第三次作品：分類器的原理與評比實驗(資料三)

學號：410978002

姓名：謝元皓

作品描述：

本專題計畫執行這篇講義描述的分類器比較，即採用三種分類器分別對三組資料進行分類學習與測試。 其中分類器包括

- 多元羅吉斯回歸 (Multinomial Logistic Regression)
- 支援向量機 (Support Vector Machine)
- 神經網路 (Neural Network)

此作業包含共三筆資料，分成三份檔案呈現，每份資料分別運用三種分類器進行分類學習與測試。

資料描述： 第三筆為來自 Yale Face 38 人的人臉影像共 2410 張，每張大小 192×168。

資料讀取

- 將資料拆成分成訓練資料跟測試資料。
- 訓練和測試資料比例為 $4:1$。
- 先將原始資料標準化(必須將訓練和測試資料分開標準化)。
- 再運用PCA降維創造出主成分資料。
- 順便列印出訓練資料跟測試資料的大小。

In [ ]:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import scipy.io
import os

D = scipy.io.loadmat('allFaces.mat')
X = D['faces'].T # 32256 x 2410, each column represents an image
y = np.ndarray.flatten(D['nfaces'])
m = D['m'].item() # 168
n = D['n'].item() # 192
n_persons = D['person'].item() # 38

y_labels = np.repeat(np.arange(len(y)), y)
# 分割資料集
X_train, X_test, y_train, y_test = train_test_split(X, y_labels, test_size=0.2)

# Standardize data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)
```

```
# 主成分分析 (PCA)
proportion = 0.8
pca = PCA(proportion)
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)

print(X_train_scaled.shape, y_train.shape)
print(X_test_scaled.shape, y_test.shape)
print(X_train_pca.shape)
print(X_test_pca.shape)
```

```
(1928, 32256) (1928,)
(482, 32256) (482,)
(1928, 7)
(482, 7)
```

分類器一: 多元邏吉斯回歸

- 需使用套件 sklearn 中的 LogisticRegression
- 運用Cross validation 的方式，找出最佳參數。
- 需使用到sklearn 中 GridSearchCV套件。
- 找出最好參數後，再用此訓練及測試資料。

---

step1 運用 Cross validation 的方式，找出最佳的參數

套件介紹

1. LogisticRegression 為Scikit-learn庫中的Logistic Regression模型
2. GridSearchCV, StratifiedShuffleSpli 為 Scikit-learn庫中的網格搜索(GridSearchCV)和分層洗牌分割(StratifiedShuffleSplit)模組。網格搜索用於選擇最佳的超參數，分層洗牌分割用於交叉驗證。

程式碼解析

這段程式碼使用了 scikit-learn 中的 LogisticRegression 和 GridSearchCV 來進行邏輯回歸模型的超參數調校和交叉驗證。我來解釋一下程式碼的各個部分：

1. **匯入庫**：

   ```
   from sklearn.linear_model import LogisticRegression
   from sklearn.model_selection import GridSearchCV,
   StratifiedShuffleSplit
   ```
   匯入了 scikit-learn 中的 LogisticRegression 類別和 GridSearchCV、StratifiedShuffleSplit 類別。

2. **定義參數**：

   ```
   opts = dict(tol=1e-3, max_iter=int(1e3))
   ```
   設置了 LogisticRegression 的參數，包括收斂容忍度(tol)和最大迭代次數(max_iter)。

3. **定義參數網格**：

   ```
   parameters = {'solver': ['lbfgs', 'liblinear', 'newton-cg'], 'C': [0.1, 1, 10]}
   ```
   設置了要調校的超參數網格，包括 solver 和 C。

4. **定義交叉驗證策略**：

   ```
   cv = StratifiedShuffleSplit(n_splits=5, test_size=0.2, random_state=0)
   ```
   使用 StratifiedShuffleSplit 來定義交叉驗證策略，其中包括了 5 折交叉驗證，訓練集測試集比例為 0.2，並且設置了隨機種子。

5. **建立 GridSearchCV 物件**：

   ```
   grid = GridSearchCV(estimator=LogisticRegression(**opts), param_grid=parameters, cv=cv, scoring=['accuracy','f1_macro'], refit="accuracy")
   ```
   使用 GridSearchCV 來建立一個網格搜尋物件，其中包括了 LogisticRegression 作為基礎估計器，設置了要調校的參數網格、交叉驗證策略、評估指標為準確率和 f1_macro，並且設置了 refit 為準確率，意味著將根據準確率來重新擬合最佳模型。

6. **進行模型訓練與評估**：

   ```
   grid.fit(X_train, y_train)
   ```
   使用訓練數據 X_train 和 y_train 來進行模型訓練和參數調校。

7. **查看最佳結果**：

   ```
   print(grid.best_params_)
   print(grid.best_score_)
   print(grid.best_estimator_)
   ```
   輸出網格搜尋後得到的最佳參數、最佳分數以及最佳估計器（最佳模型）。

In [ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import  GridSearchCV, \
                                StratifiedShuffleSplit

opts = dict(tol = 1e-3, max_iter = int(1e3)) # parameters for LogisticRegression
parameters = {'solver':['lbfgs', 'liblinear', 'newton-cg'],
              'C':[0.1, 1, 10]} # parameters for GridSearchCV
cv = StratifiedShuffleSplit(n_splits=5, test_size=0.2, \
                            random_state=0) # 5-fold CV
grid = GridSearchCV(estimator=LogisticRegression(**opts), \
                    param_grid=parameters, cv=cv,
                    scoring=['accuracy','f1_macro'], refit="accuracy")
grid.fit(X_train, y_train)
# grid.fit(X, y)
cv_logistic = pd.DataFrame(data = grid.cv_results_)
print(grid.best_params_)
print(grid.best_score_)
print(grid.best_estimator_)
```

<結果與討論>

- 試跑過上述程式，直行約一整晚沒有結果。

- 改用預設參數訓練及測試資料。

step2 測試資料之準確率回報

- 運用上一步找出最佳參數訓練資料，計算各項準確率

In [ ]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
# 初始化羅吉斯回歸分類器並使用上一步找出的最佳參數
logreg = LogisticRegression(C=1, solver='lbfgs',tol = 1e-6, max_iter = int(1e6),

# 使用原始資料進行訓練及測試
logreg.fit(X_train_scaled, y_train)
y_pred = logreg.predict(X_test_scaled)
accuracy_orig = accuracy_score(y_test, y_pred)

# 使用主成分資料進行訓練及測試
logreg.fit(X_train_pca, y_train)
y_pred_pca = logreg.predict(X_test_pca)
accuracy_pca = accuracy_score(y_test, y_pred_pca)

print("Accuracy using original data:", accuracy_orig)
print("Accuracy using principal components:", accuracy_pca)
```

```
Accuracy using original data: 0.9688796680497925
Accuracy using principal components: 0.3630705394190871
```

< 結果與討論 >

- 上述這段程式碼將使用原始資料和使用PCA降維後的主成分資料來訓練和測試多元羅吉斯回歸分類器。
- 它會列印出兩種情況下的準確率(原始資料的準確率為96.88%，主成分(n = 7)後的資料準確率為36.3%)

分類器二: 支援向量機 (Support Vector Machine)

step1 運用 Cross validation 的方式，找出最佳的參數

套件介紹

1. SVC 是 Scikit-learn 中的支持向量機分類器，
2. GridSearchCV 是用於超參數優化的網格搜索類別，
3. StratifiedShuffleSplit 用於生成分層隨機分割的交叉驗證集，
4. datetime 是 Python 的日期和時間處理類別，
5. pandas 是 Python 中用於數據處理的庫。

In [ ]:
```
from sklearn.svm import SVC
from sklearn.model_selection import  GridSearchCV, \
```

```python
                         StratifiedShuffleSplit
opts = dict(tol = 1e-3, max_iter = int(1e3)) # parameters for LogisticRegression
parameters = {'kernel':['linear', 'rbf', 'poly', 'sigmoid'],
              'C':[0.1, 1, 10]} # parameters for GridSearchCV
cv = StratifiedShuffleSplit(n_splits=5, test_size=0.2, \
                            random_state=0) # 5-fold CV
grid = GridSearchCV(estimator=SVC(**opts), \
                param_grid=parameters, cv=cv,
                scoring=['accuracy','f1_macro'], refit="accuracy")
grid.fit(X_train, y_train)
# grid.fit(X, y)
cv_logistic = pd.DataFrame(data = grid.cv_results_)
print(grid.best_params_)
print(grid.best_score_)
print(grid.best_estimator_)
```

<結果與討論>

- 試跑過上述程式，直行約一整晚沒有結果。
- 改用預設參數訓練及測試資料。

---

step2 測試資料之準確率回報

- 運用上一步找出最佳參數練資料，同時計算各項準確率

程式碼解析

這段程式碼演示了如何使用 Scikit-learn 中的支持向量機 (SVM) 來進行分類任務，並使用 `classification_report` 函數生成分類報告。讓我們一步步來解釋它：

1. **導入必要的庫和類別**：

   ```python
   from sklearn.svm import SVC, LinearSVC
   from sklearn.metrics import classification_report
   ```
   這些語句導入了必要的庫和類別。`SVC` 是 Scikit-learn 中的支持向量機分類器，`LinearSVC` 是用於線性支持向量機的類別，`classification_report` 是用於生成分類報告的函數。

2. **設置模型參數**：

   ```python
   C = 0.1  # SVM regularization parameter
   opts = dict(C=C, tol=1e-6, max_iter=int(1e6))
   ```
   這裡設置了一個正則化參數 `C`，以及一些其他的 SVC 參數，如容忍度 `tol` 和最大迭代次數 `max_iter`。這些參數將用於初始化 SVC 分類器。

3. **初始化 SVC 分類器**：

   ```python
   clf_svm = SVC(kernel='linear', **opts)
   ```
   這行程式碼初始化了一個 SVC 分類器。`kernel` 參數指定了要使用的核函數，這裡使用線性核函數 ('linear')。`**opts` 將之前設置的參數傳遞給 SVC 分類器。

4. **模型訓練**：

```
clf_svm.fit(X_train, y_train)
```
這行程式碼使用訓練集 `X_train` 和對應的標籤 `y_train` 來訓練 SVC 分類器。

5. **進行預測**：

```
predictions = clf_svm.predict(X_test)
```
這行程式碼使用訓練好的 SVC 分類器對測試集 `X_test` 進行預測，並將預測結果存儲在 `predictions` 變量中。

6. **生成分類報告**：

```
print(classification_report(y_test, predictions))
```
這行程式碼使用 `classification_report` 函數生成一個分類報告，該報告包含了精度、召回率、F1 值等指標，用於評估模型的性能。`y_test` 是測試集的真實標籤，`predictions` 是模型的預測結果。

In [ ]:
```
#原始資料標準化後的生成分類報告
from sklearn.svm import SVC, LinearSVC
from sklearn.metrics import classification_report
C = 0.1 # SVM regularization parameter
opts = dict(C = C, tol = 1e-6, max_iter = int(1e6))
clf_svm = SVC(kernel='linear', **opts)
clf_svm.fit(X_train_scaled, y_train)
predictions = clf_svm.predict(X_test_scaled)
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.95      0.91      0.93        22
           1       0.73      0.80      0.76        10
           2       0.79      1.00      0.88        11
           3       0.94      1.00      0.97        16
           4       0.75      0.60      0.67         5
           5       0.93      1.00      0.96        13
           6       0.95      1.00      0.97        19
           7       1.00      1.00      1.00        10
           8       1.00      1.00      1.00        14
           9       0.88      0.88      0.88         8
          10       0.93      0.93      0.93        14
          11       1.00      0.94      0.97        16
          12       0.78      0.78      0.78         9
          13       0.92      1.00      0.96        12
          14       0.81      1.00      0.90        13
          15       1.00      0.83      0.91        12
          16       1.00      1.00      1.00        14
          17       1.00      0.90      0.95        10
          18       1.00      1.00      1.00        16
          19       1.00      1.00      1.00         9
          20       1.00      0.91      0.95        22
          21       1.00      0.94      0.97        17
          22       1.00      1.00      1.00        12
          23       1.00      0.93      0.96        14
          24       1.00      0.93      0.97        15
          25       0.92      1.00      0.96        11
          26       1.00      1.00      1.00        17
          27       1.00      1.00      1.00        11
          28       1.00      1.00      1.00        10
          29       0.88      1.00      0.93         7
          30       1.00      0.94      0.97        16
          31       0.92      1.00      0.96        12
          32       1.00      1.00      1.00        10
          33       1.00      1.00      1.00        11
          34       0.83      0.77      0.80        13
          35       0.80      1.00      0.89         4
          36       1.00      0.83      0.91        12
          37       1.00      1.00      1.00        15

    accuracy                           0.95       482
   macro avg       0.94      0.94      0.94       482
weighted avg       0.95      0.95      0.95       482
```

In [ ]:

```
# 主成分資料的生成分類報告
from sklearn.svm import SVC, LinearSVC
from sklearn.metrics import classification_report
C = 0.1 # SVM regularization parameter
opts = dict(C = C, tol = 1e-6, max_iter = int(1e6))
clf_svm = SVC(kernel='linear', **opts)
clf_svm.fit(X_train_pca, y_train)
predictions = clf_svm.predict(X_test_pca)
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.47      0.32      0.38        22
           1       0.14      0.20      0.17        10
           2       0.40      0.55      0.46        11
           3       0.39      0.56      0.46        16
           4       0.18      0.40      0.25         5
           5       0.43      0.46      0.44        13
           6       0.70      0.37      0.48        19
           7       0.39      0.70      0.50        10
           8       0.20      0.14      0.17        14
           9       0.33      0.50      0.40         8
          10       0.40      0.57      0.47        14
          11       0.44      0.44      0.44        16
          12       0.36      0.56      0.43         9
          13       0.40      0.33      0.36        12
          14       0.35      0.46      0.40        13
          15       0.44      0.33      0.38        12
          16       0.17      0.07      0.10        14
          17       0.16      0.30      0.21        10
          18       0.44      0.25      0.32        16
          19       0.12      0.11      0.12         9
          20       0.79      0.50      0.61        22
          21       0.25      0.06      0.10        17
          22       0.35      0.50      0.41        12
          23       0.70      0.50      0.58        14
          24       0.75      0.40      0.52        15
          25       0.50      0.09      0.15        11
          26       0.65      0.65      0.65        17
          27       0.50      0.45      0.48        11
          28       0.60      0.30      0.40        10
          29       0.15      0.29      0.20         7
          30       0.77      0.62      0.69        16
          31       0.26      0.50      0.34        12
          32       0.40      0.40      0.40        10
          33       0.31      0.45      0.37        11
          34       0.29      0.31      0.30        13
          35       0.20      0.50      0.29         4
          36       0.57      0.33      0.42        12
          37       0.42      0.53      0.47        15

    accuracy                           0.40       482
   macro avg       0.40      0.40      0.38       482
weighted avg       0.44      0.40      0.39       482
```

