第三次作品(專題)(3-2)：淺度機器學習分類器的評比實驗

學號：410978040

姓名：黃冠翔

作品目標：

- 利用多元羅吉斯回歸、支援向量機 SVM 和神經網路對資料進行分類學習與測試。
- 學習分類器的原理並進行評比實驗。

本專題計畫執行分類器比較，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： (1)多元羅吉斯回歸 (Multinomial Logistic Regression) (2)支援向量機 (Support Vector Machine) (3)神經網路 (Neural Network)

三組資料包括： (1)來自 3 個產區，178 瓶葡萄酒，含 13 種葡萄酒成分。 (2)來自 AT&T 40 個人的人臉影像共 400 張，每張大小 64×64。 (3)來自 Yale Face 38 人的人臉影像共 2410 張，每張大小 192×168。

此檔案以 AT&T 40 個人的人臉影像資料進行分類學習與測試。

先讀取資料並設定變數，同時將資料標準化。

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# Read data
df = pd.read_csv('D:\\vs\\venv_name\\face_data.csv')
X = np.array(df.iloc[:, :-1]) # 排除最後一欄標籤
y = np.array(df.iloc[:, -1]) # 標籤欄
# Split data into training and testing data
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.30)
# Standardize data
scaler = StandardScaler()
X_train_ = scaler.fit_transform(X_train)
X_test_ = scaler.fit_transform(X_test)
```

(1)多元羅吉斯回歸 (Multinomial Logistic Regression)

(a)原始資料

1.使用 lbfgs 的演算法

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

opts = dict(tol = 1e-6, max_iter = int(1e6), verbose=1)
solver = 'lbfgs' # 'lbfgs' is the default
```

```
# solver = 'liblinear'
# solver = 'newton-cg'
clf_original = LogisticRegression(solver = solver, **opts)
clf_original.fit(X_train_, y_train)
y_pred = clf_original.predict(X_test_)
# 測試資料之準確率回報
print(f"{accuracy_score(y_test, y_pred):.2%}\n")
print(f"{clf_original.score(X_test_, y_test):.2%}\n")
print(classification_report(y_test, y_pred))
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1
concurrent workers.
```

```
93.33%

93.33%
```

	precision	recall	f1-score	support
0	1.00	0.75	0.86	4
1	1.00	1.00	1.00	4
2	1.00	1.00	1.00	2
3	0.80	0.80	0.80	5
4	1.00	0.80	0.89	5
5	1.00	1.00	1.00	2
6	1.00	1.00	1.00	4
7	1.00	1.00	1.00	1
8	1.00	1.00	1.00	3
9	1.00	1.00	1.00	2
10	1.00	1.00	1.00	6
11	1.00	1.00	1.00	4
12	1.00	0.83	0.91	6
13	1.00	1.00	1.00	1
14	1.00	1.00	1.00	2
15	0.50	1.00	0.67	1
16	1.00	1.00	1.00	1
17	1.00	1.00	1.00	3
18	1.00	1.00	1.00	5
19	1.00	1.00	1.00	4
21	0.00	0.00	0.00	1
22	0.80	1.00	0.89	4
23	1.00	1.00	1.00	2
24	0.50	1.00	0.67	1
25	1.00	0.60	0.75	5
26	1.00	1.00	1.00	2
27	1.00	1.00	1.00	3
28	1.00	1.00	1.00	4
29	0.67	1.00	0.80	2
30	1.00	0.80	0.89	5
31	1.00	1.00	1.00	4

```
          32       1.00      1.00      1.00         2
          33       1.00      1.00      1.00         4
          34       1.00      1.00      1.00         3
          35       1.00      1.00      1.00         3
          36       1.00      1.00      1.00         4
          37       1.00      1.00      1.00         3
          38       0.67      1.00      0.80         2
          39       0.33      1.00      0.50         1

    accuracy                           0.93       120
   macro avg       0.90      0.94      0.91       120
weighted avg       0.95      0.93      0.93       120


[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:   22.3s finished
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

2.使用 liblinear 的演算法

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

opts = dict(tol = 1e-6, max_iter = int(1e6), verbose=1)
solver = 'liblinear'
# solver = 'newton-cg'
clf_original = LogisticRegression(solver = solver, **opts)
clf_original.fit(X_train_, y_train)
y_pred = clf_original.predict(X_test_)
# 測試資料之準確率回報
print(f"{accuracy_score(y_test, y_pred):.2%}\n")
print(f"{clf_original.score(X_test_, y_test):.2%}\n")
print(classification_report(y_test, y_pred))

[LibLinear]86.67%

86.67%
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         4
           1       1.00      1.00      1.00         4
           2       1.00      1.00      1.00         2
           3       1.00      0.60      0.75         5
           4       1.00      0.80      0.89         5
           5       1.00      1.00      1.00         2
           6       1.00      1.00      1.00         4
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         3
           9       1.00      1.00      1.00         2
          10       1.00      1.00      1.00         6
          11       1.00      1.00      1.00         4
          12       1.00      0.33      0.50         6
          13       0.25      1.00      0.40         1
          14       1.00      1.00      1.00         2
          15       0.20      1.00      0.33         1
          16       0.50      1.00      0.67         1
          17       1.00      1.00      1.00         3
          18       1.00      1.00      1.00         5
          19       0.80      1.00      0.89         4
          21       0.00      0.00      0.00         1
          22       1.00      1.00      1.00         4
          23       1.00      1.00      1.00         2
          24       1.00      1.00      1.00         1
          25       1.00      0.60      0.75         5
          26       1.00      1.00      1.00         2
          27       0.50      1.00      0.67         3
          28       1.00      1.00      1.00         4
          29       1.00      1.00      1.00         2
          30       1.00      0.80      0.89         5
          31       1.00      1.00      1.00         4
          32       1.00      1.00      1.00         2
          33       1.00      1.00      1.00         4
          34       1.00      0.67      0.80         3
          35       0.75      1.00      0.86         3
          36       1.00      1.00      1.00         4
          37       0.75      1.00      0.86         3
          38       0.67      1.00      0.80         2
          39       1.00      1.00      1.00         1

    accuracy                           0.87       120
   macro avg       0.86      0.89      0.85       120
weighted avg       0.90      0.87      0.86       120


d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
```

```
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

3.使用 newton-cg 的演算法

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

opts = dict(tol = 1e-6, max_iter = int(1e6), verbose=1)
solver = 'newton-cg'
clf_original = LogisticRegression(solver = solver, **opts)
clf_original.fit(X_train_, y_train)
y_pred = clf_original.predict(X_test_)
# 測試資料之準確率回報
print(f"{accuracy_score(y_test, y_pred):.2%}\n")
print(f"{clf_original.score(X_test_, y_test):.2%}\n")
print(classification_report(y_test, y_pred))
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1
concurrent workers.

93.33%

93.33%

              precision    recall  f1-score   support

           0       1.00      0.75      0.86         4
           1       1.00      1.00      1.00         4
           2       1.00      1.00      1.00         2
           3       0.80      0.80      0.80         5
           4       1.00      0.80      0.89         5
           5       1.00      1.00      1.00         2
           6       1.00      1.00      1.00         4
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         3
           9       1.00      1.00      1.00         2
          10       1.00      1.00      1.00         6
          11       1.00      1.00      1.00         4
```

```
          12       1.00      0.83      0.91         6
          13       1.00      1.00      1.00         1
          14       1.00      1.00      1.00         2
          15       0.50      1.00      0.67         1
          16       1.00      1.00      1.00         1
          17       1.00      1.00      1.00         3
          18       1.00      1.00      1.00         5
          19       1.00      1.00      1.00         4
          21       0.00      0.00      0.00         1
          22       0.80      1.00      0.89         4
          23       1.00      1.00      1.00         2
          24       0.50      1.00      0.67         1
          25       1.00      0.60      0.75         5
          26       1.00      1.00      1.00         2
          27       1.00      1.00      1.00         3
          28       1.00      1.00      1.00         4
          29       0.67      1.00      0.80         2
          30       1.00      0.80      0.89         5
          31       1.00      1.00      1.00         4
          32       1.00      1.00      1.00         2
          33       1.00      1.00      1.00         4
          34       1.00      1.00      1.00         3
          35       1.00      1.00      1.00         3
          36       1.00      1.00      1.00         4
          37       1.00      1.00      1.00         3
          38       0.67      1.00      0.80         2
          39       0.33      1.00      0.50         1

    accuracy                           0.93       120
   macro avg       0.90      0.94      0.91       120
weighted avg       0.95      0.93      0.93       120


[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:   11.2s finished
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

討論：

- 使用 lbfgs 的演算法時，準確率為 93.33%。
- 使用 liblinear 的演算法時，準確率為 86.67%。
- 使用 newton-cg 的演算法時，準確率為 93.33%。
- 綜上所述，lbfgs 和 newton-cg 的準確率相同，且高於 liblinear。

(b)主成分資料

1.取 10 個主成分並使用 lbfgs 的演算法

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 10).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)
opts = dict(tol = 1e-6, max_iter = int(1e6), verbose=1)
solver = 'lbfgs' # 'lbfgs' is the default
clf_PCA = LogisticRegression(solver = solver, **opts)
clf_PCA.fit(Z_train, y_train)
y_pred = clf_PCA.predict(Z_test)
print(f"{clf_PCA.score(Z_test, y_test):.2%}\n")
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1
concurrent workers.

82.50%


[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.6s finished
```

2.取 30 個主成分並使用 lbfgs 的演算法

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 30).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)
opts = dict(tol = 1e-6, max_iter = int(1e6), verbose=1)
solver = 'lbfgs' # 'lbfgs' is the default
clf_PCA = LogisticRegression(solver = solver, **opts)
clf_PCA.fit(Z_train, y_train)
y_pred = clf_PCA.predict(Z_test)
print(f"{clf_PCA.score(Z_test, y_test):.2%}\n")
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1
concurrent workers.

93.33%


[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.4s finished
```

3.取 50 個主成分並使用 lbfgs 的演算法

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 50).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)
opts = dict(tol = 1e-6, max_iter = int(1e6), verbose=1)
solver = 'lbfgs' # 'lbfgs' is the default
clf_PCA = LogisticRegression(solver = solver, **opts)
clf_PCA.fit(Z_train, y_train)
y_pred = clf_PCA.predict(Z_test)
print(f"{clf_PCA.score(Z_test, y_test):.2%}\n")
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1
concurrent workers.

94.17%


[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.4s finished
```

4.取 10 個主成分並使用 liblinear 的演算法

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 10).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)
opts = dict(tol = 1e-6, max_iter = int(1e6), verbose=1)
solver = 'liblinear' # 'lbfgs' is the default
clf_PCA = LogisticRegression(solver = solver, **opts)
clf_PCA.fit(Z_train, y_train)
y_pred = clf_PCA.predict(Z_test)
print(f"{clf_PCA.score(Z_test, y_test):.2%}\n")
```

```
[LibLinear]72.50%
```

5.取 30 個主成分並使用 liblinear 的演算法

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 30).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)
opts = dict(tol = 1e-6, max_iter = int(1e6), verbose=1)
solver = 'liblinear' # 'lbfgs' is the default
clf_PCA = LogisticRegression(solver = solver, **opts)
clf_PCA.fit(Z_train, y_train)
y_pred = clf_PCA.predict(Z_test)
print(f"{clf_PCA.score(Z_test, y_test):.2%}\n")
```

```
[LibLinear]90.00%
```

6.取 50 個主成分並使用 liblinear 的演算法

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 50).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)
opts = dict(tol = 1e-6, max_iter = int(1e6), verbose=1)
solver = 'liblinear' # 'lbfgs' is the default
clf_PCA = LogisticRegression(solver = solver, **opts)
clf_PCA.fit(Z_train, y_train)
y_pred = clf_PCA.predict(Z_test)
print(f"{clf_PCA.score(Z_test, y_test):.2%}\n")
```

```
[LibLinear]92.50%
```

7.取 10 個主成分並使用 newton-cg 的演算法

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 10).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)
opts = dict(tol = 1e-6, max_iter = int(1e6), verbose=1)
solver = 'newton-cg' # 'lbfgs' is the default
clf_PCA = LogisticRegression(solver = solver, **opts)
clf_PCA.fit(Z_train, y_train)
y_pred = clf_PCA.predict(Z_test)
print(f"{clf_PCA.score(Z_test, y_test):.2%}\n")
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1
concurrent workers.

82.50%


[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.8s finished
```

