國立中山大學資訊工程學系 - 期中專題報告

陳闈霆	劉佳瑜	黃奕瑋
B124020027	B124020018	B094020046

郭楨君	王濬瑋
B094020017	B074011005

摘要

本次報告使用 KNN(K-近鄰演算法)、Decision Tree(決策樹)、Random Forest(隨機森林)、Logistic Regression(邏輯回歸)、K-means(k-平均演算法)以及 MLP(深度學習-多層感知器)等演算法，針對觀測者是否患有糖尿病進行分類。鑒於資料量偏少，我們採用了集成學習方法 Bagging(自助聚合)，建構由多個決策樹組成的隨機森林模型。且透過代價敏感學習和使用 sklearn 套件的 GridSearchCV 函式，搜索最佳參數以優化模型的泛化能力。本文將以視覺化方式呈現模型結果，並深入探討不同演算法之模型表現。 **關鍵詞：KNN、Logistic Regression、Decision Tree、Random Forest、MLP。**

1. 簡介

1.1 研究背景

本次報告使用之資料集有實驗 A 與實驗 B，其中訓練集與測試集個數分別為 567/201、668/100，資料集總數皆為 768，而實驗 B 的測試集為實驗 A 的半數。其中，有 Pregnancies(懷孕次數)、Glucose(血糖濃度)、BloodPressure(血壓)、SkinThickness(脂肪含量)、Insulin(胰島素)、BMI、DiabetesPedigreeFunction(糖尿病函數)、Age、Outcome(1:有糖尿病 ; 0:無糖尿病)等特徵。

1.2 研究動機

當今社會，資料科學已成為醫療保健、行銷、零售、政治和金融等各領域中不可或缺的技術。透過機器學習，我們能夠提前做好對未知情況的準備，因此，我們運用不同演算法，以將課堂所學應用於實務，並加深我們對資料科學的認識。

1.3 研究目的

觀察資料集數據，並使用 KNN、邏輯回歸、決策樹、隨機森林、K-means 以及 MLP 等演算法，透過特徵變數訓練模型，以預測觀測者是否患有糖尿病，最後評估模型準確率並且進行相應的改善。

1.4 研究流程

a. 資料前處理(補缺失值、資料正規化)
b. 訓練模型
c. 計算準確率
d. 調整參數與權重
e. 模型比較

2. 相關研究

Changsheng Zhu, Christian Uwa Idemudia, Wenfang Feng.(2019) "Improved logistic regression model for diabetes prediction by integrating PCA and K-means techniques."Informatics in Medicine Unlocked.

本篇論文透過 PCA、K-means 和邏輯回歸模型的整合使用，大幅提升糖尿病預測模型的性能。本論文研究使用 Pima Indian Diabetes 數據集，此數據集包含 768 筆樣本，取樣自美國亞利桑那州的女性。實驗流程為透過 PCA 進行特徵篩選後，使用 K-means 做聚類分析，然後透過邏輯回歸模型進行糖尿病預測，其分析結果的準確率高達 0.97。此外本論文還做了: XGBoost、SVM、Naïve Bayes 等模型的分析，效果不如上述，但也都證明了，整合了 PCA 和 Kmeans 的模型表現皆勝於單獨使用。

本篇論文所使用的資料集欄位與本次作業資料集欄位相同，因此在資料的預處裡上，有值得借鏡的幾個重點:

- **缺失值**：雖然本次作業並未有缺失值，但有許多有 0 值的特徵 (e.g.血糖濃度、血壓等)，以常理來說這些值在醫學上不可能為 0，因此需要進行處理。
- **正規化**：本篇論文有實行數據正規化，將其調整為 0 到 1 之間，有助於減少模型在訓練時的運行時間並且提高模型性能。

- **轉換數據**：本篇論文提及「懷孕次數」這個欄位對於資料分析沒有太大助益，因此參考了別篇論文的數據處理方式，將「懷孕次數」轉換為名義特徵（0 或 1，表示是否懷孕過）。