```
c:\Users\W9H40\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\svm\_base.py:297: ConvergenceWarning: Solver terminated early (max_iter=1000000).  Consider pre-processing your data with StandardScaler or MinMaxScaler.
  warnings.warn(
```

各項指標的描述

這是分類報告，用於評估支持向量機 (SVM) 分類器在測試集上的性能。以下是報告中各個指標的解釋：

1. **precision**（精度）：指的是在所有被分類為某類的樣本中，真正屬於該類別的樣本所占的比例。比如對類別1而言，精度為0.95，意味著有95%的真正屬於類別1的樣本被成功地分類為類別1。

2. **recall**（召回率）：指的是在所有真正屬於某類的樣本中，被正確分類為該類別的樣本所占的比例。比如對類別2而言，召回率為0.73，意味著有73%的真正屬於類別2的樣本被成功地分類為類別。

3. **f1-score**（F1 值）：精度和召回率的加權調和平均值，是一個綜合考慮精度和召回率的指標。F1 值越高，模型的性能越好。

4. **support**（支援度）：每個類別在測試集中的樣本數量。

5. **accuracy**（準確率）：模型在測試集上的整體準確率，即所有正確預測的樣本數佔所有樣本數的比例。

6. **macro avg**（宏平均）：對所有類別的精度、召回率和 F1 值進行算術平均。它對每個類別的性能給予相同的權重，不考慮類別不平衡的問題。

7. **weighted avg**（加權平均）：對所有類別的精度、召回率和 F1 值進行加權平均，權重為每個類別的支援度（樣本數量）。這反映了不同類別在測試集中的重要性。

分類器三: 神經網路 (Neural Network)

step 1 **一樣先執行cross validation**

這段程式碼是用來進行 MLPClassifier 的參數調優。逐行解釋：

1. `from sklearn.neural_network import MLPClassifier`: 從 Scikit-Learn 中導入 MLPClassifier，這是一個多層感知機分類器，用於進行神經網路相關的分類任務。

2. `from sklearn.model_selection import GridSearchCV`: 從 Scikit-Learn 中導入 GridSearchCV，用於進行參數的網格搜索和交叉驗證。

3. `opts = dict(tol=1e-6, max_iter=int(1e6))`: 創建了一個字典 `opts`，其中包含了 MLPClassifier 的一些參數設置，例如 `tol`（容忍度）和 `max_iter`（最大迭代次數）。

4. `param_grid`: 定義了一個參數網格，包含了我們想要調優的參數及其可能的取值範圍。這些參數包括隱藏層大小、激活函數、求解器和 alpha（正則化參數）。

5. `cv`: 定義了交叉驗證的策略。在這裡，使用了 StratifiedShuffleSplit，它會將數據集分成 5 個子集，每個子集都保持類別分佈的一致性，並且使用 20% 的數據作為測試集。

6. `grid = GridSearchCV(estimator=MLPClassifier(**opts), param_grid=param_grid, cv=cv, scoring=['accuracy', 'f1_macro'], refit="accuracy")`: 初始化了一個 GridSearchCV 對象，指定了要搜索的模型（MLPClassifier），參數網格，交叉驗證策略，評估指標（這裡使用了 accuracy 和 f1_macro），以及要根據哪個指標來選擇最佳模型（在這裡是 accuracy）。

7. `grid.fit(X_train_scaled, y_train)`：調用 GridSearchCV 對象的 fit 方法，開始在訓練集上進行參數搜索和交叉驗證。

8. `print(grid.best_params_)`：打印出搜索過程中得到的最佳參數組合。

9. `print(grid.best_score_)`：打印出使用最佳參數組合在交叉驗證中獲得的最佳得分。

10. `print(grid.best_estimator_)`：打印出具有最佳參數的最佳估算器（即最佳模型）的詳細信息。

總的來說，這段程式碼是一個完整的參數調優流程，通過網格搜索和交叉驗證，找到了 MLPClassifier 的最佳參數組合，以提高其在給定數據集上的性能。

In [ ]:
```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV
opts = dict(tol = 1e-6, max_iter = int(1e6))
param_grid = {
    'hidden_layer_sizes': [(100,), (50, 50), (25, 25, 25)],
    'activation': ['relu', 'tanh', 'logistic'],
    'solver': ['sgd', 'adam'],
    'alpha': [0.0001, 0.001, 0.01],
}
cv = StratifiedShuffleSplit(n_splits=5, test_size=0.2, \
                            random_state=0) # 5-fold CV
grid = GridSearchCV(estimator=MLPClassifier(**opts), \
                    param_grid=param_grid, cv=cv,
                    scoring=['accuracy','f1_macro'], refit="accuracy")
grid.fit(X_train_scaled, y_train)

print(grid.best_params_)
print(grid.best_score_)
print(grid.best_estimator_)
```

<結果與討論>

- 試跑過上述程式，執行約一整晚沒有結果。
- 改用預設參數訓練及測試資料。

程式碼解析

這段程式碼使用了Scikit-learn中的MLPClassifier來建立一個多層感知器（MLP）分類模型，並使用該模型對測試數據進行預測，最後印出分類報告。

讓我們一步步解釋這段程式碼：

1. `from sklearn.neural_network import MLPClassifier`：導入MLPClassifier類別，該類別實現了多層感知器分類器，用於構建和訓練MLP模型。

2. `from sklearn.metrics import classification_report`：導入 classification_report函數，該函數用於生成分類模型的分類報告，其中包含準確率、召回率、F1分數等指標。

3. `hidden_layers = (30,)` 和 `activation = 'logistic'`：這裡指定了MLP模型的一些參數，包括隱藏層的大小和激活函數。在這個例子中，隱藏層只有30個單元，激活函數為logistic。

4. `opts = dict(hidden_layer_sizes=hidden_layers, verbose=True, activation=activation, tol=1e-6, max_iter=int(1e6))`：定義了一個字典opts，其中包含了MLPClassifier的參數設置。這些參數包括隱藏層的大小、是否打印訓練信息、激活函數、容忍誤差以及最大迭代次數等。

5. `solver = 'adam'`：指定了MLP模型的優化器為adam。adam是一種常用的優化算法，通常在深度學習中表現良好。

6. `clf_MLP = MLPClassifier(solver=solver, **opts)`：通過MLPClassifier類別創建了一個MLP分類器，並傳入了solver參數以及opts字典中的其他參數。

7. `clf_MLP.fit(X_train_scaled, y_train)`：使用訓練數據X_train_scaled和標籤y_train來訓練MLP模型。

8. `predictions = clf_MLP.predict(X_test_scaled)`：使用訓練好的模型對測試數據X_test_scaled進行預測，並將預測結果保存在predictions變量中。

9. `print(classification_report(y_test, predictions))`：印出分類報告，其中包含了模型在測試數據上的準確率、召回率、F1分數等評估指標。

In [ ]:
```
#原始資料標準化
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report
# hidden_layers = (512,) # one hidden layer
# activation = 'relu' # the default
hidden_layers = (30,)
activation = 'logistic'
opts = dict(hidden_layer_sizes = hidden_layers, verbose = True, \
activation = activation, tol = 1e-3, max_iter = int(1e3))
solver = 'adam' # default solver
clf_MLP = MLPClassifier(solver = solver, **opts)
clf_MLP.fit(X_train_scaled, y_train)
predictions = clf_MLP.predict(X_test_scaled)
print(classification_report(y_test, predictions))
```

```
Iteration 1, loss = 3.64319965
Iteration 2, loss = 3.54765900
Iteration 3, loss = 3.49198809
Iteration 4, loss = 3.43729481
Iteration 5, loss = 3.36159762
Iteration 6, loss = 3.27215046
Iteration 7, loss = 3.19420647
Iteration 8, loss = 3.10147534
Iteration 9, loss = 3.01172107
Iteration 10, loss = 2.89128738
Iteration 11, loss = 2.80600540
Iteration 12, loss = 2.70661404
Iteration 13, loss = 2.60029344
Iteration 14, loss = 2.50891366
Iteration 15, loss = 2.41386389
Iteration 16, loss = 2.31933274
Iteration 17, loss = 2.22154302
Iteration 18, loss = 2.11845942
Iteration 19, loss = 2.04248446
Iteration 20, loss = 1.95215152
Iteration 21, loss = 1.87605080
Iteration 22, loss = 1.80808720
Iteration 23, loss = 1.73274810
Iteration 24, loss = 1.68232123
Iteration 25, loss = 1.59497367
Iteration 26, loss = 1.52294851
Iteration 27, loss = 1.47003857
Iteration 28, loss = 1.38553878
Iteration 29, loss = 1.31585134
Iteration 30, loss = 1.27108981
Iteration 31, loss = 1.20985500
Iteration 32, loss = 1.15294393
Iteration 33, loss = 1.11314658
Iteration 34, loss = 1.06947628
Iteration 35, loss = 1.02602575
Iteration 36, loss = 0.98037716
Iteration 37, loss = 0.92990861
Iteration 38, loss = 0.88934823
Iteration 39, loss = 0.85521335
Iteration 40, loss = 0.82247320
Iteration 41, loss = 0.80864006
Iteration 42, loss = 0.80392095
Iteration 43, loss = 0.75879944
Iteration 44, loss = 0.70367611
Iteration 45, loss = 0.67754426
Iteration 46, loss = 0.65299909
Iteration 47, loss = 0.61803477
Iteration 48, loss = 0.59273230
Iteration 49, loss = 0.58646686
Iteration 50, loss = 0.58617300
Iteration 51, loss = 0.56605226
Iteration 52, loss = 0.53472244
Iteration 53, loss = 0.52637564
Iteration 54, loss = 0.50058244
Iteration 55, loss = 0.46143375
Iteration 56, loss = 0.46403675
Iteration 57, loss = 0.43723306
Iteration 58, loss = 0.42423022
Iteration 59, loss = 0.40449246
Iteration 60, loss = 0.39251926
```

```
Iteration 61, loss = 0.39046124
Iteration 62, loss = 0.39170031
Iteration 63, loss = 0.37760328
Iteration 64, loss = 0.36917937
Iteration 65, loss = 0.35065516
Iteration 66, loss = 0.34003423
Iteration 67, loss = 0.34660699
Iteration 68, loss = 0.35625367
Iteration 69, loss = 0.31845936
Iteration 70, loss = 0.32399758
Iteration 71, loss = 0.32696785
Iteration 72, loss = 0.31525593
Iteration 73, loss = 0.30413576
Iteration 74, loss = 0.30719912
Iteration 75, loss = 0.29322711
Iteration 76, loss = 0.29997394
Iteration 77, loss = 0.30410506
Iteration 78, loss = 0.25644109
Iteration 79, loss = 0.25215718
Iteration 80, loss = 0.25115584
Iteration 81, loss = 0.23318135
Iteration 82, loss = 0.22262760
Iteration 83, loss = 0.21236687
Iteration 84, loss = 0.22213037
Iteration 85, loss = 0.21051086
Iteration 86, loss = 0.26199108
Iteration 87, loss = 0.22199747
Iteration 88, loss = 0.21638899
Iteration 89, loss = 0.22626107
Iteration 90, loss = 0.24226706
Iteration 91, loss = 0.23122002
Iteration 92, loss = 0.22990489
Iteration 93, loss = 0.20114565
Iteration 94, loss = 0.19224105
Iteration 95, loss = 0.20870721
Iteration 96, loss = 0.19870156
Iteration 97, loss = 0.20260018
Iteration 98, loss = 0.18628277
Iteration 99, loss = 0.18540760
Iteration 100, loss = 0.18013726
Iteration 101, loss = 0.19209107
Iteration 102, loss = 0.17092262
Iteration 103, loss = 0.17304052
Iteration 104, loss = 0.15928953
Iteration 105, loss = 0.17987527
Iteration 106, loss = 0.19098370
Iteration 107, loss = 0.20251211
Iteration 108, loss = 0.20710799
Iteration 109, loss = 0.20033877
Iteration 110, loss = 0.17798489
Iteration 111, loss = 0.17108745
Iteration 112, loss = 0.15735443
Iteration 113, loss = 0.14171940
Iteration 114, loss = 0.15109438
Iteration 115, loss = 0.16407415
Iteration 116, loss = 0.15765824
Iteration 117, loss = 0.16678965
Iteration 118, loss = 0.17262709
Iteration 119, loss = 0.15020351
Iteration 120, loss = 0.12785518
```

```
Iteration 121, loss = 0.17826698
Iteration 122, loss = 0.15631448
Iteration 123, loss = 0.15926900
Iteration 124, loss = 0.14791477
Iteration 125, loss = 0.13863034
Iteration 126, loss = 0.14703455
Iteration 127, loss = 0.14872375
Iteration 128, loss = 0.14777523
Iteration 129, loss = 0.15841735
Iteration 130, loss = 0.15062164
Iteration 131, loss = 0.13835015
Training loss did not improve more than tol=0.001000 for 10 consecutive epochs. S
topping.
              precision    recall  f1-score   support

           0       1.00      0.86      0.93        22
           1       0.69      0.90      0.78        10
           2       1.00      1.00      1.00        11
           3       1.00      0.94      0.97        16
           4       1.00      0.60      0.75         5
           5       1.00      1.00      1.00        13
           6       0.95      1.00      0.97        19
           7       1.00      1.00      1.00        10
           8       0.93      1.00      0.97        14
           9       1.00      0.75      0.86         8
          10       1.00      0.86      0.92        14
          11       1.00      0.94      0.97        16
          12       0.90      1.00      0.95         9
          13       0.92      1.00      0.96        12
          14       0.87      1.00      0.93        13
          15       1.00      0.75      0.86        12
          16       0.93      1.00      0.97        14
          17       1.00      0.90      0.95        10
          18       0.94      1.00      0.97        16
          19       0.90      1.00      0.95         9
          20       1.00      0.77      0.87        22
          21       0.89      0.94      0.91        17
          22       1.00      1.00      1.00        12
          23       1.00      0.86      0.92        14
          24       1.00      0.87      0.93        15
          25       1.00      1.00      1.00        11
          26       0.89      0.94      0.91        17
          27       1.00      1.00      1.00        11
          28       0.77      1.00      0.87        10
          29       1.00      1.00      1.00         7
          30       1.00      0.88      0.93        16
          31       0.75      1.00      0.86        12
          32       1.00      1.00      1.00        10
          33       1.00      1.00      1.00        11
          34       0.82      0.69      0.75        13
          35       0.36      1.00      0.53         4
          36       0.92      0.92      0.92        12
          37       1.00      1.00      1.00        15

