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 :來自 Yale Face 38 人的人臉影像共 2410 張，每張大小 192×168。

程式碼說明:

1. 讀取來自 Yale Face 38 人的人臉影像共 2410 張。
2. 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%。
3. 對訓練集和測試集進行標準化處理。

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.io
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

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

# Create labels for each image
y = np.repeat(np.arange(n_persons), y)

# 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)

print("Shape of X:", X.shape)
print("Shape of y:", y.shape)
```

```
Shape of X: (2410, 32256)
Shape of y: (2410,)
```

(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. 使用 LogisticRegression 來建立模型。
3. 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. 最後選擇模型演算法(lbfgs)，給出完整的報告(classification_report)。

演算法(lbfgs)

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

```python
def train_LR(solver, X_train, y_train, X_test, y_test):
    opts = dict(tol = 1e-6, max_iter = int(1e6), 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)
```

```
96.52%

96.52%

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        22
           1       1.00      1.00      1.00        18
           2       0.88      1.00      0.93        14
           3       0.79      1.00      0.88        11
           4       1.00      1.00      1.00        16
           5       1.00      1.00      1.00        12
           6       1.00      1.00      1.00        12
           7       1.00      0.95      0.97        20
           8       1.00      0.94      0.97        16
           9       0.93      1.00      0.97        14
          10       1.00      1.00      1.00        14
          11       1.00      0.94      0.97        17
          12       1.00      0.86      0.92        21
          13       1.00      1.00      1.00        16
          14       1.00      1.00      1.00        12
          15       1.00      1.00      1.00        18
          16       1.00      1.00      1.00        12
          17       1.00      1.00      1.00        16
          18       1.00      1.00      1.00        20
          19       1.00      0.79      0.88        19
          20       0.94      1.00      0.97        17
          21       1.00      0.93      0.96        14
          22       1.00      1.00      1.00        11
          23       1.00      0.94      0.97        18
          24       1.00      0.95      0.98        21
          25       1.00      1.00      1.00         9
          26       1.00      0.94      0.97        17
          27       1.00      1.00      1.00        13
          28       0.92      0.92      0.92        13
          29       0.93      0.93      0.93        14
          30       1.00      1.00      1.00        23
```

```
           31       0.88      1.00      0.93        14
           32       0.95      0.95      0.95        19
           33       0.81      0.93      0.87        14
           34       0.94      0.89      0.92        19
           35       0.74      1.00      0.85        14
           36       1.00      1.00      1.00        18
           37       1.00      0.93      0.97        15

    accuracy                           0.97       603
   macro avg       0.97      0.97      0.97       603
weighted avg       0.97      0.97      0.97       603
```

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. 測試資料之準確率大約 97%。
2. 在加權平均的評估指標中，精確率、召回率和 F1 分數都為 0.9 7 ，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。

分類器性能：

1. 演算法 lbfgs 表現最好，測試資料之準確率大約為 97%。
2. liblinear 演算法是一種針對小數據集的優化算法，對於大數據集，它可能會花費較長的時間，跑了 440 分鐘都還沒跑完；newton-cg 演算法需要計算整個海森矩陣（Hessian matrix）的逆矩陣。這在特徵數量非常大時可能會導致問題。因此先不考慮這兩個演算法。

(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

演算法(lbfgs)

程式碼說明 :

1. 使用 PCA 對訓練數據 X_train_ 進行擬合，並只保留前 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。

```
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 = "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))
```

```
91.54%

              precision    recall  f1-score   support

           0       0.80      1.00      0.89        12
           1       1.00      1.00      1.00        13
           2       0.94      0.94      0.94        16
           3       0.94      0.94      0.94        16
           4       1.00      1.00      1.00        11
           5       0.92      0.92      0.92        24
           6       0.90      0.90      0.90        20
           7       0.76      0.93      0.84        14
           8       1.00      1.00      1.00        18
           9       1.00      0.80      0.89        20
          10       1.00      0.86      0.92        14
          11       1.00      0.92      0.96        13
          12       1.00      0.88      0.93        16
          13       0.94      0.89      0.91        18
          14       0.73      0.89      0.80         9
          15       0.81      0.93      0.87        14
          16       0.86      0.95      0.90        19
          17       1.00      1.00      1.00        12
          18       0.59      1.00      0.74        13
          19       0.94      0.94      0.94        16
          20       1.00      0.95      0.97        20
          21       0.90      0.75      0.82        12
          22       0.93      0.93      0.93        15
          23       1.00      0.69      0.82        13
          24       1.00      1.00      1.00        20
          25       1.00      0.93      0.97        15
          26       0.95      0.95      0.95        19
          27       0.95      0.78      0.86        23
          28       0.95      1.00      0.97        19
          29       1.00      0.93      0.97        15
```

