#Use Autoencoder to implement anomaly detection. Build the model by using

#a. Import required libraries

#b. Upload/access the dataset

#c.Encoder converts it into latent representation

#d. Decoder networks convert it back to the original input

#e. Compile the models with Optimizer, Loss, and Evaluation

#a. Import required libraries

import numpy as np

import tensorflow as tf

from tensorflow import keras

from sklearn.model\_selection import train\_test\_split

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

#b. Upload/access the dataset

# Load the Credit Card dataset

data = pd.read\_csv("C:/Users/HP/Downloads/creditcard.csv")

# Extract features (input data) and labels

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

y = data['Class']

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Standardize input features

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

#c.Encoder converts it into latent representation

latent\_dim = 32

encoder = keras.Sequential([

    keras.layers.Input(shape=(X\_train.shape[1],)),

    keras.layers.Dense(64, activation='relu'),

    keras.layers.Dense(latent\_dim, activation='relu'),

])

#d. Decoder networks convert it back to the original input

decoder = keras.Sequential([

    keras.layers.Input(shape=(latent\_dim,)),

    keras.layers.Dense(64, activation='relu'),

    keras.layers.Dense(X\_train.shape[1], activation='sigmoid'),

])

#e. Compile the models with Optimizer, Loss, and Evaluation

autoencoder = keras.Sequential([encoder, decoder])

autoencoder.compile(optimizer='adam', loss='mean\_squared\_error')

# Train the autoencoder

autoencoder.fit(X\_train, X\_train, epochs=50, batch\_size=32)

# Reconstruct data

reconstructed\_data = autoencoder.predict(X\_test)

# Calculate reconstruction error

mse = np.mean(np.power(X\_test - reconstructed\_data, 2), axis=1)

# Define a threshold for anomaly detection

threshold = 0.05

# Identify anomalies

anomalies = mse > threshold

# Evaluate the model (e.g., accuracy, precision, recall)

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score

accuracy = accuracy\_score(y\_test, anomalies)

precision = precision\_score(y\_test, anomalies)

recall = recall\_score(y\_test, anomalies)

print(f'Accuracy: {accuracy}, Precision: {precision}, Recall: {recall}')