    accuracy                           0.93       482
   macro avg       0.93      0.93      0.92       482
weighted avg       0.95      0.93      0.93       482
```

輸出內容解析

這段程式輸出了一系列迭代次數和損失值。這個程式可能是一個訓練過程的一部分，通常是在機器學習或深度學習中使用。這裡的「loss」代表了模型在每個迭代中的損失值，損失值通常用來衡量模型預測和實際值之間的差異。

這些迭代和損失值給了我們一個了解模型訓練過程的指標。在迭代的早期，損失值相對較高，逐漸降低直到穩定。這是因為模型通過梯度下降等優化算法逐步調整參數以最小化損失值。

在這個例子中，隨著迭代次數的增加，損失值從3.64逐漸降低到1.59。這可能表示模型的性能不斷改善，損失越來越小，模型對於訓練數據的擬合越來越好。

接下來的輸出可能是模型在測試數據上的表現評估結果。準確率、精確率、召回率和F1值等指標提供了對模型在不同類別上表現的評價。在這個例子中，總體準確率為0.93，這意味著模型在測試數據上的表現相當不錯。

總的來說，這段程式展示了一個訓練過程的動態，從損失值的改善到最終的模型性能評估。

畫出測試資料的confusion matrix

這段程式碼使用了Scikit-learn中的ConfusionMatrixDisplay模組和Matplotlib來繪製混淆矩陣。讓我們一步步解釋：

1. `from sklearn.metrics import ConfusionMatrixDisplay`：導入ConfusionMatrixDisplay模組，該模組提供了一個便捷的方法來可視化混淆矩陣。
2. `import matplotlib.pyplot as plt`：導入Matplotlib庫，用於繪製圖表。
3. `fig, ax = plt.subplots(1, 1, figsize=(12,12))`：創建一個12x12大小的子圖，用於顯示混淆矩陣。
4. `score = 100*clf_MLP.score(X_test_scaled, y_test)`：使用MLP模型的score方法計算測試數據的準確率，並將其轉換為百分比形式。
5. `title = 'Testing score ={:.2f}%'.format(score)`：根據測試準確率創建標題，並將準確率插入標題字符串中。
6. `disp = ConfusionMatrixDisplay.from_estimator`：使用ConfusionMatrixDisplay模組的from_estimator方法來創建混淆矩陣的可視化。它需要以下參數：
    - `clf_MLP`：要評估的分類器（MLP模型）。
    - `X_test_scaled`：測試數據。
    - `y_test`：測試數據的真實標籤。
    - `xticks_rotation=45`：x軸刻度標籤的旋轉角度，這裡設置為45度。
    - `cmap=plt.cm.Blues`：顏色映射，用於設置混淆矩陣的顏色。

- `normalize='true'`：是否對混淆矩陣進行正規化，這裡設置為true，表示對每一行進行正規化。
- `ax=ax`：指定子圖來繪製混淆矩陣。

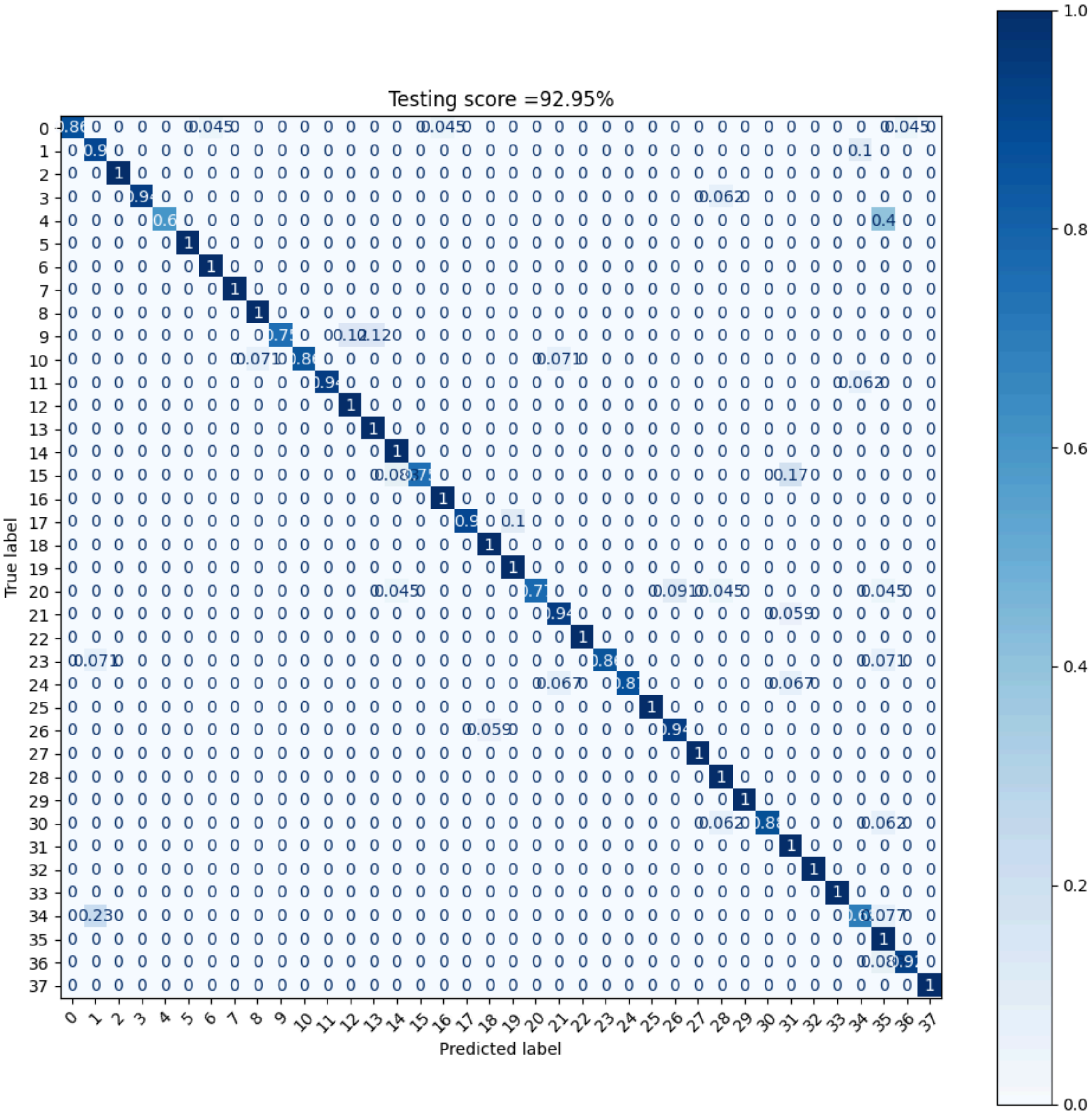
7. `disp.ax_.set_title(title)`：設置子圖的標題為剛剛創建的標題字符串。

8. `plt.show()`：顯示混淆矩陣圖表。

總的來說，這段程式碼用於創建並顯示測試數據的混淆矩陣，並在圖表中包含了測試準確率的信息。

In [ ]:

```
from sklearn.metrics import ConfusionMatrixDisplay
import matplotlib.pyplot as plt
fig, ax = plt.subplots(1, 1, figsize=(12,12))
score = 100*clf_MLP.score(X_test_scaled, y_test)
title = 'Testing score ={:.2f}%'.format(score)
disp = ConfusionMatrixDisplay.from_estimator(
clf_MLP,
X_test_scaled,
y_test,
xticks_rotation=45, #’vertical’,
# display_labels=class_names,
cmap=plt.cm.Blues,
normalize='true',
ax = ax
)
disp.ax_.set_title(title)
plt.show()
```



In [ ]:

```
# 主成分資料
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report
# hidden_layers = (512,) # one hidden layer
# activation = ’relu’ # the default
hidden_layers = (30,)
activation = 'logistic'
opts = dict(hidden_layer_sizes = hidden_layers, verbose = True, \
activation = activation, tol = 1e-6, max_iter = int(1e6))
# solver = ’sgd’ # not efficient, need more tuning
# solver = ’lbfgs’ # not suitable here
solver = 'adam' # default solver
clf_MLP = MLPClassifier(solver = solver, **opts)
clf_MLP.fit(X_train_pca, y_train)
predictions = clf_MLP.predict(X_test_pca)
print(classification_report(y_test, predictions))
```

```
Iteration 1, loss = 3.68092551
Iteration 2, loss = 3.65262925
Iteration 3, loss = 3.63046674
Iteration 4, loss = 3.61212955
Iteration 5, loss = 3.59525881
Iteration 6, loss = 3.58005730
Iteration 7, loss = 3.56465225
Iteration 8, loss = 3.55012403
Iteration 9, loss = 3.53616502
Iteration 10, loss = 3.52110034
Iteration 11, loss = 3.50567719
Iteration 12, loss = 3.48936998
Iteration 13, loss = 3.47216204
Iteration 14, loss = 3.45457609
Iteration 15, loss = 3.43636286
Iteration 16, loss = 3.41744342
Iteration 17, loss = 3.39742614
Iteration 18, loss = 3.37673078
Iteration 19, loss = 3.35634749
Iteration 20, loss = 3.33563049
Iteration 21, loss = 3.31595154
Iteration 22, loss = 3.29610160
Iteration 23, loss = 3.27586665
Iteration 24, loss = 3.25504276
Iteration 25, loss = 3.23450647
Iteration 26, loss = 3.21241019
Iteration 27, loss = 3.19220795
Iteration 28, loss = 3.17099463
Iteration 29, loss = 3.15033160
Iteration 30, loss = 3.12940708
Iteration 31, loss = 3.10870415
Iteration 32, loss = 3.08859082
Iteration 33, loss = 3.06912294
Iteration 34, loss = 3.04936018
Iteration 35, loss = 3.03020020
Iteration 36, loss = 3.01216647
Iteration 37, loss = 2.99405448
Iteration 38, loss = 2.97647286
Iteration 39, loss = 2.95854838
Iteration 40, loss = 2.94241163
Iteration 41, loss = 2.92658927
Iteration 42, loss = 2.91155393
Iteration 43, loss = 2.89658792
Iteration 44, loss = 2.88241804
Iteration 45, loss = 2.86882082
Iteration 46, loss = 2.85598825
Iteration 47, loss = 2.84291243
Iteration 48, loss = 2.83033583
Iteration 49, loss = 2.81777797
Iteration 50, loss = 2.80615642
Iteration 51, loss = 2.79337996
Iteration 52, loss = 2.78179153
Iteration 53, loss = 2.77058947
Iteration 54, loss = 2.75928028
Iteration 55, loss = 2.74867018
Iteration 56, loss = 2.73816812
Iteration 57, loss = 2.72771699
Iteration 58, loss = 2.71735085
Iteration 59, loss = 2.70667688
Iteration 60, loss = 2.69634541
```

```
Iteration 61, loss = 2.68672254
Iteration 62, loss = 2.67746939
Iteration 63, loss = 2.66864074
Iteration 64, loss = 2.65951736
Iteration 65, loss = 2.65047283
Iteration 66, loss = 2.64196875
Iteration 67, loss = 2.63294889
Iteration 68, loss = 2.62484086
Iteration 69, loss = 2.61665679
Iteration 70, loss = 2.60852731
Iteration 71, loss = 2.60090207
Iteration 72, loss = 2.59266204
Iteration 73, loss = 2.58568947
Iteration 74, loss = 2.57809424
Iteration 75, loss = 2.57107240
Iteration 76, loss = 2.56417295
Iteration 77, loss = 2.55738968
Iteration 78, loss = 2.55070680
Iteration 79, loss = 2.54433717
Iteration 80, loss = 2.53743768
Iteration 81, loss = 2.53141659
Iteration 82, loss = 2.52546063
Iteration 83, loss = 2.51928895
Iteration 84, loss = 2.51339104
Iteration 85, loss = 2.50691166
Iteration 86, loss = 2.50088701
Iteration 87, loss = 2.49528369
Iteration 88, loss = 2.48946126
Iteration 89, loss = 2.48440019
Iteration 90, loss = 2.47870557
Iteration 91, loss = 2.47362661
Iteration 92, loss = 2.46733554
Iteration 93, loss = 2.46276143
Iteration 94, loss = 2.45748387
Iteration 95, loss = 2.45247379
Iteration 96, loss = 2.44713397
Iteration 97, loss = 2.44251582
Iteration 98, loss = 2.43756657
Iteration 99, loss = 2.43255161
Iteration 100, loss = 2.42826378
Iteration 101, loss = 2.42318684
Iteration 102, loss = 2.41892600
Iteration 103, loss = 2.41423092
Iteration 104, loss = 2.40975807
Iteration 105, loss = 2.40543147
Iteration 106, loss = 2.40070519
Iteration 107, loss = 2.39684367
Iteration 108, loss = 2.39225274
Iteration 109, loss = 2.38848142
Iteration 110, loss = 2.38424761
Iteration 111, loss = 2.37995459
Iteration 112, loss = 2.37572626
Iteration 113, loss = 2.37182222
Iteration 114, loss = 2.36790433
Iteration 115, loss = 2.36406715
Iteration 116, loss = 2.36008587
Iteration 117, loss = 2.35603389
Iteration 118, loss = 2.35239787
Iteration 119, loss = 2.34907675
Iteration 120, loss = 2.34505947
```

```
Iteration 121, loss = 2.34087926
Iteration 122, loss = 2.33735313
Iteration 123, loss = 2.33422072
Iteration 124, loss = 2.33040970
Iteration 125, loss = 2.32634412
Iteration 126, loss = 2.32282030
Iteration 127, loss = 2.31982041
Iteration 128, loss = 2.31684506
Iteration 129, loss = 2.31303059
Iteration 130, loss = 2.30969529
Iteration 131, loss = 2.30614327
Iteration 132, loss = 2.30308456
Iteration 133, loss = 2.29949542
Iteration 134, loss = 2.29582137
Iteration 135, loss = 2.29282798
Iteration 136, loss = 2.29008929
Iteration 137, loss = 2.28651916
Iteration 138, loss = 2.28356643
Iteration 139, loss = 2.27990506
Iteration 140, loss = 2.27678997
Iteration 141, loss = 2.27438006
Iteration 142, loss = 2.27030740
Iteration 143, loss = 2.26711761
Iteration 144, loss = 2.26383119
Iteration 145, loss = 2.26135284
Iteration 146, loss = 2.25793479
Iteration 147, loss = 2.25562236
Iteration 148, loss = 2.25182308
Iteration 149, loss = 2.24948162
Iteration 150, loss = 2.24634642
Iteration 151, loss = 2.24333927
Iteration 152, loss = 2.24085537
Iteration 153, loss = 2.23784305
Iteration 154, loss = 2.23461677
Iteration 155, loss = 2.23221182
Iteration 156, loss = 2.22916363
Iteration 157, loss = 2.22639057
Iteration 158, loss = 2.22336605
Iteration 159, loss = 2.22121590
Iteration 160, loss = 2.21823281
Iteration 161, loss = 2.21556346
Iteration 162, loss = 2.21279178
Iteration 163, loss = 2.20978469
Iteration 164, loss = 2.20749962
Iteration 165, loss = 2.20483020
Iteration 166, loss = 2.20220475
Iteration 167, loss = 2.20012125
Iteration 168, loss = 2.19744103
Iteration 169, loss = 2.19453879
Iteration 170, loss = 2.19244099
Iteration 171, loss = 2.18897497
Iteration 172, loss = 2.18685007
Iteration 173, loss = 2.18455772
Iteration 174, loss = 2.18214985
Iteration 175, loss = 2.17936585
Iteration 176, loss = 2.17717964
Iteration 177, loss = 2.17460757
Iteration 178, loss = 2.17236804
Iteration 179, loss = 2.16971572
Iteration 180, loss = 2.16748565
```