```
          30       0.84      0.94      0.89        17
          31       0.80      0.94      0.86        17
          32       1.00      0.95      0.97        19
          33       0.83      0.67      0.74        15
          34       0.86      1.00      0.92        18
          35       1.00      0.80      0.89        15
          36       0.91      1.00      0.95        10
          37       1.00      0.92      0.96        13

    accuracy                           0.92       603
   macro avg       0.92      0.92      0.91       603
weighted avg       0.93      0.92      0.92       603
```

結果說明：

1. 測試資料之準確率 92%。
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、0.92，、0.92，這表示考慮到每個類別的樣本數量後，模型的整體性表還可以。

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]88.23%

              precision    recall  f1-score   support

           0       0.67      0.83      0.74        12
           1       1.00      0.85      0.92        13
           2       0.88      0.94      0.91        16
           3       0.75      0.94      0.83        16
           4       0.92      1.00      0.96        11
           5       0.84      0.88      0.86        24
           6       0.73      0.80      0.76        20
           7       0.87      0.93      0.90        14
           8       1.00      0.94      0.97        18
           9       0.93      0.70      0.80        20
          10       1.00      0.71      0.83        14
          11       1.00      0.77      0.87        13
```

```
           12       1.00      0.75      0.86        16
           13       1.00      0.83      0.91        18
           14       0.82      1.00      0.90         9
           15       0.78      1.00      0.88        14
           16       0.86      0.95      0.90        19
           17       0.92      1.00      0.96        12
           18       0.73      0.85      0.79        13
           19       0.91      0.62      0.74        16
           20       1.00      0.90      0.95        20
           21       1.00      0.83      0.91        12
           22       1.00      0.93      0.97        15
           23       0.82      0.69      0.75        13
           24       0.95      1.00      0.98        20
           25       0.94      1.00      0.97        15
           26       1.00      1.00      1.00        19
           27       0.95      0.83      0.88        23
           28       0.89      0.84      0.86        19
           29       0.82      0.93      0.87        15
           30       0.89      0.94      0.91        17
           31       0.76      0.94      0.84        17
           32       0.90      0.95      0.92        19
           33       0.75      0.80      0.77        15
           34       0.89      0.94      0.92        18
           35       0.92      0.80      0.86        15
           36       0.71      1.00      0.83        10
           37       1.00      1.00      1.00        13

    accuracy                           0.88       603
   macro avg       0.89      0.88      0.88       603
weighted avg       0.89      0.88      0.88       603
```

結果說明：

1. 測試資料之準確率 88%。
2. 在加權平均的評估指標中，精確率、召回率和 F1 分數都為 88，這表示考慮到每個類別的樣本數量後，模型的整體性表還可以。

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))
```

```
92.21%

              precision    recall  f1-score   support

           0       0.80      1.00      0.89        12
           1       1.00      1.00      1.00        13
           2       0.94      0.94      0.94        16
           3       0.94      0.94      0.94        16
           4       1.00      1.00      1.00        11
           5       0.95      0.88      0.91        24
           6       0.90      0.90      0.90        20
           7       0.82      1.00      0.90        14
           8       1.00      1.00      1.00        18
           9       1.00      0.80      0.89        20
          10       1.00      0.86      0.92        14
          11       1.00      1.00      1.00        13
          12       1.00      0.88      0.93        16
          13       0.94      0.89      0.91        18
          14       0.73      0.89      0.80         9
          15       0.81      0.93      0.87        14
          16       0.90      0.95      0.92        19
          17       1.00      1.00      1.00        12
          18       0.57      1.00      0.72        13
          19       0.94      0.94      0.94        16
          20       1.00      0.95      0.97        20
          21       1.00      0.75      0.86        12
          22       0.94      1.00      0.97        15
          23       1.00      0.77      0.87        13
          24       1.00      1.00      1.00        20
          25       1.00      0.93      0.97        15
          26       0.95      0.95      0.95        19
          27       0.95      0.78      0.86        23
          28       0.95      1.00      0.97        19
          29       1.00      0.93      0.97        15
          30       0.84      0.94      0.89        17
          31       0.81      1.00      0.89        17
          32       1.00      0.95      0.97        19
          33       0.83      0.67      0.74        15
          34       0.90      1.00      0.95        18
          35       1.00      0.80      0.89        15
          36       0.91      1.00      0.95        10
          37       1.00      0.92      0.96        13