此外，本論文模型評估時使用：

1. 交叉驗證：10 折交叉驗證。
2. 混淆矩陣
3. 其他性能指標，如準確率、召回率。

3. 文獻探討

本章節將介紹本次報告使用的機器學習模型。

3.1 KNN(K-Nearest Neighbors)

KNN 是一種不需要訓練的分類演算法，其時間複雜度為 O(n)，n 為訓練資料的大小。KNN 的概念是相同類別的資料比較容易待在一起，因此你附近的資料是甚麼類別，你就高機率會是同個類別。KNN 演算法實行流程可分為三個步驟：1. 計算 test data 與每個 train data 之間的距離；2. 輸入 k 值以決定找出距離 test data 最短的 k 個 train data；3. 統計這 k 個樣本中最常見的類別，並且 KNN 演算法支援多種計算距離的方法，例如明氏距離、曼哈頓距離、及歐幾里得距離等。在本次報告中，我們使用歐氏距離作為 KNN 的距離計算方式（如圖 2-1 所示）。

$$D = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

圖 2-1 歐氏距離公式
（資料來源:PyInvest）

3.2 Logistic Regression

Logistic Regression，又稱邏輯回歸，與一般線性回歸不同之處在於，邏輯回歸透過 Sigmoid 函數將應變數映射於 0 到 1 之間，以有效區分不同類別的數據。如圖 2-2 所示，此圖示描述了邏輯回歸將輸入轉換為機率的過程，其中，變數 x 代表資料集的特徵變數、w 值表示權重、b 為偏權值，即模型的偏移量。這三者組合得出 z 值，再將 z 帶入 Sigmoid 函數轉換為介於 0 到 1 之間的數值 y_hat。透過這個過程，邏輯回歸便可將特徵變數的線性組合轉換為機率值，最後依據這些機率值加以將數據進行分類。

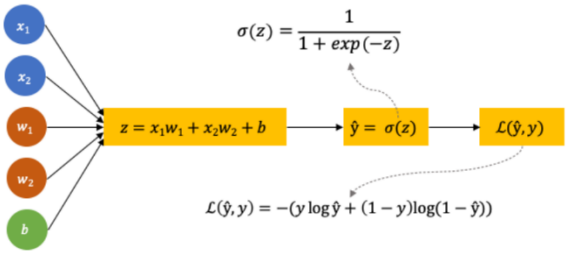

圖 2-2 Logistic Regression 公式
（資料來源: https://reurl.cc/OMOQOD）

3.3 Decision Tree

本研究將啟發式演算法套用並實作成排課系統，故選擇較經典且具不同特色的三種演算法。爬山演算法確認落入區域解的程度；利用模擬退火演算法以及基因遺傳演算法跳出區域解，且兩者分別屬於單解型 (single-solution) 與多解型 (multi-solution) 的啟發式演算法。

3.4 Random Forest

隨機森林演算法利用集成學習中的 Bagging 自助聚合 (Bootstrap aggregating)，由多個 CART 決策樹組成。這些決策樹是由同一母體透過 Bootstrap 抽樣方式形成，隨機抽樣固定數樣本組成獨立的決策樹，並且隨機採用不同特徵進行訓練，這些資料可重複使用。經過多次 Bootstrap 抽樣後，將以投票方式進行分類。相較單獨使用決策樹，隨機森林較不易產生 Overfitting 的情況。

4. 程式設計方式

4.1 KNN(K-近鄰演算法)

圖 4-1 KNN 程式流程圖

在我們的 KNN 程式中(圖 4-1)，有兩個主要的功能，第一個是看特定 K 值下的 KNN 的以下我們會以 KNN 的步驟來介紹我們的程式。

首先是資料前處理的部份，資料前處理分為兩個部分，補缺失值以及正規化資料。雖然這份資料中看起來沒有缺失值，但血液中葡萄糖濃度、

舒張壓、三頭肌皮摺厚度、胰島素濃度以及 BMI 值這五個欄位中，裡面有包含 0 的值，由於在醫學上這些值不可能是 0，因此這些其實就是缺失值，所以我們就平均值去取代這些值。由於 KNN 是計算資料間的距離，因此我們需要進行資料正規化，避免因為值域不同所產生的誤差。我們使用的正規化方式是將資料範圍縮放到 0~1 之間。

第二步驟是要決定 K 值，決定 K 值算是 KNN 中最重要的一個步驟，身為 KNN 的新手，我們也不知道如何設定最佳的參數。因此，我們設計了一個 function，他會將 K 值全部跑一遍，並將結果視覺化呈現(圖 4-2)，目的是找出最佳的 K 值。

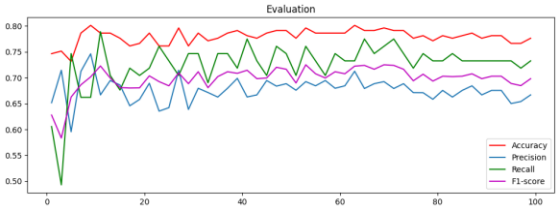

圖 4-2 KNN 準確率

第三步驟是要算出 test data 跟所有 train data 之間的距離。距離計算公式方面，我們選用的是歐式距離（Euclidean Distance）。決定距離公式後，我們要將 test data 對每個 train data 的距離以及 train data 的 Outcome 存進 list 中，方便後續計算。