```
Iteration 181, loss = 2.16452963
Iteration 182, loss = 2.16317919
Iteration 183, loss = 2.16019537
Iteration 184, loss = 2.15820882
Iteration 185, loss = 2.15555917
Iteration 186, loss = 2.15487125
Iteration 187, loss = 2.15140303
Iteration 188, loss = 2.14881020
Iteration 189, loss = 2.14703930
Iteration 190, loss = 2.14445615
Iteration 191, loss = 2.14189987
Iteration 192, loss = 2.14032602
Iteration 193, loss = 2.13798039
Iteration 194, loss = 2.13622922
Iteration 195, loss = 2.13363749
Iteration 196, loss = 2.13114920
Iteration 197, loss = 2.12845101
Iteration 198, loss = 2.12663290
Iteration 199, loss = 2.12550851
Iteration 200, loss = 2.12239031
Iteration 201, loss = 2.12051089
Iteration 202, loss = 2.11832646
Iteration 203, loss = 2.11693130
Iteration 204, loss = 2.11416309
Iteration 205, loss = 2.11281673
Iteration 206, loss = 2.11087686
Iteration 207, loss = 2.10857371
Iteration 208, loss = 2.10689931
Iteration 209, loss = 2.10481015
Iteration 210, loss = 2.10253659
Iteration 211, loss = 2.10119400
Iteration 212, loss = 2.09922594
Iteration 213, loss = 2.09677019
Iteration 214, loss = 2.09512874
Iteration 215, loss = 2.09228867
Iteration 216, loss = 2.09040802
Iteration 217, loss = 2.08949393
Iteration 218, loss = 2.08770403
Iteration 219, loss = 2.08493687
Iteration 220, loss = 2.08299088
Iteration 221, loss = 2.08177529
Iteration 222, loss = 2.07939789
Iteration 223, loss = 2.07832396
Iteration 224, loss = 2.07584691
Iteration 225, loss = 2.07373377
Iteration 226, loss = 2.07247891
Iteration 227, loss = 2.07067950
Iteration 228, loss = 2.06830334
Iteration 229, loss = 2.06687978
Iteration 230, loss = 2.06473962
Iteration 231, loss = 2.06361580
Iteration 232, loss = 2.06080336
Iteration 233, loss = 2.06008866
Iteration 234, loss = 2.05781589
Iteration 235, loss = 2.05643192
Iteration 236, loss = 2.05395849
Iteration 237, loss = 2.05245924
Iteration 238, loss = 2.05052466
Iteration 239, loss = 2.04959173
Iteration 240, loss = 2.04739770
```

```
Iteration 241, loss = 2.04611516
Iteration 242, loss = 2.04414961
Iteration 243, loss = 2.04243711
Iteration 244, loss = 2.04068470
Iteration 245, loss = 2.03890491
Iteration 246, loss = 2.03747360
Iteration 247, loss = 2.03592119
Iteration 248, loss = 2.03353682
Iteration 249, loss = 2.03249289
Iteration 250, loss = 2.03091391
Iteration 251, loss = 2.02904626
Iteration 252, loss = 2.02779321
Iteration 253, loss = 2.02635406
Iteration 254, loss = 2.02505219
Iteration 255, loss = 2.02284946
Iteration 256, loss = 2.02108691
Iteration 257, loss = 2.01984007
Iteration 258, loss = 2.01781644
Iteration 259, loss = 2.01652831
Iteration 260, loss = 2.01461997
Iteration 261, loss = 2.01346293
Iteration 262, loss = 2.01208220
Iteration 263, loss = 2.01048050
Iteration 264, loss = 2.00902494
Iteration 265, loss = 2.00690827
Iteration 266, loss = 2.00565959
Iteration 267, loss = 2.00407787
Iteration 268, loss = 2.00281956
Iteration 269, loss = 2.00131324
Iteration 270, loss = 1.99960699
Iteration 271, loss = 1.99773978
Iteration 272, loss = 1.99680945
Iteration 273, loss = 1.99543145
Iteration 274, loss = 1.99384657
Iteration 275, loss = 1.99233889
Iteration 276, loss = 1.99139064
Iteration 277, loss = 1.98962637
Iteration 278, loss = 1.98798288
Iteration 279, loss = 1.98687536
Iteration 280, loss = 1.98532485
Iteration 281, loss = 1.98371234
Iteration 282, loss = 1.98211562
Iteration 283, loss = 1.98137377
Iteration 284, loss = 1.97977878
Iteration 285, loss = 1.97933634
Iteration 286, loss = 1.97730081
Iteration 287, loss = 1.97569226
Iteration 288, loss = 1.97416606
Iteration 289, loss = 1.97274144
Iteration 290, loss = 1.97120473
Iteration 291, loss = 1.96990407
Iteration 292, loss = 1.96870961
Iteration 293, loss = 1.96756589
Iteration 294, loss = 1.96592961
Iteration 295, loss = 1.96513392
Iteration 296, loss = 1.96352709
Iteration 297, loss = 1.96319896
Iteration 298, loss = 1.96065862
Iteration 299, loss = 1.95972509
Iteration 300, loss = 1.95840045
```

```
Iteration 301, loss = 1.95631953
Iteration 302, loss = 1.95529730
Iteration 303, loss = 1.95394460
Iteration 304, loss = 1.95212537
Iteration 305, loss = 1.95122468
Iteration 306, loss = 1.95085423
Iteration 307, loss = 1.94959904
Iteration 308, loss = 1.94751914
Iteration 309, loss = 1.94531493
Iteration 310, loss = 1.94507183
Iteration 311, loss = 1.94333217
Iteration 312, loss = 1.94244725
Iteration 313, loss = 1.94135441
Iteration 314, loss = 1.93944501
Iteration 315, loss = 1.93890320
Iteration 316, loss = 1.93745053
Iteration 317, loss = 1.93649845
Iteration 318, loss = 1.93446308
Iteration 319, loss = 1.93364157
Iteration 320, loss = 1.93224058
Iteration 321, loss = 1.93136753
Iteration 322, loss = 1.92984407
Iteration 323, loss = 1.92888283
Iteration 324, loss = 1.92794063
Iteration 325, loss = 1.92629713
Iteration 326, loss = 1.92602302
Iteration 327, loss = 1.92445739
Iteration 328, loss = 1.92309507
Iteration 329, loss = 1.92167226
Iteration 330, loss = 1.92059925
Iteration 331, loss = 1.91936776
Iteration 332, loss = 1.91889140
Iteration 333, loss = 1.91692371
Iteration 334, loss = 1.91664532
Iteration 335, loss = 1.91519687
Iteration 336, loss = 1.91367761
Iteration 337, loss = 1.91302905
Iteration 338, loss = 1.91149743
Iteration 339, loss = 1.91089101
Iteration 340, loss = 1.90965515
Iteration 341, loss = 1.90831803
Iteration 342, loss = 1.90842663
Iteration 343, loss = 1.90634581
Iteration 344, loss = 1.90611955
Iteration 345, loss = 1.90492569
Iteration 346, loss = 1.90293852
Iteration 347, loss = 1.90237529
Iteration 348, loss = 1.90070488
Iteration 349, loss = 1.90000652
Iteration 350, loss = 1.89949214
Iteration 351, loss = 1.89798856
Iteration 352, loss = 1.89703517
Iteration 353, loss = 1.89586177
Iteration 354, loss = 1.89450447
Iteration 355, loss = 1.89369943
Iteration 356, loss = 1.89258592
Iteration 357, loss = 1.89188375
Iteration 358, loss = 1.89059546
Iteration 359, loss = 1.89030523
Iteration 360, loss = 1.88878135
```

```
Iteration 361, loss = 1.88767679
Iteration 362, loss = 1.88698534
Iteration 363, loss = 1.88624079
Iteration 364, loss = 1.88455822
Iteration 365, loss = 1.88373485
Iteration 366, loss = 1.88282630
Iteration 367, loss = 1.88150196
Iteration 368, loss = 1.88143907
Iteration 369, loss = 1.88079307
Iteration 370, loss = 1.87886051
Iteration 371, loss = 1.87821453
Iteration 372, loss = 1.87710276
Iteration 373, loss = 1.87578385
Iteration 374, loss = 1.87541086
Iteration 375, loss = 1.87441543
Iteration 376, loss = 1.87351219
Iteration 377, loss = 1.87269015
Iteration 378, loss = 1.87109711
Iteration 379, loss = 1.87071705
Iteration 380, loss = 1.86959921
Iteration 381, loss = 1.86874930
Iteration 382, loss = 1.86798780
Iteration 383, loss = 1.86666189
Iteration 384, loss = 1.86616621
Iteration 385, loss = 1.86476007
Iteration 386, loss = 1.86384577
Iteration 387, loss = 1.86402885
Iteration 388, loss = 1.86299199
Iteration 389, loss = 1.86156237
Iteration 390, loss = 1.86038988
Iteration 391, loss = 1.86054437
Iteration 392, loss = 1.85899581
Iteration 393, loss = 1.85826281
Iteration 394, loss = 1.85702483
Iteration 395, loss = 1.85669923
Iteration 396, loss = 1.85538341
Iteration 397, loss = 1.85430745
Iteration 398, loss = 1.85331176
Iteration 399, loss = 1.85279286
Iteration 400, loss = 1.85236191
Iteration 401, loss = 1.85114623
Iteration 402, loss = 1.85004635
Iteration 403, loss = 1.84945228
Iteration 404, loss = 1.84864767
Iteration 405, loss = 1.84794836
Iteration 406, loss = 1.84695224
Iteration 407, loss = 1.84658646
Iteration 408, loss = 1.84558431
Iteration 409, loss = 1.84453076
Iteration 410, loss = 1.84373187
Iteration 411, loss = 1.84382721
Iteration 412, loss = 1.84196983
Iteration 413, loss = 1.84129494
Iteration 414, loss = 1.84015940
Iteration 415, loss = 1.83968902
Iteration 416, loss = 1.83854411
Iteration 417, loss = 1.83674711
Iteration 418, loss = 1.83700634
Iteration 419, loss = 1.83664994
Iteration 420, loss = 1.83540210
```

```
Iteration 421, loss = 1.83466485
Iteration 422, loss = 1.83352581
Iteration 423, loss = 1.83253639
Iteration 424, loss = 1.83186307
Iteration 425, loss = 1.83180837
Iteration 426, loss = 1.83005702
Iteration 427, loss = 1.82980604
Iteration 428, loss = 1.82904317
Iteration 429, loss = 1.82742617
Iteration 430, loss = 1.82811344
Iteration 431, loss = 1.82665615
Iteration 432, loss = 1.82629255
Iteration 433, loss = 1.82548501
Iteration 434, loss = 1.82365655
Iteration 435, loss = 1.82376880
Iteration 436, loss = 1.82264834
Iteration 437, loss = 1.82243057
Iteration 438, loss = 1.82150396
Iteration 439, loss = 1.82037011
Iteration 440, loss = 1.82008685
Iteration 441, loss = 1.81914899
Iteration 442, loss = 1.81852695
Iteration 443, loss = 1.81759026
Iteration 444, loss = 1.81659214
Iteration 445, loss = 1.81553464
Iteration 446, loss = 1.81470303
Iteration 447, loss = 1.81435776
Iteration 448, loss = 1.81364321
Iteration 449, loss = 1.81278199
Iteration 450, loss = 1.81245875
Iteration 451, loss = 1.81132587
Iteration 452, loss = 1.81045290
Iteration 453, loss = 1.81045015
Iteration 454, loss = 1.81005110
Iteration 455, loss = 1.80868955
Iteration 456, loss = 1.80812622
Iteration 457, loss = 1.80709236
Iteration 458, loss = 1.80615401
Iteration 459, loss = 1.80617047
Iteration 460, loss = 1.80532833
Iteration 461, loss = 1.80424652
Iteration 462, loss = 1.80324940
Iteration 463, loss = 1.80301403
Iteration 464, loss = 1.80291820
Iteration 465, loss = 1.80123341
Iteration 466, loss = 1.80096022
Iteration 467, loss = 1.80067403
Iteration 468, loss = 1.79863377
Iteration 469, loss = 1.79904888
Iteration 470, loss = 1.79815452
Iteration 471, loss = 1.79806450
Iteration 472, loss = 1.79689082
Iteration 473, loss = 1.79613931
Iteration 474, loss = 1.79588070
Iteration 475, loss = 1.79436499
Iteration 476, loss = 1.79397401
Iteration 477, loss = 1.79329005
Iteration 478, loss = 1.79309236
Iteration 479, loss = 1.79160953
Iteration 480, loss = 1.79142052
```

```
Iteration 481, loss = 1.79047629
Iteration 482, loss = 1.79006118
Iteration 483, loss = 1.78899967
Iteration 484, loss = 1.78820503
Iteration 485, loss = 1.78778898
Iteration 486, loss = 1.78645002
Iteration 487, loss = 1.78615088
Iteration 488, loss = 1.78665385
Iteration 489, loss = 1.78488625
Iteration 490, loss = 1.78393368
Iteration 491, loss = 1.78382522
Iteration 492, loss = 1.78284842
Iteration 493, loss = 1.78217373
Iteration 494, loss = 1.78138895
Iteration 495, loss = 1.78036605
Iteration 496, loss = 1.78066791
Iteration 497, loss = 1.77888059
Iteration 498, loss = 1.77893493
Iteration 499, loss = 1.77908833
Iteration 500, loss = 1.77744292
Iteration 501, loss = 1.77788816
Iteration 502, loss = 1.77641981
Iteration 503, loss = 1.77585092
Iteration 504, loss = 1.77579936
Iteration 505, loss = 1.77491565
Iteration 506, loss = 1.77366469
Iteration 507, loss = 1.77304398
Iteration 508, loss = 1.77250605
Iteration 509, loss = 1.77124956
Iteration 510, loss = 1.77093682
Iteration 511, loss = 1.77057459
Iteration 512, loss = 1.76980566
Iteration 513, loss = 1.77003514
Iteration 514, loss = 1.76852751
Iteration 515, loss = 1.76820755
Iteration 516, loss = 1.76844594
Iteration 517, loss = 1.76707209
Iteration 518, loss = 1.76594973
Iteration 519, loss = 1.76565974
Iteration 520, loss = 1.76528603
Iteration 521, loss = 1.76520410
Iteration 522, loss = 1.76374155
Iteration 523, loss = 1.76380403
Iteration 524, loss = 1.76326932
Iteration 525, loss = 1.76214440
Iteration 526, loss = 1.76114457
Iteration 527, loss = 1.76112448
Iteration 528, loss = 1.76030419
Iteration 529, loss = 1.76002893
Iteration 530, loss = 1.75918237
Iteration 531, loss = 1.75877364
Iteration 532, loss = 1.75794978
Iteration 533, loss = 1.75805104
Iteration 534, loss = 1.75709671
Iteration 535, loss = 1.75566692
Iteration 536, loss = 1.75589952
Iteration 537, loss = 1.75568070
Iteration 538, loss = 1.75408868
Iteration 539, loss = 1.75386226
Iteration 540, loss = 1.75277899
```

```
Iteration 541, loss = 1.75271851
Iteration 542, loss = 1.75203087
Iteration 543, loss = 1.75215941
Iteration 544, loss = 1.75141093
Iteration 545, loss = 1.74966685
Iteration 546, loss = 1.74948448
Iteration 547, loss = 1.74975833
Iteration 548, loss = 1.74923430
Iteration 549, loss = 1.74813222
Iteration 550, loss = 1.74805213
Iteration 551, loss = 1.74719092
Iteration 552, loss = 1.74649428
Iteration 553, loss = 1.74584103
Iteration 554, loss = 1.74544446
Iteration 555, loss = 1.74552209
Iteration 556, loss = 1.74449838
Iteration 557, loss = 1.74450033
Iteration 558, loss = 1.74296174
Iteration 559, loss = 1.74230232
Iteration 560, loss = 1.74118181
Iteration 561, loss = 1.74077563
Iteration 562, loss = 1.74030158
Iteration 563, loss = 1.74080043
Iteration 564, loss = 1.73935872
Iteration 565, loss = 1.73884435
Iteration 566, loss = 1.73886445
Iteration 567, loss = 1.73897156
Iteration 568, loss = 1.73815383
Iteration 569, loss = 1.73658356
Iteration 570, loss = 1.73649224
Iteration 571, loss = 1.73569707
Iteration 572, loss = 1.73526949
Iteration 573, loss = 1.73491741
Iteration 574, loss = 1.73440718
Iteration 575, loss = 1.73451609
Iteration 576, loss = 1.73262194
Iteration 577, loss = 1.73255344
Iteration 578, loss = 1.73140171
Iteration 579, loss = 1.73146358
Iteration 580, loss = 1.73162666
Iteration 581, loss = 1.72948960
Iteration 582, loss = 1.73000140
Iteration 583, loss = 1.72940413
Iteration 584, loss = 1.72904493
Iteration 585, loss = 1.72855047
Iteration 586, loss = 1.72836619
Iteration 587, loss = 1.72786612
Iteration 588, loss = 1.72701383
Iteration 589, loss = 1.72636861
Iteration 590, loss = 1.72554245
Iteration 591, loss = 1.72557553
Iteration 592, loss = 1.72395321
Iteration 593, loss = 1.72408807
Iteration 594, loss = 1.72332108
Iteration 595, loss = 1.72280180
Iteration 596, loss = 1.72240670
Iteration 597, loss = 1.72166674
Iteration 598, loss = 1.72202437
Iteration 599, loss = 1.72098066
Iteration 600, loss = 1.72076282
```