    accuracy                           0.92       603
   macro avg       0.93      0.92      0.92       603
weighted avg       0.93      0.92      0.92       603
```

結果說明：

1. 測試資料之準確率 92%。
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、0.92、0.92，這表示考慮到每個類別的樣本數量後，模型的整體性表還可以。

分類器性能：

演算法 lbfgs 和 newton-cg 在只保留前 45 個主成分的表現最好，測試資料之準確率大約為 92%。

(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

程式碼說明：

1. 函數定義：train_and_evaluate 函數接收一個分類器（clf）和訓練集與測試集的數據。這個函數的目的是訓練分類器並評估其在測試集上的性能。
2. 訓練模型，其中 X_train 和 y_train 分別是訓練數據的特徵和標籤。
3. 對測試數據進行預測，並將預測結果存儲在 predictions 變量中。
4. 使用 SVC(kernel="＊＊＊", **opts)來建立 SVM 分類器並選定演算法，最後產出結果報告。

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))

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

```
Accuracy Score: 90.71%
              precision    recall  f1-score   support

           0       0.83      0.83      0.83        12
           1       1.00      1.00      1.00        13
           2       0.84      1.00      0.91        16
           3       0.84      1.00      0.91        16
           4       0.85      1.00      0.92        11
           5       0.89      1.00      0.94        24
           6       0.95      0.95      0.95        20
           7       0.52      0.93      0.67        14
           8       0.94      0.89      0.91        18
           9       0.74      0.70      0.72        20
          10       1.00      0.93      0.96        14
          11       1.00      0.92      0.96        13
          12       1.00      0.88      0.93        16
          13       1.00      0.89      0.94        18
          14       0.89      0.89      0.89         9
          15       0.88      1.00      0.93        14
          16       1.00      1.00      1.00        19
          17       0.92      1.00      0.96        12
          18       0.68      1.00      0.81        13
          19       0.83      0.94      0.88        16
          20       1.00      0.95      0.97        20
          21       1.00      0.92      0.96        12
          22       1.00      0.93      0.97        15
          23       0.93      1.00      0.96        13
          24       1.00      0.95      0.97        20
          25       1.00      0.93      0.97        15
          26       1.00      0.84      0.91        19
          27       1.00      0.83      0.90        23
          28       0.83      1.00      0.90        19
          29       0.78      0.93      0.85        15
          30       1.00      0.94      0.97        17
          31       0.94      1.00      0.97        17
          32       1.00      0.79      0.88        19
          33       1.00      0.53      0.70        15
          34       0.94      0.89      0.91        18
          35       1.00      0.80      0.89        15
          36       1.00      0.90      0.95        10
          37       1.00      0.62      0.76        13

    accuracy                           0.91       603
   macro avg       0.92      0.91      0.91       603
weighted avg       0.92      0.91      0.91       603
```

結果說明：

1. 測試資料之準確率 91%。

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 0.92、0.91、0.91，這表示考慮到每個類別的樣本數量後，模型的整體性表還可以。

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: 1.66%
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        12
           1       0.00      0.00      0.00        13
           2       0.00      0.00      0.00        16
           3       0.00      0.00      0.00        16
           4       0.00      0.00      0.00        11
           5       0.00      0.00      0.00        24
           6       0.00      0.00      0.00        20
           7       0.00      0.00      0.00        14
           8       0.00      0.00      0.00        18
           9       0.00      0.00      0.00        20
          10       0.00      0.00      0.00        14
          11       0.00      0.00      0.00        13
          12       0.00      0.00      0.00        16
          13       0.00      0.00      0.00        18
          14       0.00      0.00      0.00         9
          15       0.00      0.00      0.00        14
          16       0.00      0.00      0.00        19
          17       0.00      0.00      0.00        12
          18       0.00      0.00      0.00        13
          19       0.00      0.00      0.00        16
          20       0.00      0.00      0.00        20
          21       0.00      0.00      0.00        12
          22       0.00      0.00      0.00        15
          23       0.00      0.00      0.00        13
          24       0.00      0.00      0.00        20
          25       0.00      0.00      0.00        15
          26       0.00      0.00      0.00        19
          27       0.00      0.00      0.00        23
          28       0.00      0.00      0.00        19
          29       0.00      0.00      0.00        15
          30       0.00      0.00      0.00        17
          31       0.00      0.00      0.00        17
          32       0.00      0.00      0.00        19
          33       0.00      0.00      0.00        15
          34       0.00      0.00      0.00        18
          35       0.00      0.00      0.00        15
          36       0.02      1.00      0.03        10
          37       0.00      0.00      0.00        13
```

```
    accuracy                           0.02       603
   macro avg       0.00      0.03      0.00       603
weighted avg       0.00      0.02      0.00       603
```