8.取 30 個主成分並使用 newton-cg 的演算法

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 30).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)
opts = dict(tol = 1e-6, max_iter = int(1e6), verbose=1)
solver = 'newton-cg' # 'lbfgs' is the default
clf_PCA = LogisticRegression(solver = solver, **opts)
clf_PCA.fit(Z_train, y_train)
y_pred = clf_PCA.predict(Z_test)
print(f"{clf_PCA.score(Z_test, y_test):.2%}\n")
```

```
92.50%


[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1
concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.1s finished
```

9.取 50 個主成分並使用 newton-cg 的演算法

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 50).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)
opts = dict(tol = 1e-6, max_iter = int(1e6), verbose=1)
solver = 'newton-cg' # 'lbfgs' is the default
clf_PCA = LogisticRegression(solver = solver, **opts)
clf_PCA.fit(Z_train, y_train)
y_pred = clf_PCA.predict(Z_test)
print(f"{clf_PCA.score(Z_test, y_test):.2%}\n")
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1
concurrent workers.

95.00%


[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.2s finished
```

討論：

使用 lbfgs 的演算法：

- 取 10 個主成分時，準確率為 82.50%。
- 取 30 個主成分時，準確率為 93.33%。
- 取 50 個主成分時，準確率為 94.17%。

使用 liblinear 的演算法：

- 取 10 個主成分時，準確率為 72.50%。
- 取 30 個主成分時，準確率為 90.00%。
- 取 50 個主成分時，準確率為 92.50%。

使用 newton-cg 的演算法：

- 取 10 個主成分時，準確率為 82.50%。
- 取 30 個主成分時，準確率為 92.50%。
- 取 50 個主成分時，準確率為 95.00%。

小結：

- 使用 liblinear 的演算法時，準確率最低，使用 lbfgs 的演算法和使用 newton-cg 的演算法所得之準確率差不多。
- 取愈多主成分，準確率愈高。

(2)支援向量機 (Support Vector Machine)

(a)原始資料

1.使用 kernel="linear"

```
from sklearn.svm import SVC, LinearSVC
C = 1 # SVM regularization parameter
opts = dict(C = C, tol = 1e-6, max_iter = int(1e6))
# opts = dict(C = C, decision_function_shape = 'ovo', \
# tol = 1e-6, max_iter = int(1e6))
clf_svm = SVC(kernel="linear", **opts)
# clf_svm = SVC(kernel="rbf", gamma=0.2, **opts)
# clf_svm = SVC(kernel="poly", degree=3, gamma="auto", **opts)
# clf_svm = LinearSVC(**opts) # one vs the rest
clf_svm.fit(X_train_, y_train)
predictions = clf_svm.predict(X_test_)
print(f"{accuracy_score(y_test, predictions):.2%}\n")
print(classification_report(y_test, predictions))
```

```
91.67%

              precision    recall  f1-score   support

           0       1.00      0.75      0.86         4
           1       1.00      1.00      1.00         4
           2       1.00      0.50      0.67         2
           3       0.80      0.80      0.80         5
           4       1.00      0.80      0.89         5
           5       1.00      1.00      1.00         2
           6       1.00      1.00      1.00         4
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         3
           9       1.00      1.00      1.00         2
          10       1.00      1.00      1.00         6
          11       1.00      1.00      1.00         4
          12       1.00      0.67      0.80         6
          13       1.00      1.00      1.00         1
          14       0.67      1.00      0.80         2
          15       0.50      1.00      0.67         1
          16       1.00      1.00      1.00         1
          17       1.00      1.00      1.00         3
          18       1.00      1.00      1.00         5
          19       1.00      1.00      1.00         4
          20       0.00      0.00      0.00         0
          21       0.00      0.00      0.00         1
          22       0.80      1.00      0.89         4
```

```
           23       1.00      1.00      1.00         2
           24       0.50      1.00      0.67         1
           25       0.75      0.60      0.67         5
           26       1.00      1.00      1.00         2
           27       1.00      1.00      1.00         3
           28       1.00      1.00      1.00         4
           29       0.67      1.00      0.80         2
           30       1.00      0.80      0.89         5
           31       1.00      1.00      1.00         4
           32       1.00      1.00      1.00         2
           33       1.00      1.00      1.00         4
           34       1.00      1.00      1.00         3
           35       1.00      1.00      1.00         3
           36       1.00      1.00      1.00         4
           37       1.00      1.00      1.00         3
           38       0.67      1.00      0.80         2
           39       0.50      1.00      0.67         1

    accuracy                           0.92       120
   macro avg       0.87      0.90      0.87       120
weighted avg       0.94      0.92      0.92       120


d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
```

```
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

2.使用 kernel="rbf"

```
from sklearn.svm import SVC, LinearSVC
C = 1 # SVM regularization parameter
opts = dict(C = C, tol = 1e-6, max_iter = int(1e6))
# opts = dict(C = C, decision_function_shape = 'ovo', \
# tol = 1e-6, max_iter = int(1e6))
#clf_svm = SVC(kernel="linear", **opts)
clf_svm = SVC(kernel="rbf", gamma='auto', **opts)
# clf_svm = SVC(kernel="poly", degree=3, gamma="auto", **opts)
# clf_svm = LinearSVC(**opts) # one vs the rest
clf_svm.fit(X_train_, y_train)
predictions = clf_svm.predict(X_test_)
print(f"{accuracy_score(y_test, predictions):.2%}\n")
print(classification_report(y_test, predictions))
```

```
84.17%

              precision    recall  f1-score   support

           0       1.00      0.50      0.67         4
           1       1.00      1.00      1.00         4
           2       1.00      0.50      0.67         2
           3       0.67      0.40      0.50         5
           4       1.00      0.80      0.89         5
           5       1.00      1.00      1.00         2
           6       1.00      1.00      1.00         4
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         3
           9       1.00      1.00      1.00         2
          10       1.00      1.00      1.00         6
          11       1.00      1.00      1.00         4
          12       1.00      0.50      0.67         6
          13       1.00      1.00      1.00         1
          14       1.00      1.00      1.00         2
          15       0.33      1.00      0.50         1
          16       1.00      1.00      1.00         1
          17       1.00      1.00      1.00         3
          18       1.00      1.00      1.00         5
          19       1.00      1.00      1.00         4
          20       0.00      0.00      0.00         0
          21       0.00      0.00      0.00         1
          22       0.80      1.00      0.89         4
          23       1.00      1.00      1.00         2
          24       0.33      1.00      0.50         1
```

```
          25       1.00      0.60      0.75         5
          26       1.00      1.00      1.00         2
          27       1.00      1.00      1.00         3
          28       1.00      0.75      0.86         4
          29       1.00      1.00      1.00         2
          30       1.00      0.20      0.33         5
          31       1.00      1.00      1.00         4
          32       1.00      1.00      1.00         2
          33       1.00      1.00      1.00         4
          34       1.00      0.67      0.80         3
          35       1.00      1.00      1.00         3
          36       1.00      1.00      1.00         4
          37       1.00      1.00      1.00         3
          38       0.50      1.00      0.67         2
          39       0.12      1.00      0.22         1

    accuracy                           0.84       120
   macro avg       0.87      0.85      0.82       120
weighted avg       0.94      0.84      0.86       120


d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
```

```
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

3.使用 kernel="poly"

```
from sklearn.svm import SVC, LinearSVC
C = 1 # SVM regularization parameter
opts = dict(C = C, tol = 1e-6, max_iter = int(1e6))
# opts = dict(C = C, decision_function_shape = 'ovo', \
# tol = 1e-6, max_iter = int(1e6))
#clf_svm = SVC(kernel="linear", **opts)
# clf_svm = SVC(kernel="rbf", gamma=0.2, **opts)
clf_svm = SVC(kernel="poly", degree=1, gamma="auto", **opts)
# clf_svm = LinearSVC(**opts) # one vs the rest
clf_svm.fit(X_train_, y_train)
predictions = clf_svm.predict(X_test_)
print(f"{accuracy_score(y_test, predictions):.2%}\n")
print(classification_report(y_test, predictions))
```

```
82.50%

              precision    recall  f1-score   support

           0       1.00      0.50      0.67         4
           1       1.00      1.00      1.00         4
           2       1.00      0.50      0.67         2
           3       0.67      0.40      0.50         5
           4       1.00      0.80      0.89         5
           5       1.00      1.00      1.00         2
           6       1.00      1.00      1.00         4
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         3
           9       1.00      1.00      1.00         2
          10       1.00      1.00      1.00         6
          11       1.00      1.00      1.00         4
          12       1.00      0.33      0.50         6
          13       1.00      1.00      1.00         1
          14       0.67      1.00      0.80         2
          15       0.33      1.00      0.50         1
          16       1.00      1.00      1.00         1
          17       1.00      1.00      1.00         3
          18       1.00      1.00      1.00         5
          19       1.00      1.00      1.00         4
          20       0.00      0.00      0.00         0
          21       0.00      0.00      0.00         1
          22       0.80      1.00      0.89         4
          23       1.00      1.00      1.00         2
          24       0.33      1.00      0.50         1
          25       1.00      0.60      0.75         5
```

```
          26       1.00      1.00      1.00         2
          27       1.00      1.00      1.00         3
          28       1.00      0.50      0.67         4
          29       0.67      1.00      0.80         2
          30       1.00      0.20      0.33         5
          31       1.00      1.00      1.00         4
          32       1.00      1.00      1.00         2
          33       1.00      1.00      1.00         4
          34       1.00      0.67      0.80         3
          35       1.00      1.00      1.00         3
          36       1.00      1.00      1.00         4
          37       1.00      1.00      1.00         3
          38       0.40      1.00      0.57         2
          39       0.12      1.00      0.22         1

    accuracy                           0.82       120
   macro avg       0.85      0.84      0.80       120
weighted avg       0.93      0.82      0.84       120


d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

討論：

- 使用 kernel="linear"時，準確率為 91.67%。
- 使用 kernel="rbf"時，準確率為 84.17%。
- 使用 kernel="poly"時，準確率為 82.50%。
- 綜上所述，kernel="linear"之準確率最高，kernel="rbf"次之，kernel="poly"之準確率最低。

(b)主成分資料

1.取 10 個主成分並使用 kernel="linear"

```
from sklearn.decomposition import PCA
from sklearn.svm import SVC, LinearSVC

pca = PCA(n_components = 10).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)
C = 1 # SVM regularization parameter
opts = dict(C = C, tol = 1e-6, max_iter = int(1e6))
clf_svm = SVC(kernel="linear", **opts)
clf_svm.fit(Z_train, y_train)
predictions = clf_svm.predict(Z_test)
print(f"{clf_svm.score(Z_test, y_test):.2%}\n")
print(classification_report(y_test, predictions))
```

```
85.83%

              precision    recall  f1-score   support

           0       1.00      1.00      1.00         4
           1       1.00      1.00      1.00         4
           2       1.00      1.00      1.00         2
           3       0.50      0.40      0.44         5
           4       1.00      0.80      0.89         5
           5       0.67      1.00      0.80         2
           6       1.00      0.75      0.86         4
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         3
           9       1.00      1.00      1.00         2
          10       1.00      0.83      0.91         6
          11       1.00      1.00      1.00         4
          12       0.50      0.33      0.40         6
          13       1.00      1.00      1.00         1
          14       0.50      1.00      0.67         2
          15       0.33      1.00      0.50         1
          16       1.00      1.00      1.00         1
          17       1.00      1.00      1.00         3
          18       0.83      1.00      0.91         5
          19       1.00      0.75      0.86         4
          20       0.00      0.00      0.00         0
```