```
Iteration 601, loss = 1.72011892
Iteration 602, loss = 1.72005816
Iteration 603, loss = 1.71946047
Iteration 604, loss = 1.71878292
Iteration 605, loss = 1.71843372
Iteration 606, loss = 1.71763007
Iteration 607, loss = 1.71770739
Iteration 608, loss = 1.71637349
Iteration 609, loss = 1.71592097
Iteration 610, loss = 1.71599745
Iteration 611, loss = 1.71499679
Iteration 612, loss = 1.71457491
Iteration 613, loss = 1.71454282
Iteration 614, loss = 1.71364942
Iteration 615, loss = 1.71418700
Iteration 616, loss = 1.71226643
Iteration 617, loss = 1.71289085
Iteration 618, loss = 1.71178859
Iteration 619, loss = 1.71166910
Iteration 620, loss = 1.71114104
Iteration 621, loss = 1.71036831
Iteration 622, loss = 1.71057078
Iteration 623, loss = 1.70925324
Iteration 624, loss = 1.70892186
Iteration 625, loss = 1.70845387
Iteration 626, loss = 1.70830890
Iteration 627, loss = 1.70770087
Iteration 628, loss = 1.70719977
Iteration 629, loss = 1.70694171
Iteration 630, loss = 1.70672396
Iteration 631, loss = 1.70485062
Iteration 632, loss = 1.70535253
Iteration 633, loss = 1.70379516
Iteration 634, loss = 1.70423172
Iteration 635, loss = 1.70398001
Iteration 636, loss = 1.70364339
Iteration 637, loss = 1.70295850
Iteration 638, loss = 1.70248569
Iteration 639, loss = 1.70227015
Iteration 640, loss = 1.70155925
Iteration 641, loss = 1.70092012
Iteration 642, loss = 1.70059412
Iteration 643, loss = 1.70038867
Iteration 644, loss = 1.69924402
Iteration 645, loss = 1.69922374
Iteration 646, loss = 1.69851172
Iteration 647, loss = 1.69816144
Iteration 648, loss = 1.69726841
Iteration 649, loss = 1.69756425
Iteration 650, loss = 1.69735294
Iteration 651, loss = 1.69627518
Iteration 652, loss = 1.69612585
Iteration 653, loss = 1.69463952
Iteration 654, loss = 1.69524421
Iteration 655, loss = 1.69446246
Iteration 656, loss = 1.69385224
Iteration 657, loss = 1.69400696
Iteration 658, loss = 1.69321344
Iteration 659, loss = 1.69234172
Iteration 660, loss = 1.69198777
```

```
Iteration 661, loss = 1.69193406
Iteration 662, loss = 1.69134702
Iteration 663, loss = 1.69072282
Iteration 664, loss = 1.69059480
Iteration 665, loss = 1.68987160
Iteration 666, loss = 1.68916040
Iteration 667, loss = 1.68983492
Iteration 668, loss = 1.68872587
Iteration 669, loss = 1.68826418
Iteration 670, loss = 1.68830439
Iteration 671, loss = 1.68787667
Iteration 672, loss = 1.68753418
Iteration 673, loss = 1.68662715
Iteration 674, loss = 1.68599448
Iteration 675, loss = 1.68603537
Iteration 676, loss = 1.68496592
Iteration 677, loss = 1.68549973
Iteration 678, loss = 1.68564729
Iteration 679, loss = 1.68411526
Iteration 680, loss = 1.68422414
Iteration 681, loss = 1.68356529
Iteration 682, loss = 1.68284008
Iteration 683, loss = 1.68279019
Iteration 684, loss = 1.68251047
Iteration 685, loss = 1.68185710
Iteration 686, loss = 1.68136743
Iteration 687, loss = 1.68111749
Iteration 688, loss = 1.68017817
Iteration 689, loss = 1.68020279
Iteration 690, loss = 1.67964007
Iteration 691, loss = 1.67899389
Iteration 692, loss = 1.67953670
Iteration 693, loss = 1.67846896
Iteration 694, loss = 1.67793060
Iteration 695, loss = 1.67752021
Iteration 696, loss = 1.67736929
Iteration 697, loss = 1.67666202
Iteration 698, loss = 1.67728697
Iteration 699, loss = 1.67614272
Iteration 700, loss = 1.67535064
Iteration 701, loss = 1.67545373
Iteration 702, loss = 1.67459324
Iteration 703, loss = 1.67381075
Iteration 704, loss = 1.67441344
Iteration 705, loss = 1.67369368
Iteration 706, loss = 1.67274959
Iteration 707, loss = 1.67305853
Iteration 708, loss = 1.67228540
Iteration 709, loss = 1.67236529
Iteration 710, loss = 1.67199525
Iteration 711, loss = 1.67095544
Iteration 712, loss = 1.67126542
Iteration 713, loss = 1.67007671
Iteration 714, loss = 1.66977656
Iteration 715, loss = 1.67042412
Iteration 716, loss = 1.66879849
Iteration 717, loss = 1.66877494
Iteration 718, loss = 1.66854820
Iteration 719, loss = 1.66813890
Iteration 720, loss = 1.66823413
```

```
Iteration 721, loss = 1.66809909
Iteration 722, loss = 1.66748456
Iteration 723, loss = 1.66676511
Iteration 724, loss = 1.66711614
Iteration 725, loss = 1.66628036
Iteration 726, loss = 1.66528298
Iteration 727, loss = 1.66646142
Iteration 728, loss = 1.66579558
Iteration 729, loss = 1.66485982
Iteration 730, loss = 1.66381148
Iteration 731, loss = 1.66361224
Iteration 732, loss = 1.66302751
Iteration 733, loss = 1.66354008
Iteration 734, loss = 1.66236274
Iteration 735, loss = 1.66215260
Iteration 736, loss = 1.66199244
Iteration 737, loss = 1.66097688
Iteration 738, loss = 1.66049787
Iteration 739, loss = 1.65967357
Iteration 740, loss = 1.66061455
Iteration 741, loss = 1.65960880
Iteration 742, loss = 1.65922716
Iteration 743, loss = 1.65986015
Iteration 744, loss = 1.65939466
Iteration 745, loss = 1.65830867
Iteration 746, loss = 1.65834291
Iteration 747, loss = 1.65743804
Iteration 748, loss = 1.65833527
Iteration 749, loss = 1.65763094
Iteration 750, loss = 1.65657895
Iteration 751, loss = 1.65621051
Iteration 752, loss = 1.65595532
Iteration 753, loss = 1.65590202
Iteration 754, loss = 1.65567111
Iteration 755, loss = 1.65491397
Iteration 756, loss = 1.65409565
Iteration 757, loss = 1.65390629
Iteration 758, loss = 1.65456580
Iteration 759, loss = 1.65318935
Iteration 760, loss = 1.65334786
Iteration 761, loss = 1.65229561
Iteration 762, loss = 1.65231442
Iteration 763, loss = 1.65185874
Iteration 764, loss = 1.65161515
Iteration 765, loss = 1.65100764
Iteration 766, loss = 1.65039978
Iteration 767, loss = 1.65060702
Iteration 768, loss = 1.65040677
Iteration 769, loss = 1.64961581
Iteration 770, loss = 1.64958521
Iteration 771, loss = 1.64926379
Iteration 772, loss = 1.64918687
Iteration 773, loss = 1.64801865
Iteration 774, loss = 1.64773555
Iteration 775, loss = 1.64776404
Iteration 776, loss = 1.64802098
Iteration 777, loss = 1.64643750
Iteration 778, loss = 1.64735287
Iteration 779, loss = 1.64614217
Iteration 780, loss = 1.64554578
```

```
Iteration 781, loss = 1.64585740
Iteration 782, loss = 1.64571888
Iteration 783, loss = 1.64468550
Iteration 784, loss = 1.64483127
Iteration 785, loss = 1.64440949
Iteration 786, loss = 1.64382286
Iteration 787, loss = 1.64297501
Iteration 788, loss = 1.64343102
Iteration 789, loss = 1.64296163
Iteration 790, loss = 1.64332914
Iteration 791, loss = 1.64313600
Iteration 792, loss = 1.64185210
Iteration 793, loss = 1.64105227
Iteration 794, loss = 1.64125287
Iteration 795, loss = 1.64107703
Iteration 796, loss = 1.64029376
Iteration 797, loss = 1.63950646
Iteration 798, loss = 1.63970151
Iteration 799, loss = 1.63988022
Iteration 800, loss = 1.63876037
Iteration 801, loss = 1.63838989
Iteration 802, loss = 1.63860635
Iteration 803, loss = 1.63851597
Iteration 804, loss = 1.63781864
Iteration 805, loss = 1.63823971
Iteration 806, loss = 1.63730015
Iteration 807, loss = 1.63717250
Iteration 808, loss = 1.63726776
Iteration 809, loss = 1.63660194
Iteration 810, loss = 1.63616437
Iteration 811, loss = 1.63605761
Iteration 812, loss = 1.63708108
Iteration 813, loss = 1.63479043
Iteration 814, loss = 1.63560825
Iteration 815, loss = 1.63476986
Iteration 816, loss = 1.63378576
Iteration 817, loss = 1.63352063
Iteration 818, loss = 1.63349299
Iteration 819, loss = 1.63389098
Iteration 820, loss = 1.63255573
Iteration 821, loss = 1.63233451
Iteration 822, loss = 1.63248401
Iteration 823, loss = 1.63177865
Iteration 824, loss = 1.63161437
Iteration 825, loss = 1.63123545
Iteration 826, loss = 1.63077272
Iteration 827, loss = 1.63088784
Iteration 828, loss = 1.62957524
Iteration 829, loss = 1.63002663
Iteration 830, loss = 1.62953721
Iteration 831, loss = 1.62928802
Iteration 832, loss = 1.62916474
Iteration 833, loss = 1.62902179
Iteration 834, loss = 1.62964786
Iteration 835, loss = 1.62920705
Iteration 836, loss = 1.62805400
Iteration 837, loss = 1.62726014
Iteration 838, loss = 1.62730884
Iteration 839, loss = 1.62636280
Iteration 840, loss = 1.62744513
```

```
Iteration 841, loss = 1.62588250
Iteration 842, loss = 1.62586498
Iteration 843, loss = 1.62590182
Iteration 844, loss = 1.62568508
Iteration 845, loss = 1.62542645
Iteration 846, loss = 1.62435862
Iteration 847, loss = 1.62467373
Iteration 848, loss = 1.62460060
Iteration 849, loss = 1.62337107
Iteration 850, loss = 1.62349469
Iteration 851, loss = 1.62367258
Iteration 852, loss = 1.62273715
Iteration 853, loss = 1.62241128
Iteration 854, loss = 1.62262581
Iteration 855, loss = 1.62212516
Iteration 856, loss = 1.62151751
Iteration 857, loss = 1.62145280
Iteration 858, loss = 1.62078335
Iteration 859, loss = 1.62114918
Iteration 860, loss = 1.62096398
Iteration 861, loss = 1.62051332
Iteration 862, loss = 1.62040775
Iteration 863, loss = 1.61990101
Iteration 864, loss = 1.61922681
Iteration 865, loss = 1.61877031
Iteration 866, loss = 1.61831926
Iteration 867, loss = 1.61866983
Iteration 868, loss = 1.61872262
Iteration 869, loss = 1.61773818
Iteration 870, loss = 1.61816058
Iteration 871, loss = 1.61757986
Iteration 872, loss = 1.61751791
Iteration 873, loss = 1.61678948
Iteration 874, loss = 1.61695715
Iteration 875, loss = 1.61607011
Iteration 876, loss = 1.61610448
Iteration 877, loss = 1.61615412
Iteration 878, loss = 1.61546719
Iteration 879, loss = 1.61551149
Iteration 880, loss = 1.61560545
Iteration 881, loss = 1.61494485
Iteration 882, loss = 1.61474390
Iteration 883, loss = 1.61345042
Iteration 884, loss = 1.61373407
Iteration 885, loss = 1.61443805
Iteration 886, loss = 1.61338179
Iteration 887, loss = 1.61227156
Iteration 888, loss = 1.61364007
Iteration 889, loss = 1.61133415
Iteration 890, loss = 1.61202886
Iteration 891, loss = 1.61173042
Iteration 892, loss = 1.61145506
Iteration 893, loss = 1.61160305
Iteration 894, loss = 1.61101816
Iteration 895, loss = 1.61122608
Iteration 896, loss = 1.60988474
Iteration 897, loss = 1.61061754
Iteration 898, loss = 1.60981138
Iteration 899, loss = 1.60998719
Iteration 900, loss = 1.60961672
```

```
Iteration 901, loss = 1.60933218
Iteration 902, loss = 1.60898183
Iteration 903, loss = 1.60872995
Iteration 904, loss = 1.60878647
Iteration 905, loss = 1.60772211
Iteration 906, loss = 1.60838993
Iteration 907, loss = 1.60691468
Iteration 908, loss = 1.60737955
Iteration 909, loss = 1.60720660
Iteration 910, loss = 1.60705271
Iteration 911, loss = 1.60712170
Iteration 912, loss = 1.60619758
Iteration 913, loss = 1.60709713
Iteration 914, loss = 1.60562540
Iteration 915, loss = 1.60509232
Iteration 916, loss = 1.60562991
Iteration 917, loss = 1.60435159
Iteration 918, loss = 1.60449802
Iteration 919, loss = 1.60466770
Iteration 920, loss = 1.60512856
Iteration 921, loss = 1.60470934
Iteration 922, loss = 1.60371960
Iteration 923, loss = 1.60338821
Iteration 924, loss = 1.60326832
Iteration 925, loss = 1.60267689
Iteration 926, loss = 1.60221001
Iteration 927, loss = 1.60343560
Iteration 928, loss = 1.60181485
Iteration 929, loss = 1.60214542
Iteration 930, loss = 1.60137868
Iteration 931, loss = 1.60143954
Iteration 932, loss = 1.60178477
Iteration 933, loss = 1.60107048
Iteration 934, loss = 1.60025121
Iteration 935, loss = 1.59993166
Iteration 936, loss = 1.60025253
Iteration 937, loss = 1.59979725
Iteration 938, loss = 1.59986738
Iteration 939, loss = 1.59915783
Iteration 940, loss = 1.59935820
Iteration 941, loss = 1.59939778
Iteration 942, loss = 1.59880768
Iteration 943, loss = 1.59835703
Iteration 944, loss = 1.59800107
Iteration 945, loss = 1.59759129
Iteration 946, loss = 1.59777306
Iteration 947, loss = 1.59678022
Iteration 948, loss = 1.59676843
Iteration 949, loss = 1.59681331
Iteration 950, loss = 1.59609376
Iteration 951, loss = 1.59744360
Iteration 952, loss = 1.59567435
Iteration 953, loss = 1.59651804
Iteration 954, loss = 1.59595893
Iteration 955, loss = 1.59542727
Iteration 956, loss = 1.59437171
Iteration 957, loss = 1.59498646
Iteration 958, loss = 1.59473989
Iteration 959, loss = 1.59475736
Iteration 960, loss = 1.59472969
```

