# 淺度機器學習作品三:分類器的原理與評比實驗

學號:411072054 姓名:黃暐宸

# 作品目標:

用三種分類器分別對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)

#### 目標:

- 1. 比較三種分類器在原始資料和主成分資料上的表現,找出在各種情況下表現最好的分類器和參數。
- 2. 機器學習最基本的概念是使機器能對多變量資料進行分類,在找表現最好的分類器 和參數的過程中,初步了解機器學習到底在做甚麼事情。
- 3. 透過對不同大小的資料集進行分類學習與測試,了解資料量對機器學習造成的影響。

# 分類器介紹

- 1. 多元羅吉斯回歸 (Multinomial Logistic Regression)是一種用於處理多類別問題的監督學習算法。它使用羅吉斯函數來估計一個觀察值屬於某一類別的概率。最後,模型將觀察值分類到概率最高的類別。
- 2. 支援向量機 (Support Vector Machine):是一種二元分類器,其目標是找到一個超平面來最大化兩個類別之間的邊界。對於非線性問題,它可以使用核函數將資料映射到一個更高維度的空間,使得資料在這個空間中是線性可分的。
- 3. 神經網路 (Neural Network)是一種模仿人腦神經元工作方式的模型,由多個層次的節點(或稱為"神經元")組成。每個節點將前一層的輸出進行加權總和,然後通過一個非線性函數(如 ReLU 或 sigmoid)來產生自己的輸出。透過反向傳播和梯度下降等方法來學習權重。

# (一)準備資料: AT&T 人臉資料。這組影像資料共有 40 人(類),每人 10 張影像,每張影像大小為 64 × 64。

#### 程式碼說明:

- 1. 訓練資料與測試資料必須分開標準化,而非標準化後再分成訓練與測試資料,將測試資料 規劃為 25%。
- 2. 對訓練集和測試集進行標準化處理

```
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("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=0.25)
# Standardize data
scaler = StandardScaler()
X train = scaler.fit transform(X train)
X test = scaler.fit transform(X test)
num samples = df.shape[0]
print(f"Number of samples: {num samples}")
Number of samples: 400
```

# (二)用 logistic regression 分類

(1)以標準化後之原始資料的訓練資料學習,並以測試資料測試準確率 程式碼說明:

- 1. opts = dict(tol = 1e-6, max\_iter = int(1e6), verbose=1)創建了一個名為 opts 的字典,設定 模型參數(tol: 容忍值, max\_iter: 最大迭代次數, verbose: 是否顯示訓練過程)。
- 2. 分別設定不同的模型演算法(lbfgs、liblinear、newton-cg)。
- 訓練模型,並使用訓練好的模型來預測測試數據。
- 4. 回報測試資料對於訓練完成的分類器的分類準確率,以兩種不同方式呈現,其中 accuracy\_score 比對了測試資料的標籤(y\_test)與分類預測值(y\_pred),而 clf\_original.score 直接給出準確率。兩者結果是一樣的。
- 5. 最後給出較完整的報告(classification\_report)。

#### 演算法(lbfgs)

```
from sklearn.linear_model import LogisticRegression from sklearn.metrics import accuracy_score, classification_report def train_LR(solver, X_train, y_train, X_test, y_test):
    opts = dict(tol = le-6, max_iter = int(le6), verbose=1)    clf_LR = LogisticRegression(solver = solver, **opts) # 建立模型    clf_LR.fit(X_train, y_train) # 訓練模型    y_pred = clf_LR.predict(X_test) # 預測測試資料 # 測試資料之準確率回報    print(f"{accuracy_score(y_test, y_pred):.2%}\n")
```

```
print(f"{clf_LR.score(X_test, y_test):.2%}\n")
    print(classification_report(y_test, y_pred))
# 使用函數訓練模型
train_LR("lbfgs", X_train_, y_train, X_test_, y_test)
93.33%
93.33%
                precision
                              recall f1-score
                                                    support
                                 0.60
                                                           5
            0
                     1.00
                                            0.75
            1
                     1.00
                                 1.00
                                            1.00
                                                           6
            2
                                                           2
                     0.67
                                 1.00
                                            0.80
            3
                                                           3
                     1.00
                                 0.67
                                            0.80
            4
                     1.00
                                 0.75
                                            0.86
                                                           4
            5
                                                           3
                     1.00
                                 1.00
                                            1.00
            6
                                                           5
                     1.00
                                 1.00
                                            1.00
                                                           2
            7
                                 0.50
                     1.00
                                            0.67
                                                           3
            8
                     1.00
                                 1.00
                                            1.00
                                                           3
            9
                     1.00
                                 0.67
                                            0.80
                                                           3
           10
                     1.00
                                 1.00
                                            1.00
                                                           4
           11
                     1.00
                                 1.00
                                            1.00
                                                           3
           12
                     0.67
                                 0.67
                                            0.67
           13
                                                           3
                     1.00
                                 1.00
                                            1.00
                                                           1
           14
                     0.50
                                 1.00
                                            0.67
                                                           4
           15
                     1.00
                                 1.00
                                            1.00
                                                           3
           16
                     1.00
                                 1.00
                                            1.00
                                                           1
           17
                     1.00
                                 1.00
                                            1.00
                                                           2
           18
                     1.00
                                 1.00
                                            1.00
           19
                     1.00
                                 1.00
                                            1.00
                                                           1
           20
                     1.00
                                 1.00
                                            1.00
                                                           3
                                                           3
           21
                     1.00
                                 1.00
                                            1.00
                                                           3
           22
                     1.00
                                 1.00
                                            1.00
           23
                                                           5
                     1.00
                                 0.80
                                            0.89
                                                           2
           24
                     1.00
                                 1.00
                                            1.00
                                                           3
2
           25
                     1.00
                                 1.00
                                            1.00
           26
                     1.00
                                 1.00
                                            1.00
                                                           3
           27
                     1.00
                                 1.00
                                            1.00
                                                           3
           28
                     1.00
                                 1.00
                                            1.00
                                                           3
           29
                     1.00
                                 1.00
                                            1.00
           30
                                            1.00
                                                           1
                     1.00
                                 1.00
           31
                                 1.00
                                            1.00
                                                           4
                     1.00
           32
                     1.00
                                 1.00
                                            1.00
                                                           4
                                                           2
           33
                     1.00
                                 1.00
                                            1.00
                                                           5
           34
                     1.00
                                 1.00
                                            1.00
                                                           4
           35
                     1.00
                                 1.00
                                            1.00
                                                           2
           36
                     1.00
                                 1.00
                                            1.00
```

37

1.00

1.00

1.00

38	0.67	1.00	0.80	2
39	0.43	1.00	0.60	3
accuracy macro avg weighted avg	0.95 0.96	0.94 0.93	0.93 0.93 0.94	120 120 120

[LibLinear]91.67%

91.67%

	precision	recall	f1-score	support
	preezszen	10000	11 30010	Suppor c
0	1.00	0.40	0.57	5
1	1.00	0.83	0.91	6
2	1.00	1.00	1.00	2 3
3	1.00	0.67	0.80	3
4	1.00	0.75	0.86	4
5	1.00	1.00	1.00	3
6 7	1.00	1.00	1.00	ე ე
8	0.40 1.00	1.00 1.00	0.57	2
9	1.00	1.00	1.00 1.00	3
10	1.00	1.00	1.00	3
11	1.00	1.00	1.00	4
12	1.00	0.67	0.80	3
13	1.00	1.00	1.00	4 3 5 2 3 3 4 3 3
14	0.50	1.00	0.67	1
15	1.00	1.00	1.00	4
16	0.75	1.00	0.86	3 1
17	1.00	1.00	1.00	
18	0.50	1.00	0.67	2
19	1.00	1.00	1.00	1
20	1.00	0.67	0.80	3
21	1.00	1.00	1.00	3
22 23	$1.00 \\ 1.00$	1.00 0.80	1.00	5
24	1.00	1.00	0.89 1.00	2
25	1.00	1.00	1.00	3
26	1.00	1.00	1.00	2
27	1.00	1.00	1.00	2 1 3 3 5 2 3 2 3 3 1
28	1.00	1.00	1.00	3
29	1.00	1.00	1.00	3
30	0.50	1.00	0.67	
31	1.00	1.00	1.00	4
32	1.00	1.00	1.00	4
33	1.00	1.00	1.00	2
34	1.00	1.00	1.00	5
35	0.75	0.75	0.75	4 2
36	1.00	1.00	1.00	2

37 38	1.00 0.67	1.00 1.00	1.00 0.80	2 2
39	1.00	1.00	1.00	3
accuracy			0.92	120
macro avg weighted avg	0.93 0.95	0.94 0.92	0.92 0.92	120 120
93.33%				
93.33%				
	precision	recall	f1-score	support
0	1.00	0.60	0.75	5
1 2	1.00 0.67	$1.00 \\ 1.00$	1.00 0.80	6 2
3	1.00	0.67	0.80	3
4 5	$1.00 \\ 1.00$	0.75 1.00	0.86 1.00	4
6	1.00	1.00	1.00	
7	1.00	0.50	0.67	5 2
8	$1.00 \\ 1.00$	1.00 0.67	1.00 0.80	3 3 3
10	1.00	1.00	1.00	3
11	1.00	1.00	1.00	4
12	0.67	0.67	0.67	3
13 14	1.00 0.50	$1.00 \\ 1.00$	1.00 0.67	1
15	1.00	1.00	1.00	4
16	1.00	1.00	1.00	3
17 18	$1.00 \\ 1.00$	$1.00 \\ 1.00$	1.00 1.00	1 2
19	1.00	1.00	1.00	1
20	1.00	1.00	1.00	
21 22	$1.00 \\ 1.00$	$1.00 \\ 1.00$	1.00 1.00	3
23	1.00	0.80	0.89	3 3 5 2 3 2
24	1.00	1.00	1.00	2
25	1.00	1.00	1.00	3
26 27	$1.00 \\ 1.00$	$1.00 \\ 1.00$	1.00 1.00	2
28	1.00	1.00	1.00	3
29	1.00	1.00	1.00	3 3 3 1
30 31	$1.00 \\ 1.00$	$1.00 \\ 1.00$	1.00 1.00	1 4
32	1.00	1.00	1.00	4
33	1.00	1.00	1.00	2
34 35	$1.00 \\ 1.00$	$1.00 \\ 1.00$	1.00 1.00	5 4
23	1.00	1.00	1.00	4

36	1.00	1.00	1.00	2
37	1.00	1.00	1.00	2
38	0.67	1.00	0.80	2
39	0.43	1.00	0.60	3
accuracy macro avg weighted avg	0.95 0.96	0.94 0.93	0.93 0.93 0.94	120 120 120