```
          21       1.00      1.00      1.00         1
          22       0.80      1.00      0.89         4
          23       1.00      1.00      1.00         2
          24       0.50      1.00      0.67         1
          25       1.00      0.60      0.75         5
          26       1.00      1.00      1.00         2
          27       1.00      1.00      1.00         3
          28       1.00      0.75      0.86         4
          29       1.00      1.00      1.00         2
          30       1.00      1.00      1.00         5
          31       1.00      1.00      1.00         4
          32       1.00      1.00      1.00         2
          33       1.00      1.00      1.00         4
          34       1.00      1.00      1.00         3
          35       0.50      0.33      0.40         3
          36       1.00      1.00      1.00         4
          37       1.00      0.67      0.80         3
          38       0.67      1.00      0.80         2
          39       0.50      1.00      0.67         1

    accuracy                           0.86       120
   macro avg       0.86      0.88      0.85       120
weighted avg       0.89      0.86      0.86       120


d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

2.取 30 個主成分並使用 kernel="linear"

```
from sklearn.decomposition import PCA
from sklearn.svm import SVC, LinearSVC

pca = PCA(n_components = 30).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)
```

```
C = 1 # SVM regularization parameter
opts = dict(C = C, tol = 1e-6, max_iter = int(1e6))
clf_svm = SVC(kernel="linear", **opts)
clf_svm.fit(Z_train, y_train)
predictions = clf_svm.predict(Z_test)
print(f"{clf_svm.score(Z_test, y_test):.2%}\n")
print(classification_report(y_test, predictions))
```

```
93.33%
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	4
1	1.00	1.00	1.00	4
2	0.50	0.50	0.50	2
3	1.00	0.80	0.89	5
4	1.00	0.80	0.89	5
5	0.67	1.00	0.80	2
6	1.00	1.00	1.00	4
7	1.00	1.00	1.00	1
8	1.00	1.00	1.00	3
9	1.00	1.00	1.00	2
10	1.00	1.00	1.00	6
11	1.00	1.00	1.00	4
12	1.00	0.67	0.80	6
13	1.00	1.00	1.00	1
14	0.67	1.00	0.80	2
15	1.00	1.00	1.00	1
16	1.00	1.00	1.00	1
17	1.00	1.00	1.00	3
18	1.00	1.00	1.00	5
19	1.00	1.00	1.00	4
21	0.00	0.00	0.00	1
22	0.80	1.00	0.89	4
23	1.00	1.00	1.00	2
24	0.50	1.00	0.67	1
25	0.75	0.60	0.67	5
26	1.00	1.00	1.00	2
27	1.00	1.00	1.00	3
28	1.00	1.00	1.00	4
29	0.67	1.00	0.80	2
30	1.00	1.00	1.00	5
31	1.00	1.00	1.00	4
32	1.00	1.00	1.00	2
33	1.00	1.00	1.00	4
34	1.00	1.00	1.00	3
35	1.00	1.00	1.00	3
36	1.00	1.00	1.00	4
37	1.00	1.00	1.00	3
38	0.67	1.00	0.80	2

```
          39       1.00      1.00      1.00         1

    accuracy                           0.93       120
   macro avg       0.90      0.93      0.91       120
weighted avg       0.94      0.93      0.93       120


d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

3.取 50 個主成分並使用 kernel="linear"

```
from sklearn.decomposition import PCA
from sklearn.svm import SVC, LinearSVC

pca = PCA(n_components = 50).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)
C = 1 # SVM regularization parameter
opts = dict(C = C, tol = 1e-6, max_iter = int(1e6))
clf_svm = SVC(kernel="linear", **opts)
clf_svm.fit(Z_train, y_train)
predictions = clf_svm.predict(Z_test)
print(f"{clf_svm.score(Z_test, y_test):.2%}\n")
print(classification_report(y_test, predictions))
```

```
94.17%

              precision    recall  f1-score   support

           0       1.00      1.00      1.00         4
           1       1.00      1.00      1.00         4
           2       1.00      1.00      1.00         2
           3       0.80      0.80      0.80         5
           4       1.00      0.80      0.89         5
           5       0.67      1.00      0.80         2
           6       1.00      1.00      1.00         4
```

```
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         3
           9       1.00      1.00      1.00         2
          10       1.00      1.00      1.00         6
          11       1.00      1.00      1.00         4
          12       1.00      0.67      0.80         6
          13       1.00      1.00      1.00         1
          14       0.67      1.00      0.80         2
          15       1.00      1.00      1.00         1
          16       1.00      1.00      1.00         1
          17       1.00      1.00      1.00         3
          18       1.00      1.00      1.00         5
          19       1.00      1.00      1.00         4
          21       0.00      0.00      0.00         1
          22       0.80      1.00      0.89         4
          23       1.00      1.00      1.00         2
          24       0.50      1.00      0.67         1
          25       1.00      0.60      0.75         5
          26       1.00      1.00      1.00         2
          27       1.00      1.00      1.00         3
          28       1.00      1.00      1.00         4
          29       0.67      1.00      0.80         2
          30       1.00      1.00      1.00         5
          31       1.00      1.00      1.00         4
          32       1.00      1.00      1.00         2
          33       1.00      1.00      1.00         4
          34       1.00      1.00      1.00         3
          35       1.00      1.00      1.00         3
          36       1.00      1.00      1.00         4
          37       1.00      1.00      1.00         3
          38       0.67      1.00      0.80         2
          39       1.00      1.00      1.00         1

    accuracy                           0.94       120
   macro avg       0.92      0.95      0.92       120
weighted avg       0.95      0.94      0.94       120


d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
```

```
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

4.取 10 個主成分並使用 kernel="rbf"

```
from sklearn.decomposition import PCA
from sklearn.svm import SVC, LinearSVC

pca = PCA(n_components = 10).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)
C = 1 # SVM regularization parameter
opts = dict(C = C, tol = 1e-6, max_iter = int(1e6))
clf_svm = SVC(kernel="rbf", **opts)
clf_svm.fit(Z_train, y_train)
predictions = clf_svm.predict(Z_test)
print(f"{clf_svm.score(Z_test, y_test):.2%}\n")
print(classification_report(y_test, predictions))
```

```
60.83%

              precision    recall  f1-score   support

           0       1.00      0.50      0.67         4
           1       1.00      1.00      1.00         4
           2       0.20      0.50      0.29         2
           3       0.50      0.20      0.29         5
           4       1.00      0.40      0.57         5
           5       1.00      1.00      1.00         2
           6       0.60      0.75      0.67         4
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         3
           9       1.00      1.00      1.00         2
          10       1.00      0.33      0.50         6
          11       1.00      1.00      1.00         4
          12       1.00      0.17      0.29         6
          13       1.00      1.00      1.00         1
          14       0.40      1.00      0.57         2
          15       0.00      0.00      0.00         1
          16       0.50      1.00      0.67         1
          17       1.00      1.00      1.00         3
          18       0.20      0.20      0.20         5
          19       0.50      0.25      0.33         4
          20       0.00      0.00      0.00         0
          21       1.00      1.00      1.00         1
          22       0.50      0.50      0.50         4
          23       0.67      1.00      0.80         2
          24       0.33      1.00      0.50         1
```

```
          25       0.00      0.00      0.00         5
          26       1.00      1.00      1.00         2
          27       0.67      0.67      0.67         3
          28       1.00      0.25      0.40         4
          29       0.67      1.00      0.80         2
          30       1.00      0.20      0.33         5
          31       0.67      1.00      0.80         4
          32       1.00      1.00      1.00         2
          33       1.00      1.00      1.00         4
          34       1.00      0.33      0.50         3
          35       0.50      0.33      0.40         3
          36       1.00      1.00      1.00         4
          37       0.75      1.00      0.86         3
          38       0.40      1.00      0.57         2
          39       0.17      1.00      0.29         1

    accuracy                           0.61       120
   macro avg       0.71      0.69      0.64       120
weighted avg       0.75      0.61      0.61       120


d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
```

```
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

5.取 30 個主成分並使用 kernel="rbf"

```
from sklearn.decomposition import PCA
from sklearn.svm import SVC, LinearSVC

pca = PCA(n_components = 30).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)
C = 1 # SVM regularization parameter
opts = dict(C = C, tol = 1e-6, max_iter = int(1e6))
clf_svm = SVC(kernel="rbf", **opts)
clf_svm.fit(Z_train, y_train)
predictions = clf_svm.predict(Z_test)
print(f"{clf_svm.score(Z_test, y_test):.2%}\n")
print(classification_report(y_test, predictions))
```

```
83.33%

              precision    recall  f1-score   support

           0       1.00      0.50      0.67         4
           1       1.00      1.00      1.00         4
           2       1.00      0.50      0.67         2
           3       0.75      0.60      0.67         5
           4       1.00      0.80      0.89         5
           5       1.00      1.00      1.00         2
           6       1.00      1.00      1.00         4
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         3
           9       1.00      1.00      1.00         2
          10       1.00      1.00      1.00         6
          11       1.00      1.00      1.00         4
          12       1.00      0.33      0.50         6
          13       1.00      1.00      1.00         1
          14       0.67      1.00      0.80         2
          15       0.33      1.00      0.50         1
          16       1.00      1.00      1.00         1
          17       1.00      1.00      1.00         3
          18       1.00      1.00      1.00         5
          19       1.00      1.00      1.00         4
          20       0.00      0.00      0.00         0
          21       0.00      0.00      0.00         1
          22       0.80      1.00      0.89         4
          23       1.00      1.00      1.00         2
          24       0.33      1.00      0.50         1
          25       1.00      0.60      0.75         5
```

```
          26       1.00      1.00      1.00         2
          27       1.00      1.00      1.00         3
          28       1.00      0.75      0.86         4
          29       1.00      1.00      1.00         2
          30       1.00      0.20      0.33         5
          31       1.00      1.00      1.00         4
          32       1.00      1.00      1.00         2
          33       1.00      1.00      1.00         4
          34       1.00      0.33      0.50         3
          35       1.00      1.00      1.00         3
          36       1.00      1.00      1.00         4
          37       1.00      1.00      1.00         3
          38       0.50      1.00      0.67         2
          39       0.17      1.00      0.29         1

    accuracy                           0.83       120
   macro avg       0.86      0.84      0.81       120
weighted avg       0.94      0.83      0.85       120


d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

6.取 50 個主成分並使用 kernel="rbf"

```
from sklearn.decomposition import PCA
from sklearn.svm import SVC, LinearSVC

pca = PCA(n_components = 50).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)
C = 1 # SVM regularization parameter
opts = dict(C = C, tol = 1e-6, max_iter = int(1e6))
clf_svm = SVC(kernel="rbf", **opts)
clf_svm.fit(Z_train, y_train)
predictions = clf_svm.predict(Z_test)
print(f"{clf_svm.score(Z_test, y_test):.2%}\n")
print(classification_report(y_test, predictions))
```

```
84.17%

              precision    recall  f1-score   support

           0       1.00      0.50      0.67         4
           1       1.00      1.00      1.00         4
           2       1.00      0.50      0.67         2
           3       0.75      0.60      0.67         5
           4       1.00      0.80      0.89         5
           5       1.00      1.00      1.00         2
           6       1.00      1.00      1.00         4
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         3
           9       1.00      1.00      1.00         2
          10       1.00      1.00      1.00         6
          11       1.00      1.00      1.00         4
          12       1.00      0.33      0.50         6
          13       1.00      1.00      1.00         1
          14       0.67      1.00      0.80         2
          15       0.33      1.00      0.50         1
          16       1.00      1.00      1.00         1
          17       1.00      1.00      1.00         3
          18       1.00      1.00      1.00         5
          19       1.00      1.00      1.00         4
          20       0.00      0.00      0.00         0
          21       0.00      0.00      0.00         1
          22       0.80      1.00      0.89         4
          23       1.00      1.00      1.00         2
          24       0.33      1.00      0.50         1
          25       1.00      0.60      0.75         5
          26       1.00      1.00      1.00         2
          27       1.00      1.00      1.00         3
          28       1.00      0.75      0.86         4
          29       1.00      1.00      1.00         2
```

```
          30       1.00      0.20      0.33         5
          31       1.00      1.00      1.00         4
          32       1.00      1.00      1.00         2
          33       1.00      1.00      1.00         4
          34       1.00      0.67      0.80         3
          35       1.00      1.00      1.00         3
          36       1.00      1.00      1.00         4
          37       1.00      1.00      1.00         3
          38       0.50      1.00      0.67         2
          39       0.17      1.00      0.29         1

    accuracy                           0.84       120
   macro avg       0.86      0.85      0.82       120
weighted avg       0.94      0.84      0.86       120


d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

7.取 10 個主成分並使用 kernel="poly"

```
from sklearn.decomposition import PCA
from sklearn.svm import SVC, LinearSVC

pca = PCA(n_components = 10).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)
C = 1 # SVM regularization parameter
opts = dict(C = C, tol = 1e-6, max_iter = int(1e6))
clf_svm = SVC(kernel="poly", degree=1, gamma="auto", **opts)
clf_svm.fit(Z_train, y_train)
predictions = clf_svm.predict(Z_test)
print(f"{clf_svm.score(Z_test, y_test):.2%}\n")
print(classification_report(y_test, predictions))
```

```
85.83%
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	4
1	1.00	1.00	1.00	4
2	1.00	1.00	1.00	2
3	0.50	0.40	0.44	5
4	1.00	0.80	0.89	5
5	0.67	1.00	0.80	2
6	1.00	0.75	0.86	4
7	1.00	1.00	1.00	1
8	1.00	1.00	1.00	3
9	1.00	1.00	1.00	2
10	1.00	0.83	0.91	6
11	1.00	1.00	1.00	4
12	0.50	0.33	0.40	6
13	1.00	1.00	1.00	1
14	0.50	1.00	0.67	2
15	0.33	1.00	0.50	1
16	1.00	1.00	1.00	1
17	1.00	1.00	1.00	3
18	0.83	1.00	0.91	5
19	1.00	0.75	0.86	4
20	0.00	0.00	0.00	0
21	1.00	1.00	1.00	1
22	0.80	1.00	0.89	4
23	1.00	1.00	1.00	2
24	0.50	1.00	0.67	1
25	1.00	0.60	0.75	5
26	1.00	1.00	1.00	2
27	1.00	1.00	1.00	3
28	1.00	0.75	0.86	4
29	1.00	1.00	1.00	2
30	1.00	1.00	1.00	5
31	1.00	1.00	1.00	4