```
Iteration 961, loss = 1.59440379
Iteration 962, loss = 1.59446426
Iteration 963, loss = 1.59355117
Iteration 964, loss = 1.59369078
Iteration 965, loss = 1.59276993
Iteration 966, loss = 1.59263559
Iteration 967, loss = 1.59216408
Iteration 968, loss = 1.59270128
Iteration 969, loss = 1.59184859
Iteration 970, loss = 1.59284091
Iteration 971, loss = 1.59202562
Iteration 972, loss = 1.59143014
Iteration 973, loss = 1.59074880
Iteration 974, loss = 1.59143682
Iteration 975, loss = 1.59038509
Iteration 976, loss = 1.58992210
Iteration 977, loss = 1.59029532
Iteration 978, loss = 1.58954998
Iteration 979, loss = 1.58961451
Iteration 980, loss = 1.59032951
Iteration 981, loss = 1.58941262
Iteration 982, loss = 1.58864729
Iteration 983, loss = 1.58891477
Iteration 984, loss = 1.58821953
Iteration 985, loss = 1.58815038
Iteration 986, loss = 1.58815141
Iteration 987, loss = 1.58772336
Iteration 988, loss = 1.58740098
Iteration 989, loss = 1.58660535
Iteration 990, loss = 1.58703079
Iteration 991, loss = 1.58699669
Iteration 992, loss = 1.58690385
Iteration 993, loss = 1.58635708
Iteration 994, loss = 1.58609838
Iteration 995, loss = 1.58522916
Iteration 996, loss = 1.58480055
Iteration 997, loss = 1.58520214
Iteration 998, loss = 1.58573541
Iteration 999, loss = 1.58522532
Iteration 1000, loss = 1.58515004
Iteration 1001, loss = 1.58411028
Iteration 1002, loss = 1.58433931
Iteration 1003, loss = 1.58426868
Iteration 1004, loss = 1.58415885
Iteration 1005, loss = 1.58323382
Iteration 1006, loss = 1.58353067
Iteration 1007, loss = 1.58357913
Iteration 1008, loss = 1.58350726
Iteration 1009, loss = 1.58192433
Iteration 1010, loss = 1.58241370
Iteration 1011, loss = 1.58226314
Iteration 1012, loss = 1.58107035
Iteration 1013, loss = 1.58232741
Iteration 1014, loss = 1.58152926
Iteration 1015, loss = 1.58114356
Iteration 1016, loss = 1.58096545
Iteration 1017, loss = 1.58011812
Iteration 1018, loss = 1.58098315
Iteration 1019, loss = 1.58073625
Iteration 1020, loss = 1.58013636
```

```
Iteration 1021, loss = 1.58012379
Iteration 1022, loss = 1.57917205
Iteration 1023, loss = 1.57961199
Iteration 1024, loss = 1.57913274
Iteration 1025, loss = 1.57882380
Iteration 1026, loss = 1.58011776
Iteration 1027, loss = 1.57898406
Iteration 1028, loss = 1.57959415
Iteration 1029, loss = 1.57876510
Iteration 1030, loss = 1.57791365
Iteration 1031, loss = 1.57752167
Iteration 1032, loss = 1.57715608
Iteration 1033, loss = 1.57717700
Iteration 1034, loss = 1.57711076
Iteration 1035, loss = 1.57633405
Iteration 1036, loss = 1.57655071
Iteration 1037, loss = 1.57644664
Iteration 1038, loss = 1.57743344
Iteration 1039, loss = 1.57518681
Iteration 1040, loss = 1.57580967
Iteration 1041, loss = 1.57541778
Iteration 1042, loss = 1.57550334
Iteration 1043, loss = 1.57594125
Iteration 1044, loss = 1.57504394
Iteration 1045, loss = 1.57435262
Iteration 1046, loss = 1.57439884
Iteration 1047, loss = 1.57408800
Iteration 1048, loss = 1.57450866
Iteration 1049, loss = 1.57336168
Iteration 1050, loss = 1.57384760
Iteration 1051, loss = 1.57333629
Iteration 1052, loss = 1.57354441
Iteration 1053, loss = 1.57308460
Iteration 1054, loss = 1.57229320
Iteration 1055, loss = 1.57209425
Iteration 1056, loss = 1.57270381
Iteration 1057, loss = 1.57303766
Iteration 1058, loss = 1.57209802
Iteration 1059, loss = 1.57197732
Iteration 1060, loss = 1.57137295
Iteration 1061, loss = 1.57147281
Iteration 1062, loss = 1.57129435
Iteration 1063, loss = 1.57061913
Iteration 1064, loss = 1.57076821
Iteration 1065, loss = 1.57037915
Iteration 1066, loss = 1.57074113
Iteration 1067, loss = 1.57004430
Iteration 1068, loss = 1.56986243
Iteration 1069, loss = 1.56984850
Iteration 1070, loss = 1.56939856
Iteration 1071, loss = 1.56961293
Iteration 1072, loss = 1.56881896
Iteration 1073, loss = 1.56857021
Iteration 1074, loss = 1.56858400
Iteration 1075, loss = 1.56890616
Iteration 1076, loss = 1.56831227
Iteration 1077, loss = 1.56790885
Iteration 1078, loss = 1.56858482
Iteration 1079, loss = 1.56760981
Iteration 1080, loss = 1.56787810
```

```
Iteration 1081, loss = 1.56725026
Iteration 1082, loss = 1.56689574
Iteration 1083, loss = 1.56613011
Iteration 1084, loss = 1.56635805
Iteration 1085, loss = 1.56644675
Iteration 1086, loss = 1.56658885
Iteration 1087, loss = 1.56591544
Iteration 1088, loss = 1.56555395
Iteration 1089, loss = 1.56473276
Iteration 1090, loss = 1.56536556
Iteration 1091, loss = 1.56496793
Iteration 1092, loss = 1.56473698
Iteration 1093, loss = 1.56436917
Iteration 1094, loss = 1.56511406
Iteration 1095, loss = 1.56448433
Iteration 1096, loss = 1.56439857
Iteration 1097, loss = 1.56413036
Iteration 1098, loss = 1.56327224
Iteration 1099, loss = 1.56384073
Iteration 1100, loss = 1.56341647
Iteration 1101, loss = 1.56292962
Iteration 1102, loss = 1.56313016
Iteration 1103, loss = 1.56255514
Iteration 1104, loss = 1.56277261
Iteration 1105, loss = 1.56300916
Iteration 1106, loss = 1.56242327
Iteration 1107, loss = 1.56272099
Iteration 1108, loss = 1.56184839
Iteration 1109, loss = 1.56176494
Iteration 1110, loss = 1.56110896
Iteration 1111, loss = 1.56127576
Iteration 1112, loss = 1.56132550
Iteration 1113, loss = 1.56059443
Iteration 1114, loss = 1.56057952
Iteration 1115, loss = 1.56088111
Iteration 1116, loss = 1.55970202
Iteration 1117, loss = 1.55991200
Iteration 1118, loss = 1.56103673
Iteration 1119, loss = 1.55943302
Iteration 1120, loss = 1.55918158
Iteration 1121, loss = 1.55991950
Iteration 1122, loss = 1.55918548
Iteration 1123, loss = 1.55938923
Iteration 1124, loss = 1.55809549
Iteration 1125, loss = 1.55856254
Iteration 1126, loss = 1.55850352
Iteration 1127, loss = 1.55865384
Iteration 1128, loss = 1.55808849
Iteration 1129, loss = 1.55827933
Iteration 1130, loss = 1.55726836
Iteration 1131, loss = 1.55791331
Iteration 1132, loss = 1.55703531
Iteration 1133, loss = 1.55620148
Iteration 1134, loss = 1.55674648
Iteration 1135, loss = 1.55795079
Iteration 1136, loss = 1.55724655
Iteration 1137, loss = 1.55567230
Iteration 1138, loss = 1.55650946
Iteration 1139, loss = 1.55634824
Iteration 1140, loss = 1.55653790
```

```
Iteration 1141, loss = 1.55556458
Iteration 1142, loss = 1.55615395
Iteration 1143, loss = 1.55516138
Iteration 1144, loss = 1.55550378
Iteration 1145, loss = 1.55622681
Iteration 1146, loss = 1.55480439
Iteration 1147, loss = 1.55475195
Iteration 1148, loss = 1.55418945
Iteration 1149, loss = 1.55395671
Iteration 1150, loss = 1.55411025
Iteration 1151, loss = 1.55518721
Iteration 1152, loss = 1.55364540
Iteration 1153, loss = 1.55321314
Iteration 1154, loss = 1.55403843
Iteration 1155, loss = 1.55343786
Iteration 1156, loss = 1.55312092
Iteration 1157, loss = 1.55223955
Iteration 1158, loss = 1.55163290
Iteration 1159, loss = 1.55261585
Iteration 1160, loss = 1.55196251
Iteration 1161, loss = 1.55198075
Iteration 1162, loss = 1.55218963
Iteration 1163, loss = 1.55134691
Iteration 1164, loss = 1.55122566
Iteration 1165, loss = 1.55154071
Iteration 1166, loss = 1.55022634
Iteration 1167, loss = 1.55035822
Iteration 1168, loss = 1.55138517
Iteration 1169, loss = 1.55082411
Iteration 1170, loss = 1.55051091
Iteration 1171, loss = 1.54995502
Iteration 1172, loss = 1.55112934
Iteration 1173, loss = 1.55014235
Iteration 1174, loss = 1.55057516
Iteration 1175, loss = 1.54998847
Iteration 1176, loss = 1.55030511
Iteration 1177, loss = 1.54907239
Iteration 1178, loss = 1.54917819
Iteration 1179, loss = 1.54901988
Iteration 1180, loss = 1.54887725
Iteration 1181, loss = 1.54869200
Iteration 1182, loss = 1.54819213
Iteration 1183, loss = 1.54757508
Iteration 1184, loss = 1.54810993
Iteration 1185, loss = 1.54810142
Iteration 1186, loss = 1.54791677
Iteration 1187, loss = 1.54764779
Iteration 1188, loss = 1.54735879
Iteration 1189, loss = 1.54734744
Iteration 1190, loss = 1.54756051
Iteration 1191, loss = 1.54690755
Iteration 1192, loss = 1.54678560
Iteration 1193, loss = 1.54642498
Iteration 1194, loss = 1.54733194
Iteration 1195, loss = 1.54776250
Iteration 1196, loss = 1.54652408
Iteration 1197, loss = 1.54513142
Iteration 1198, loss = 1.54638988
Iteration 1199, loss = 1.54621067
Iteration 1200, loss = 1.54520124
```

```
Iteration 1201, loss = 1.54522586
Iteration 1202, loss = 1.54397823
Iteration 1203, loss = 1.54484416
Iteration 1204, loss = 1.54536430
Iteration 1205, loss = 1.54446043
Iteration 1206, loss = 1.54469291
Iteration 1207, loss = 1.54364423
Iteration 1208, loss = 1.54407973
Iteration 1209, loss = 1.54417830
Iteration 1210, loss = 1.54385600
Iteration 1211, loss = 1.54349040
Iteration 1212, loss = 1.54281912
Iteration 1213, loss = 1.54336410
Iteration 1214, loss = 1.54233210
Iteration 1215, loss = 1.54217172
Iteration 1216, loss = 1.54215840
Iteration 1217, loss = 1.54224724
Iteration 1218, loss = 1.54285703
Iteration 1219, loss = 1.54149466
Iteration 1220, loss = 1.54225524
Iteration 1221, loss = 1.54198591
Iteration 1222, loss = 1.54175998
Iteration 1223, loss = 1.54124223
Iteration 1224, loss = 1.54110055
Iteration 1225, loss = 1.54069630
Iteration 1226, loss = 1.54023506
Iteration 1227, loss = 1.54109960
Iteration 1228, loss = 1.54007677
Iteration 1229, loss = 1.54038699
Iteration 1230, loss = 1.54001942
Iteration 1231, loss = 1.54044305
Iteration 1232, loss = 1.53886499
Iteration 1233, loss = 1.53989608
Iteration 1234, loss = 1.53931955
Iteration 1235, loss = 1.53976934
Iteration 1236, loss = 1.53885581
Iteration 1237, loss = 1.53892659
Iteration 1238, loss = 1.53856676
Iteration 1239, loss = 1.53968346
Iteration 1240, loss = 1.53893451
Iteration 1241, loss = 1.53926511
Iteration 1242, loss = 1.53834669
Iteration 1243, loss = 1.53830851
Iteration 1244, loss = 1.53767181
Iteration 1245, loss = 1.53888011
Iteration 1246, loss = 1.53881226
Iteration 1247, loss = 1.53695361
Iteration 1248, loss = 1.53754405
Iteration 1249, loss = 1.53657168
Iteration 1250, loss = 1.53676246
Iteration 1251, loss = 1.53658170
Iteration 1252, loss = 1.53706759
Iteration 1253, loss = 1.53603028
Iteration 1254, loss = 1.53625281
Iteration 1255, loss = 1.53578434
Iteration 1256, loss = 1.53633097
Iteration 1257, loss = 1.53719750
Iteration 1258, loss = 1.53615770
Iteration 1259, loss = 1.53475312
Iteration 1260, loss = 1.53524980
```

```
Iteration 1261, loss = 1.53487782
Iteration 1262, loss = 1.53529207
Iteration 1263, loss = 1.53536765
Iteration 1264, loss = 1.53484319
Iteration 1265, loss = 1.53377356
Iteration 1266, loss = 1.53474737
Iteration 1267, loss = 1.53434536
Iteration 1268, loss = 1.53397054
Iteration 1269, loss = 1.53400331
Iteration 1270, loss = 1.53312197
Iteration 1271, loss = 1.53475065
Iteration 1272, loss = 1.53248727
Iteration 1273, loss = 1.53295860
Iteration 1274, loss = 1.53291789
Iteration 1275, loss = 1.53274597
Iteration 1276, loss = 1.53238251
Iteration 1277, loss = 1.53267371
Iteration 1278, loss = 1.53190715
Iteration 1279, loss = 1.53219469
Iteration 1280, loss = 1.53108034
Iteration 1281, loss = 1.53247985
Iteration 1282, loss = 1.53130373
Iteration 1283, loss = 1.53206617
Iteration 1284, loss = 1.53119668
Iteration 1285, loss = 1.53119977
Iteration 1286, loss = 1.53152611
Iteration 1287, loss = 1.53110870
Iteration 1288, loss = 1.53066279
Iteration 1289, loss = 1.53076116
Iteration 1290, loss = 1.53037807
Iteration 1291, loss = 1.53007476
Iteration 1292, loss = 1.53008894
Iteration 1293, loss = 1.53087322
Iteration 1294, loss = 1.53010685
Iteration 1295, loss = 1.52969449
Iteration 1296, loss = 1.53044422
Iteration 1297, loss = 1.52927619
Iteration 1298, loss = 1.52873015
Iteration 1299, loss = 1.52913979
Iteration 1300, loss = 1.52972545
Iteration 1301, loss = 1.52847532
Iteration 1302, loss = 1.52792107
Iteration 1303, loss = 1.52810439
Iteration 1304, loss = 1.52800652
Iteration 1305, loss = 1.52809519
Iteration 1306, loss = 1.52761375
Iteration 1307, loss = 1.52885772
Iteration 1308, loss = 1.52752249
Iteration 1309, loss = 1.52722951
Iteration 1310, loss = 1.52804961
Iteration 1311, loss = 1.52712646
Iteration 1312, loss = 1.52745773
Iteration 1313, loss = 1.52743757
Iteration 1314, loss = 1.52697655
Iteration 1315, loss = 1.52602912
Iteration 1316, loss = 1.52628688
Iteration 1317, loss = 1.52565942
Iteration 1318, loss = 1.52617512
Iteration 1319, loss = 1.52581471
Iteration 1320, loss = 1.52491023
```