結果說明：

1. 測試資料之準確率 2%。
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、0.02、0.00，模型的整體性表現非常糟糕。

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: 67.00%
              precision    recall  f1-score   support

           0       0.40      0.83      0.54        12
           1       1.00      0.23      0.38        13
           2       0.65      0.69      0.67        16
           3       0.87      0.81      0.84        16
           4       0.53      0.91      0.67        11
           5       0.63      0.92      0.75        24
           6       0.60      0.75      0.67        20
           7       0.53      0.71      0.61        14
           8       1.00      0.50      0.67        18
           9       0.41      0.35      0.38        20
          10       0.67      0.86      0.75        14
          11       0.85      0.85      0.85        13
          12       1.00      0.31      0.48        16
          13       0.65      0.83      0.73        18
          14       0.57      0.89      0.70         9
          15       0.74      1.00      0.85        14
          16       0.82      0.95      0.88        19
          17       0.31      0.92      0.46        12
          18       0.91      0.77      0.83        13
          19       0.69      0.56      0.62        16
          20       0.72      0.90      0.80        20
          21       0.43      0.75      0.55        12
          22       0.61      0.93      0.74        15
          23       0.75      0.46      0.57        13
          24       0.72      0.90      0.80        20
          25       0.73      0.73      0.73        15
          26       1.00      0.53      0.69        19
          27       0.80      0.70      0.74        23
          28       0.75      0.79      0.77        19
          29       0.58      0.93      0.72        15
          30       1.00      0.53      0.69        17
```

```
           31       0.81      0.76      0.79        17
           32       1.00      0.53      0.69        19
           33       1.00      0.13      0.24        15
           34       1.00      0.17      0.29        18
           35       1.00      0.40      0.57        15
           36       1.00      0.40      0.57        10
           37       0.75      0.23      0.35        13

    accuracy                           0.67       603
   macro avg       0.75      0.67      0.65       603
weighted avg       0.76      0.67      0.65       603
```

結果說明：

1. 測試資料之準確率 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 0.76、0.67、0.65，這表示考慮到每個類別的樣本數量後，模型的整體性表現有待加強。

分類器性能：

kernel(linear)表現最好，測試資料之準確率大約為 91%。

(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: 92.87%
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       0.81      0.89      0.85        19
           2       0.89      0.89      0.89        19
           3       1.00      1.00      1.00        14
           4       0.88      0.82      0.85        17
           5       0.80      1.00      0.89        12
           6       0.91      1.00      0.95        21
           7       0.93      0.93      0.93        15
           8       1.00      0.89      0.94        18
           9       0.67      0.89      0.76        18
          10       1.00      1.00      1.00        12
          11       1.00      1.00      1.00        17
          12       0.94      0.83      0.88        18
          13       1.00      1.00      1.00        12
          14       0.94      0.94      0.94        17
          15       1.00      0.85      0.92        13
          16       1.00      1.00      1.00        18
          17       1.00      1.00      1.00        15
          18       0.94      1.00      0.97        17
          19       0.90      1.00      0.95         9
          20       0.71      1.00      0.83        10
          21       1.00      1.00      1.00        16
          22       1.00      1.00      1.00        13
          23       0.94      0.84      0.89        19
          24       0.86      1.00      0.92        12
          25       0.92      0.92      0.92        13
          26       0.94      1.00      0.97        15
          27       1.00      0.95      0.97        19
          28       0.94      0.84      0.89        19
          29       0.83      1.00      0.90        19
```

```
           30       0.86      1.00      0.92        12
           31       0.96      1.00      0.98        23
           32       1.00      1.00      1.00        11
           33       1.00      0.80      0.89        15
           34       1.00      0.80      0.89        20
           35       1.00      0.79      0.88        14
           36       1.00      0.67      0.80        15
           37       1.00      0.90      0.95        20

    accuracy                           0.93       603
   macro avg       0.94      0.93      0.93       603
weighted avg       0.94      0.93      0.93       603
```