```
          32       1.00      1.00      1.00         2
          33       1.00      1.00      1.00         4
          34       1.00      1.00      1.00         3
          35       0.50      0.33      0.40         3
          36       1.00      1.00      1.00         4
          37       1.00      0.67      0.80         3
          38       0.67      1.00      0.80         2
          39       0.50      1.00      0.67         1

    accuracy                           0.86       120
   macro avg       0.86      0.88      0.85       120
weighted avg       0.89      0.86      0.86       120


d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

8.取 30 個主成分並使用 kernel="poly"

```python
from sklearn.decomposition import PCA
from sklearn.svm import SVC, LinearSVC

pca = PCA(n_components = 30).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)
C = 1 # SVM regularization parameter
opts = dict(C = C, tol = 1e-6, max_iter = int(1e6))
clf_svm = SVC(kernel="poly", degree=1, gamma="auto", **opts)
clf_svm.fit(Z_train, y_train)
predictions = clf_svm.predict(Z_test)
print(f"{clf_svm.score(Z_test, y_test):.2%}\n")
print(classification_report(y_test, predictions))
```

```
93.33%

              precision    recall  f1-score   support
```

```
           0       1.00      1.00      1.00         4
           1       1.00      1.00      1.00         4
           2       0.50      0.50      0.50         2
           3       1.00      0.80      0.89         5
           4       1.00      0.80      0.89         5
           5       0.67      1.00      0.80         2
           6       1.00      1.00      1.00         4
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         3
           9       1.00      1.00      1.00         2
          10       1.00      1.00      1.00         6
          11       1.00      1.00      1.00         4
          12       1.00      0.67      0.80         6
          13       1.00      1.00      1.00         1
          14       0.67      1.00      0.80         2
          15       1.00      1.00      1.00         1
          16       1.00      1.00      1.00         1
          17       1.00      1.00      1.00         3
          18       1.00      1.00      1.00         5
          19       1.00      1.00      1.00         4
          21       0.00      0.00      0.00         1
          22       0.80      1.00      0.89         4
          23       1.00      1.00      1.00         2
          24       0.50      1.00      0.67         1
          25       0.75      0.60      0.67         5
          26       1.00      1.00      1.00         2
          27       1.00      1.00      1.00         3
          28       1.00      1.00      1.00         4
          29       0.67      1.00      0.80         2
          30       1.00      1.00      1.00         5
          31       1.00      1.00      1.00         4
          32       1.00      1.00      1.00         2
          33       1.00      1.00      1.00         4
          34       1.00      1.00      1.00         3
          35       1.00      1.00      1.00         3
          36       1.00      1.00      1.00         4
          37       1.00      1.00      1.00         3
          38       0.67      1.00      0.80         2
          39       1.00      1.00      1.00         1

    accuracy                           0.93       120
   macro avg       0.90      0.93      0.91       120
weighted avg       0.94      0.93      0.93       120


d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

9.取 50 個主成分並使用 kernel="poly"

```
from sklearn.decomposition import PCA
from sklearn.svm import SVC, LinearSVC

pca = PCA(n_components = 50).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)
C = 1 # SVM regularization parameter
opts = dict(C = C, tol = 1e-6, max_iter = int(1e6))
clf_svm = SVC(kernel="poly", degree=1, gamma="auto", **opts)
clf_svm.fit(Z_train, y_train)
predictions = clf_svm.predict(Z_test)
print(f"{clf_svm.score(Z_test, y_test):.2%}\n")
print(classification_report(y_test, predictions))
```

```
94.17%

              precision    recall  f1-score   support

           0       1.00      1.00      1.00         4
           1       1.00      1.00      1.00         4
           2       1.00      1.00      1.00         2
           3       0.80      0.80      0.80         5
           4       1.00      0.80      0.89         5
           5       0.67      1.00      0.80         2
           6       1.00      1.00      1.00         4
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         3
           9       1.00      1.00      1.00         2
          10       1.00      1.00      1.00         6
          11       1.00      1.00      1.00         4
          12       1.00      0.67      0.80         6
          13       1.00      1.00      1.00         1
          14       0.67      1.00      0.80         2
          15       1.00      1.00      1.00         1
          16       1.00      1.00      1.00         1
          17       1.00      1.00      1.00         3
```

```
          18       1.00      1.00      1.00         5
          19       1.00      1.00      1.00         4
          21       0.00      0.00      0.00         1
          22       0.80      1.00      0.89         4
          23       1.00      1.00      1.00         2
          24       0.50      1.00      0.67         1
          25       1.00      0.60      0.75         5
          26       1.00      1.00      1.00         2
          27       1.00      1.00      1.00         3
          28       1.00      1.00      1.00         4
          29       0.67      1.00      0.80         2
          30       1.00      1.00      1.00         5
          31       1.00      1.00      1.00         4
          32       1.00      1.00      1.00         2
          33       1.00      1.00      1.00         4
          34       1.00      1.00      1.00         3
          35       1.00      1.00      1.00         3
          36       1.00      1.00      1.00         4
          37       1.00      1.00      1.00         3
          38       0.67      1.00      0.80         2
          39       1.00      1.00      1.00         1

    accuracy                           0.94       120
   macro avg       0.92      0.95      0.92       120
weighted avg       0.95      0.94      0.94       120

d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

討論：

使用 kernel="linear"：

- 取 10 個主成分時，準確率為 85.83%。
- 取 30 個主成分時，準確率為 93.33%。
- 取 50 個主成分時，準確率為 94.17%。

使用 kernel="rbf"：

- 取 10 個主成分時，準確率為 60.83%。
- 取 30 個主成分時，準確率為 83.33%。
- 取 50 個主成分時，準確率為 84.17%。

使用 kernel="poly"：

- 取 10 個主成分時，準確率為 85.83%。
- 取 30 個主成分時，準確率為 93.33%。
- 取 50 個主成分時，準確率為 94.17%。

小結：

- 使用 kernel="linear"和使用 kernel="poly"之準確率相同，使用 kernel="rbf"準確率最低。
- 取愈多主成分，準確率愈高。

(3)神經網路 (Neural Network)

(a)原始資料

1.使用 activation = 'logistic'且 hidden_layers = (30,)

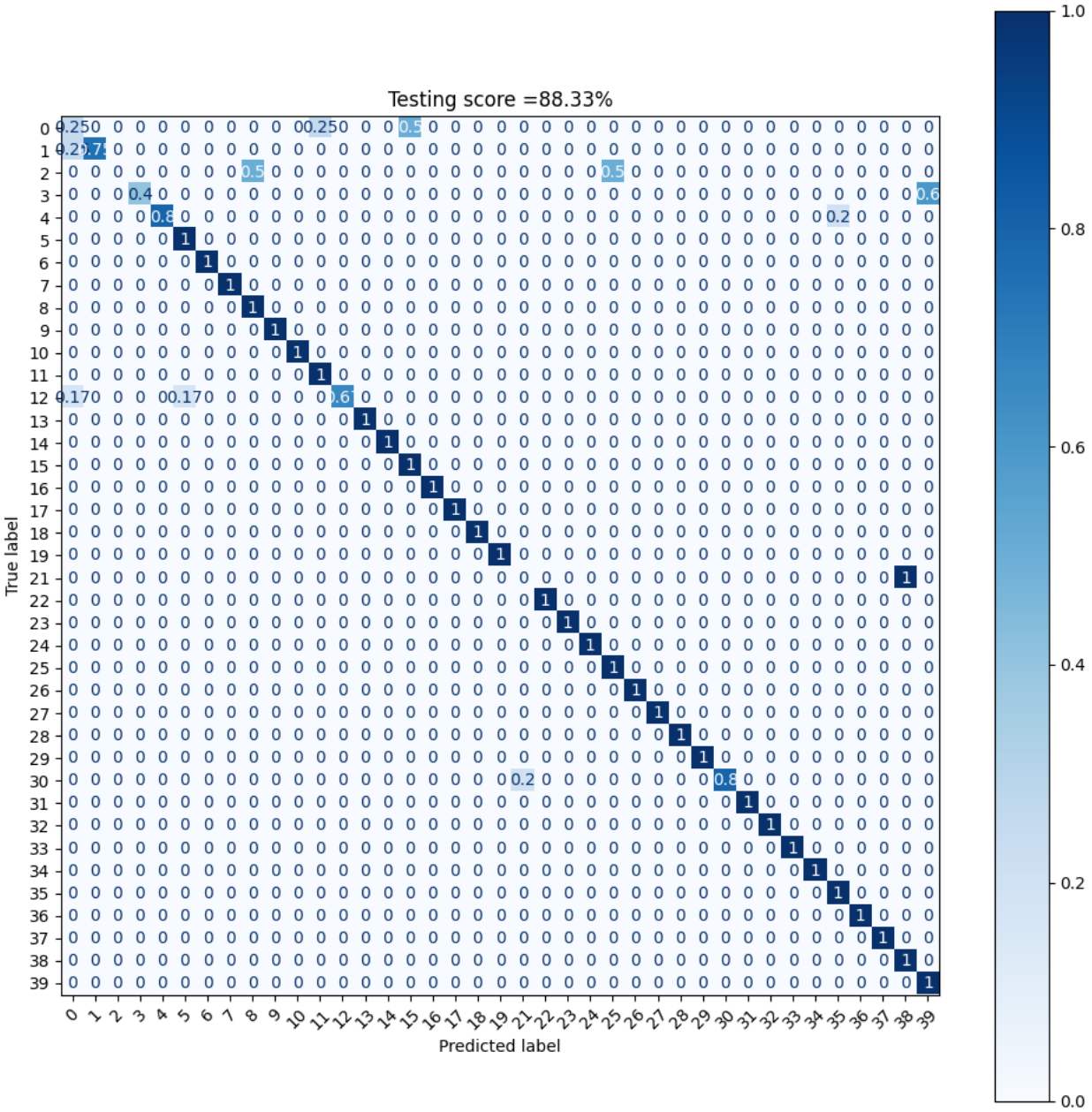
```
from sklearn.neural_network import MLPClassifier
# hidden_layers = (512,) # one hidden layer
# activation = 'relu' # the default
hidden_layers = (30,)
activation = 'logistic'
opts = dict(hidden_layer_sizes = hidden_layers , verbose = False, \
activation = activation, tol = 1e-6, max_iter = int(1e6))
# 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_, y_train)
predictions = clf_MLP.predict(X_test_)
print(f"{accuracy_score(y_test, predictions):.2%}\n")
print(classification_report(y_test, predictions))
```

```
88.33%

              precision    recall  f1-score   support

           0       0.33      0.25      0.29         4
           1       1.00      0.75      0.86         4
           2       0.00      0.00      0.00         2
           3       1.00      0.40      0.57         5
           4       1.00      0.80      0.89         5
           5       0.67      1.00      0.80         2
           6       1.00      1.00      1.00         4
           7       1.00      1.00      1.00         1
```

```
           8       0.75      1.00      0.86         3
           9       1.00      1.00      1.00         2
          10       1.00      1.00      1.00         6
          11       0.80      1.00      0.89         4
          12       1.00      0.67      0.80         6
          13       1.00      1.00      1.00         1
          14       1.00      1.00      1.00         2
          15       0.33      1.00      0.50         1
          16       1.00      1.00      1.00         1
          17       1.00      1.00      1.00         3
          18       1.00      1.00      1.00         5
          19       1.00      1.00      1.00         4
          21       0.00      0.00      0.00         1
          22       1.00      1.00      1.00         4
          23       1.00      1.00      1.00         2
          24       1.00      1.00      1.00         1
          25       0.83      1.00      0.91         5
          26       1.00      1.00      1.00         2
          27       1.00      1.00      1.00         3
          28       1.00      1.00      1.00         4
          29       1.00      1.00      1.00         2
          30       1.00      0.80      0.89         5
          31       1.00      1.00      1.00         4
          32       1.00      1.00      1.00         2
          33       1.00      1.00      1.00         4
          34       1.00      1.00      1.00         3
          35       0.75      1.00      0.86         3
          36       1.00      1.00      1.00         4
          37       1.00      1.00      1.00         3
          38       0.67      1.00      0.80         2
          39       0.25      1.00      0.40         1

    accuracy                           0.88       120
   macro avg       0.86      0.89      0.85       120
weighted avg       0.90      0.88      0.88       120


d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
```

```
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay
fig, ax = plt.subplots(1, 1, figsize=(12,12))
score = 100*clf_MLP.score(X_test_, y_test)
title = 'Testing score ={:.2f}%'.format(score)
disp = ConfusionMatrixDisplay.from_estimator(
clf_MLP,
X_test_,
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()
```