# sklearn 分類報告的項目說明:

- 1. Accuracy:模型預測正確數量所佔整體的比例。
- 2. Precision:精確率,被預測為 Positive 的資料中,有多少是真的 Positive。
- 3. Recall: 召回率,它是原本是 Positive 的資料,它能夠召回多少,也就是說在原本 Positive 的資料中被預測出多少。
- 4. F1-score: Precision 與 Recall 調和平均數,模型越接近1,模型越好。
- 5. support:告訴測試資料集中有多少項目屬於每個類別。
- 6. macro avg: 對每個類別的 precision、recall、f1-score 加起來求平均。
- 7. weighted avg : 按照 support 的權重,對每個類別的 precision、recall、f1-score 加起來求平均。

#### 結果說明:

- 1. 測試資料之準確率 93.33%。
- 2. 在加權平均的評估指標中,模型的精確率、召回率和 F1 分數分別為 0.96、0.93、0.94, 這表示模型在每個類別的預測性能還不錯。

train\_LR("liblinear", X\_train\_, y\_train, X\_test\_, y\_test)

[LibLinear]91.67%

#### 91.67%

	precision	recall	f1-score	support
0	1.00	0.40	0.57	5
1	1.00	0.83	0.91	6
2	1.00	1.00	1.00	2
3	1.00	0.67	0.80	3
4	1.00	0.75	0.86	4
5	1.00	1.00	1.00	3
6	1.00	1.00	1.00	5
7	0.40	1.00	0.57	2
8	1.00	1.00	1.00	3
9	1.00	1.00	1.00	3
10	1.00	1.00	1.00	3
11	1.00	1.00	1.00	4
12	1.00	0.67	0.80	3
13	1.00	1.00	1.00	3

14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36	0.50 1.00 0.75 1.00 0.50 1.00 1.00 1.00 1.00 1.00 1.0	1.00 1.00 1.00 1.00 1.00 0.67 1.00 0.80 1.00 1.00 1.00 1.00 1.00 1.00	0.67 1.00 0.86 1.00 0.67 1.00 0.89 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.0	1 4 3 1 2 1 3 3 5 2 3 2 3 3 3 1 4 4 4 2 5 4 2	
				4	
accuracy macro avg weighted avg	0.93 0.95	0.94 0.92	0.92 0.92 0.92	120 120 120	

- 1. 測試資料之準確率 91.67%。
- 2. 在加權平均的評估指標中,模型的精確率、召回率和 F1 分數分別為 0.95、0.92、0.92, 這表示模型在每個類別的預測性能還不錯。

```
train_LR("newton-cg", X_train_, y_train, X_test_, y_test)
93.33%
```

### 93.33%

	precision	recall	f1-score	support
0	1.00	0.60	0.75	5
1	1.00	1.00	1.00	6
2	0.67	1.00	0.80	2
3	1.00	0.67	0.80	3

	4	1.00	0.75	0.86	4	
	5	1.00	1.00	1.00	3	
	6	1.00	1.00	1.00	5	
	7	1.00	0.50	0.67	5 2 3 3	
	8	1.00	1.00	1.00	3	
	9	1.00	0.67	0.80	3	
	10	1.00	1.00	1.00	3	
	11	1.00	1.00	1.00	4	
	12	0.67	0.67	0.67	3	
	13	1.00	1.00	1.00	3 1	
	14	0.50	1.00	0.67		
	15	1.00	1.00	1.00	4	
	16	1.00	1.00	1.00	3 1	
	17	1.00	1.00	1.00		
	18	1.00	1.00	1.00	2	
	19	1.00	1.00	1.00	1	
	20	1.00	1.00	1.00	3	
	21	1.00	1.00	1.00	3 3 5 2 3	
	22	1.00	1.00	1.00	3	
	23	1.00	0.80	0.89	5	
	24	1.00	1.00	1.00	2	
	25	1.00	1.00	1.00	3	
	26	1.00	1.00	1.00	2	
	27	1.00	1.00	1.00	3	
	28	1.00	1.00	1.00	3 3	
	29	1.00	1.00	1.00	3	
	30	1.00	1.00	1.00	1	
	31	1.00	1.00	1.00	4	
	32	1.00	1.00	1.00	4	
	33 34	1.00	$1.00 \\ 1.00$	1.00 1.00	2 5	
	35	1.00 1.00	1.00	1.00	4	
	36	1.00	1.00	1.00	2	
	30 37	1.00	1.00	1.00	2	
	38	0.67	1.00	0.80	_	
	39	0.43	1.00	0.60	2	
	39	0.40	1.00	0.00	5	
accu	racv			0.93	120	
macro	_	0.95	0.94	0.93	120	
weighted		0.96	0.93	0.94	120	
<b>J</b>	3					

- 1. 測試資料之準確率 93.33%。
- 2. 在加權平均的評估指標中,模型的精確率、召回率和 F1 分數分別為 0.96、0.93、0.94, 這表示模型在每個類別的預測性能還不錯。

# 使用網格搜索(GridSearchCV)來尋找最佳的參數組合

#### 程式碼說明:

- 1. 設定參數網格
- 2. 進行網格搜索(GridSearchCV)
- 3. 輸出最佳參數:網格搜索完成後,輸出找到的最佳參數組合
- 4. 預測並評估模型:使用訓練好的模型對測試數據進行預測,並輸出模型在測試數據上的準確度和分類報告

```
from sklearn.model selection import GridSearchCV
from sklearn.neural network import MLPClassifier
param grid = [
    {
        'solver': ['newton-cg', 'lbfgs', 'liblinear'],
        'tol': [1e-3, 1e-4, 1e-5, 1e-6],
        'max iter': [int(1e3), int(1e4), int(1e5), int(1e6)]
    }
1
clf LR = LogisticRegression()
clf LR gs = GridSearchCV(clf LR, param grid, verbose=1, cv=3, n jobs=-
1)
clf LR gs.fit(X train , y train)
print("Best Parameters: ", clf LR gs.best params )
y pred = clf LR gs.predict(X test )
print(f"{accuracy score(y test, y pred):.2%}\n")
print(f"{clf LR gs.score(X test , y test):.2%}\n")
print(classification report(y test, y pred))
Fitting 3 folds for each of 48 candidates, totalling 144 fits
Best Parameters: {'max iter': 1000, 'solver': 'newton-cg', 'tol':
0.001
97.00%
97.00%
              precision
                           recall f1-score
                                               support
                                                     2
           0
                   1.00
                             1.00
                                        1.00
                                                     3
           1
                   1.00
                             1.00
                                        1.00
           2
                                                     3
                   0.75
                             1.00
                                        0.86
           3
                                                     3
                   1.00
                             1.00
                                        1.00
           4
                             0.80
                                        0.89
                                                     5
                   1.00
           5
                   0.67
                             1.00
                                        0.80
                                                     2
```

6	1.00	1.00	1.00	1	
7	1.00	1.00	1.00	3	
8	1.00	1.00	1.00	2	
9	1.00	0.67	0.80		
10	1.00	1.00	1.00	3	
11	1.00	1.00	1.00	3 3 2	
12	1.00	1.00	1.00	2	
13	1.00	1.00	1.00	4	
14	1.00	1.00	1.00	2	
15	1.00	1.00	1.00	4	
16	1.00	1.00	1.00	1	
17	1.00	1.00	1.00	3	
18	1.00	1.00	1.00	2	
19	1.00	1.00	1.00	3	
20	1.00	1.00	1.00	2	
21	1.00	1.00	1.00	2	
22	1.00	1.00	1.00	5	
23	1.00	1.00	1.00	3	
24	1.00	1.00	1.00	2	
25	1.00	0.67	0.80	3	
26	1.00	1.00	1.00	2	
28	1.00	1.00	1.00	3	
29	1.00	1.00	1.00	2	
30	1.00	1.00	1.00	4	
33	1.00	1.00	1.00	2	
34	1.00	1.00	1.00	2	
35	1.00	1.00	1.00	1	
36	1.00	1.00	1.00	4	
37	1.00	1.00	1.00	4	
38	1.00	1.00	1.00	3	
39	0.75	1.00	0.86	3	
		-			
accuracy			0.97	100	
macro avg	0.98	0.98	0.97	100	
weighted avg	0.98	0.97	0.97	100	

三種演算法表現最好的是 newton-cg , 參數設定是('max\_iter': 1000, 'solver': 'newton-cg', 'tol': 0.001) , 測試資料之準確率達到 97%。

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.io

D = scipy.io.loadmat("allFaces.mat")
X = D["faces"].T # 32256 x 2410, each column represents an image
y = np.ndarray.flatten(D["nfaces"])
```

```
m = D["m"].item() # 168
n = D["n"].item() # 192
n_persons = D["person"].item() # 38

# prepare y labels
y_labels = np.repeat(np.arange(len(y)), y)
```

# (2)以標準化後之原始資料的主成分之訓練資料學習,並以測試資料測試準確率

#### 程式碼說明:

- 1. 使用 PCA 對訓練數據 X\_train\_ 進行擬合,並將其降維到 45 個主成分。
- 2. 使用訓練好的 PCA 模型將訓練數據和測試數據轉換到新的低維空間,得到 Z\_train 和 Z\_test。
- 3. 設定並訓練羅吉斯迴歸模型。
- 4. 使用模型預測測試數據並計算準確率。
- 5. 輸出模型的分類報告。

#### 演算法(lbfgs)

