```
# -------------------------------------------

# 🧠 Autoencoder for Anomaly (Fraud) Detection

# -------------------------------------------


import pandas as pd

import numpy as np

import tensorflow as tf

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import confusion_matrix, classification_report

import matplotlib.pyplot as plt

import seaborn as sns

import os


# ✅ 1. Load dataset (auto-download if missing)

file_path = "creditcard.csv"


if not os.path.exists(file_path):

    print("🔽 Downloading dataset...")

    url = "https://storage.googleapis.com/download.tensorflow.org/data/creditcard.csv"

    data = pd.read_csv(url)

    data.to_csv(file_path, index=False)

else:

    print("✅ Found local dataset.")

    data = pd.read_csv(file_path)
```

```
print("Data Loaded. Shape:", data.shape)

print(data.head())


# ✅ 2. Preprocessing

scaler = StandardScaler()

data[['Time', 'Amount']] = scaler.fit_transform(data[['Time', 'Amount']])


X = data.drop('Class', axis=1)

y = data['Class'].astype(bool)


# Split into train/test

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Train only on normal data

X_train_norm = X_train[~y_train]

X_test_norm = X_test[~y_test]

X_test_fraud = X_test[y_test]


print(f"Training on normal samples: {len(X_train_norm)}")

print(f"Testing on normal samples: {len(X_test_norm)} | Fraud samples: {len(X_test_fraud)}")


# ✅ 3. Build Autoencoder

input_dim = X_train_norm.shape[1]

input_layer = tf.keras.layers.Input(shape=(input_dim,))

encoder = tf.keras.layers.Dense(16, activation='relu')(input_layer)

encoder = tf.keras.layers.Dense(8, activation='relu')(encoder)
```

```
latent = tf.keras.layers.Dense(4, activation='relu')(encoder)

decoder = tf.keras.layers.Dense(8, activation='relu')(latent)

decoder = tf.keras.layers.Dense(16, activation='relu')(decoder)

output_layer = tf.keras.layers.Dense(input_dim, activation='sigmoid')(decoder)


autoencoder = tf.keras.Model(inputs=input_layer, outputs=output_layer)

autoencoder.compile(optimizer='adam', loss='mse', metrics=['mae'])

autoencoder.summary()


# ✅ 4. Train Autoencoder

history = autoencoder.fit(

    X_train_norm, X_train_norm,

    epochs=5,

    batch_size=64,

    validation_data=(X_test_norm, X_test_norm),

    verbose=1

)


# ✅ 5. Reconstruction Error (Anomaly Detection)

recon = autoencoder.predict(X_test)

mse = np.mean(np.power(X_test - recon, 2), axis=1)


threshold = np.percentile(mse, 95)  # Top 5% errors as anomaly

y_pred = (mse > threshold).astype(int)


# ✅ 6. Visualization — Histogram of reconstruction error
```

```
plt.figure(figsize=(8, 5))

plt.hist(mse[y_test == 0], bins=50, alpha=0.6, label='Normal')

plt.hist(mse[y_test == 1], bins=50, alpha=0.6, label='Fraud')

plt.axvline(threshold, color='r', linestyle='--', label='Threshold')

plt.title("Histogram of Reconstruction Error")

plt.xlabel("Reconstruction error (MSE)")

plt.ylabel("Number of samples")

plt.legend()

plt.show()


# ✅ 8. Plot Training Curves

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val_loss'], label='Validation Loss')

plt.title("Autoencoder Training Performance")

plt.xlabel("Epochs")

plt.ylabel("Loss")

plt.legend()

plt.show()


# ✅ 9. Anomaly Detection Graph (NEW ADDITION)

# Scatter plot showing reconstruction error per sample

plt.figure(figsize=(10, 6))

plt.scatter(range(len(mse)), mse, c=y_pred, cmap='coolwarm', alpha=0.6)

plt.axhline(threshold, color='r', linestyle='--', label='Threshold')

plt.title("Anomaly Detection Graph (Reconstruction Error per Sample)")

plt.xlabel("Sample Index")
plt.ylabel("Reconstruction Error (MSE)")

plt.legend()

plt.show()


# ✅ 10. Show how fraud samples stand out

plt.figure(figsize=(8, 5))

sns.kdeplot(mse[y_test == 0], label="Normal", fill=True)

sns.kdeplot(mse[y_test == 1], label="Fraud", fill=True)

plt.axvline(threshold, color='r', linestyle='--', label='Threshold')

plt.title("Density Plot: Normal vs Fraud Reconstruction Errors")

plt.xlabel("Reconstruction Error (MSE)")

plt.ylabel("Density")

plt.legend()

plt.show()
```