```python

# importing Libraries

import numpy as np

import matplotlib.pyplot as plt

from sklearn.neural_network import BernoulliRBM

from sklearn.datasets import fetch_openml

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import classification_report, accuracy_score
```

```python
# Load the MNIST dataset

mnist = fetch_openml("mnist_784")
# Scale pixel values to [0, 1]

X = mnist.data / 255.0

y = mnist.target.astype(int)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/datasets/_openml.py:968: FutureWarning: The de
  warn(
```

```python
# Split the dataset into training and test sets

X_train, X_test, y_train, y_test = train_test_split(

    X, y, test_size=0.2, random_state=42)
```

```python
# Create and configure the BernoulliRBM

rbm = BernoulliRBM(n_components=64, learning_rate=0.1,

                   n_iter=10, random_state=0, verbose=True)

# Fit the RBM to the training data
rbm.fit(X_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
  warnings.warn(
[BernoulliRBM] Iteration 1, pseudo-likelihood = -106.63, time = 9.08s
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
  warnings.warn(
[BernoulliRBM] Iteration 2, pseudo-likelihood = -104.03, time = 11.17s
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
  warnings.warn(
[BernoulliRBM] Iteration 3, pseudo-likelihood = -102.23, time = 11.65s
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
  warnings.warn(
[BernoulliRBM] Iteration 4, pseudo-likelihood = -99.64, time = 12.04s
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
  warnings.warn(
[BernoulliRBM] Iteration 5, pseudo-likelihood = -101.11, time = 11.76s
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
  warnings.warn(
[BernoulliRBM] Iteration 6, pseudo-likelihood = -97.81, time = 11.81s
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
  warnings.warn(
[BernoulliRBM] Iteration 7, pseudo-likelihood = -98.99, time = 11.88s
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
  warnings.warn(
[BernoulliRBM] Iteration 8, pseudo-likelihood = -98.25, time = 11.84s
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
  warnings.warn(
[BernoulliRBM] Iteration 9, pseudo-likelihood = -97.02, time = 10.38s
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
  warnings.warn(
[BernoulliRBM] Iteration 10, pseudo-likelihood = -94.00, time = 10.89s
```

```
▾                  BernoulliRBM
BernoulliRBM(n_components=64, random_state=0, verbose=True)
```

```python
# Transform the training and test data into the hidden representations

X_train_encoded = rbm.transform(X_train)

X_test_encoded = rbm.transform(X_test)

# Train a classifier (Logistic Regression) on the encoded data

classifier = LogisticRegression(max_iter=100)
classifier.fit(X_train_encoded, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarni
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
▾ LogisticRegression
LogisticRegression()
```

```python
# Evaluate the classifier on the test data

y_pred = classifier.predict(X_test_encoded)

accuracy = accuracy_score(y_test, y_pred)

print("Classifier Accuracy:", accuracy)
```

```
Classifier Accuracy: 0.9115
```

```
classification_rep = classification_report(y_test, y_pred)

print("Classification Report:")

print(classification_rep)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.97      0.96      1343
           1       0.97      0.98      0.97      1600
           2       0.91      0.91      0.91      1380
           3       0.87      0.87      0.87      1433
           4       0.89      0.86      0.88      1295
           5       0.91      0.87      0.89      1273
           6       0.94      0.96      0.95      1396
           7       0.93      0.92      0.93      1503
           8       0.88      0.89      0.89      1357
           9       0.84      0.87      0.85      1420

    accuracy                           0.91     14000
   macro avg       0.91      0.91      0.91     14000
weighted avg       0.91      0.91      0.91     14000
```