```
from sklearn.decomposition import PCA
pca = PCA(n components = 45).fit(X train)
Z_{train} = \overline{pca.transform}(X train)
Z test = pca.transform(X test )
opts = dict(tol = 1e-6, max iter = int(1e6), verbose=1)
solver = "lbfgs"
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")
print(classification report(y test, y pred))
93.33%
              precision
                            recall f1-score
                                                support
                    1.00
                              0.60
                                         0.75
                                                       5
           0
           1
                    1.00
                              1.00
                                         1.00
                                                       6
           2
                              1.00
                                         0.80
                                                       2
                    0.67
           3
                                                       3
                    1.00
                                         0.80
                              0.67
           4
                    1.00
                              1.00
                                         1.00
                                                       4
           5
                                                       3
                              1.00
                    1.00
                                         1.00
           6
                                                       5
                    1.00
                              1.00
                                         1.00
           7
                    1.00
                                                       2
                              0.50
                                         0.67
           8
                                                       3
                    1.00
                              1.00
                                         1.00
           9
                    1.00
                              0.67
                                         0.80
                                                       3
          10
                    1.00
                              1.00
                                         1.00
                                                       3
          11
                    1.00
                              1.00
                                         1.00
                                                       4
```

12 13	1.00	0.67 1.00	0.57 1.00	3	
14		1.00	0.67	1	
15		1.00	1.00	4	
16		1.00	1.00	3	
17		1.00	1.00	1	
18		1.00	1.00	2	
19		1.00	1.00	1	
20		1.00	1.00	3 3	
21		1.00	1.00	3	
22		1.00	1.00	3	
23 24		0.80 1.00	0.89 1.00	5 2	
24 25		1.00	1.00	2	
26		1.00	1.00	3 2	
27		1.00	1.00	3	
28		1.00	1.00	3	
29		1.00	0.86	3 3	
30		1.00	1.00	1	
31		1.00	1.00	4	
32		1.00	1.00	4	
33		1.00	1.00	2	
34		1.00	1.00	5	
35	1.00	0.75	0.86	4	
36	1.00	1.00	1.00	2	
37	1.00	1.00	1.00	2	
38		1.00	0.80	2	
39	0.60	1.00	0.75	3	
			0.00	100	
accuracy		0.04	0.93	120	
macro avg		0.94	0.93	120	
weighted avg	0.96	0.93	0.93	120	

- 1. 測試資料之準確率 93.33%。
- 2. 在加權平均的評估指標中,精確率、召回率和 F1 分數分別為 0.96、 0.93、 0.93,這表示考慮到每個類別的樣本數量後,模型的整體性能仍然非常好。

#### 演算法(liblinear)

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 45).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"
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")
print(classification_report(y_test, y_pred))
```

[LibLinear]91.67%

[				
	precision	recall	f1-score	support
0 1 2 3 4 5 6	1.00 1.00 0.67 1.00 1.00 1.00	0.60 1.00 1.00 0.67 1.00 1.00 0.80	0.75 1.00 0.80 0.80 1.00 1.00 0.89	5 6 2 3 4 3 5
7 8 9 10 11 12 13	0.50 1.00 1.00 1.00 0.80 0.67 1.00	0.50 1.00 0.67 1.00 1.00 0.67 1.00	0.50 1.00 0.80 1.00 0.89 0.67 1.00	3 4 3 5 2 3 3 4 3 3
14 15 16 17 18 19 20 21	1.00 1.00 0.33 1.00 1.00 1.00	1.00 1.00 1.00 1.00 1.00 1.00 1.00	1.00 1.00 1.00 0.50 1.00 1.00	4 3 1
22 23 24 25 26 27 28	1.00 1.00 1.00 1.00 1.00 1.00	1.00 0.80 1.00 1.00 1.00 1.00	1.00 0.89 1.00 1.00 1.00 1.00	2 1 3 3 5 2 3 2 3 3 1
29 30 31 32 33 34 35	0.75 1.00 1.00 1.00 1.00 1.00	1.00 1.00 1.00 1.00 1.00 1.00	0.86 1.00 1.00 1.00 1.00 0.86	3 1 4 2 5 4 2 2 2
36 37 38 39	1.00 0.67 1.00 0.50	1.00 1.00 1.00 0.67	1.00 0.80 1.00 0.57	2 2 3

macro avg 0.92 0.93 0.91 120 ighted avg 0.94 0.92 0.92 120
--

- 1. 測試資料之準確率 91.67%。
- 2. 在加權平均的評估指標中,精確率、召回率和 F1 分數分別為 0.94、0.92、0.92,這表示考慮到每個類別的樣本數量後,模型的整體性能仍然非常好。

#### 演算法(newton-cg)

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 45).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"
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")
print(classification_report(y_test, y_pred))
```

#### 94.17%

		precision	recall	f1-score	support
	_	1 00	0.00	0.75	_
	0	1.00	0.60	0.75	5
	1	1.00	1.00	1.00	6
	2	0.67	1.00	0.80	2
	3	1.00	0.67	0.80	3
	4	1.00	1.00	1.00	4
	5	1.00	1.00	1.00	3 5
	6	1.00	1.00	1.00	5
	7	1.00	0.50	0.67	2
	8	1.00	1.00	1.00	3
	9	1.00	0.67	0.80	2 3 3 4 3 3 1
1	L0	1.00	1.00	1.00	3
1	l1	1.00	1.00	1.00	4
1	<b>L</b> 2	0.50	0.67	0.57	3
1	L3	1.00	1.00	1.00	3
1	L4	0.50	1.00	0.67	1
1	L5	1.00	1.00	1.00	4
1	L6	1.00	1.00	1.00	
1	L7	1.00	1.00	1.00	3 1
1	18	1.00	1.00	1.00	2
	L9	1.00	1.00	1.00	1
	20	1.00	1.00	1.00	3

	21 22 23	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00	3 3 5
	24	1.00	1.00	1.00	2
	25 26	1.00 1.00	1.00 1.00	1.00 1.00	3 2
	27	1.00	1.00	1.00	3
	28	1.00	1.00	1.00	3 3 3
	29	0.75	1.00	0.86	
	30 31	1.00 1.00	1.00 1.00	1.00 1.00	1 4
	32	1.00	1.00	1.00	4
	33	1.00	1.00	1.00	2
	34 35	1.00 1.00	1.00 0.75	1.00 0.86	5 4
	36	1.00	1.00	1.00	2
	37	1.00	1.00	1.00	2
	38	1.00	1.00	1.00	2
	39	0.60	1.00	0.75	3
accur	-			0.94	120
macro a	_	0.95 0.96	0.95 0.94	0.94 0.94	120 120
weighted (	avy	0.90	0.34	0.54	120

- 1. 測試資料之準確率 94.17%。
- 2. 在加權平均的評估指標中,精確率、召回率和 F1 分數分別為 0.96、0.94、0.94,這表示考慮到每個類別的樣本數量後,模型的整體性能仍然非常好。

# 使用網格搜索(GridSearchCV)來尋找最佳的參數組合

#### 程式碼說明:

- 1. 設定參數網格
- 2. 進行網格搜索(GridSearchCV)
- 3. 輸出最佳參數:網格搜索完成後,輸出找到的最佳參數組合
- 4. 預測並評估模型:使用訓練好的模型對測試數據進行預測,並輸出模型在測試數據上的準確度和分類報告

```
from sklearn.model_selection import GridSearchCV from sklearn.linear_model import LogisticRegression from sklearn.decomposition import PCA from sklearn.metrics import accuracy_score, classification_report # 定義要搜索的參數 param_grid = {
    'solver': ['newton-cg', 'lbfgs', 'liblinear'],
    'tol': [1e-4, 1e-5, 1e-6],
```

```
'max iter': [ int(1e4), int(1e5), int(1e6)]
}
# 使用 PCA 降維
pca = PCA(n components = 45).fit(X train)
Z train = pca.transform(X train )
Z test = pca.transform(X test )
# 建立模型
clf PCA = LogisticRegression()
# 建立 GridSearchCV
clf PCA gs = GridSearchCV(clf PCA, param grid, verbose=1, cv=3,
n jobs=-1
# 訓練模型
clf_PCA_gs.fit(Z_train, y_train)
# 打印最佳參數
print("Best Parameters: ", clf_PCA_gs.best_params_)
# 預測測試資料
y pred = clf PCA gs.predict(Z test)
# 測試資料之準確率回報
print(f"{accuracy score(y test, y pred):.2%}\n")
print(f"{clf PCA gs.score(Z test, y test):.2%}\n")
print(classification report(y test, y pred))
Fitting 3 folds for each of 27 candidates, totalling 81 fits
Best Parameters: {'max_iter': 10000, 'solver': 'liblinear', 'tol':
0.0001
95.00%
95.00%
              precision
                           recall f1-score
                                              support
                                                     2
           0
                   1.00
                             1.00
                                       1.00
           1
                   1.00
                             1.00
                                       1.00
                                                     3
           2
                   0.75
                             1.00
                                                     3
                                       0.86
                                                     3
           3
                             0.67
                   1.00
                                       0.80
           4
                                                     5
                   1.00
                             0.80
                                       0.89
           5
                                                     2
                   1.00
                             1.00
                                       1.00
           6
                   1.00
                             1.00
                                       1.00
                                                     1
                                                     3
           7
                   1.00
                             1.00
                                       1.00
                                                     2
           8
                   1.00
                             1.00
                                       1.00
           9
                                                     3
                   1.00
                             0.67
                                       0.80
                   1.00
                             1.00
                                       1.00
                                                     3
          10
          11
                   1.00
                             1.00
                                       1.00
                                                     2
```

12						
14       0.67       1.00       0.80       2         15       1.00       1.00       1.00       4         16       1.00       1.00       1.00       1         17       1.00       1.00       1.00       3         18       1.00       1.00       1.00       2         19       1.00       1.00       1.00       3         20       1.00       1.00       1.00       2         21       1.00       1.00       1.00       2         22       1.00       0.80       0.89       5         23       1.00       1.00       1.00       3         24       1.00       1.00       1.00       2         25       1.00       0.67       0.80       3         26       1.00       1.00       1.00       2         28       1.00       1.00       1.00       2         30       1.00       1.00       1.00       2         34       1.00       1.00       1.00       2         34       1.00       1.00       1.00       2         35       0.50       1.00       0.67 <td< td=""><td></td><td></td><td></td><td></td><td></td><td></td></td<>						
15						
16					0.80	
17						
19		16	1.00	1.00	1.00	1
19		17	1.00	1.00	1.00	3
21		18	1.00	1.00	1.00	2
21		19	1.00	1.00	1.00	3
22		20	1.00	1.00	1.00	
25		21	1.00	1.00	1.00	
25		22	1.00	0.80	0.89	5
25		23	1.00	1.00	1.00	3
28		24	1.00	1.00	1.00	2
28		25	1.00	0.67	0.80	3
30					1.00	2
30		28	1.00	1.00	1.00	3
33 1.00 1.00 1.00 2 34 1.00 1.00 1.00 2 35 0.50 1.00 0.67 1 36 1.00 1.00 1.00 4 37 0.80 1.00 0.89 4 38 1.00 1.00 1.00 3 39 0.75 1.00 0.86 3  accuracy 0.95 100 macro avg 0.96 0.96 0.95 100		29	1.00	1.00	1.00	2
34 1.00 1.00 1.00 2 35 0.50 1.00 0.67 1 36 1.00 1.00 1.00 4 37 0.80 1.00 0.89 4 38 1.00 1.00 1.00 3 39 0.75 1.00 0.86 3  accuracy 0.95 100 macro avg 0.96 0.96 0.95 100		30	1.00	1.00	1.00	
35 0.50 1.00 0.67 1 36 1.00 1.00 1.00 4 37 0.80 1.00 0.89 4 38 1.00 1.00 1.00 3 39 0.75 1.00 0.86 3  accuracy 0.95 100 macro avg 0.96 0.96 0.95 100		33	1.00	1.00	1.00	2
36 1.00 1.00 1.00 4 37 0.80 1.00 0.89 4 38 1.00 1.00 1.00 3 39 0.75 1.00 0.86 3  accuracy 0.95 100 macro avg 0.96 0.96 0.95 100		34	1.00	1.00	1.00	
37 0.80 1.00 0.89 4 38 1.00 1.00 1.00 3 39 0.75 1.00 0.86 3  accuracy 0.95 100 macro avg 0.96 0.96 0.95 100		35	0.50	1.00	0.67	
38 1.00 1.00 1.00 3 39 0.75 1.00 0.86 3  accuracy 0.95 100 macro avg 0.96 0.96 0.95 100		36		1.00	1.00	
39 0.75 1.00 0.86 3  accuracy 0.95 100 macro avg 0.96 0.96 0.95 100		37	0.80	1.00	0.89	
accuracy 0.95 100 macro avg 0.96 0.96 0.95 100		38	1.00		1.00	3
macro avg 0.96 0.96 0.95 100		39	0.75	1.00	0.86	3
macro avg 0.96 0.96 0.95 100						
		_				
weighted avg 0.97 0.95 0.95 100						
	weighted	avg	0.97	0.95	0.95	100

三種演算法表現最好的是 liblinear, 參數設定是('max\_iter': 10000, 'solver': 'liblinear', 'tol': 0.0001), 測試資料之準確率達到 95%。

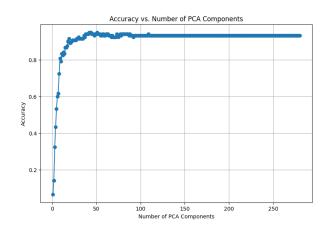
將主成分逐漸調高(直到 281),觀察準確率的變化,並畫一張折線圖來比較(與 scree plot 對比)。

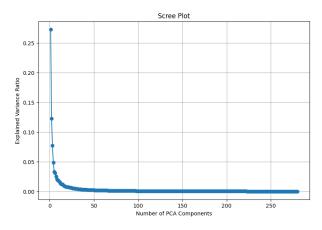
選用 lbfqs 演算法

