# Machine Learning

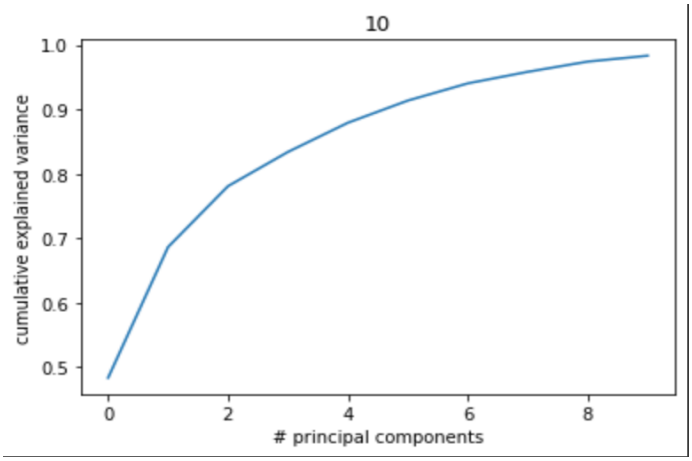
Homework 3 report b09508004 陳祈曄

（如果助教在跑 code 的時候出問題，再麻煩跟我說，因為我是在 colab 上跑的，有些路徑可能需要調整，謝謝！）

1.

用 PCA 的方式做 Dimension reduction，透過以下程式碼去選擇能讓 PCA 投影後 cumulative variance 大於 95%以上的 feature 數，決定要降到幾維，我最後是以 10 個 features 去做分類：因為他的 cumulative variance 到 0.983，代表這 10 個 eigenvectors 的投影結果可以達到全部 variance 的 98%，而且也降了一半的維數，達到降維的目的。

```
# 選feature
cov_mat = np.dot(scaled_data.T,scaled_data)
eigen_vals, eigen_vecs = np.linalg.eig(cov_mat)
np.round(eigen_vals,3)
np.round(eigen_vecs,3)
sum = 0
summation = eigen_vals.sum(axis = 0)
for i in range(len(eigen_vals)):
  sum += eigen_vals[i]
  if (sum/summation) >= 0.9:
    print((i+1))
    print((sum/summation))
```




```
6
0.9136153255180326
7
0.9403011034065631
8
0.958204255738999
9
0.9740781614350074
10
0.9832010350929792
11
0.9890648841814761
12
0.9930602931449081
13
0.9957449752937764
14
0.9973922962050938
15
0.9983282294405039
16
0.9991580079767666
17
0.9997679236187792
18
0.9999848299118933
19
0.9999997291964614
20
1.0
```

2.

下面是我自己建的 ANN model：我建了兩層 hidden layer 和一層 output layer，常見的 hidden layer 的 activation function 為 relu，而 output layer 的 activation function 則為 softmax，適合多分類使用。

至於層數和 neuron 數，我是嘗試幾個後找到一個訓練結果的 accuracy 和 validation accuracy 都表現的還不錯的。

```
X_train, X_test, y_train, y_test = train_test_split(L, encoded_label, test_size=0.2, random_state=None)

model = Sequential()
# add hidden layer
model.add(Dense(units=32, kernel_initializer='normal', activation='relu'))
model.add(Dense(units=64, kernel_initializer='normal', activation='relu'))
# Add output layer
model.add(Dense(units=16, kernel_initializer='normal', activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy',optimizer='adam', metrics=['accuracy'])

model.fit(X_train, y_train, validation_split=0.2, epochs=25, batch_size=100)

%cd '/content/gdrive/MyDrive/Colab Notebooks/機器學習/Hw3 陳祈曄 b09508004'
model.save('Ann_model')

ann = load_model('Ann_model')
```

```
Epoch 10/25
1/1 [==============================] - 0s 29ms/step - loss: 2.7003 - accuracy: 0.7846 - val_loss: 2.6894 - val_accuracy: 0.6471
Epoch 11/25
1/1 [==============================] - 0s 31ms/step - loss: 2.6894 - accuracy: 0.8154 - val_loss: 2.6776 - val_accuracy: 0.7059
Epoch 12/25
1/1 [==============================] - 0s 31ms/step - loss: 2.6777 - accuracy: 0.8308 - val_loss: 2.6650 - val_accuracy: 0.7059
Epoch 13/25
1/1 [==============================] - 0s 31ms/step - loss: 2.6652 - accuracy: 0.8462 - val_loss: 2.6515 - val_accuracy: 0.8235
Epoch 14/25
1/1 [==============================] - 0s 31ms/step - loss: 2.6518 - accuracy: 0.8462 - val_loss: 2.6373 - val_accuracy: 0.8824
Epoch 15/25
1/1 [==============================] - 0s 38ms/step - loss: 2.6374 - accuracy: 0.8615 - val_loss: 2.6220 - val_accuracy: 0.8824
Epoch 16/25
1/1 [==============================] - 0s 31ms/step - loss: 2.6218 - accuracy: 0.8769 - val_loss: 2.6056 - val_accuracy: 0.8824
Epoch 17/25
1/1 [==============================] - 0s 31ms/step - loss: 2.6051 - accuracy: 0.8769 - val_loss: 2.5879 - val_accuracy: 0.8824
Epoch 18/25
1/1 [==============================] - 0s 37ms/step - loss: 2.5871 - accuracy: 0.8769 - val_loss: 2.5690 - val_accuracy: 0.8824
Epoch 19/25
1/1 [==============================] - 0s 33ms/step - loss: 2.5677 - accuracy: 0.8769 - val_loss: 2.5488 - val_accuracy: 0.8824
Epoch 20/25
1/1 [==============================] - 0s 35ms/step - loss: 2.5468 - accuracy: 0.8769 - val_loss: 2.5272 - val_accuracy: 0.8824
Epoch 21/25
1/1 [==============================] - 0s 30ms/step - loss: 2.5244 - accuracy: 0.8769 - val_loss: 2.5043 - val_accuracy: 0.9412
Epoch 22/25
1/1 [==============================] - 0s 28ms/step - loss: 2.5003 - accuracy: 0.8769 - val_loss: 2.4799 - val_accuracy: 0.9412
Epoch 23/25
1/1 [==============================] - 0s 44ms/step - loss: 2.4745 - accuracy: 0.8769 - val_loss: 2.4538 - val_accuracy: 0.9412
Epoch 24/25
1/1 [==============================] - 0s 30ms/step - loss: 2.4469 - accuracy: 0.8769 - val_loss: 2.4261 - val_accuracy: 0.9412
Epoch 25/25
1/1 [==============================] - 0s 31ms/step - loss: 2.4175 - accuracy: 0.8615 - val_loss: 2.3968 - val_accuracy: 0.9412
```

