```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, roc_auc_score, confusion_matrix
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, Model, Input, optimizers

df = pd.read_csv("/content/creditcard.csv")



X = df.drop(columns=["Class"])
y = df["Class"].values

X_normal = X[y == 0].copy()
X_anom   = X[y == 1].copy()

X_train_norm, X_val_norm = train_test_split(X_normal, test_size=0.2, random_state=42,
shuffle=True)

scaler = StandardScaler()
scaler.fit(X_train_norm)
X_train = scaler.transform(X_train_norm)
X_val   = scaler.transform(X_val_norm)
X_test_norm = scaler.transform(X_normal.sample(n=min(10000, len(X_normal)),
random_state=42))
X_test_anom = scaler.transform(X_anom)

X_test = np.vstack([X_test_norm, X_test_anom])
y_test = np.hstack([np.zeros(len(X_test_norm)), np.ones(len(X_test_anom))])

input_dim = X_train.shape[1]
encoding_dim = 16

inp = Input(shape=(input_dim,), name="encoder_input")
x = layers.Dense(64, activation="relu")(inp)
x = layers.Dense(32, activation="relu")(x)
latent = layers.Dense(encoding_dim, activation="relu", name="latent")(x)

x = layers.Dense(32, activation="relu")(latent)
x = layers.Dense(64, activation="relu")(x)
out = layers.Dense(input_dim, activation="linear", name="reconstruction")(x)

autoencoder = Model(inputs=inp, outputs=out, name="autoencoder")
encoder = Model(inputs=inp, outputs=latent, name="encoder")
```

```
autoencoder.compile(optimizer=optimizers.Adam(learning_rate=1e-3), loss="mse")
autoencoder.summary()

history = autoencoder.fit(
    X_train, X_train,
    epochs=50,
    batch_size=256,
    validation_data=(X_val, X_val),
    verbose=2
)

recon_val = autoencoder.predict(X_val, verbose=0)
mse_val = np.mean(np.square(recon_val - X_val), axis=1)

threshold = np.percentile(mse_val, 99)
print(f"Chosen anomaly threshold (99th percentile of val normal MSE): {threshold:.6f}")

recon_test = autoencoder.predict(X_test, verbose=0)
mse_test = np.mean(np.square(recon_test - X_test), axis=1)

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

print("\nClassification report (autoencoder threshold-based):")
print(classification_report(y_test, y_pred, digits=4))

auc = roc_auc_score(y_test, mse_test)
print(f"ROC AUC (reconstruction score): {auc:.4f}")

cm = confusion_matrix(y_test, y_pred)
print("Confusion matrix:\n", cm)

plt.figure(figsize=(8,4))
plt.hist(mse_test[y_test == 0], bins=50, alpha=0.6, label="normal (test)")
plt.hist(mse_test[y_test == 1], bins=50, alpha=0.6, label="anomaly (test)")
plt.axvline(threshold, color='r', linestyle='--', label='threshold')
plt.title("Reconstruction error (MSE) distribution")
plt.xlabel("MSE"); plt.ylabel("Count"); plt.legend()
plt.show()

plt.figure(figsize=(10,5))
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss (MSE)')
plt.legend()
plt.title('Autoencoder Training and Validation Loss')
plt.show()
```

🔹 Objectives:

1. To understand and implement an **unsupervised deep learning** method for anomaly detection.
2. To train an **autoencoder** using only normal transactions.
3. To classify anomalies based on **reconstruction error**.
4. To evaluate the model using **ROC-AUC score, confusion matrix, and classification report**.

🔹 Theory:

1. Introduction

Credit card fraud detection is a highly imbalanced classification problem where fraudulent transactions are very rare compared to normal ones. Traditional supervised learning models require labeled data, which is often limited or unavailable.
To overcome this, we use an **Autoencoder**, an unsupervised neural network that learns to reconstruct normal data. Any large deviation (error) in reconstruction indicates a potential anomaly or fraud.

2. Autoencoder Concept

An **Autoencoder** is a type of **neural network** that attempts to learn a compressed representation of input data and then reconstructs it back to its original form.

It consists of two main parts:

- **Encoder:** Compresses the input into a smaller latent (bottleneck) representation.
- **Decoder:** Reconstructs the input from the latent representation.

When trained only on **normal data**, the autoencoder becomes good at reconstructing normal patterns but performs poorly on unseen or abnormal (fraudulent) data.
Thus, reconstruction **error (Mean Squared Error)** becomes a signal for detecting anomalies.

---

3. Working Principle

1. **Training Phase:**
    - Train the autoencoder using only normal transactions (non-fraud).
    - The network learns to reconstruct normal patterns.
2. **Validation Phase:**
    - Compute the **Mean Squared Error (MSE)** between input and output.
    - Determine a **threshold** value based on the 99th percentile of MSE from the validation set.
3. **Testing Phase:**
    - Compute reconstruction errors on test data (both normal and fraud).
    - If the error > threshold → classify as **fraud (anomaly)**, otherwise **normal**.

🔹 Algorithm / Steps:

1. Import necessary libraries (NumPy, Pandas, Matplotlib, TensorFlow, Sklearn).
2. Load the dataset `creditcard.csv`.
3. Separate features (`X`) and labels (`y`).
4. Split the dataset into normal and anomalous data.
5. Scale the features using `StandardScaler`.
6. Build the **autoencoder model** (Encoder + Decoder).
7. Compile the model using **Adam optimizer** and **MSE loss**.
8. Train the autoencoder on **normal data** only.
9. Compute reconstruction errors on validation data to find a **threshold**.
10. Test on combined (normal + fraud) data and classify using threshold.
11. Evaluate the performance using metrics and visualize results.

🔹 Applications:

- Fraud detection in finance and banking.
- Network intrusion detection.
- Fault detection in manufacturing systems.
- Medical anomaly detection.

Conclusion:

An **Autoencoder-based anomaly detection model** effectively identifies fraudulent credit card transactions without requiring labeled data during training.
This demonstrates the power of **unsupervised deep learning** in detecting anomalies in highly imbalanced datasets.