```
import matplotlib.pyplot as plt

# Initialize an empty list to store accuracy for each number of components
accuracy_scores = []
solver = "lbfgs"
# Loop over number of components from 1 to 13
for i in range(1, 281):
```

```
pca = PCA(n components=i).fit(X_train_)
    Z train = pca.transform(X train )
    Z test = pca.transform(X test )
    clf PCA = LogisticRegression(solver=solver, **opts)
    clf PCA.fit(Z train, y train)
    y_pred = clf_PCA.predict(Z_test)
    accuracy = clf PCA.score(Z test, y test)
    accuracy scores.append(accuracy)
# Create a figure and a grid of subplots
fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))
# Plot the accuracy scores on the first subplot
ax1.plot(range(1, 281), accuracy scores, marker='o', linestyle='-')
ax1.set title('Accuracy vs. Number of PCA Components')
ax1.set xlabel('Number of PCA Components')
ax1.set ylabel('Accuracy')
ax1.grid(True)
# Fit PCA on the training data
pca = PCA().fit(X train )
# Calculate explained variance ratio for each component
explained variance ratio = pca.explained variance ratio
# Plot the explained variance ratio on the second subplot
ax2.plot(range(1, 281), explained variance ratio, marker='o',
linestyle='-')
ax2.set title('Scree Plot')
ax2.set xlabel('Number of PCA Components')
ax2.set ylabel('Explained Variance Ratio')
ax2.grid(True)
# Display the figure with the two subplots
plt.show()
```





- 1. 從 Scree Plot 來看,將主成分逐漸調高,Explained Variance Ratio 隨之降低。
- 2. 從左圖來看,只採用第一個主成分時,測試資料之準確率大約等於10%,隨著主成分逐漸調高,測試資料之準確率皆維持在大約等於90幾%。
- 3. 以上說明採用越多主成分,Explained Variance Ratio 隨之降低,測試資料之準確率隨之提升。

# (三)SVM 分群

(1)以標準化後之原始資料的訓練資料學習,並以測試資料測試準確率

kernel(linear)

```
from sklearn.metrics import accuracy score, classification report
from sklearn.svm import SVC
from sklearn.svm import LinearSVC
def train and evaluate(clf, X train, X test, y train, y test):
    # Fit the classifier with the data
    clf.fit(X train, y train)
    # Predict the labels of the test data
    predictions = clf.predict(X_test)
    # Calculate and print the accuracy score
    acc_score = accuracy_score(y_test, predictions)
    print(f"Accuracy Score: {acc score:.2%}")
    # Print the classification report
    print(classification report(y test, predictions, zero division=0))
# Define the SVM classifier
C = 1 # SVM regularization parameter
opts = dict(C = C, tol = 1e-6, max iter = int(1e6))
```

#### kernel(linear)

```
clf svm = SVC(kernel="linear", **opts)
train and evaluate(clf svm, X train, X test, y train, y test)
Accuracy Score: 94.17%
              precision
                            recall f1-score
                                               support
                              0.60
                                                     5
           0
                   1.00
                                        0.75
           1
                   1.00
                              1.00
                                        1.00
                                                     6
           2
                                        1.00
                                                     2
                   1.00
                              1.00
           3
                                                     3
                   1.00
                                        0.80
                              0.67
           4
                   1.00
                              0.75
                                        0.86
                                                     4
                                        1.00
           5
                                                     3
                   1.00
                             1.00
                   1.00
                              1.00
                                        1.00
                                                     5
```

7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35	1.00 1.00 1.00 1.00 0.67 1.00 0.50 1.00 1.00 1.00 1.00 1.00 1.00	0.50 1.00 0.67 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.0	0.67 1.00 0.80 1.00 1.00 0.67 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.0	2 3 3 4 3 1 4 3 1 2 1 3 3 5 2 3 3 3 1 4 4 2 5 4 4 2 5 4 4 4 2 5 4 4 4 4 4 4 4	
34	0.83	1.00	0.91	5	
accuracy macro avg weighted avg	0.95 0.96	0.95 0.94	0.94 0.94 0.94	120 120 120	

- 1. 測試資料之準確率 94.17%。
- 2. 在加權平均的評估指標中,精確率、召回率和 F1 分數分別為 0.96、0.94、0.94,這表示考慮到每個類別的樣本數量後,模型的整體性能仍然非常好。

#### kernel(rbf)

```
clf_svm = SVC(kernel="rbf", gamma=0.2, **opts)
train_and_evaluate(clf_svm, X_train, X_test, y_train, y_test)
```

A c c u c c c :	C a a :-	o. 2 E00			
Accuracy	Scor		naca11	fl ccoro	cuppont
		precision	recatt	f1-score	support
	۵	0.00	0 00	0.00	5
	0 1		0.00		
		0.00	0.00	0.00	6
	2	0.00	0.00	0.00	2
	3	0.00	0.00	0.00	3
	4	1.00	0.25	0.40	4
	5	0.00	0.00	0.00	3
	6	0.00	0.00	0.00	5
	7	0.00	0.00	0.00	2
	8	0.00	0.00	0.00	3
	9	0.00	0.00	0.00	3 3 3
	10	0.00	0.00	0.00	
	11	0.00	0.00	0.00	4
	12	0.00	0.00	0.00	3 3
	13	0.00	0.00	0.00	
	14	0.00	0.00	0.00	1
	15	0.00	0.00	0.00	4
	16	0.00	0.00	0.00	3
	17	1.00	1.00	1.00	1
	18	0.00	0.00	0.00	2
	19	0.01	1.00	0.02	1
	20	0.00	0.00	0.00	
	21	0.00	0.00	0.00	3 3
	22	0.00	0.00	0.00	3
	23	0.00	0.00	0.00	5
	24	0.00	0.00	0.00	2
	25	0.00	0.00	0.00	3
	26	0.00	0.00	0.00	2
	27	0.00	0.00	0.00	3
	28	0.00	0.00	0.00	3 3
	29	0.00	0.00	0.00	3
	30	0.00	0.00	0.00	1
	31	0.00	0.00	0.00	4
	32	0.00	0.00	0.00	4
	33	0.00	0.00	0.00	2
	34	0.00	0.00	0.00	5
	35	0.00	0.00	0.00	4
	36	0.00	0.00	0.00	2
	30 37	0.00	0.00	0.00	2
	38	0.00	0.00	0.00	2
	39	0.00	0.00	0.00	3
	29	0.00	שטיט	0.00	3
accur	2CV			0.03	120
accur macro	-	0.05	0.06	0.03	120
weighted	_	0.04	0.03	0.04	120
weighted	avy	0.04	0.03	0.02	120

1. 測試資料之準確率 2.5%。

37

0.00

0.00

0.00

2. 在加權平均的評估指標中,精確率、召回率和 F1 分數分別為 0.04、0.03、0.02,這表示考慮到每個類別的樣本數量後,模型的整體性能。

#### kernel(poly)

clf\_svm = SVC(kernel="poly", degree=3, gamma="auto", \*\*opts) train\_and\_evaluate(clf\_svm, X\_train, X\_test, y\_train, y\_test) Accuracy Score: 3.33% recall f1-score precision support 0 0.00 0.00 0.00 5 0.00 1 0.00 0.00 6 2 2 0.00 0.00 0.00 3 3 0.00 0.00 0.00 4 0.00 0.00 0.00 4 5 3 0.00 0.00 0.00 6 5 0.00 0.00 0.00 2 7 0.00 0.00 0.00 8 0.00 0.00 3 0.00 3 9 0.00 0.00 0.00 3 10 0.00 0.00 0.00 4 11 0.00 0.00 0.00 3 12 0.00 0.00 0.00 3 13 0.00 0.00 0.00 1 14 0.02 1.00 0.05 15 0.00 0.00 0.00 4 3 16 0.00 0.00 0.00 1 17 0.04 1.00 0.08 2 18 0.00 0.00 0.00 1 0.14 1.00 19 0.25 20 0.00 0.00 0.00 3 3 21 0.00 0.00 0.00 3 22 0.00 0.00 0.00 5 23 0.00 0.00 0.00 2 24 0.00 0.00 0.00 25 3 0.00 0.00 0.00 2 26 0.00 0.00 0.00 27 0.00 0.00 0.00 3 3 28 0.00 0.00 0.00 3 29 0.00 0.00 0.00 1 30 0.02 1.00 0.04 4 31 0.00 0.00 0.00 32 0.00 0.00 0.00 4 2 33 0.00 0.00 0.00 0.00 0.00 5 34 0.00 4 35 0.00 0.00 0.00 2 0.00 0.00 36 0.00

2

	38 39	0.00 0.00	0.00 0.00	0.00 0.00	2
accu macro weighted	avg	0.01 0.00	0.10 0.03	0.03 0.01 0.00	120 120 120

- 1. 測試資料之準確率 3.33%。
- 2. 在加權平均的評估指標中,精確率、召回率和 F1 分數分別為 0.00、0.03、0.0,這表示考慮到每個類別的樣本數量後,模型的整體性能。

kernel(LinearSVC : "one-vs-rest")

clf\_svm = LinearSVC(dual=False, \*\*opts) # one vs the rest
train and evaluate(clf sym X train X test v train v test)

train_and	l_eva	luate(clf_svm,	X_train	, X_test,	y_train,	y_test)
Accuracy	Score	e: 92.50%				
		precision	recall	f1-score	support	
	0	1.00	0.60	0.75	5	
	1	1.00	1.00	1.00	6	
	2	0.67	1.00	0.80	2	
	3	1.00	0.67	0.80	2	
	4	1.00	0.75	0.86	4	
	5	1.00	1.00	1.00	3	
	6	1.00	1.00	1.00	5	
	7	0.50	0.50	0.50	2	
	8 9	1.00	1.00	1.00	3	
	10	1.00 1.00	0.67 1.00	0.80 1.00	5 2 3 3 3	
	11	1.00	1.00	1.00	4	
	12	1.00	0.67	0.80	3	
	13	1.00	1.00	1.00	3	
	14	1.00	1.00	1.00		
	15	1.00	1.00	1.00	4	
	16	1.00	1.00	1.00	3 1	
	17 18	1.00	1.00	1.00	2	
	19	1.00 1.00	$1.00 \\ 1.00$	1.00 1.00	1	
	20	1.00	1.00	1.00		
	21	1.00	1.00	1.00	3 3 3	
	22	1.00	1.00	1.00	3	
	23	1.00	0.80	0.89	5 2	
	24	1.00	1.00	1.00	2	
	25	1.00	1.00	1.00	3	
	26 27	1.00 1.00	$1.00 \\ 1.00$	$1.00 \\ 1.00$	2	
	21	1.00	1.00	1.00	3	

28 29 30	1.00 0.75 1.00	1.00 1.00 1.00	1.00 0.86 1.00	3 3 1	
31 32	1.00 1.00	1.00	1.00	4	
33 34	1.00 1.00	1.00	1.00	2	
35 36	0.75 1.00	0.75 1.00	0.75 1.00	4 2	
37 38	0.67 1.00	1.00	0.80 1.00	2	
39	0.43	1.00	0.60	3	
accuracy macro avg weighted avg	0.94 0.95	0.93 0.93	0.93 0.93 0.93	120 120 120	

- 1. 測試資料之準確率 92.50%。
- 2. 在加權平均的評估指標中,精確率、召回率和 F1 分數分別為 0.95、0.93、0.93,這表示考慮到每個類別的樣本數量後,模型的整體性能仍然非常好。

# 使用網格搜索(GridSearchCV)來尋找最佳的參數組合

#### 程式碼說明:

- 1. 設定參數網格
- 2. 進行網格搜索(GridSearchCV)
- 3. 輸出最佳參數:網格搜索完成後,輸出找到的最佳參數組合
- 4. 預測並評估模型:使用訓練好的模型對測試數據進行預測,並輸出模型在測試數據上的準確度和分類報告