透過製造 confusion matrix 來計算 accuracy, sensitivity, specificity，並把五組的這三個值存到三個陣列，計算平均值和標準差。

Logistic regression

```
[[11  1]
 [ 2  7]]
Accuracy :  0.8571428571428571
Sensitivity :  0.9166666666666666
Specificity :  0.7777777777777778
[[10  2]
 [ 3  6]]
Accuracy :  0.7619047619047619
Sensitivity :  0.8333333333333334
Specificity :  0.6666666666666666
[[13  0]
 [ 1  7]]
Accuracy :  0.9523809523809523
Sensitivity :  1.0
Specificity :  0.875
[[12  0]
 [ 2  6]]
Accuracy :  0.9
Sensitivity :  1.0
Specificity :  0.75
[[13  0]
 [ 1  6]]
Accuracy :  0.95
Sensitivity :  1.0
Specificity :  0.8571428571428571


logi_acc : 0.88(+-)0.07
logi_sensi : 0.95(+-)0.07
logi_speci : 0.79(+-)0.08
```

SVM

```
[[12  0]
 [ 2  7]]
Accuracy :  0.9047619047619048
Sensitivity :  1.0
Specificity :  0.7777777777777778
[[10  2]
 [ 2  7]]
Accuracy :  0.8095238095238095
Sensitivity :  0.8333333333333334
Specificity :  0.7777777777777778
[[13  0]
 [ 1  7]]
Accuracy :  0.9523809523809523
Sensitivity :  1.0
Specificity :  0.875
[[12  0]
 [ 2  6]]
Accuracy :  0.9
Sensitivity :  1.0
Specificity :  0.75
[[13  0]
 [ 1  6]]
Accuracy :  0.95
Sensitivity :  1.0
Specificity :  0.8571428571428571


svm_acc : 0.9(+-)0.05
svm_sensi : 0.97(+-)0.07
svm_speci : 0.81(+-)0.05
```

ANN model

```
[[12  0]
 [ 3  6]]
Accuracy :  0.8571428571428571
Sensitivity :  1.0
Specificity :  0.6666666666666666
3/3 [==============================] - 0s 5ms/step - loss: 2.4030 - accuracy: 0.9024
1/1 [==============================] - 0s 19ms/step
[[12  0]
 [ 3  6]]
Accuracy :  0.8571428571428571
Sensitivity :  1.0
Specificity :  0.6666666666666666
3/3 [==============================] - 0s 5ms/step - loss: 2.3505 - accuracy: 0.8780
1/1 [==============================] - 0s 17ms/step
[[13  0]
 [ 1  7]]
Accuracy :  0.9523809523809523
Sensitivity :  1.0
Specificity :  0.875
3/3 [==============================] - 0s 5ms/step - loss: 2.2722 - accuracy: 0.8675
1/1 [==============================] - 0s 20ms/step
[[12  0]
 [ 2  6]]
Accuracy :  0.9
Sensitivity :  1.0
Specificity :  0.75
3/3 [==============================] - 0s 6ms/step - loss: 2.1547 - accuracy: 0.8554
1/1 [==============================] - 0s 19ms/step
[[13  0]
 [ 2  5]]
Accuracy :  0.9
Sensitivity :  1.0
Specificity :  0.7142857142857143
ann_acc : 0.89(+-)0.04
ann_sensi : 1.0(+-)0.0
ann_speci : 0.73(+-)0.08
```

3.

| | Logistic | SVM | ANN |
|---|---|---|---|
| Accuracy | $0.88 \pm 0.07$ | $0.9 \pm 0.05$ | $0.89 \pm 0.04$ |
| Sensitivity | $0.95 \pm 0.07$ | $0.97 \pm 0.07$ | $1.0 \pm 0.0$ |
| Specificity | $0.79 \pm 0.08$ | $0.81 \pm 0.05$ | $0.73 \pm 0.08$ |

三者的 accuracy 表現都差不多在 90%上下，sensitivity 則是以我們的 ANN 最高，表示能靈敏判斷真正有病症的患者，但在 Specification 中，ANN 的表現卻是最差的，表示沒病症的患者有可能被檢驗錯誤。

就結果而言，我認為三者之中效果最好的是 SVM，accuracy 和 Specification 都最高，sensitivity 也足夠好，而我認為自己建立的 ANN 模型還不足夠穩定，數值變動較大，可能還有待改善！