2.使用 activation = 'relu'且 hidden_layers = (512,)

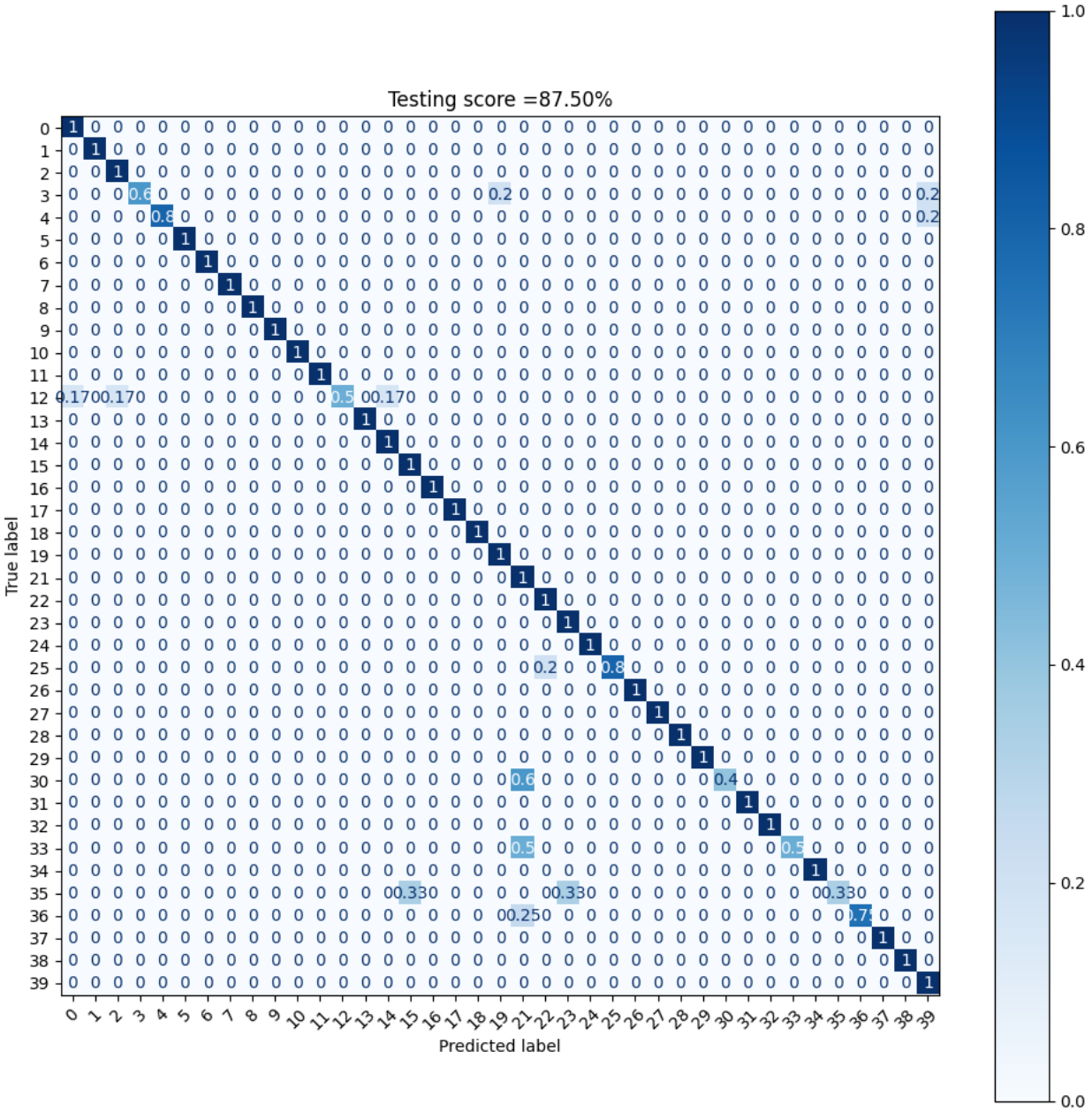
```
from sklearn.neural_network import MLPClassifier
hidden_layers = (512,) # one hidden layer
activation = 'relu' # the default
# hidden_layers = (30,)
# activation = 'logistic'
opts = dict(hidden_layer_sizes = hidden_layers , verbose = False, \
activation = activation, tol = 1e-6, max_iter = int(1e6))
# 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_, y_train)
predictions = clf_MLP.predict(X_test_)
print(f"{accuracy_score(y_test, predictions):.2%}\n")
print(classification_report(y_test, predictions))
```

```
87.50%

              precision    recall  f1-score   support

           0       0.80      1.00      0.89         4
           1       1.00      1.00      1.00         4
           2       0.67      1.00      0.80         2
           3       1.00      0.60      0.75         5
           4       1.00      0.80      0.89         5
           5       1.00      1.00      1.00         2
           6       1.00      1.00      1.00         4
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         3
           9       1.00      1.00      1.00         2
          10       1.00      1.00      1.00         6
          11       1.00      1.00      1.00         4
          12       1.00      0.50      0.67         6
          13       1.00      1.00      1.00         1
          14       0.67      1.00      0.80         2
          15       0.50      1.00      0.67         1
          16       1.00      1.00      1.00         1
          17       1.00      1.00      1.00         3
          18       1.00      1.00      1.00         5
          19       0.80      1.00      0.89         4
          21       0.14      1.00      0.25         1
          22       0.80      1.00      0.89         4
          23       0.67      1.00      0.80         2
          24       1.00      1.00      1.00         1
          25       1.00      0.80      0.89         5
          26       1.00      1.00      1.00         2
          27       1.00      1.00      1.00         3
          28       1.00      1.00      1.00         4
          29       1.00      1.00      1.00         2
          30       1.00      0.40      0.57         5
          31       1.00      1.00      1.00         4
          32       1.00      1.00      1.00         2
          33       1.00      0.50      0.67         4
          34       1.00      1.00      1.00         3
          35       1.00      0.33      0.50         3
          36       1.00      0.75      0.86         4
          37       1.00      1.00      1.00         3
          38       1.00      1.00      1.00         2
          39       0.33      1.00      0.50         1
```

```
    accuracy                           0.88       120
   macro avg       0.91      0.91      0.88       120
weighted avg       0.95      0.88      0.88       120
```

```
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay
fig, ax = plt.subplots(1, 1, figsize=(12,12))
score = 100*clf_MLP.score(X_test_, y_test)
title = 'Testing score ={:.2f}%'.format(score)
disp = ConfusionMatrixDisplay.from_estimator(
clf_MLP,
X_test_,
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()
```



討論：

- 使用 activation = 'logistic'且 hidden_layers = (30,)時，準確率為 88.33%。
- 使用 activation = 'relu'且 hidden_layers = (512,)時，準確率為 87.50%。
- 綜上所述，使用 activation = 'logistic'且 hidden_layers = (30,)之準確率較高。

(b)主成分資料

1.取 10 個主成分並使用 activation = 'logistic'且 hidden_layers = (30,)

```
from sklearn.decomposition import PCA
from sklearn.neural_network import MLPClassifier
```

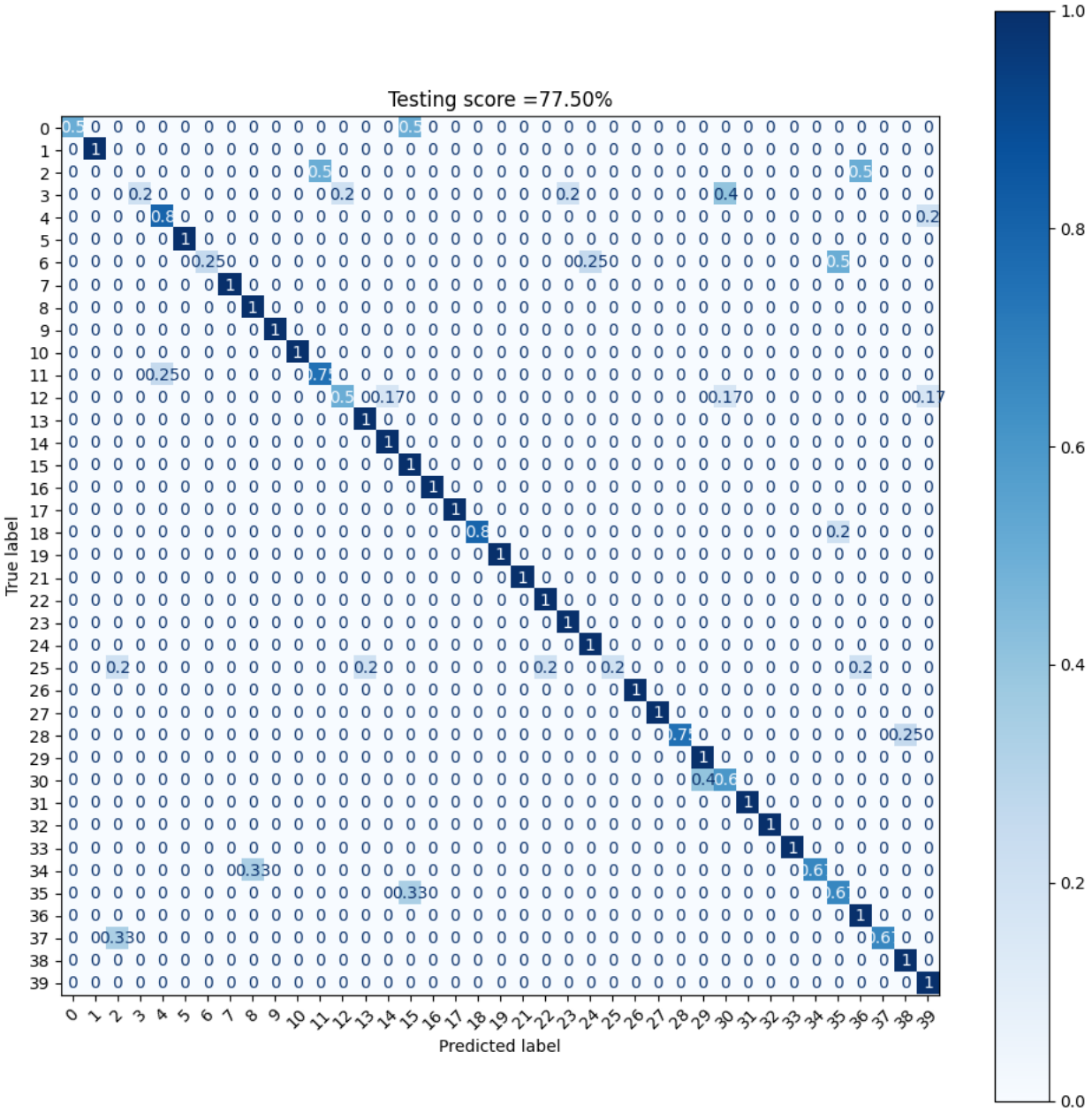
```
pca = PCA(n_components = 10).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)

hidden_layers = (30,)
activation = 'logistic'
opts = dict(hidden_layer_sizes = hidden_layers , verbose = False, \
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(Z_train, y_train)
predictions = clf_MLP.predict(Z_test)
print(f"{clf_MLP.score(Z_test, y_test):.2%}\n")
print(classification_report(y_test, predictions))
```

```
77.50%
```

```
              precision    recall  f1-score   support

           0       1.00      0.50      0.67         4
           1       1.00      1.00      1.00         4
           2       0.00      0.00      0.00         2
           3       1.00      0.20      0.33         5
           4       0.80      0.80      0.80         5
           5       1.00      1.00      1.00         2
           6       1.00      0.25      0.40         4
           7       1.00      1.00      1.00         1
           8       0.75      1.00      0.86         3
           9       1.00      1.00      1.00         2
          10       1.00      1.00      1.00         6
          11       0.75      0.75      0.75         4
          12       0.75      0.50      0.60         6
          13       0.50      1.00      0.67         1
          14       0.67      1.00      0.80         2
          15       0.25      1.00      0.40         1
          16       1.00      1.00      1.00         1
          17       1.00      1.00      1.00         3
          18       1.00      0.80      0.89         5
          19       1.00      1.00      1.00         4
          21       1.00      1.00      1.00         1
          22       0.80      1.00      0.89         4
          23       0.67      1.00      0.80         2
          24       0.50      1.00      0.67         1
          25       1.00      0.20      0.33         5
          26       1.00      1.00      1.00         2
          27       1.00      1.00      1.00         3
          28       1.00      0.75      0.86         4
```

```
           29       0.50      1.00      0.67         2
           30       0.50      0.60      0.55         5
           31       1.00      1.00      1.00         4
           32       1.00      1.00      1.00         2
           33       1.00      1.00      1.00         4
           34       1.00      0.67      0.80         3
           35       0.40      0.67      0.50         3
           36       0.67      1.00      0.80         4
           37       1.00      0.67      0.80         3
           38       0.67      1.00      0.80         2
           39       0.33      1.00      0.50         1

    accuracy                           0.78       120
   macro avg       0.81      0.83      0.77       120
weighted avg       0.85      0.78      0.76       120
```

```python
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay
fig, ax = plt.subplots(1, 1, figsize=(12,12))
score = 100*clf_MLP.score(Z_test, y_test)
title = 'Testing score ={:.2f}%'.format(score)
disp = ConfusionMatrixDisplay.from_estimator(
clf_MLP,
Z_test,
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()
```