```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC, LinearSVC
from sklearn.metrics import accuracy_score, classification_report

# Define the parameter grid for GridSearchCV
param_grid = {
    'tol': [le-4, le-5, le-6],
    'max_iter': [int(le4), int(le5), int(le6)]
}

# Define the classifiers
classifiers = {
    'SVC_linear': SVC(kernel='linear', max_iter=int(le6)),
    'SVC_rbf': SVC(kernel='rbf', max_iter=int(le6)),
    'SVC_poly': SVC(kernel='poly', max_iter=int(le6)),
    'LinearSVC': LinearSVC(max_iter=int(le6), dual=False)
```

```
}
# Initialize the best accuracy and classifier
best accuracy = 0
best classifier = None
best params = None
best report = None
# For each classifier
for name, clf in classifiers.items():
    # Define the GridSearchCV
    clf gs = GridSearchCV(clf, param grid, verbose=1, cv=3, n jobs=-1)
    # Train the model
    clf gs.fit(X train, y train)
    # Predict the test data
    y_pred = clf_gs.predict(X_test)
    # Calculate the accuracy score
    accuracy = accuracy_score(y_test, y_pred)
    # If the accuracy is better than the current best, update the best
accuracy, classifier, parameters, and report
    if accuracy > best accuracy:
        best accuracy = accuracy
        best classifier = name
        best params = clf gs.best_params_
        best report = classification report(y test, y pred)
# Print the best results
print(f"Best Classifier: {best classifier}")
print(f"Best Parameters: ", best params)
print(f"Best Accuracy: {best accuracy:.2%}\n")
print(f"Classification Report for {best classifier}:\n", best report)
Fitting 3 folds for each of 9 candidates, totalling 27 fits
Fitting 3 folds for each of 9 candidates, totalling 27 fits
Fitting 3 folds for each of 9 candidates, totalling 27 fits
Fitting 3 folds for each of 9 candidates, totalling 27 fits
Best Classifier: SVC linear
Best Parameters: {'max iter': 10000, 'tol': 0.0001}
Best Accuracy: 98.00%
Classification Report for SVC linear:
               precision recall f1-score
                                               support
                                                     2
           0
                   1.00
                             1.00
                                       1.00
                                                     3
           1
                   1.00
                             1.00
                                       1.00
           2
                   1.00
                             1.00
                                       1.00
                                                     3
```

3	1.00	1.00	1.00	3
4	1.00	0.80	0.89	5
5	1.00	1.00	1.00	2
6	1.00	1.00	1.00	1
7	1.00	1.00	1.00	3
8	1.00	1.00	1.00	3 2
9	1.00	0.67	0.80	3
10	1.00	1.00	1.00	3
11	1.00	1.00	1.00	3 2
12	1.00	1.00	1.00	2
13	1.00	1.00	1.00	4
14	1.00	1.00	1.00	2
15	1.00	1.00	1.00	4
16	1.00	1.00	1.00	1
17	1.00	1.00	1.00	3
18	1.00	1.00	1.00	2
19	1.00	1.00	1.00	3
20	1.00	1.00	1.00	2
21	1.00	1.00	1.00	2
22	1.00	1.00	1.00	5
23	1.00	1.00	1.00	3 2
24	1.00	1.00	1.00	2
25	1.00	1.00	1.00	3
26	1.00	1.00	1.00	2
28	1.00	1.00	1.00	3
29	1.00	1.00	1.00	3 2
30	1.00	1.00	1.00	4
33	1.00	1.00	1.00	2
34	1.00	1.00	1.00	2
35	1.00	1.00	1.00	1
36	1.00	1.00	1.00	4
37	1.00	1.00	1.00	4
38	1.00	1.00	1.00	3
39	0.60	1.00	0.75	3
accuracy			0.98	100
macro avg	0.99	0.99	0.98	100
weighted avg	0.99	0.98	0.98	100

三種 kernel 表現最好的是 SVC\_linear , 參數設定是('max\_iter': 10000, 'tol': 0.0001) , 測試資料之 準確率達到 98%。

(2)以標準化後之原始資料的主成分之訓練資料學習,並以測試資料測試準確率

# 程式碼說明:

- 使用 PCA 對訓練數據 X\_train\_ 進行擬合,並將其降維到兩個主成分。
- 2. 使用訓練好的 PCA 模型將訓練數據和測試數據轉換到新的低維空間,得到 Z\_train 和 Z\_test。
- 3. 定義 SVM 分類器。設置 SVM 的正則化參數 C 為 1, 並將其與其他參數一起存儲在 opts 中。
- 4. 使用 PCA 轉換後的訓練數據 Z\_train 和對應的標籤 y\_train 來訓練 SVM 分類器。
- 5. 輸出模型的分類報告。

```
from sklearn.decomposition import PCA
from sklearn.metrics import accuracy score, classification report
def train_and_evaluate(clf_svm, X_train_, X_test_, y_train, y_test):
    # Apply PCA to the training data and transform it
    pca = PCA(n components = 45).fit(X_train_)
    Z train = pca.transform(X train )
    Z test = pca.transform(X test )
    # Fit the SVM classifier with the PCA transformed data
    clf svm.fit(Z train, y train)
    # Predict the labels of the PCA transformed test data
    predictions = clf svm.predict(Z test)
    # Calculate and print the accuracy score
    acc score = accuracy score(y test, predictions)
    print(f"Accuracy Score: {acc score:.2%}")
    # Print the classification report
    print(classification report(y test, predictions, zero division=0))
# Define the SVM classifier
C = 1 # SVM regularization parameter
opts = dict(C = C, tol = 1e-6, max iter = int(1e6))
```

#### kernel(linear)

```
clf svm = SVC(kernel="linear", **opts)
train_and_evaluate(clf_svm, X_train_, X_test_, y_train, y_test)
Accuracy Score: 92.50%
              precision recall f1-score
                                                support
                                                       5
           0
                    1.00
                              0.60
                                         0.75
           1
                    1.00
                              1.00
                                         1.00
                                                       6
           2
                                                       2
                    1.00
                              1.00
                                         1.00
           3
                                                       3
                    1.00
                              0.67
                                         0.80
           4
                                                      4
                    0.75
                              0.75
                                         0.75
                                                       3
           5
                    1.00
                              1.00
                                         1.00
           6
                    1.00
                              1.00
                                         1.00
                                                       5
           7
                    1.00
                              0.50
                                         0.67
                                                       2
```

	8	1.00	1.00	1.00	3
	9	1.00	0.67	0.80	3 3
1	L0	1.00	1.00	1.00	3
	L1	1.00	1.00	1.00	4
1	L2	0.67	0.67	0.67	3
1	L3	1.00	1.00	1.00	3
	L <b>4</b>	0.50	1.00	0.67	1
1	L5	1.00	1.00	1.00	4
1	L6	1.00	1.00	1.00	3
	L7	0.50	1.00	0.67	1
	L8	1.00	1.00	1.00	2
	L9	1.00	1.00	1.00	1
2	20	1.00	1.00	1.00	3
2	21	1.00	1.00	1.00	3
2	22	1.00	1.00	1.00	3 3 3
2	23	1.00	0.80	0.89	5
2	24	0.67	1.00	0.80	2
2	25	1.00	1.00	1.00	3
2	26	1.00	1.00	1.00	2
2	27	1.00	1.00	1.00	3
2	28	1.00	1.00	1.00	3
2	29	1.00	1.00	1.00	3
3	30	1.00	1.00	1.00	1
3	31	1.00	1.00	1.00	4
3	32	1.00	1.00	1.00	4
3	33	1.00	1.00	1.00	2
3	34	0.83	1.00	0.91	5
3	35	1.00	1.00	1.00	4
3	36	1.00	1.00	1.00	2
3	37	0.67	1.00	0.80	2
3	38	1.00	1.00	1.00	2
3	39	0.50	0.67	0.57	3
accurac	-			0.93	120
macro av		0.93	0.93	0.92	120
weighted av	/g	0.94	0.93	0.93	120

- 1. 測試資料之準確率 92.50%。
- 2. 在加權平均的評估指標中,精確率、召回率和 F1 分數分別為 0.94、0.93、0.93,這表示考慮到每個類別的樣本數量後,模型的整體性能仍然非常好。

#### kernel(rbf)