```
Iteration 1321, loss = 1.52583386
Iteration 1322, loss = 1.52521383
Iteration 1323, loss = 1.52504751
Iteration 1324, loss = 1.52472386
Iteration 1325, loss = 1.52486536
Iteration 1326, loss = 1.52510088
Iteration 1327, loss = 1.52460174
Iteration 1328, loss = 1.52429881
Iteration 1329, loss = 1.52444719
Iteration 1330, loss = 1.52296904
Iteration 1331, loss = 1.52437362
Iteration 1332, loss = 1.52329168
Iteration 1333, loss = 1.52413792
Iteration 1334, loss = 1.52242189
Iteration 1335, loss = 1.52294762
Iteration 1336, loss = 1.52397413
Iteration 1337, loss = 1.52210363
Iteration 1338, loss = 1.52275634
Iteration 1339, loss = 1.52297737
Iteration 1340, loss = 1.52241714
Iteration 1341, loss = 1.52245183
Iteration 1342, loss = 1.52268550
Iteration 1343, loss = 1.52289247
Iteration 1344, loss = 1.52219375
Iteration 1345, loss = 1.52093737
Iteration 1346, loss = 1.52139244
Iteration 1347, loss = 1.52098698
Iteration 1348, loss = 1.52129059
Iteration 1349, loss = 1.52122803
Iteration 1350, loss = 1.52156735
Iteration 1351, loss = 1.52130411
Iteration 1352, loss = 1.52057100
Iteration 1353, loss = 1.52064555
Iteration 1354, loss = 1.52066163
Iteration 1355, loss = 1.52072795
Iteration 1356, loss = 1.52009789
Iteration 1357, loss = 1.52035451
Iteration 1358, loss = 1.51925304
Iteration 1359, loss = 1.51977312
Iteration 1360, loss = 1.51971893
Iteration 1361, loss = 1.51934954
Iteration 1362, loss = 1.51946078
Iteration 1363, loss = 1.51866985
Iteration 1364, loss = 1.51854771
Iteration 1365, loss = 1.51995749
Iteration 1366, loss = 1.51817756
Iteration 1367, loss = 1.51864048
Iteration 1368, loss = 1.51789909
Iteration 1369, loss = 1.51841029
Iteration 1370, loss = 1.51763599
Iteration 1371, loss = 1.51797622
Iteration 1372, loss = 1.51827770
Iteration 1373, loss = 1.51701606
Iteration 1374, loss = 1.51712940
Iteration 1375, loss = 1.51741974
Iteration 1376, loss = 1.51603708
Iteration 1377, loss = 1.51798610
Iteration 1378, loss = 1.51669636
Iteration 1379, loss = 1.51748208
Iteration 1380, loss = 1.51627259
```

```
Iteration 1381, loss = 1.51664660
Iteration 1382, loss = 1.51691954
Iteration 1383, loss = 1.51664322
Iteration 1384, loss = 1.51556583
Iteration 1385, loss = 1.51644386
Iteration 1386, loss = 1.51542201
Iteration 1387, loss = 1.51590341
Iteration 1388, loss = 1.51663316
Iteration 1389, loss = 1.51525865
Iteration 1390, loss = 1.51532005
Iteration 1391, loss = 1.51414019
Iteration 1392, loss = 1.51517460
Iteration 1393, loss = 1.51536691
Iteration 1394, loss = 1.51471397
Iteration 1395, loss = 1.51545355
Iteration 1396, loss = 1.51475636
Iteration 1397, loss = 1.51410504
Iteration 1398, loss = 1.51352336
Iteration 1399, loss = 1.51373183
Iteration 1400, loss = 1.51318203
Iteration 1401, loss = 1.51378346
Iteration 1402, loss = 1.51375178
Iteration 1403, loss = 1.51299151
Iteration 1404, loss = 1.51423622
Iteration 1405, loss = 1.51342954
Iteration 1406, loss = 1.51312944
Iteration 1407, loss = 1.51325553
Iteration 1408, loss = 1.51282219
Iteration 1409, loss = 1.51302623
Iteration 1410, loss = 1.51187060
Iteration 1411, loss = 1.51203027
Iteration 1412, loss = 1.51249416
Iteration 1413, loss = 1.51149219
Iteration 1414, loss = 1.51146161
Iteration 1415, loss = 1.51262602
Iteration 1416, loss = 1.51122137
Iteration 1417, loss = 1.51221336
Iteration 1418, loss = 1.51143633
Iteration 1419, loss = 1.51210599
Iteration 1420, loss = 1.51127725
Iteration 1421, loss = 1.51092216
Iteration 1422, loss = 1.51126820
Iteration 1423, loss = 1.51021715
Iteration 1424, loss = 1.51108676
Iteration 1425, loss = 1.50974296
Iteration 1426, loss = 1.51047611
Iteration 1427, loss = 1.50970867
Iteration 1428, loss = 1.50999305
Iteration 1429, loss = 1.50949266
Iteration 1430, loss = 1.50975249
Iteration 1431, loss = 1.51018059
Iteration 1432, loss = 1.50992636
Iteration 1433, loss = 1.50889090
Iteration 1434, loss = 1.50975409
Iteration 1435, loss = 1.50892250
Iteration 1436, loss = 1.50803770
Iteration 1437, loss = 1.50910921
Iteration 1438, loss = 1.50862810
Iteration 1439, loss = 1.50740090
Iteration 1440, loss = 1.50912716
```

```
Iteration 1441, loss = 1.50785578
Iteration 1442, loss = 1.50794348
Iteration 1443, loss = 1.50846846
Iteration 1444, loss = 1.50817461
Iteration 1445, loss = 1.50802803
Iteration 1446, loss = 1.50739349
Iteration 1447, loss = 1.50730481
Iteration 1448, loss = 1.50642206
Iteration 1449, loss = 1.50722973
Iteration 1450, loss = 1.50753931
Iteration 1451, loss = 1.50639671
Iteration 1452, loss = 1.50652895
Iteration 1453, loss = 1.50682646
Iteration 1454, loss = 1.50604826
Iteration 1455, loss = 1.50676508
Iteration 1456, loss = 1.50621696
Iteration 1457, loss = 1.50631818
Iteration 1458, loss = 1.50614876
Iteration 1459, loss = 1.50525695
Iteration 1460, loss = 1.50565666
Iteration 1461, loss = 1.50562212
Iteration 1462, loss = 1.50505726
Iteration 1463, loss = 1.50503275
Iteration 1464, loss = 1.50622470
Iteration 1465, loss = 1.50507548
Iteration 1466, loss = 1.50481210
Iteration 1467, loss = 1.50531256
Iteration 1468, loss = 1.50543379
Iteration 1469, loss = 1.50459784
Iteration 1470, loss = 1.50436889
Iteration 1471, loss = 1.50461910
Iteration 1472, loss = 1.50404988
Iteration 1473, loss = 1.50339926
Iteration 1474, loss = 1.50460368
Iteration 1475, loss = 1.50379152
Iteration 1476, loss = 1.50409573
Iteration 1477, loss = 1.50303552
Iteration 1478, loss = 1.50374329
Iteration 1479, loss = 1.50414992
Iteration 1480, loss = 1.50334857
Iteration 1481, loss = 1.50210215
Iteration 1482, loss = 1.50231665
Iteration 1483, loss = 1.50298137
Iteration 1484, loss = 1.50326055
Iteration 1485, loss = 1.50212307
Iteration 1486, loss = 1.50215292
Iteration 1487, loss = 1.50199292
Iteration 1488, loss = 1.50297540
Iteration 1489, loss = 1.50216563
Iteration 1490, loss = 1.50147528
Iteration 1491, loss = 1.50189806
Iteration 1492, loss = 1.50113871
Iteration 1493, loss = 1.50206524
Iteration 1494, loss = 1.50132398
Iteration 1495, loss = 1.50208018
Iteration 1496, loss = 1.50216128
Iteration 1497, loss = 1.50120236
Iteration 1498, loss = 1.50010069
Iteration 1499, loss = 1.50024681
Iteration 1500, loss = 1.50143347
```

```
Iteration 1501, loss = 1.49977991
Iteration 1502, loss = 1.49971640
Iteration 1503, loss = 1.49997643
Iteration 1504, loss = 1.49998246
Iteration 1505, loss = 1.49997730
Iteration 1506, loss = 1.49942753
Iteration 1507, loss = 1.49967941
Iteration 1508, loss = 1.49944226
Iteration 1509, loss = 1.50009268
Iteration 1510, loss = 1.49903590
Iteration 1511, loss = 1.50004498
Iteration 1512, loss = 1.49958187
Iteration 1513, loss = 1.49974310
Iteration 1514, loss = 1.49881423
Iteration 1515, loss = 1.49809416
Iteration 1516, loss = 1.49804939
Iteration 1517, loss = 1.49781069
Iteration 1518, loss = 1.49822919
Iteration 1519, loss = 1.49847103
Iteration 1520, loss = 1.49759252
Iteration 1521, loss = 1.49798944
Iteration 1522, loss = 1.49786592
Iteration 1523, loss = 1.49804912
Iteration 1524, loss = 1.49706644
Iteration 1525, loss = 1.49712699
Iteration 1526, loss = 1.49736952
Iteration 1527, loss = 1.49744057
Iteration 1528, loss = 1.49724384
Iteration 1529, loss = 1.49660289
Iteration 1530, loss = 1.49640431
Iteration 1531, loss = 1.49721140
Iteration 1532, loss = 1.49778515
Iteration 1533, loss = 1.49681706
Iteration 1534, loss = 1.49621358
Iteration 1535, loss = 1.49632698
Iteration 1536, loss = 1.49652485
Iteration 1537, loss = 1.49644488
Iteration 1538, loss = 1.49637747
Iteration 1539, loss = 1.49700410
Iteration 1540, loss = 1.49646502
Iteration 1541, loss = 1.49610654
Iteration 1542, loss = 1.49495424
Iteration 1543, loss = 1.49573489
Iteration 1544, loss = 1.49546291
Iteration 1545, loss = 1.49589976
Iteration 1546, loss = 1.49505155
Iteration 1547, loss = 1.49485931
Iteration 1548, loss = 1.49524928
Iteration 1549, loss = 1.49452246
Iteration 1550, loss = 1.49426613
Iteration 1551, loss = 1.49426040
Iteration 1552, loss = 1.49417937
Iteration 1553, loss = 1.49389718
Iteration 1554, loss = 1.49431159
Iteration 1555, loss = 1.49391526
Iteration 1556, loss = 1.49405567
Iteration 1557, loss = 1.49342205
Iteration 1558, loss = 1.49371681
Iteration 1559, loss = 1.49352192
Iteration 1560, loss = 1.49278959
```

```
Iteration 1561, loss = 1.49412876
Iteration 1562, loss = 1.49277353
Iteration 1563, loss = 1.49242661
Iteration 1564, loss = 1.49245538
Iteration 1565, loss = 1.49445400
Iteration 1566, loss = 1.49251156
Iteration 1567, loss = 1.49250502
Iteration 1568, loss = 1.49290897
Iteration 1569, loss = 1.49256520
Iteration 1570, loss = 1.49293389
Iteration 1571, loss = 1.49286479
Iteration 1572, loss = 1.49174747
Iteration 1573, loss = 1.49218623
Iteration 1574, loss = 1.49138770
Iteration 1575, loss = 1.49213663
Iteration 1576, loss = 1.49192375
Iteration 1577, loss = 1.49208869
Iteration 1578, loss = 1.49157715
Iteration 1579, loss = 1.49082752
Iteration 1580, loss = 1.49218631
Iteration 1581, loss = 1.49063067
Iteration 1582, loss = 1.49088168
Iteration 1583, loss = 1.49080670
Iteration 1584, loss = 1.49003449
Iteration 1585, loss = 1.49070309
Iteration 1586, loss = 1.48974915
Iteration 1587, loss = 1.49008482
Iteration 1588, loss = 1.48911905
Iteration 1589, loss = 1.48977941
Iteration 1590, loss = 1.49011245
Iteration 1591, loss = 1.49009273
Iteration 1592, loss = 1.49090276
Iteration 1593, loss = 1.48955214
Iteration 1594, loss = 1.49052270
Iteration 1595, loss = 1.48953540
Iteration 1596, loss = 1.48851902
Iteration 1597, loss = 1.48909203
Iteration 1598, loss = 1.48954527
Iteration 1599, loss = 1.48988724
Iteration 1600, loss = 1.48853999
Iteration 1601, loss = 1.48883737
Iteration 1602, loss = 1.48949296
Iteration 1603, loss = 1.48878953
Iteration 1604, loss = 1.48799396
Iteration 1605, loss = 1.48931458
Iteration 1606, loss = 1.48871816
Iteration 1607, loss = 1.48872793
Iteration 1608, loss = 1.48765301
Iteration 1609, loss = 1.48844295
Iteration 1610, loss = 1.48679939
Iteration 1611, loss = 1.48803798
Iteration 1612, loss = 1.48788117
Iteration 1613, loss = 1.48706678
Iteration 1614, loss = 1.48828429
Iteration 1615, loss = 1.48784856
Iteration 1616, loss = 1.48611297
Iteration 1617, loss = 1.48668904
Iteration 1618, loss = 1.48733858
Iteration 1619, loss = 1.48645004
Iteration 1620, loss = 1.48688453
```