第四步驟我們需要找出距離最短的 K 個 train data，我們先將前面所存的 list 依照距離 sort 過，因此現在 list 中的前 K 個值就會是距離最短的 K 個 data。

第五步驟是要找哪個類別最多。找到距離最短的 K 個 data 後，我們需要找出這 K 個 data 中，哪個類別比較多。由於我們的 Outcome 只有 0 和 1，因此我們只需要將 list 加總就可以得知有幾個 1。此外，我們也有設定 threshold，只要 true 的數量大於（K / threshold），我們就會將該 test data 判定為 true，反之則判定為 false，最後就可以得出我們所預測的答案。

最後一步驟是計算準確率，我們的程式中有計算 Accuracy、Precision、Recall、F1-score 以及 Confusion Matrix。程式中我們會先計算出 test data 的 True Positive、False Positive、True Negative 以及 False Negative，並根據這些值計算 Accuracy、Precision、Recall、F1-score 的值，以及利用視覺化圖表呈現 Confusion Matrix(圖 4-3)。

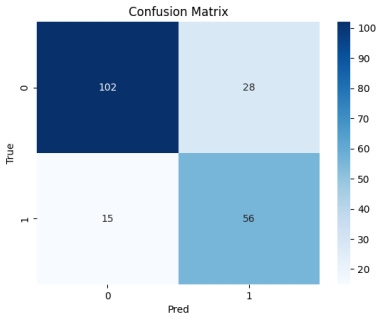

圖 4-3 KNN Confusion Matrix

- **參數設定以及實驗結果：**

在我們 KNN 的程式中，主要可以設定的參數有判定 true、false 的 threshold 值以及 K 值。

首先先來講我們是如何選定 threshold 值，只要 true 的數量大於（K / threshold），我們就會將該 test data 判定為 true，反之則判定為 false。

一開始我們 threshold 值設定為 2，也就是一半以上為 true 就判定為 true，但我們就發現出現 Recall 值太低的問題，在觀察資料集後，我們認為可能的原因是 train data 中，Outcome 為 false 的資料數量大約是 true 的兩倍(圖 4-4)，也就是說在平均分配的情況下，身邊會有三分之一的 train data 為 true，三分之二為 false，因此就有比較高的機率將 Outcome 判定為 false。為了解決這個問題，我們就將 threshold 值設定為 3，也就是只要超過三分之一為 true，就判定為 true，也可以觀察到 Recall 值上升(表 4-1、表 4-2)。

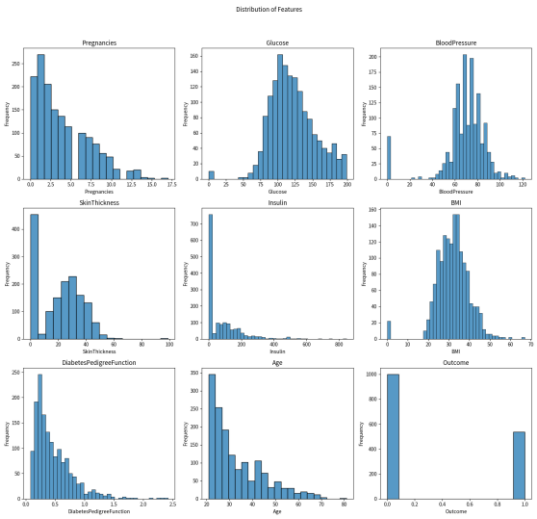

圖 4-4 資料分佈圖

實驗 A, KNN	Accuracy	Precision	Recall	F1-score
Threshol	0.83084	0.813559	0.67605	0.73846

d = 2, K = 25	5		6	1
Threshold = 2, K = 17	0.820895	0.830188	0.619718	0.709677
Threshold = 3, K = 9	0.810945	0.698795	0.816901	0.753246
Threshold = 3, K = 23	0.771144	0.619047	0.915492	0.738636

表 4-1 KNN 在不同參數下準確率，黃底為該準確率下最高值（實驗 A）

實驗 B, KNN	Accuracy	Precision	Recall	F1-score
Threshold = 2, K = 61	0.790000	0.900000	0.486486	0.631578
Threshold = 2, K = 13	0.780000	0.727272	0.648648	0.685714
Threshold = 3, K = 9	0.800000	0.697674	0.810810	0.750000
Threshold = 3, K = 35	0.780000	0.647058	0.891891	0.749999