2.取 30 個主成分並使用 activation = 'logistic'且 hidden_layers = (30,)

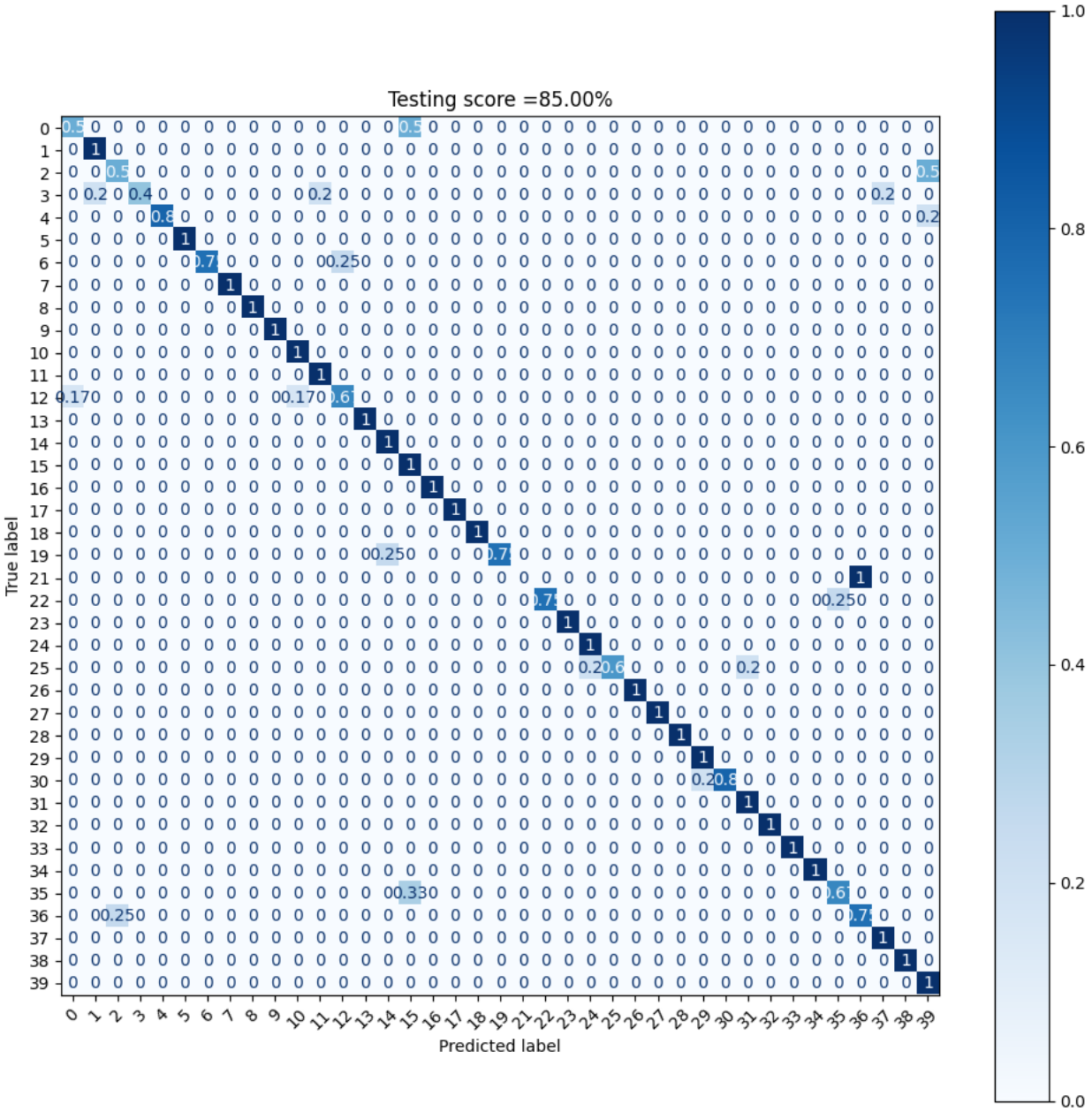
```
from sklearn.decomposition import PCA
from sklearn.neural_network import MLPClassifier

pca = PCA(n_components = 30).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)

hidden_layers = (30,)
activation = 'logistic'
opts = dict(hidden_layer_sizes = hidden_layers , verbose = False, \
```

```
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(Z_train, y_train)
predictions = clf_MLP.predict(Z_test)
print(f"{clf_MLP.score(Z_test, y_test):.2%}\n")
print(classification_report(y_test, predictions))
```

```
85.00%

              precision    recall  f1-score   support

           0       0.67      0.50      0.57         4
           1       0.80      1.00      0.89         4
           2       0.50      0.50      0.50         2
           3       1.00      0.40      0.57         5
           4       1.00      0.80      0.89         5
           5       1.00      1.00      1.00         2
           6       1.00      0.75      0.86         4
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         3
           9       1.00      1.00      1.00         2
          10       0.86      1.00      0.92         6
          11       0.80      1.00      0.89         4
          12       0.80      0.67      0.73         6
          13       1.00      1.00      1.00         1
          14       0.67      1.00      0.80         2
          15       0.25      1.00      0.40         1
          16       1.00      1.00      1.00         1
          17       1.00      1.00      1.00         3
          18       1.00      1.00      1.00         5
          19       1.00      0.75      0.86         4
          21       0.00      0.00      0.00         1
          22       1.00      0.75      0.86         4
          23       1.00      1.00      1.00         2
          24       0.50      1.00      0.67         1
          25       1.00      0.60      0.75         5
          26       1.00      1.00      1.00         2
          27       1.00      1.00      1.00         3
          28       1.00      1.00      1.00         4
          29       0.67      1.00      0.80         2
          30       1.00      0.80      0.89         5
          31       0.80      1.00      0.89         4
          32       1.00      1.00      1.00         2
          33       1.00      1.00      1.00         4
          34       1.00      1.00      1.00         3
          35       0.67      0.67      0.67         3
          36       0.75      0.75      0.75         4
```

```
          37       0.75      1.00      0.86         3
          38       1.00      1.00      1.00         2
          39       0.33      1.00      0.50         1

    accuracy                           0.85       120
   macro avg       0.84      0.87      0.83       120
weighted avg       0.89      0.85      0.85       120


d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```python
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay
fig, ax = plt.subplots(1, 1, figsize=(12,12))
score = 100*clf_MLP.score(Z_test, y_test)
title = 'Testing score ={:.2f}%'.format(score)
disp = ConfusionMatrixDisplay.from_estimator(
clf_MLP,
Z_test,
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()
```



3.取 50 個主成分並使用 activation = 'logistic'且 hidden_layers = (30,)

```
from sklearn.decomposition import PCA
from sklearn.neural_network import MLPClassifier

pca = PCA(n_components = 50).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)

hidden_layers = (30,)
activation = 'logistic'
opts = dict(hidden_layer_sizes = hidden_layers , verbose = False, \
```

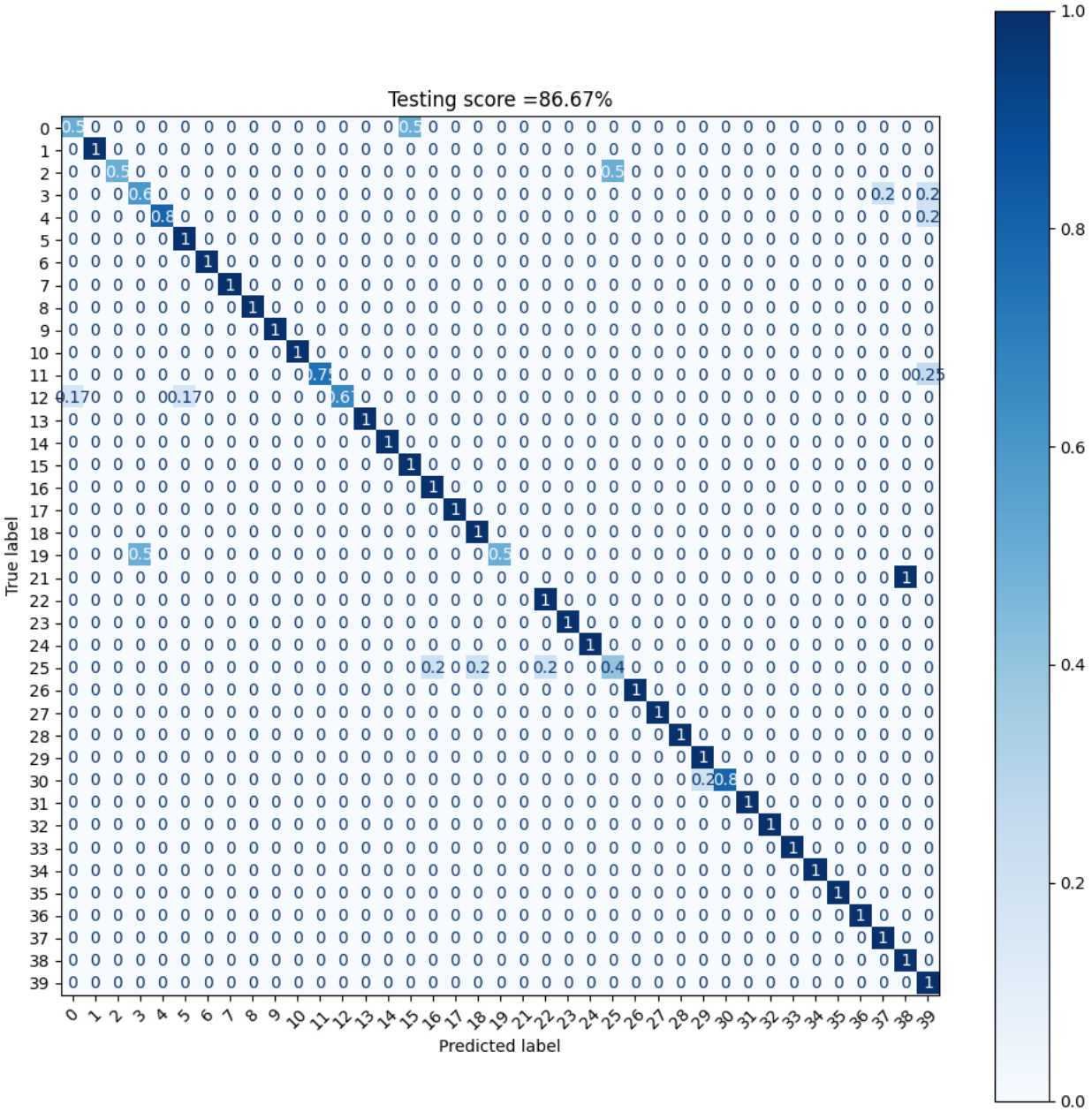
```
activation = activation, tol = 1e-6, max_iter = int(1e6))
# solver = 'sgd' # not efficient, need more tuning
# solver = 'lbfgs' # not suitable here
solver = 'adam' # default solver
clf_MLP = MLPClassifier(solver = solver, **opts)
clf_MLP.fit(Z_train, y_train)
predictions = clf_MLP.predict(Z_test)
print(f"{clf_MLP.score(Z_test, y_test):.2%}\n")
print(classification_report(y_test, predictions))
```

```
86.67%
```

```
              precision    recall  f1-score   support

           0       0.67      0.50      0.57         4
           1       1.00      1.00      1.00         4
           2       1.00      0.50      0.67         2
           3       0.60      0.60      0.60         5
           4       1.00      0.80      0.89         5
           5       0.67      1.00      0.80         2
           6       1.00      1.00      1.00         4
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         3
           9       1.00      1.00      1.00         2
          10       1.00      1.00      1.00         6
          11       1.00      0.75      0.86         4
          12       1.00      0.67      0.80         6
          13       1.00      1.00      1.00         1
          14       1.00      1.00      1.00         2
          15       0.33      1.00      0.50         1
          16       0.50      1.00      0.67         1
          17       1.00      1.00      1.00         3
          18       0.83      1.00      0.91         5
          19       1.00      0.50      0.67         4
          21       0.00      0.00      0.00         1
          22       0.80      1.00      0.89         4
          23       1.00      1.00      1.00         2
          24       1.00      1.00      1.00         1
          25       0.67      0.40      0.50         5
          26       1.00      1.00      1.00         2
          27       1.00      1.00      1.00         3
          28       1.00      1.00      1.00         4
          29       0.67      1.00      0.80         2
          30       1.00      0.80      0.89         5
          31       1.00      1.00      1.00         4
          32       1.00      1.00      1.00         2
          33       1.00      1.00      1.00         4
          34       1.00      1.00      1.00         3
          35       1.00      1.00      1.00         3
          36       1.00      1.00      1.00         4
```

```
          37       0.75      1.00      0.86         3
          38       0.67      1.00      0.80         2
          39       0.25      1.00      0.40         1

    accuracy                           0.87       120
   macro avg       0.86      0.89      0.85       120
weighted avg       0.90      0.87      0.87       120


d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```python
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay
fig, ax = plt.subplots(1, 1, figsize=(12,12))
score = 100*clf_MLP.score(Z_test, y_test)
title = 'Testing score ={:.2f}%'.format(score)
disp = ConfusionMatrixDisplay.from_estimator(
clf_MLP,
Z_test,
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()
```