```
clf_svm = SVC(kernel="rbf", gamma=0.2, **opts)
train_and_evaluate(clf_svm, X_train_, X_test_, y_train, y_test)
```

Accuracy	Score	a· 0 83%			
Accuracy	5001	precision	recall	f1-score	support
	0	0.00	0.00	0.00	5
	1	0.00	0.00	0.00	6
	2	0.00	0.00	0.00	2 3
	3	0.00	0.00	0.00	3
	4	0.00	0.00	0.00	4
	5	0.00	0.00	0.00	3
	6 7	0.00	0.00	0.00	5 2
	8	0.00	0.00 0.00	0.00	2
	9	0.00 0.00	0.00	0.00 0.00	3 3 3
	10	0.00	0.00	0.00	3
	11	0.00	0.00	0.00	4
	12	0.00	0.00	0.00	3
	13	0.00	0.00	0.00	3
	14	0.00	0.00	0.00	1
	15	0.00	0.00	0.00	4
	16	0.00	0.00	0.00	3
	17	0.01	1.00	0.02	1
	18	0.00	0.00	0.00	2
	19	0.00	0.00	0.00	1
	20	0.00	0.00	0.00	3
	21	0.00	0.00	0.00	3 3
	22	0.00	0.00	0.00	3
	23	0.00	0.00	0.00	5 2
	24	0.00	0.00	0.00	2
	25	0.00	0.00	0.00	3 2
	26	0.00	0.00	0.00	2
	27	0.00	0.00	0.00	3
	28	0.00	0.00	0.00	3
	29 20	0.00	0.00	0.00	3 1
	30 31	0.00 0.00	0.00 0.00	0.00 0.00	
	32	0.00	0.00	0.00	4 4
	33	0.00	0.00	0.00	2
	34	0.00	0.00	0.00	5
	35	0.00	0.00	0.00	4
	36	0.00	0.00	0.00	2
	37	0.00	0.00	0.00	2 2
	38	0.00	0.00	0.00	2
	39	0.00	0.00	0.00	3
				0.01	100
accui	_	0.00	0.02	0.01	120
macro		0.00	0.03	0.00	120
weighted	avg	0.00	0.01	0.00	120

- 1. 測試資料之準確率 0.83%。
- 2. 在加權平均的評估指標中,精確率、召回率和 F1 分數分別為 0.00、0.01、0.00,這表示考慮到每個類別的樣本數量後,模型的整體性非常糟糕。

#### kernel(poly)

```
clf_svm = SVC(kernel="poly", degree=3, gamma="auto", **opts)
train_and_evaluate(clf_svm, X_train_, X_test_, y_train, y_test)
Accuracy Score: 70.00%
                              recall f1-score
               precision
                                                   support
            0
                     1.00
                                0.40
                                           0.57
                                                          5
                                           0.67
            1
                     1.00
                                0.50
                                                          6
            2
                                                          2
                     0.67
                                1.00
                                           0.80
            3
                                                          3
                     0.00
                                0.00
                                           0.00
            4
                     0.75
                                0.75
                                           0.75
                                                          4
            5
                                                          3
                     1.00
                                1.00
                                           1.00
            6
                                                          5
                     1.00
                                0.20
                                           0.33
                                                          2
            7
                     1.00
                                0.50
                                           0.67
            8
                                                          3
                     0.67
                                0.67
                                           0.67
                                                          3
            9
                     1.00
                                0.67
                                           0.80
                                                          3
           10
                     1.00
                                1.00
                                           1.00
                                0.50
                                                          4
           11
                     1.00
                                           0.67
                                                          3
           12
                     0.50
                                0.33
                                           0.40
                                                          3
           13
                     1.00
                                0.67
                                           0.80
                                                          1
           14
                     0.04
                                1.00
                                           0.08
           15
                     0.75
                                0.75
                                           0.75
                                                          4
                                                          3
           16
                     1.00
                                1.00
                                           1.00
                                                          1
           17
                     0.33
                                1.00
                                           0.50
                                                          2
           18
                     1.00
                                1.00
                                           1.00
                                                          1
           19
                     1.00
                                1.00
                                           1.00
           20
                     0.75
                                1.00
                                           0.86
                                                          3
                                                          3
           21
                     1.00
                                1.00
                                           1.00
                                                          3
           22
                     1.00
                                1.00
                                           1.00
                                                          5
           23
                     1.00
                                0.80
                                           0.89
                                                          2
           24
                     0.67
                                1.00
                                           0.80
           25
                                                          3
                     1.00
                                0.33
                                           0.50
                                                          2
           26
                     1.00
                                1.00
                                           1.00
           27
                     0.00
                                0.00
                                           0.00
                                                          3
                                                          3
           28
                     1.00
                                1.00
                                           1.00
                                                          3
           29
                     1.00
                                1.00
                                           1.00
                                                          1
           30
                     1.00
                                1.00
                                           1.00
                                                          4
           31
                     1.00
                                0.75
                                           0.86
           32
                     1.00
                                1.00
                                           1.00
                                                          4
                                                          2
           33
                     1.00
                                1.00
                                           1.00
                                                          5
           34
                     0.75
                                0.60
                                           0.67
                                                          4
           35
                     1.00
                                0.25
                                           0.40
                                                          2
           36
                     1.00
                                1.00
                                           1.00
                                                          2
           37
                     1.00
                                1.00
                                           1.00
```

	38 39	1.00 0.50	1.00 0.67	1.00 0.57	2
accu macro weighted	avg	0.83 0.86	0.76 0.70	0.70 0.75 0.73	120 120 120

- 1. 測試資料之準確率 70%。
- 2. 在加權平均的評估指標中,精確率、召回率和 F1 分數分別為 0.86、0.70、0.73,這表示考慮到每個類別的樣本數量後,模型的整體性能。

kernel(LinearSVC : "one-vs-rest")

clf\_svm = LinearSVC(dual=False, \*\*opts) # one vs the rest
train and evaluate(clf svm. X train . X test . v train. v test)

train_and_eva	lluate(clf_svm	, X_train	n_, X_test_	, y_train, y_t	test)
Accuracy Scor					
	precision	recall	f1-score	support	
Θ	0.75	0.60	0.67	5	
1	1.00	1.00	1.00	6	
2	0.67	1.00	0.80	2	
3	1.00	0.67	0.80	2 3	
4	1.00	1.00	1.00	4	
5	1.00	1.00	1.00	3	
6	1.00	0.80	0.89	5	
7	0.50	0.50	0.50	2	
8 9	1.00 1.00	1.00 0.67	1.00 0.80	5 2 3 3 3	
10	1.00	1.00	1.00	3	
11	0.80	1.00	0.89	4	
12	0.67	0.67	0.67	3	
13	1.00	1.00	1.00	3	
14	1.00	1.00	1.00		
15	1.00	0.75	0.86	4	
16	1.00	1.00	1.00	3 1	
17 18	0.50 1.00	$1.00 \\ 1.00$	0.67 1.00	2	
19	1.00	1.00	1.00	1	
20	1.00	1.00	1.00		
21	1.00	1.00	1.00	3 3 3	
22	1.00	1.00	1.00	3	
23	1.00	0.80	0.89	5 2	
24	0.67	1.00	0.80	2	
25	1.00	1.00	1.00	3	
26 27	$egin{array}{c} 1.00 \ 1.00 \end{array}$	$1.00 \\ 1.00$	$1.00 \\ 1.00$	2	
21	1.00	1.00	1.00	5	

28 29	1.00 1.00	1.00 1.00	1.00 1.00	3 3	
30	1.00	1.00	1.00	1	
31	1.00	1.00	1.00	4	
32	1.00	1.00	1.00	4	
33	1.00	1.00	1.00	2	
34	1.00	1.00	1.00	5	
35	1.00	0.75	0.86	4	
36	1.00	1.00	1.00	2	
37	0.67	1.00	0.80	2	
38	1.00	1.00	1.00	2	
39	0.40	0.67	0.50	3	
			0.01	120	
accuracy	0.02	0.02	0.91	120	
macro avg	0.92 0.93	0.92 0.91	0.91 0.91	120 120	
weighted avg	0.95	0.91	0.91	120	

- 1. 測試資料之準確率 90.83%。
- 2. 在加權平均的評估指標中,精確率、召回率和 F1 分數分別為 0.93、0.91、0.91,這表示考慮到每個類別的樣本數量後,模型的整體性能很好。

# 使用網格搜索(GridSearchCV)來尋找最佳的參數組合

#### 程式碼說明:

- 1. 設定參數網格
- 2. 進行網格搜索(GridSearchCV)
- 3. 輸出最佳參數:網格搜索完成後,輸出找到的最佳參數組合
- 4. 預測並評估模型:使用訓練好的模型對測試數據進行預測,並輸出模型在測試數據上的準確度和分類報告

```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC, LinearSVC

def train_and_evaluate(X_train_, X_test_, y_train, y_test):
    # Apply PCA to the training data and transform it
    pca = PCA(n_components = 45).fit(X_train_)
    Z_train = pca.transform(X_train_)
    Z_test = pca.transform(X_test_)

# Define the parameter grid for GridSearchCV
param_grid = {
    'tol': [le-4, le-5, le-6],
    'max_iter': [int(le4), int(le5), int(le6)]
}