表 4-2 KNN 在不同參數下準確率，黃底為該準確率下最高值（實驗 B）

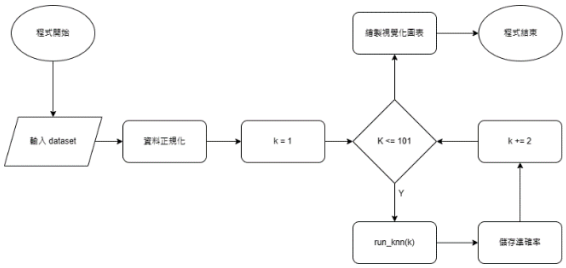

圖 4-5 find_best_k_value 流程圖

再來是我們如何設定 K 值。決定 K 值算是 KNN 中最重要的一個步驟，身為 KNN 的新手，我們也不知道如何設定最佳的參數。因此，我們設計了一個 function(圖 4-5)，他會將 K 值全部跑一遍，並且將結果以視覺化圖表呈現出來，我們就可以依照我們的需求來選擇 K 值(圖 4-6、圖 4-7、圖 4-8、圖 4-9)。

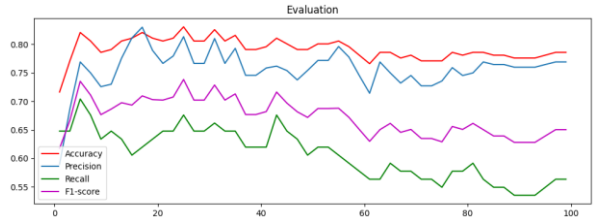

圖 4-6 Threshold = 2，不同 K 值下的準確率(實驗 A)

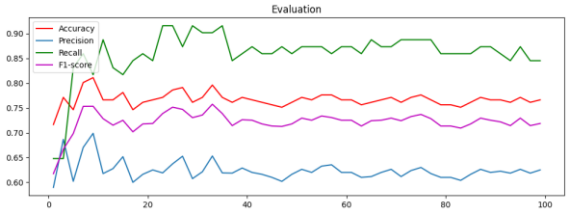

圖 4-7 Threshold = 3，不同 K 值下的準確率(實驗 A)

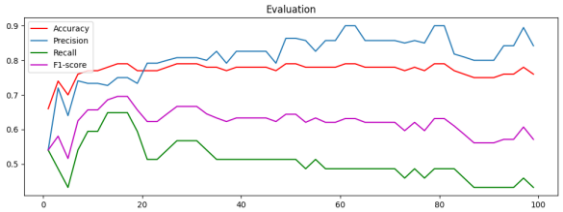

圖 4-8 Threshold = 2，不同 K 值下的準確率(實驗 B)

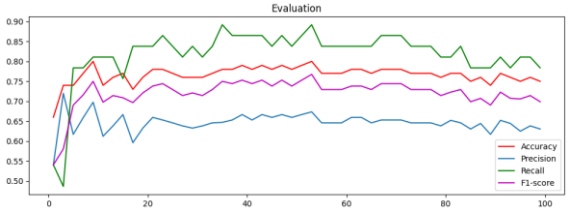

圖 4-9 Threshold = 3，不同 K 值下的準確率(實驗 B)

- **結論：**

  我們可以根據結果，查看特定 K 值的準確率以及 Confusion Matrix。

  在實驗 A 中，從表 4-1 中可以看出，當 Threshold = 2、K = 25，Accuracy 可以達到最高 0.830845；當 Threshold = 2、K = 17，Prcesion 可以達到最高 0.830188；當 Threshold = 3、K = 23，Recall 最高為 0.915492；當 Threshold = 2、K = 9，F1-score 最高為 0.753246。

  在實驗 B 中，從表 4-2 中可以看出，當 Threshold = 3、K = 9，Accuracy 可以達到最高 0.8，F1-score 也能達到最高為 0.75；當 Threshold = 2、K = 61，Prcesion 可以達到最高 0.9；當 Threshold = 3、K = 35，Recall 最高為 0.891891。

  從實驗 A、B 中可以發現，在 Threshold = 2 時，Precesion 的值會比 Recall 來的高，相反的 Threshold = 3 時 Recall 值會比較高，同時我們也可以發現，Precision 較高時就會相對的犧牲一些 Recall 值，如果我們希望能夠盡量將得到糖尿病的患者都找出來，我們可以選擇讓 Recall 值較高；反之如果我們希望不要誤判他得了糖尿病，則需要選擇讓 Precision 值較高。