結果說明：

1. 測試資料之準確率 93%。
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，模型的整體性表現還不錯。

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: 1.49%
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        17
           1       0.00      0.00      0.00        19
           2       0.00      0.00      0.00        19
           3       0.00      0.00      0.00        14
           4       0.00      0.00      0.00        17
           5       0.00      0.00      0.00        12
           6       0.00      0.00      0.00        21
           7       0.00      0.00      0.00        15
           8       0.00      0.00      0.00        18
           9       0.00      0.00      0.00        18
          10       0.00      0.00      0.00        12
          11       0.00      0.00      0.00        17
          12       0.00      0.00      0.00        18
          13       0.00      0.00      0.00        12
          14       0.00      0.00      0.00        17
          15       0.00      0.00      0.00        13
          16       0.00      0.00      0.00        18
          17       0.00      0.00      0.00        15
          18       0.00      0.00      0.00        17
          19       0.01      1.00      0.03         9
          20       0.00      0.00      0.00        10
          21       0.00      0.00      0.00        16
```

```
           22       0.00      0.00      0.00        13
           23       0.00      0.00      0.00        19
           24       0.00      0.00      0.00        12
           25       0.00      0.00      0.00        13
           26       0.00      0.00      0.00        15
           27       0.00      0.00      0.00        19
           28       0.00      0.00      0.00        19
           29       0.00      0.00      0.00        19
           30       0.00      0.00      0.00        12
           31       0.00      0.00      0.00        23
           32       0.00      0.00      0.00        11
           33       0.00      0.00      0.00        15
           34       0.00      0.00      0.00        20
           35       0.00      0.00      0.00        14
           36       0.00      0.00      0.00        15
           37       0.00      0.00      0.00        20

    accuracy                           0.01       603
   macro avg       0.00      0.03      0.00       603
weighted avg       0.00      0.01      0.00       603
```

結果說明：

1. 測試資料之準確率 1%。
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，模型的整體性表現很糟。

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: 60.36%
              precision    recall  f1-score   support

           0       0.80      0.71      0.75        17
           1       1.00      0.68      0.81        19
           2       1.00      0.32      0.48        19
           3       0.09      0.93      0.17        14
           4       1.00      0.76      0.87        17
           5       0.78      0.58      0.67        12
           6       0.50      0.05      0.09        21
           7       0.50      0.53      0.52        15
           8       1.00      0.56      0.71        18
           9       1.00      0.56      0.71        18
          10       1.00      0.58      0.74        12
          11       1.00      0.41      0.58        17
          12       1.00      0.50      0.67        18
          13       0.89      0.67      0.76        12
```

```
           14       1.00      0.82      0.90        17
           15       1.00      0.69      0.82        13
           16       0.92      0.61      0.73        18
           17       0.85      0.73      0.79        15
           18       0.64      0.53      0.58        17
           19       1.00      0.89      0.94         9
           20       0.90      0.90      0.90        10
           21       1.00      0.56      0.72        16
           22       0.92      0.92      0.92        13
           23       1.00      0.42      0.59        19
           24       1.00      0.67      0.80        12
           25       1.00      0.54      0.70        13
           26       1.00      0.87      0.93        15
           27       0.86      0.63      0.73        19
           28       0.71      0.26      0.38        19
           29       1.00      0.47      0.64        19
           30       0.33      0.83      0.48        12
           31       0.89      0.35      0.50        23
           32       1.00      0.91      0.95        11
           33       0.26      0.73      0.39        15
           34       1.00      0.65      0.79        20
           35       0.78      0.50      0.61        14
           36       0.33      0.93      0.48        15
           37       0.93      0.65      0.76        20

    accuracy                           0.60       603
   macro avg       0.84      0.63      0.67       603
weighted avg       0.84      0.60      0.66       603
```

結果說明：

1. 測試資料之準確率 60%。
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 0.84、0.60、0.66，這表示考慮到每個類別的樣本數量後，模型的整體性表現仍需加強。

分類器性能：

kernel(linear)表現最好，測試資料之準確率大約為 93%。

(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

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

activation = "logistic"
```

```
opts = dict( verbose = False, \
activation = activation, tol = 1e-6, max_iter = int(1e8))
solver = "lbfgs" # not suitable here
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))
```