```
Iteration 1621, loss = 1.48642346
Iteration 1622, loss = 1.48647360
Iteration 1623, loss = 1.48791234
Iteration 1624, loss = 1.48606480
Iteration 1625, loss = 1.48617111
Iteration 1626, loss = 1.48531351
Iteration 1627, loss = 1.48663988
Iteration 1628, loss = 1.48654471
Iteration 1629, loss = 1.48705600
Iteration 1630, loss = 1.48648174
Iteration 1631, loss = 1.48440459
Iteration 1632, loss = 1.48531257
Iteration 1633, loss = 1.48580905
Iteration 1634, loss = 1.48459219
Iteration 1635, loss = 1.48506523
Iteration 1636, loss = 1.48564025
Iteration 1637, loss = 1.48504683
Iteration 1638, loss = 1.48540187
Iteration 1639, loss = 1.48472149
Iteration 1640, loss = 1.48392484
Iteration 1641, loss = 1.48429695
Iteration 1642, loss = 1.48470763
Iteration 1643, loss = 1.48480809
Iteration 1644, loss = 1.48498724
Iteration 1645, loss = 1.48429986
Iteration 1646, loss = 1.48384919
Iteration 1647, loss = 1.48322549
Iteration 1648, loss = 1.48438215
Iteration 1649, loss = 1.48303486
Iteration 1650, loss = 1.48310122
Iteration 1651, loss = 1.48344682
Iteration 1652, loss = 1.48378082
Iteration 1653, loss = 1.48346710
Iteration 1654, loss = 1.48316683
Iteration 1655, loss = 1.48244119
Iteration 1656, loss = 1.48307235
Iteration 1657, loss = 1.48315645
Iteration 1658, loss = 1.48233805
Iteration 1659, loss = 1.48289015
Iteration 1660, loss = 1.48233200
Iteration 1661, loss = 1.48265305
Iteration 1662, loss = 1.48155254
Iteration 1663, loss = 1.48219664
Iteration 1664, loss = 1.48195486
Iteration 1665, loss = 1.48127064
Iteration 1666, loss = 1.48228826
Iteration 1667, loss = 1.48201233
Iteration 1668, loss = 1.48138464
Iteration 1669, loss = 1.48162804
Iteration 1670, loss = 1.48127469
Iteration 1671, loss = 1.48029051
Iteration 1672, loss = 1.48136916
Iteration 1673, loss = 1.48156381
Iteration 1674, loss = 1.48024659
Iteration 1675, loss = 1.48038652
Iteration 1676, loss = 1.48042158
Iteration 1677, loss = 1.48121301
Iteration 1678, loss = 1.48046382
Iteration 1679, loss = 1.48102102
Iteration 1680, loss = 1.47947564
```

```
Iteration 1681, loss = 1.48021947
Iteration 1682, loss = 1.47955068
Iteration 1683, loss = 1.48026213
Iteration 1684, loss = 1.47969516
Iteration 1685, loss = 1.47966138
Iteration 1686, loss = 1.48023127
Iteration 1687, loss = 1.47964818
Iteration 1688, loss = 1.47979667
Iteration 1689, loss = 1.47945129
Iteration 1690, loss = 1.47946650
Iteration 1691, loss = 1.47893032
Iteration 1692, loss = 1.47880810
Iteration 1693, loss = 1.47906196
Iteration 1694, loss = 1.47856228
Iteration 1695, loss = 1.47873384
Iteration 1696, loss = 1.47991924
Iteration 1697, loss = 1.47855488
Iteration 1698, loss = 1.47902769
Iteration 1699, loss = 1.47913398
Iteration 1700, loss = 1.47745661
Iteration 1701, loss = 1.47805517
Iteration 1702, loss = 1.47897330
Iteration 1703, loss = 1.47772525
Iteration 1704, loss = 1.47820918
Iteration 1705, loss = 1.47773852
Iteration 1706, loss = 1.47736901
Iteration 1707, loss = 1.47856316
Iteration 1708, loss = 1.47764102
Iteration 1709, loss = 1.47815187
Iteration 1710, loss = 1.47723098
Iteration 1711, loss = 1.47660902
Iteration 1712, loss = 1.47706121
Iteration 1713, loss = 1.47793829
Iteration 1714, loss = 1.47629754
Iteration 1715, loss = 1.47789043
Iteration 1716, loss = 1.47658537
Iteration 1717, loss = 1.47736297
Iteration 1718, loss = 1.47753813
Iteration 1719, loss = 1.47825582
Iteration 1720, loss = 1.47577877
Iteration 1721, loss = 1.47666606
Iteration 1722, loss = 1.47590225
Iteration 1723, loss = 1.47658258
Iteration 1724, loss = 1.47554629
Iteration 1725, loss = 1.47618298
Iteration 1726, loss = 1.47578880
Iteration 1727, loss = 1.47547130
Iteration 1728, loss = 1.47537520
Iteration 1729, loss = 1.47553085
Iteration 1730, loss = 1.47554890
Iteration 1731, loss = 1.47545771
Iteration 1732, loss = 1.47547643
Iteration 1733, loss = 1.47564260
Iteration 1734, loss = 1.47525293
Iteration 1735, loss = 1.47573401
Iteration 1736, loss = 1.47414808
Iteration 1737, loss = 1.47449496
Iteration 1738, loss = 1.47473297
Iteration 1739, loss = 1.47377261
Iteration 1740, loss = 1.47387322
```

```
Iteration 1741, loss = 1.47440540
Iteration 1742, loss = 1.47361861
Iteration 1743, loss = 1.47405664
Iteration 1744, loss = 1.47351410
Iteration 1745, loss = 1.47360831
Iteration 1746, loss = 1.47490100
Iteration 1747, loss = 1.47332482
Iteration 1748, loss = 1.47436255
Iteration 1749, loss = 1.47414050
Iteration 1750, loss = 1.47436405
Iteration 1751, loss = 1.47420784
Iteration 1752, loss = 1.47294015
Iteration 1753, loss = 1.47400633
Iteration 1754, loss = 1.47331129
Iteration 1755, loss = 1.47377890
Iteration 1756, loss = 1.47376747
Iteration 1757, loss = 1.47282866
Iteration 1758, loss = 1.47234532
Iteration 1759, loss = 1.47331533
Iteration 1760, loss = 1.47192538
Iteration 1761, loss = 1.47188329
Iteration 1762, loss = 1.47201694
Iteration 1763, loss = 1.47223533
Iteration 1764, loss = 1.47389061
Iteration 1765, loss = 1.47282105
Iteration 1766, loss = 1.47194581
Iteration 1767, loss = 1.47136549
Iteration 1768, loss = 1.47180743
Iteration 1769, loss = 1.47225532
Iteration 1770, loss = 1.47212561
Iteration 1771, loss = 1.47152632
Iteration 1772, loss = 1.47118835
Iteration 1773, loss = 1.47186988
Iteration 1774, loss = 1.47116488
Iteration 1775, loss = 1.47089011
Iteration 1776, loss = 1.47139253
Iteration 1777, loss = 1.47089833
Iteration 1778, loss = 1.47083507
Iteration 1779, loss = 1.47020634
Iteration 1780, loss = 1.47092273
Iteration 1781, loss = 1.47051352
Iteration 1782, loss = 1.47027037
Iteration 1783, loss = 1.46992571
Iteration 1784, loss = 1.47056014
Iteration 1785, loss = 1.47081722
Iteration 1786, loss = 1.47023527
Iteration 1787, loss = 1.46983404
Iteration 1788, loss = 1.47027367
Iteration 1789, loss = 1.46957062
Iteration 1790, loss = 1.46960458
Iteration 1791, loss = 1.46876909
Iteration 1792, loss = 1.46950139
Iteration 1793, loss = 1.46913865
Iteration 1794, loss = 1.46871754
Iteration 1795, loss = 1.46927822
Iteration 1796, loss = 1.46947162
Iteration 1797, loss = 1.46868592
Iteration 1798, loss = 1.46788749
Iteration 1799, loss = 1.46910213
Iteration 1800, loss = 1.46781565
```

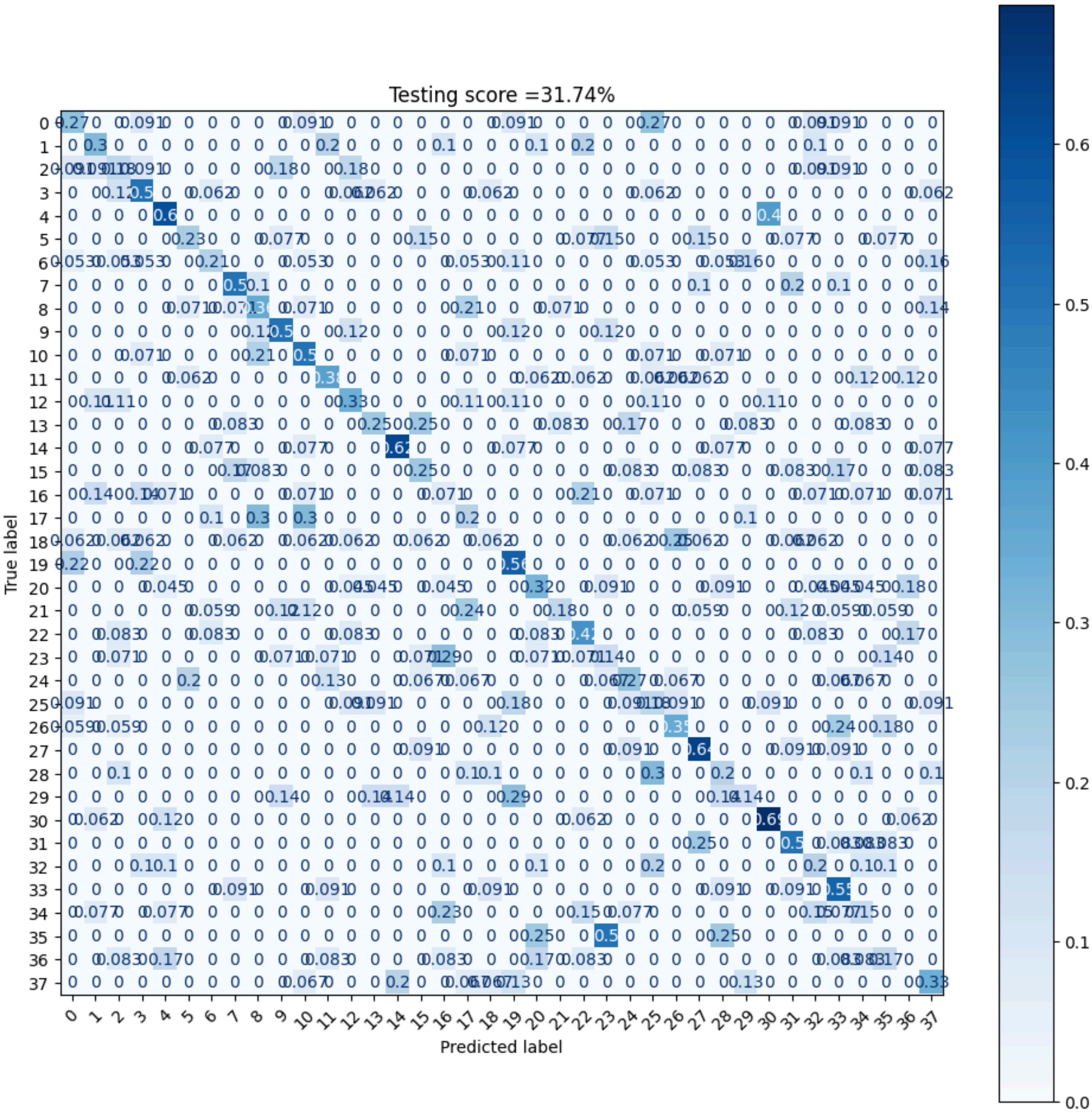
```
Iteration 1801, loss = 1.46909093
Iteration 1802, loss = 1.46850246
Iteration 1803, loss = 1.46857899
Iteration 1804, loss = 1.46891105
Iteration 1805, loss = 1.46820841
Iteration 1806, loss = 1.46800248
Iteration 1807, loss = 1.46823334
Iteration 1808, loss = 1.46813451
Iteration 1809, loss = 1.46789077
Iteration 1810, loss = 1.46825029
Iteration 1811, loss = 1.46822873
Training loss did not improve more than tol=0.000001 for 10 consecutive epochs. S
topping.
              precision    recall  f1-score   support

           0       0.46      0.27      0.34        22
           1       0.33      0.30      0.32        10
           2       0.17      0.18      0.17        11
           3       0.42      0.50      0.46        16
           4       0.27      0.60      0.38         5
           5       0.38      0.23      0.29        13
           6       0.44      0.21      0.29        19
           7       0.45      0.50      0.48        10
           8       0.36      0.36      0.36        14
           9       0.36      0.50      0.42         8
          10       0.35      0.50      0.41        14
          11       0.46      0.38      0.41        16
          12       0.27      0.33      0.30         9
          13       0.43      0.25      0.32        12
          14       0.67      0.62      0.64        13
          15       0.25      0.25      0.25        12
          16       0.08      0.07      0.08        14
          17       0.13      0.20      0.16        10
          18       0.14      0.06      0.09        16
          19       0.28      0.56      0.37         9
          20       0.47      0.32      0.38        22
          21       0.60      0.18      0.27        17
          22       0.29      0.42      0.34        12
          23       0.20      0.14      0.17        14
          24       0.36      0.27      0.31        15
          25       0.11      0.18      0.13        11
          26       0.46      0.35      0.40        17
          27       0.41      0.64      0.50        11
          28       0.20      0.20      0.20        10
          29       0.12      0.14      0.13         7
          30       0.73      0.69      0.71        16
          31       0.40      0.50      0.44        12
          32       0.17      0.20      0.18        10
          33       0.26      0.55      0.35        11
          34       0.17      0.15      0.16        13
          35       0.00      0.00      0.00         4
          36       0.00      0.00      0.00        12
          37       0.31      0.33      0.32        15

    accuracy                           0.32       482
   macro avg       0.32      0.32      0.30       482
weighted avg       0.34      0.32      0.31       482
```

In [ ]:

```python
from sklearn.metrics import ConfusionMatrixDisplay
import matplotlib.pyplot as plt
fig, ax = plt.subplots(1, 1, figsize=(12,12))
score = 100*clf_MLP.score(X_test_pca, y_test)
title = 'Testing score ={:.2f}%'.format(score)
disp = ConfusionMatrixDisplay.from_estimator(
clf_MLP,
X_test_pca,
y_test,
xticks_rotation=45, #’vertical’,
# display_labels=class_names,
cmap=plt.cm.Blues,
normalize='true',
ax = ax
)
disp.ax_.set_title(title)
plt.show()
```



<解釋混淆矩陣概念>

混淆矩陣（Confusion Matrix）是一種用於評估分類模型性能的表格，特別是在多類分類問題中。它以矩陣的形式顯示了模型在測試集上的預測結果與真實標籤之間的對應關係。

混淆矩陣的結構如下：

```
             Predicted Class 1   Predicted Class 2   ...
Predicted Class N
Actual Class 1      True Positive        False Negative   ...
False Negative
Actual Class 2      False Positive       True Positive    ...
False Negative
   ...                 ...                  ...            ...
...
Actual Class N      False Positive      False Positive   ...
True Positive
```

在混淆矩陣中，行代表真實標籤，列代表模型預測的結果。其中每個元素的含義如下：

- True Positive（TP）：真實為正類（Actual Class）且被模型預測為正類（Predicted Class）的樣本數。
- False Positive（FP）：真實為負類且被模型錯誤地預測為正類的樣本數。
- True Negative（TN）：真實為負類且被模型預測為負類的樣本數。
- False Negative（FN）：真實為正類且被模型錯誤地預測為負類的樣本數。

通過混淆矩陣，可以清晰地了解模型在每個類別上的預測表現，進而計算出各種評估指標，例如精確率（Precision）、召回率（Recall）和 F1 分數（F1 Score）等，以更全面地評估模型的性能。

在這段程式碼中，使用 `ConfusionMatrixDisplay.from_estimator` 函數從已擬合的 `clf_MLP` 模型生成混淆矩陣的可視化表示，並將其顯示在圖形中。