# Define the classifiers
```

```
classifiers = {
        'SVC linear': SVC(kernel='linear', max iter=int(1e6)),
        'SVC rbf': SVC(kernel='rbf', max iter=int(1e6)),
        'SVC poly': SVC(kernel='poly', max_iter=int(1e6)),
        'LinearSVC': LinearSVC(max iter=int(1e6), dual=False)
    }
    # Initialize the best accuracy and classifier
    best accuracy = 0
    best classifier = None
    best params = None
    best report = None
    # For each classifier
    for name, clf in classifiers.items():
        # Define the GridSearchCV
        clf gs = GridSearchCV(clf, param grid, verbose=1, cv=3,
n jobs=-1
        # Fit the classifier with the PCA transformed data
        clf gs.fit(Z train, y train)
        # Predict the labels of the PCA transformed test data
        predictions = clf gs.predict(Z test)
        # Calculate the accuracy score
        acc score = accuracy score(y test, predictions)
        # If the accuracy is better than the current best, update the
best accuracy, classifier, parameters, and report
        if acc score > best accuracy:
            best accuracy = acc score
            best_classifier = name
            best params = clf_gs.best_params_
            best report = classification report(y test, predictions)
    # Print the best classifier, parameters and accuracy
    print(f"Best Classifier: {best_classifier}")
    print(f"Best Parameters: ", best_params)
    print(f"Best Accuracy: {best_accuracy:.2%}\n")
    print(f"Classification Report for {best classifier}:\n",
best report)
train and evaluate(X train , X test , y train, y test)
Fitting 3 folds for each of 9 candidates, totalling 27 fits
Fitting 3 folds for each of 9 candidates, totalling 27 fits
Fitting 3 folds for each of 9 candidates, totalling 27 fits
Fitting 3 folds for each of 9 candidates, totalling 27 fits
Best Classifier: SVC_linear
Best Parameters: {'max iter': 10000, 'tol': 0.0001}
```

Best Accuracy: 98.00% Classification Report for SVC linear: precision recall f1-score support 1.00 1.00 1.00 2 1 1.00 1.00 1.00 3 2 0.75 1.00 3 0.86 3 3 1.00 1.00 1.00 5 4 1.00 1.00 1.00 2 5 1.00 1.00 1.00 6 1 1.00 1.00 1.00 3 7 1.00 1.00 1.00 2 8 1.00 1.00 1.00 9 3 3 2 1.00 0.67 0.80 10 1.00 1.00 1.00 11 1.00 1.00 1.00 2 12 1.00 1.00 1.00 4 13 1.00 1.00 1.00 14 1.00 1.00 1.00 2 15 1.00 1.00 4 1.00 1 16 1.00 1.00 1.00 17 1.00 1.00 1.00 3 2 18 1.00 1.00 1.00 3 19 1.00 1.00 1.00 2 1.00 20 1.00 1.00 2 21 1.00 1.00 1.00 5 22 1.00 1.00 1.00 3 23 1.00 1.00 1.00 24 1.00 1.00 1.00 3 1.00 25 0.67 0.80 26 1.00 1.00 1.00 3 28 1.00 1.00 1.00 29 1.00 1.00 2 1.00 4 30 1.00 1.00 1.00 2 33 1.00 1.00 1.00 2 34 1.00 1.00 1.00 1 35 1.00 1.00 1.00 1.00 1.00 4 36 1.00 4 37 1.00 1.00 1.00 38 1.00 1.00 1.00 3 39 0.75 1.00 0.86 0.98 100 accuracy 0.99 0.98 0.98 100 macro avg 0.98 0.98 0.98 weighted avg 100

結果說明:

三種 kernel 表現最好的是 SVC\_linear, 參數設定是('max\_iter': 10000, 'tol': 0.0001), 測試資料之準確率達到 98%。

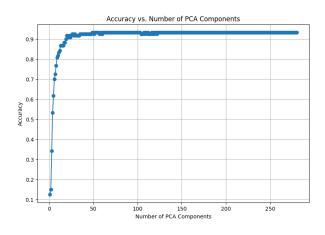
將主成分逐漸調高(直到 281),觀察準確率的變化,並畫一張折線圖來比較(與 scree plot 對比)。

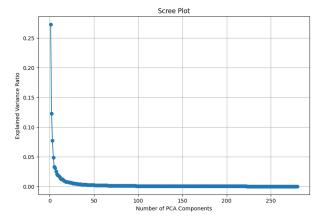
(選用 linear kernel)

```
from sklearn.svm import SVC
import warnings
from sklearn.exceptions import ConvergenceWarning
# Ignore ConvergenceWarning
warnings.filterwarnings("ignore", category=ConvergenceWarning)
# Initialize an empty list to store accuracy for each number of
components
accuracy scores = []
C = 1 \# \overline{SVM} regularization parameter
opts = dict(C = C, tol = 1e-6, max iter = int(1e6))
# Loop over number of components from 1 to 13
for i in range(1, 281):
    pca = PCA(n components=i).fit(X train )
    Z train = pca.transform(X train )
    Z test = pca.transform(X_test_)
    clf PCA = SVC(kernel="linear", **opts)
    clf_PCA.fit(Z_train, y_train)
    y pred = clf PCA.predict(Z test)
    accuracy = clf PCA.score(Z test, y test)
    accuracy scores.append(accuracy)
# Create a figure and a grid of subplots
fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))
# Plot the accuracy scores on the first subplot
ax1.plot(range(1, 281), accuracy scores, marker='o', linestyle='-')
ax1.set title('Accuracy vs. Number of PCA Components')
ax1.set_xlabel('Number of PCA Components')
ax1.set vlabel('Accuracy')
ax1.grid(True)
# Fit PCA on the training data
pca = PCA().fit(X train )
# Calculate explained variance ratio for each component
explained variance ratio = pca.explained variance ratio
# Plot the explained variance ratio on the second subplot
ax2.plot(range(1, 281), explained variance ratio, marker='o',
linestyle='-')
```

```
ax2.set_title('Scree Plot')
ax2.set_xlabel('Number of PCA Components')
ax2.set_ylabel('Explained Variance Ratio')
ax2.grid(True)

# Display the figure with the two subplots
plt.show()
```





- 1. 從 Scree Plot 來看,將主成分逐漸調高,Explained Variance Ratio 隨之降低。
- 從左圖來看,只採用第一個主成分時,測試資料之準確率大約等於10%,隨著主成分逐漸調高,測試資料之準確率皆維持在大約等於90幾%。
- 3. 以上說明採用越多主成分,Explained Variance Ratio 隨之降低,測試資料之準確率隨之提升。

# (四)神經網路(Neural Network)

# (1)以標準化後之原始資料的訓練資料學習,並以測試資料測試準確率 程式碼說明:

- 1. 定義並訓練 MLP 分類器
- 2. 進行預測,將預測結果存儲在 predictions 變量中
- 3. 評估模型,使用 classification\_report 函數來生成一個分類報告
- 4. 設定使用不同演算法

```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report

def train_MLP(solver, X_train, y_train, X_test, y_test):
    hidden_layers = (30,)
    activation = "logistic"
    opts = dict(hidden_layer_sizes = hidden_layers , verbose = False,
    activation = activation, tol = 1e-6, max_iter = int(1e6))
```

```
clf_MLP = MLPClassifier(solver = solver, **opts)
clf_MLP.fit(X_train, y_train)
predictions = clf_MLP.predict(X_test)
print(classification_report(y_test, predictions, zero_division=0))
```

# 演算法(adam)

# # 使用函數訓練模型

train_MLP("ada	am", X_train,	y_train,	X_test,	y_test)
	precision	recall	f1-score	support
Θ	1.00	0.60	0.75	5
1	1.00	1.00	1.00	6
2	1.00	1.00	1.00	2
3 4	1.00 1.00	0.67 0.75	0.80 0.86	3 4
5	1.00	1.00	1.00	3
6	1.00	1.00	1.00	5
7	1.00	0.50	0.67	5 2
8	1.00	1.00	1.00	3
9	1.00	0.67	0.80	3
10	1.00	1.00	1.00	3
11	1.00	1.00	1.00	4
12	0.60	1.00	0.75	3 3
13 14	1.00 1.00	$1.00 \\ 1.00$	1.00 1.00	1
15	1.00	1.00	1.00	4
16	1.00	1.00	1.00	3
17	0.50	1.00	0.67	1
18	1.00	1.00	1.00	2
19	1.00	1.00	1.00	1
20	1.00	1.00	1.00	3 3
21	1.00	1.00	1.00	3
22	1.00	1.00	1.00	3
23 24	1.00 1.00	$1.00 \\ 1.00$	$1.00 \\ 1.00$	ے 2
25	1.00	1.00	1.00	3
26	1.00	1.00	1.00	5 2 3 2 3 3
27	1.00	1.00	1.00	3
28	1.00	1.00	1.00	
29	1.00	1.00	1.00	3
30	0.50	1.00	0.67	1
31	1.00	1.00	1.00	4
32 33	1.00 1.00	$1.00 \\ 1.00$	1.00	4 2
34	1.00	1.00	1.00 1.00	5
35	0.80	1.00	0.89	4
55	0.00			·

36	1.00	1.00	1.00	2
37	1.00	1.00	1.00	2
38	1.00	1.00	1.00	2
39	0.75	1.00	0.86	3
accuracy macro avg weighted avg	0.95 0.97	0.95 0.95	0.95 0.94 0.95	120 120 120

- 1. 測試資料之準確率 95%。
- 在加權平均的評估指標中,模型的精確率、召回率和 F1 分數分別為 0.97、0.95、0.95, 這表示模型在每個類別的預測性能很好。

#### 演算法(sgd)

train\_MLP("sgd", X\_train, y\_train, X\_test, y\_test) recall f1-score precision support 5 0 0.60 0.75 1.00 1 1.00 1.00 1.00 6 2 2 0.67 1.00 0.80 3 1.00 0.67 0.80 3 4 4 1.00 0.75 0.86 5 3 1.00 1.00 1.00 5 6 1.00 1.00 1.00 2 7 0.50 0.50 0.50 3 1.00 8 1.00 1.00 9 3 1.00 0.67 0.80 3 10 1.00 1.00 1.00 11 1.00 1.00 1.00 3 12 1.00 0.67 0.80 3 13 1.00 1.00 1.00 0.67 0.50 1.00 1 14 4 15 1.00 1.00 1.00 3 16 1.00 1.00 1.00 17 0.50 1.00 0.67 1 18 1.00 1.00 1.00 2 1 19 1.00 1.00 1.00 3 20 1.00 1.00 1.00 21 1.00 1.00 1.00 3 3 22 1.00 1.00 1.00 5 23 1.00 0.80 0.89 24 1.00 1.00 1.00 3 25 0.75 1.00 0.86 2 26 1.00 1.00 1.00 3 27 1.00 1.00 1.00

28						
31	29	1.00	1.00	1.00	3	
32						
34 0.83 1.00 0.91 5 35 0.80 1.00 0.89 4 36 1.00 1.00 1.00 2 37 1.00 1.00 1.00 2 38 1.00 1.00 1.00 2 39 0.75 1.00 0.86 3  accuracy macro avg 0.93 0.94 0.93 120	32	1.00	1.00	1.00	4	
35 0.80 1.00 0.89 4 36 1.00 1.00 1.00 2 37 1.00 1.00 1.00 2 38 1.00 1.00 1.00 2 39 0.75 1.00 0.86 3  accuracy 0.93 0.94 0.93 120						
36 1.00 1.00 1.00 2 37 1.00 1.00 1.00 2 38 1.00 1.00 1.00 2 39 0.75 1.00 0.86 3  accuracy 0.93 120 macro avg 0.93 0.94 0.93 120						
38 1.00 1.00 1.00 2 39 0.75 1.00 0.86 3 accuracy 0.93 120 macro avg 0.93 0.94 0.93 120	36	1.00	1.00	1.00	2	
39 0.75 1.00 0.86 3  accuracy 0.93 120 macro avg 0.93 0.94 0.93 120						
accuracy 0.93 120 macro avg 0.93 0.94 0.93 120						
macro avg 0.93 0.94 0.93 120					_	
weighted avg 0.95 0.93 0.93 120	_		0.94			
	weighted avo	0.95	0.93	0.93	120	

- 1. 測試資料之準確率 93%。
- 2. 在加權平均的評估指標中,模型的精確率、召回率和 F1 分數分別為 0.95、0.93、0.93, 這表示模型在每個類別的預測性能很好。

# 演算法(lbfgs)

train_MLP("lb	fgs", X_train,	y_train,	X_test,	y_test)
	precision	recall f	1-score	support
0	1.00	0.40	0.57	5
1	1.00	0.83	0.91	6
2	1.00	1.00	1.00	2
3 4	1.00 0.60	1.00 0.75	1.00 0.67	3 4
5	1.00	1.00	1.00	
6	1.00	1.00	1.00	3
7 8	1.00 1.00	0.50 1.00	0.67 1.00	2
9	1.00	1.00	1.00	3 3 3
10	1.00	1.00	1.00	
11	1.00	1.00	1.00	4
12 13	0.67 0.75	0.67 1.00	0.67 0.86	3 3 1
14	1.00	1.00	1.00	
15	0.75	0.75	0.75	4
16 17	1.00 1.00	$1.00 \\ 1.00$	$1.00 \\ 1.00$	3 1
18	1.00	1.00	1.00	2
19	1.00	1.00	1.00	1

- 1. 測試資料之準確率 92%。
- 2. 在加權平均的評估指標中,模型的精確率、召回率和 F1 分數分別為 0.93、0.92、0.91, 這表示模型在每個類別的預測性能很好。

三種演算法表現最好的是 adamc, 測試資料之準確率 95%。

# (2)以標準化後之原始資料的主成分之訓練資料學習,並以測試資料測試準確率

#### 程式碼說明:

- 1. 定義並訓練 MLP 分類器
- 2. 進行預測,將預測結果存儲在 predictions 變量中
- 3. 設定保留前兩個主成分
- 4. 評估模型,使用 classification\_report 函數來生成一個分類報告
- 5. 設定使用不同演算法

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

