return: Tuple (prediction, probability)
    """
    if len(data) != 30:
        raise ValueError("❌ Input must have exactly 30 features.")    if not
os.path.exists(MODEL_PATH):
        raise FileNotFoundError("Trained model not found at 'model/fraud_model.pkl'.")
```

```
    model = joblib.load(MODEL_PATH)
    data_array = np.array(data).reshape(1, -1)
    prediction = model.predict(data_array)[0]
    probability = model.predict_proba(data_array)[0][1]
    return int(prediction), round(probability, 4)
# === Simulate a transaction ===
def simulate_transaction():
    print("\n=== Simulating Transaction ===")
    sample_transaction = [0.0] * 30  # Dummy input, replace with real values if needed
    try:
        result, prob = predict_transaction(sample_transaction)
        status = "FRAUDULENT" if result else "LEGITIMATE"
        print(f"🧠 Prediction: {status} | 🔢 Probability of Fraud: {prob:.2f}")
        if result == 1:
            print("🚨 ALERT: Fraud detected. Transaction blocked.")
        else:
            print("✅ Transaction approved.")
    except Exception as e:
        print(f"❌ Error: {str(e)}")
# === Run simulation ===
if __name__ == "__main__":
    simulate_transaction()
```

Output



Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12
0	-1.35981	-0.07278	2.536347	1.378155	-0.33832	0.462388	0.239599	0.098698	0.363787	0.090794	-0.5516	-0.617
0	1.191857	0.266151	0.16648	0.448154	0.060018	-0.08236	-0.0788	0.085102	-0.25543	-0.16697	1.612727	1.06523
1	-1.35835	-1.34016	1.773209	0.37978	-0.5032	1.800499	0.791461	0.247676	-1.51165	0.207643	0.624501	0.06608
1	-0.96627	-0.18523	1.792993	-0.86329	-0.01031	1.247203	0.237609	0.377436	-1.38702	-0.05495	-0.22649	0.17822
2	-1.15823	0.877737	1.548718	0.403034	-0.40719	0.095921	0.592941	-0.27053	0.817739	0.753074	-0.82284	0.53819
2	-0.42597	0.960523	1.141109	-0.16825	0.420987	-0.02973	0.476201	0.260314	-0.56867	-0.37141	1.341262	0.35989
4	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.00516	0.081213	0.46496	-0.09925	-1.41691	-0.1538
7	-0.64427	1.417964	1.07438	-0.4922	0.948934	0.428118	1.120631	-3.80786	0.615375	1.249376	-0.61947	0.29147
7	-0.89429	0.286157	-0.11319	-0.27153	2.669599	3.721818	0.370145	0.851084	-0.39205	-0.41043	-0.70512	-0.1104
9	-0.33826	1.119593	1.044367	-0.22219	0.499361	-0.24676	0.651583	0.069539	-0.73673	-0.36685	1.017614	0.8363
10	1.449044	-1.17634	0.91386	-1.37567	-1.97138	-0.62915	-1.42324	0.048456	-1.72041	1.626659	1.199644	-0.6714
10	0.384978	0.616109	-0.8743	-0.09402	2.924584	3.317027	0.470455	0.538247	-0.55889	0.309755	-0.25912	-0.3261
10	1.249999	-1.22164	0.38393	-1.2349	-1.48542	-0.75323	-0.6894	-0.22749	-2.09401	1.323729	0.227666	-0.2426
11	1.069374	0.287722	0.828613	2.71252	-0.1784	0.337544	-0.09672	0.115982	-0.22108	0.46023	-0.77366	0.32338
						0.13659	0.807596	-0.42291	-1.90711	0.755713	1.151087	0.844555	0.79294