4.取 10 個主成分並使用使用 activation = 'relu'且 hidden_layers = (512,)

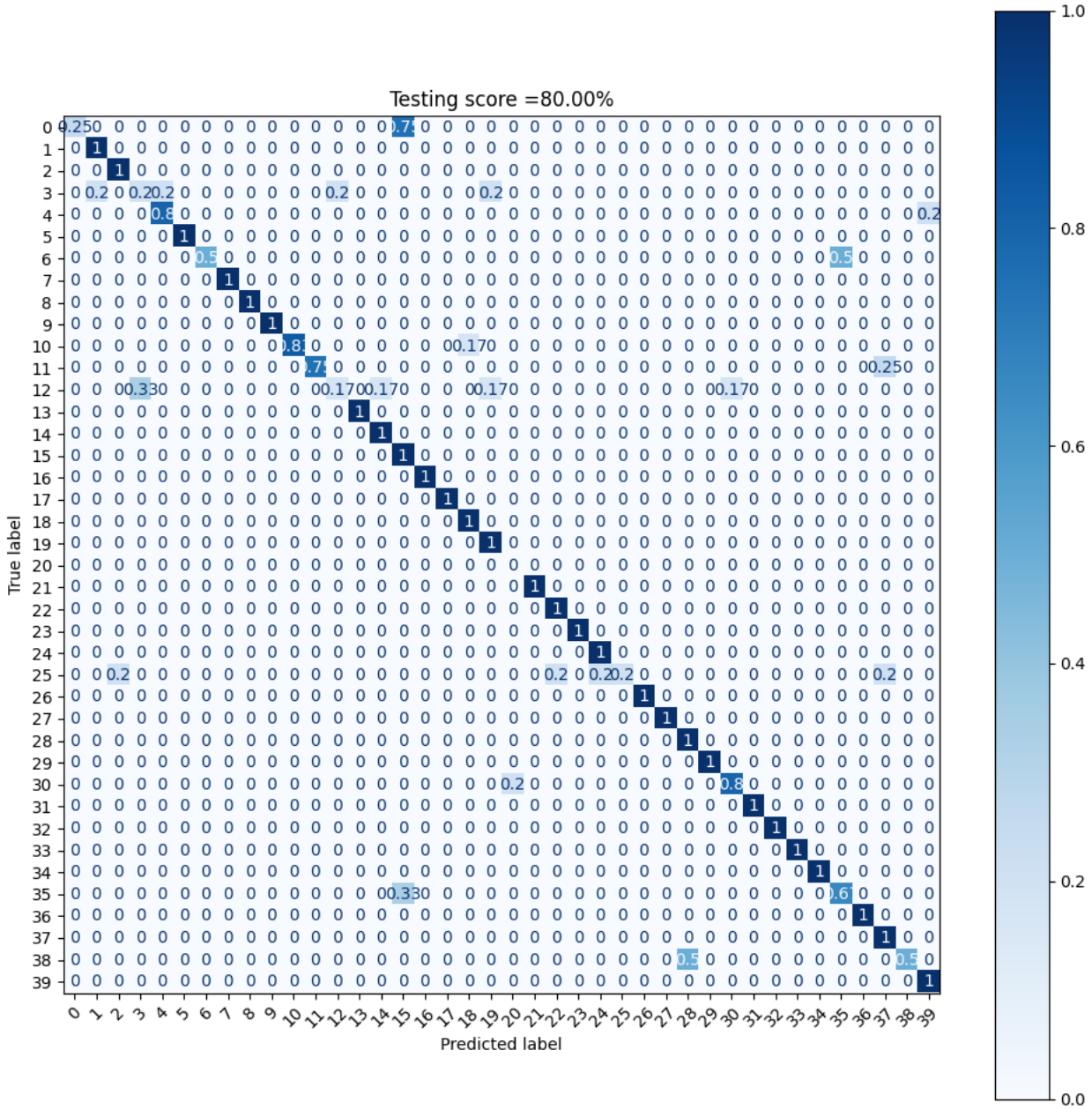
```
from sklearn.decomposition import PCA
from sklearn.neural_network import MLPClassifier

pca = PCA(n_components = 10).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)

hidden_layers = (512,)
activation = 'relu'
opts = dict(hidden_layer_sizes = hidden_layers , verbose = False, \
```

```
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(Z_train, y_train)
predictions = clf_MLP.predict(Z_test)
print(f"{clf_MLP.score(Z_test, y_test):.2%}\n")
print(classification_report(y_test, predictions))
```

```
80.00%

              precision    recall  f1-score   support

           0       1.00      0.25      0.40         4
           1       0.80      1.00      0.89         4
           2       0.67      1.00      0.80         2
           3       0.33      0.20      0.25         5
           4       0.80      0.80      0.80         5
           5       1.00      1.00      1.00         2
           6       1.00      0.50      0.67         4
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         3
           9       1.00      1.00      1.00         2
          10       1.00      0.83      0.91         6
          11       1.00      0.75      0.86         4
          12       0.50      0.17      0.25         6
          13       1.00      1.00      1.00         1
          14       0.67      1.00      0.80         2
          15       0.20      1.00      0.33         1
          16       1.00      1.00      1.00         1
          17       1.00      1.00      1.00         3
          18       0.83      1.00      0.91         5
          19       0.67      1.00      0.80         4
          20       0.00      0.00      0.00         0
          21       1.00      1.00      1.00         1
          22       0.80      1.00      0.89         4
          23       1.00      1.00      1.00         2
          24       0.50      1.00      0.67         1
          25       1.00      0.20      0.33         5
          26       1.00      1.00      1.00         2
          27       1.00      1.00      1.00         3
          28       0.80      1.00      0.89         4
          29       1.00      1.00      1.00         2
          30       0.80      0.80      0.80         5
          31       1.00      1.00      1.00         4
          32       1.00      1.00      1.00         2
          33       1.00      1.00      1.00         4
          34       1.00      1.00      1.00         3
          35       0.50      0.67      0.57         3
```

```
           36       1.00      1.00      1.00         4
           37       0.60      1.00      0.75         3
           38       1.00      0.50      0.67         2
           39       0.50      1.00      0.67         1

    accuracy                          0.80       120
   macro avg       0.82      0.84      0.80       120
weighted avg       0.84      0.80      0.78       120


d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay
fig, ax = plt.subplots(1, 1, figsize=(12,12))
score = 100*clf_MLP.score(Z_test, y_test)
title = 'Testing score ={:.2f}%'.format(score)
disp = ConfusionMatrixDisplay.from_estimator(
clf_MLP,
Z_test,
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()
```



5.取 30 個主成分並使用使用 activation = 'relu'且 hidden_layers = (512,)

```
from sklearn.decomposition import PCA
from sklearn.neural_network import MLPClassifier

pca = PCA(n_components = 30).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)

hidden_layers = (512,)
activation = 'relu'
opts = dict(hidden_layer_sizes = hidden_layers , verbose = False, \
```