```
def train MLP with PCA(solver, X train, y train, X test, y test):
    # Define the PCA transformer
    pca = PCA(n components=45)
    # Apply PCA to the training data and transform it
    X_train_pca = pca.fit_transform(X_train)
    X_test_pca = 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))
    clf MLP = MLPClassifier(solver = solver, **opts)
    clf MLP.fit(X train pca, y train)
    predictions = clf_MLP.predict(X_test_pca)
    print(classification report(y test, predictions))
```

#### 演算法(adam)

#### # 使用函數訓練模型

train_MLP_wit	h_PCA("adam",	X_train,	y_train,	X_test, y_test)	
	precision	recall	f1-score	support	
Θ	1.00	0.60	0.75	5	
1	1.00	0.83	0.91	6	
2	0.67	1.00	0.80	2	
3	0.50	0.67	0.57	3	
4	1.00	0.75	0.86	4	
5	1.00	1.00	1.00	3	
6	1.00	1.00	1.00	5	
7	1.00	0.50	0.67	2	
8 9	1.00	1.00	1.00	3	
10	$egin{array}{c} 1.00 \ 1.00 \end{array}$	0.67	0.80	5 2 3 3 3	
11	1.00	$1.00 \\ 1.00$	$1.00 \\ 1.00$	4	
12	0.67	0.67	0.67	3	
13	1.00	1.00	1.00	3 3 1	
14	0.50	1.00	0.67	1	
15	1.00	1.00	1.00	$\overline{4}$	
16	1.00	1.00	1.00	3	
17	1.00	1.00	1.00	1	
18	1.00	1.00	1.00	2	
19	1.00	1.00	1.00	1	
20	0.75	1.00	0.86	3 3 3	
21	1.00	1.00	1.00	3	
22	1.00	1.00	1.00	3	

23	1.00	1.00	1.00	5
24	0.67	1.00	0.80	2
25	1.00	1.00	1.00	3
26	0.67	1.00	0.80	2
27	1.00	1.00	1.00	3
28	1.00	1.00	1.00	3
29	1.00	1.00	1.00	3
30	1.00	1.00	1.00	1
31	1.00	1.00	1.00	4
32	1.00	1.00	1.00	4
33	1.00	1.00	1.00	2
34	1.00	1.00	1.00	5
35	1.00	0.75	0.86	4
36	1.00	1.00	1.00	2
37	1.00	1.00	1.00	2
38	1.00	1.00	1.00	2
39	0.75	1.00	0.86	3
			0.02	120
accuracy	0.02	0.04	0.93	120
macro avg	0.93	0.94	0.92	120
weighted avg	0.95	0.93	0.93	120

- 1. 測試資料之準確率 93%。
- 2. 在加權平均的評估指標中,模型的精確率、召回率和 F1 分數分別為 0.95、0.93、0.93, 這表示模型在每個類別的預測性能。

# 演算法(sgd)

# # 使用函數訓練模型

train_MLP_wit	ch_PCA("sgd",	X_train,	y_train, X	<pre>(_test, y_test)</pre>
	precision	recall	f1-score	support
•	1 00	0.60	0.75	-
0	1.00	0.60	0.75	5
1	1.00	1.00	1.00	6
2	1.00	1.00	1.00	2
3	0.67	0.67	0.67	3
4	1.00	0.75	0.86	4
5	0.75	1.00	0.86	3
6	1.00	1.00	1.00	5
7	1.00	0.50	0.67	2
8	1.00	1.00	1.00	3
9	1.00	0.67	0.80	3
10	1.00	1.00	1.00	3
11	1.00	1.00	1.00	4
12	0.67	0.67	0.67	3
13	0.75	1.00	0.86	3

14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37	0.50 0.80 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00	1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.80 1.00 0.67 1.00 1.00 1.00 1.00 1.00	0.67 0.89 1.00 1.00 1.00 0.75 1.00 0.89 1.00 0.80 1.00 0.67 1.00 1.00 1.00 1.00	1 4 3 1 2 1 3 3 5 2 3 3 3 1 4 4 2 5 4 2 2	
37 38 39	1.00 1.00 0.75	1.00 1.00 1.00	1.00 1.00 0.86	2 2 3	
accuracy macro avg weighted avg	0.92 0.94	0.92 0.92	0.92 0.91 0.92	120 120 120	

- 1. 測試資料之準確率 92%。
- 2. 在加權平均的評估指標中,模型的精確率、召回率和 F1 分數分別為 0.94、0.92、0.92, 這表示模型在每個類別的預測性能很好。

# 演算法(lbfgs)

#### # 使用函數訓練模型

train\_MLP\_with\_PCA("lbfgs", X\_train, y\_train, X\_test, y\_test)

	precision	recall	f1-score	support
0	1.00	0.60	0.75	5
1	1.00	1.00	1.00	6
2	0.67	1.00	0.80	2
3	0.67	0.67	0.67	3
4	1.00	0.75	0.86	4

	5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 37 37 37 37 37 37 37 37 37 37 37 37	0.75 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.0	1.00 1.00 0.50 1.00 0.67 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.0	0.86 1.00 0.67 1.00 0.67 1.00 0.80 1.00 0.89 0.86 0.67 1.00 1.00 0.86 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.0	3 5 2 3 3 3 4 3 1 4 3 1 2 1 3 3 5 2 3 3 3 1 4 4 2 5 4 5 4	
	34 35 36 37	1.00 1.00 1.00 1.00	1.00 0.75 1.00 1.00	1.00 0.86 1.00 1.00	5 4 2 2	
accun macro weighted	avg	1.00 1.00 0.93 0.94	1.00 1.00 0.93 0.93	1.00 1.00 0.93 0.92 0.92	2 3 120 120 120	
J. 1 = 2 4	J					

- 1. 測試資料之準確率 93%。
- 2. 在加權平均的評估指標中,模型的精確率、召回率和 F1 分數分別為 0.94、0.93、0.92, 這表示模型在每個類別的預測性能很好。

三種演算法表現最好的是 adamc 和 lbfgs,測試資料之準確率 93%。

# 對兩種資料型態與三個分類器的表現做比較

多元羅吉斯回歸 (Multinomial Logistic Regression)

- 1. 原始資料: accuracy 大約 97%
- 2. 主成分資料(取前 2 個): accuracy 大約 95%

#### 支援向量機 (Support Vector Machine)

- 1. 原始資料: accuracy 大約 98%
- 2. 主成分資料(取前 2 個): accuracy 大約 98%

#### 神經網路 (Neural Network)

- 1. 原始資料: accuracy 大約 95%
- 2. 主成分資料(取前 2 個): accuracy 大約 93%

#### 結果:

- 1. 在原始資料下,支援向量機的測試資料之準確率最高,達97%;再來是多元羅吉斯回歸,測試資料之準確率達97%;最後是神經網路,測試資料之準確率達95%。
- 2. 主成分資料(取前 45 個)下,支援向量機的測試資料最高,達 98%;多元羅吉斯回歸的測試資料之準確率 95%;最後是神經網路,測試資料之準確率達 93%。

#### 個人見解:

- 1. 支援向量機在所有情況下都有最高的準確率。
- 2. 神經網路在降維後的資料上表現較差,可能因為資訊損失或模型複雜度過高。
- 3. 對於這組較小的資料,較簡單的模型(多元羅吉斯回歸和支援向量機)表現較好,可能因為神經網路無法充分學習。