```
c:\Users\wesley\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\neural_network\_multilayer_perceptron.py:546:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res,
self.max_iter)
```

	precision	recall	f1-score	support
0	0.92	0.65	0.76	17
1	0.89	0.94	0.92	18
2	0.67	0.74	0.70	19
3	0.73	0.89	0.80	18
4	1.00	0.88	0.93	16
5	0.83	0.79	0.81	19
6	0.84	0.80	0.82	20
7	0.78	0.88	0.82	24
8	1.00	0.83	0.91	18
9	1.00	0.71	0.83	17
10	0.90	0.69	0.78	13
11	0.91	0.67	0.77	15
12	0.80	0.80	0.80	15
13	0.82	0.82	0.82	11
14	0.78	0.93	0.85	15
15	0.71	1.00	0.83	12
16	0.94	0.94	0.94	16
17	0.94	1.00	0.97	16
18	0.59	0.91	0.71	11
19	0.84	0.84	0.84	19
20	0.92	1.00	0.96	12
21	0.91	0.95	0.93	22
22	0.89	0.80	0.84	10
23	0.67	0.83	0.74	12
24	0.85	0.73	0.79	15
25	0.87	0.87	0.87	15
26	0.83	1.00	0.91	10
27	0.67	0.62	0.65	16
28	0.84	0.89	0.86	18

```
          29       0.82      0.82      0.82        17
          30       0.76      0.87      0.81        15
          31       1.00      0.67      0.80        15
          32       0.62      0.91      0.74        11
          33       0.90      0.86      0.88        21
          34       0.93      0.87      0.90        15
          35       0.92      0.92      0.92        13
          36       0.94      0.89      0.91        18
          37       1.00      0.79      0.88        19

    accuracy                           0.84       603
   macro avg       0.85      0.84      0.84       603
weighted avg       0.85      0.84      0.84       603
```

```
# 使用函數訓練模型
train_MLP("adam", X_train, y_train, X_test, y_test)
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        15
           1       0.00      0.00      0.00        14
           2       0.00      0.00      0.00        19
           3       0.00      0.00      0.00        21
           4       0.00      0.00      0.00        18
           5       0.00      0.00      0.00        17
           6       0.00      0.00      0.00        18
           7       0.00      0.00      0.00        15
           8       0.00      0.00      0.00        13
           9       0.00      0.00      0.00        16
          10       0.00      0.00      0.00         8
          11       0.00      0.00      0.00        16
          12       0.00      0.00      0.00        21
          13       0.00      0.00      0.00        11
          14       0.00      0.00      0.00        15
          15       0.00      0.00      0.00        15
          16       0.00      0.00      0.00        16
          17       0.00      0.00      0.00        15
          18       0.00      0.00      0.00        15
          19       0.00      0.00      0.00        19
          20       0.00      0.00      0.00        21
          21       0.00      0.00      0.00        13
          22       0.00      0.00      0.00        11
          23       0.00      0.00      0.00        10
          24       0.00      0.00      0.00        17
          25       0.00      0.00      0.00        17
          26       0.00      0.00      0.00        17
          27       0.00      0.00      0.00        15
          28       0.00      0.00      0.00        18
          29       0.00      0.00      0.00        19
```

```
           30       0.06      0.25      0.10        20
           31       0.00      0.00      0.00        14
           32       0.00      0.00      0.00        18
           33       0.00      0.00      0.00        16
           34       0.00      0.00      0.00        16
           35       0.00      0.00      0.00        16
           36       0.02      0.67      0.03        12
           37       0.00      0.00      0.00        16

    accuracy                           0.02       603
   macro avg       0.00      0.02      0.00       603
weighted avg       0.00      0.02      0.00       603


train_MLP("sgd", X_train, y_train, X_test, y_test)

              precision    recall  f1-score   support

           0       0.00      0.00      0.00        13
           1       0.00      0.00      0.00        19
           2       0.00      0.00      0.00        12
           3       0.00      0.00      0.00         9
           4       0.00      0.00      0.00        16
           5       0.00      0.00      0.00        12
           6       0.00      0.00      0.00        14
           7       0.00      0.00      0.00        13
           8       0.00      0.00      0.00        15
           9       0.00      0.00      0.00        13
          10       0.00      0.00      0.00        19
          11       0.00      0.00      0.00        18
          12       0.00      0.00      0.00        13
          13       0.00      0.00      0.00        22
          14       0.00      0.00      0.00        15
          15       0.00      0.00      0.00        14
          16       0.00      0.00      0.00        13
          17       0.00      0.00      0.00        19
          18       0.00      0.00      0.00        23
          19       0.00      0.00      0.00        19
          20       0.00      0.00      0.00        14
          21       0.00      0.00      0.00        24
          22       0.00      0.00      0.00        17
          23       0.00      0.00      0.00        21
          24       0.00      0.00      0.00        15
          25       0.00      0.00      0.00        14
          26       0.00      0.00      0.00        18
          27       0.00      0.00      0.00        14
          28       0.00      0.00      0.00        21
          29       0.01      1.00      0.03         8
          30       0.00      0.00      0.00        12
          31       0.00      0.00      0.00        12
```