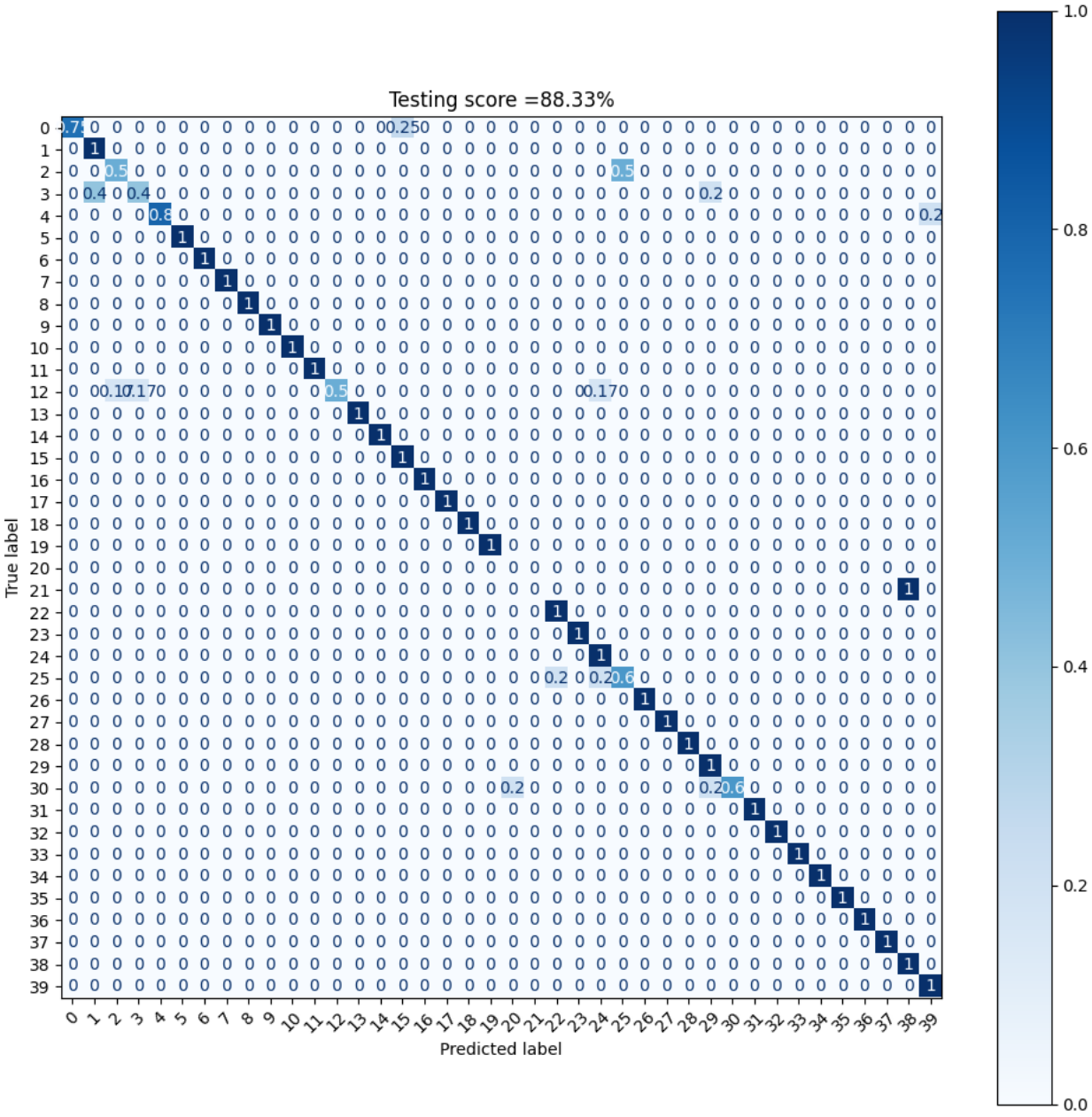
```
activation = activation, tol = 1e-6, max_iter = int(1e6))
# solver = 'sgd' # not efficient, need more tuning
# solver = 'lbfgs' # not suitable here
solver = 'adam' # default solver
clf_MLP = MLPClassifier(solver = solver, **opts)
clf_MLP.fit(Z_train, y_train)
predictions = clf_MLP.predict(Z_test)
print(f"{clf_MLP.score(Z_test, y_test):.2%}\n")
print(classification_report(y_test, predictions))
```

```
88.33%

              precision    recall  f1-score   support

           0       1.00      0.75      0.86         4
           1       0.67      1.00      0.80         4
           2       0.50      0.50      0.50         2
           3       0.67      0.40      0.50         5
           4       1.00      0.80      0.89         5
           5       1.00      1.00      1.00         2
           6       1.00      1.00      1.00         4
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         3
           9       1.00      1.00      1.00         2
          10       1.00      1.00      1.00         6
          11       1.00      1.00      1.00         4
          12       1.00      0.50      0.67         6
          13       1.00      1.00      1.00         1
          14       1.00      1.00      1.00         2
          15       0.50      1.00      0.67         1
          16       1.00      1.00      1.00         1
          17       1.00      1.00      1.00         3
          18       1.00      1.00      1.00         5
          19       1.00      1.00      1.00         4
          20       0.00      0.00      0.00         0
          21       0.00      0.00      0.00         1
          22       0.80      1.00      0.89         4
          23       1.00      1.00      1.00         2
          24       0.33      1.00      0.50         1
          25       0.75      0.60      0.67         5
          26       1.00      1.00      1.00         2
          27       1.00      1.00      1.00         3
          28       1.00      1.00      1.00         4
          29       0.50      1.00      0.67         2
          30       1.00      0.60      0.75         5
          31       1.00      1.00      1.00         4
          32       1.00      1.00      1.00         2
          33       1.00      1.00      1.00         4
          34       1.00      1.00      1.00         3
          35       1.00      1.00      1.00         3
```

```
          36       1.00      1.00      1.00         4
          37       1.00      1.00      1.00         3
          38       0.67      1.00      0.80         2
          39       0.50      1.00      0.67         1

    accuracy                           0.88       120
   macro avg       0.85      0.88      0.85       120
weighted avg       0.91      0.88      0.88       120


d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Recall and F-score
are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```python
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay
fig, ax = plt.subplots(1, 1, figsize=(12,12))
score = 100*clf_MLP.score(Z_test, y_test)
title = 'Testing score ={:.2f}%'.format(score)
disp = ConfusionMatrixDisplay.from_estimator(
clf_MLP,
Z_test,
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()
```



6.取 50 個主成分並使用使用 activation = 'relu'且 hidden_layers = (512,)

```
from sklearn.decomposition import PCA
from sklearn.neural_network import MLPClassifier

pca = PCA(n_components = 50).fit(X_train_)
Z_train = pca.transform(X_train_)
Z_test = pca.transform(X_test_)

hidden_layers = (512,)
activation = 'relu'
opts = dict(hidden_layer_sizes = hidden_layers , verbose = False, \
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(Z_train, y_train)
predictions = clf_MLP.predict(Z_test)
print(f"{clf_MLP.score(Z_test, y_test):.2%}\n")
print(classification_report(y_test, predictions))
```

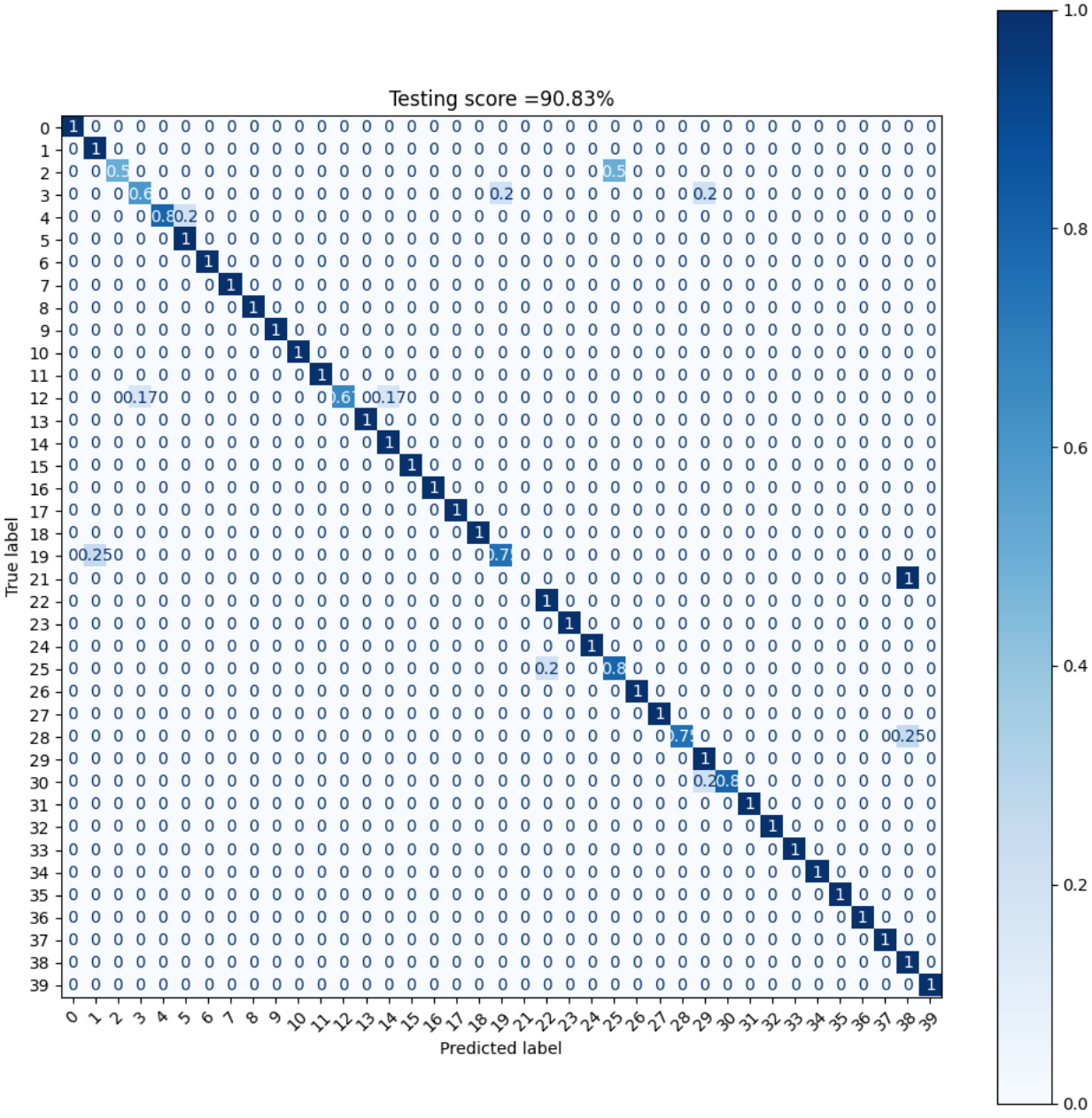
```
90.83%
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	4
1	0.80	1.00	0.89	4
2	1.00	0.50	0.67	2
3	0.75	0.60	0.67	5
4	1.00	0.80	0.89	5
5	0.67	1.00	0.80	2
6	1.00	1.00	1.00	4
7	1.00	1.00	1.00	1
8	1.00	1.00	1.00	3
9	1.00	1.00	1.00	2
10	1.00	1.00	1.00	6
11	1.00	1.00	1.00	4
12	1.00	0.67	0.80	6
13	1.00	1.00	1.00	1
14	0.67	1.00	0.80	2
15	1.00	1.00	1.00	1
16	1.00	1.00	1.00	1
17	1.00	1.00	1.00	3
18	1.00	1.00	1.00	5
19	0.75	0.75	0.75	4
21	0.00	0.00	0.00	1
22	0.80	1.00	0.89	4
23	1.00	1.00	1.00	2
24	1.00	1.00	1.00	1
25	0.80	0.80	0.80	5
26	1.00	1.00	1.00	2

```
          27       1.00      1.00      1.00         3
          28       1.00      0.75      0.86         4
          29       0.50      1.00      0.67         2
          30       1.00      0.80      0.89         5
          31       1.00      1.00      1.00         4
          32       1.00      1.00      1.00         2
          33       1.00      1.00      1.00         4
          34       1.00      1.00      1.00         3
          35       1.00      1.00      1.00         3
          36       1.00      1.00      1.00         4
          37       1.00      1.00      1.00         3
          38       0.50      1.00      0.67         2
          39       1.00      1.00      1.00         1

    accuracy                           0.91       120
   macro avg       0.90      0.91      0.90       120
weighted avg       0.92      0.91      0.91       120


d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
d:\vs\venv_name\lib\site-packages\sklearn\metrics\
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```python
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay
fig, ax = plt.subplots(1, 1, figsize=(12,12))
score = 100*clf_MLP.score(Z_test, y_test)
title = 'Testing score ={:.2f}%'.format(score)
disp = ConfusionMatrixDisplay.from_estimator(
clf_MLP,
Z_test,
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()
```



討論：

使用 activation = 'logistic'且 hidden_layers = (30,)：

- 取 10 個主成分時，準確率為 77.50%。
- 取 30 個主成分時，準確率為 85.00%。
- 取 50 個主成分時，準確率為 86.67%。

使用 activation = 'relu'且 hidden_layers = (512,)：

- 取 10 個主成分時，準確率為 80.00%。
- 取 30 個主成分時，準確率為 88.33%。
- 取 50 個主成分時，準確率為 90.83%。

小結：

- 使用 activation = 'relu'和 hidden_layers = (512,)時，準確率較高。
- 取愈多主成分，準確率愈高。

總結：

依照準確率比較：

- 多元羅吉斯回歸 (Multinomial Logistic Regression) (1)使用 lbfgs 演算法時，需要取 30 個主成分，準確率才會和原始資料相同。 (2)使用 liblinear 演算法時，需要取 30 個主成分，準確率才會比原始資料高。 (3)使用 newton-cg 演算法時，需要取 50 個主成分，準確率才會比原始資料高。 (4)使用 newton-cg 演算法且取 50 個主成分有最高的準確率 95.00%。

- 支援向量機 (Support Vector Machine) (1)使用 kernel='linear'時，需要取 30 個主成分，準確率才會比原始資料高。 (2)使用 kernel='rbf'時，需要取 50 個主成分，準確率才會和原始資料相同。 (3)使用 kernel='poly'時，只需要取 10 個主成分，準確率就比原始資料高。 (4)取 50 個主成分且使用 kernel='linear'(或 kernel='poly')有最高的準確率 94.17%。

- 神經網路 (Neural Network) (1)在原始資料中，使用 activation = 'logistic'且 hidden_layers = (30,)之準確率較使用 activation = 'relu'且 hidden_layers = (512,)高；但主成分資料卻相反。 (2)在主成分資料中，無論是何種 activation 和 hidden_layers，取愈多主成分，準確率愈高。 (3)原始資料和使用 activation = 'relu'且 hidden_layers = (512,)且取 50 個主成分有最高的準確率 90.83%。

綜上所述：

- 我認為最佳分類器為多元羅吉斯回歸中取 50 個主成分且使用 newton-cg 演算法'，因為它的準確率為所有分類器中最高(95.00%)。
- 由於資料樣本數較小，因此無論何種分類器執行時間都很短。