```
           32       0.00      0.00      0.00        19
           33       0.00      0.00      0.00        14
           34       0.00      0.00      0.00        16
           35       0.00      0.00      0.00        20
           36       0.00      0.00      0.00        14
           37       0.00      0.00      0.00        19

    accuracy                           0.01       603
   macro avg       0.00      0.03      0.00       603
weighted avg       0.00      0.01      0.00       603
```

分類器性能：

演算法(lbfgs)表現最好，測試資料之準確率大約為 84%。

(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

```
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 = (600,)
    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))


# 使用函數訓練模型
train_MLP_with_PCA("adam", X_train, y_train, X_test, y_test)
```

```
              precision    recall  f1-score   support

           0       0.76      0.87      0.81        15
           1       1.00      0.86      0.92        14
           2       0.86      1.00      0.93        19
```

```
              3       0.95      0.90      0.93        21
              4       0.95      1.00      0.97        18
              5       0.84      0.94      0.89        17
              6       1.00      0.89      0.94        18
              7       0.93      0.87      0.90        15
              8       1.00      0.92      0.96        13
              9       1.00      0.81      0.90        16
             10       1.00      1.00      1.00         8
             11       1.00      0.88      0.93        16
             12       1.00      0.90      0.95        21
             13       0.85      1.00      0.92        11
             14       0.61      0.93      0.74        15
             15       0.87      0.87      0.87        15
             16       1.00      0.94      0.97        16
             17       1.00      1.00      1.00        15
             18       0.80      0.80      0.80        15
             19       0.94      0.84      0.89        19
             20       0.95      0.86      0.90        21
             21       0.86      0.92      0.89        13
             22       0.85      1.00      0.92        11
             23       1.00      0.90      0.95        10
             24       1.00      0.94      0.97        17
             25       1.00      0.76      0.87        17
             26       0.80      0.94      0.86        17
             27       1.00      0.73      0.85        15
             28       0.94      0.83      0.88        18
             29       1.00      0.95      0.97        19
             30       1.00      0.95      0.97        20
             31       0.88      1.00      0.93        14
             32       0.82      1.00      0.90        18
             33       0.94      0.94      0.94        16
             34       0.93      0.88      0.90        16
             35       0.71      0.75      0.73        16
             36       0.75      1.00      0.86        12
             37       0.93      0.88      0.90        16

    accuracy                           0.90       603
   macro avg       0.91      0.91      0.91       603
weighted avg       0.92      0.90      0.91       603
```

結果說明：

1. 測試資料之準確率 90%。
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 0.92、0.90、0.91，這表示考慮到每個類別的樣本數量後，模型的整體性表現還不錯。

```
# 使用函數訓練模型
train_MLP_with_PCA("sgd", X_train, y_train, X_test, y_test)
```

	precision	recall	f1-score	support
0	0.38	0.53	0.44	15
1	0.52	0.93	0.67	14
2	0.61	0.74	0.67	19
3	0.59	0.48	0.53	21
4	0.88	0.78	0.82	18
5	0.65	0.65	0.65	17
6	0.75	0.50	0.60	18
7	0.59	0.67	0.62	15
8	0.60	0.92	0.73	13
9	0.67	0.62	0.65	16
10	0.73	1.00	0.84	8
11	0.83	0.62	0.71	16
12	1.00	0.62	0.76	21
13	0.54	0.64	0.58	11
14	0.75	0.60	0.67	15
15	0.58	0.47	0.52	15
16	0.83	0.62	0.71	16
17	0.69	0.60	0.64	15
18	0.67	0.53	0.59	15
19	1.00	0.79	0.88	19
20	1.00	0.52	0.69	21
21	0.67	0.46	0.55	13
22	0.62	0.73	0.67	11
23	1.00	0.60	0.75	10
24	0.82	0.53	0.64	17
25	0.73	0.47	0.57	17
26	0.64	0.82	0.72	17
27	0.53	0.53	0.53	15
28	0.89	0.44	0.59	18
29	0.80	0.42	0.55	19
30	0.62	0.75	0.68	20
31	0.50	0.79	0.61	14
32	1.00	0.56	0.71	18
33	0.25	0.69	0.37	16
34	0.53	0.56	0.55	16
35	1.00	0.56	0.72	16
36	0.42	0.83	0.56	12
37	0.39	0.69	0.50	16
accuracy			0.63	603
macro avg	0.69	0.64	0.64	603
weighted avg	0.70	0.63	0.64	603

結果說明：

1. 測試資料之準確率 63%。

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 0.70、0.63、0.64，這表示考慮到每個類別的樣本數量後，模型的整體性表現仍需加強。

```
# 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       0.57      0.87      0.68        15
           1       0.74      1.00      0.85        14
           2       0.94      0.89      0.92        19
           3       0.89      0.76      0.82        21
           4       0.80      0.89      0.84        18
           5       0.79      0.65      0.71        17
           6       0.85      0.61      0.71        18
           7       0.75      0.80      0.77        15
           8       0.86      0.92      0.89        13
           9       0.80      0.75      0.77        16
          10       0.62      1.00      0.76         8
          11       0.92      0.75      0.83        16
          12       0.93      0.67      0.78        21
          13       0.92      1.00      0.96        11
          14       0.92      0.80      0.86        15
          15       0.73      0.73      0.73        15
          16       0.80      0.75      0.77        16
          17       1.00      0.87      0.93        15
          18       0.64      0.60      0.62        15
          19       0.83      0.79      0.81        19
          20       0.75      0.71      0.73        21
          21       0.73      0.85      0.79        13
          22       0.62      0.91      0.74        11
          23       0.78      0.70      0.74        10
          24       0.69      0.65      0.67        17
          25       0.88      0.88      0.88        17
          26       0.94      0.94      0.94        17
          27       0.71      0.80      0.75        15
          28       0.79      0.61      0.69        18
          29       0.65      0.79      0.71        19
          30       0.89      0.85      0.87        20
          31       1.00      0.93      0.96        14
          32       1.00      0.78      0.88        18
          33       0.76      0.81      0.79        16
          34       0.82      0.56      0.67        16
          35       0.64      0.88      0.74        16
          36       0.73      0.92      0.81        12
          37       0.59      0.62      0.61        16

    accuracy                           0.79       603
   macro avg       0.80      0.80      0.79       603
weighted avg       0.80      0.79      0.79       603
```

結果說明 :

1. 測試資料之準確率 79%。
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 0.80、0.79、0.79，這表示考慮到每個類別的樣本數量後，模型的整體性表現仍需加強。

分類器性能：

演算法(adam)表現最好，測試資料之準確率大約為 90%。

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 45 個) : accuracy 大約 92%

支援向量機 (Support Vector Machine)

1. 原始資料 : accuracy 大約 91%
2. 主成分資料(取前 45 個) : accuracy 大約 93%

神經網路 (Neural Network)

1. 原始資料 : accuracy 大約 84%
2. 主成分資料(取前 45 個) : accuracy 大約 90%

結果:

1. 在原始資料下，多元羅吉斯回歸的測試資料之準確率最高，達 97；再來是支援向量機，測試資料之準確率達 91%；最後神經網路的測試資料之準確率也有 84%，表現也不錯。
2. 主成分資料(取前 45 個)下，支援向量機的測試資料之準確率最高，達 93%；再來是多元羅吉斯回歸，測試資料之準確率達 92%；最後神經網路的測試資料之準確率也有 90%，表現也不錯。

個人見解 :

1. 多元羅吉斯回歸和支援向量機在原始資料和主成分資料上的表現都優於神經網路，可能因為這兩種模型能更好地處理這組資料的複雜度和分佈。

2. 神經網路在降維後的主成分資料上的表現下降，可能是由於資訊損失或模型複雜度過高。

3. 雖然 Yale Face 資料集相對於其他兩個資料集大了許多，但還是不夠大，神經網路可能無法充分學習，而多元羅吉斯回歸和支援向量機作為較簡單的模型，則能在小型資料集上表現較好。

4. 資料量對機器學習花的時間影響很大，葡萄酒的資料集的每個程式一分鐘內就可以跑完，而 Yale Face 資料集的每個程式都要多花很多時間，甚至有些程式跑了好幾個小時都跑不完，因此這個資料集我沒有用網格搜索（GridSearchCV）來尋找最佳的參數組合，而是改成手動調整參數，盡量產生最佳解。