- **介紹 MLP、怎麼選 mlp、調整參數、結果:**

  深度學習由多層類神經網路（Neural network）組成，這些神經網路會模仿人類大腦的運作模式，由一個個的神經元將訊息

互相傳遞，從大量的資料內學習到特定的資訊或規律。

在這次實驗中，我們有用深度學習的方式來訓練分類器，並且利用 Pytorch 套件實作。我們所使用的模型是基礎的 MLP 模型，loss function 為用來做二元分類的 BCEWithLogitsLoss，Optimizer 選擇的是 Adam，learning rate 設定為 0.0001。

我們所使用的模型的架構為是 feature_dim -> 8 -> 1 -> RELU，只過了兩層簡單的線性層，在我們測試的過程中發現只要模型稍微加深一點，就會讓準確率下降，我們認為可能的原因是資料量少，因此太深的模型 train 不起來，所以我們也選擇較淺的模型。

在 Optimizer 中，我們有嘗試過 SGD 以及 Adam，最後發現 Adam 的成效較佳，因此就選擇使用 Adam。Learning Rate 的部份因為我們的資料量很小，train 一個 epoch 很快，所以就想說讓 lr 設定小一點，讓他可以慢慢收斂，以得到比較好的結果。

最後，我們 train 了 100 個 epoch，並以視覺化圖表呈現其 Loss、Accuracy、Precision、Recall 以及 F1-score 值。

實驗 A 的訓練成果如下，從圖表中可以看出大約在 40 個 epoch 左右準確率就差不多沒有變動了(圖 4-10、圖 4-11、圖 4-12、圖 4-13)，準確率最高的是第 14 個 epoch(圖 4-11)，Test Accuracy = 0.800995、Precision = 0.738462、Recall = 0.676056、F1-score = 0.705882。

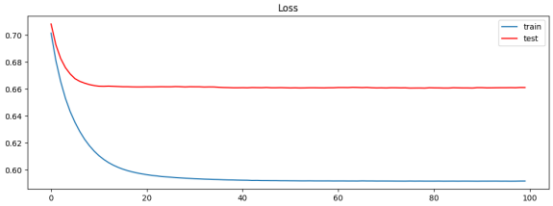

圖 4-10 Train、Test 的 Loss 變化(實驗 A)

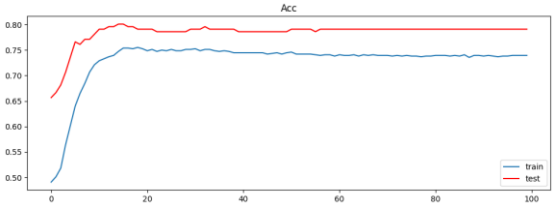

圖 4-11 Train、Test 的 Accuracy 變化(實驗 A)

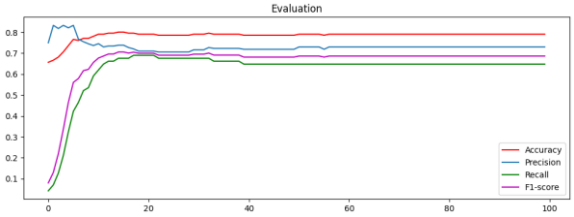

圖 4-12 Accuracy、Precision、Recall、F1-score 變化(實驗 A)

```
==================== Epoch 14 ====================
Train Acc: 0.747382 Train Loss: 0.601900
Test Acc: 0.800995 Test Loss: 0.661969
Test Precision: 0.738462
Test Recall: 0.676056
Test F1-score: 0.705882
```

圖 4-13 實驗 A 準確率最佳的 epoch

實驗 B 的訓練成果如下，從圖表中可以看出一樣大約在 40 個 epoch 左右準確率就差不多沒有變動了(圖 4-14、圖 4-15、圖 4-16)，準確率最高的是第 9 個 epoch，Test Accuracy = 0.81、Precision = 0.846154、Recall = 0.594595、F1-score = 0.698413（圖 4-17)。

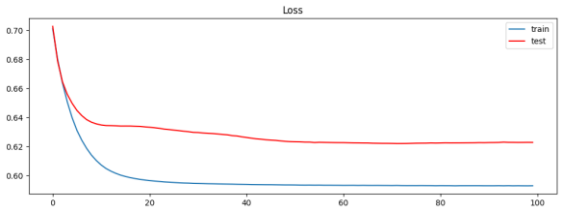

圖 4-14 Train、Test 的 Loss 變化(實驗 B)

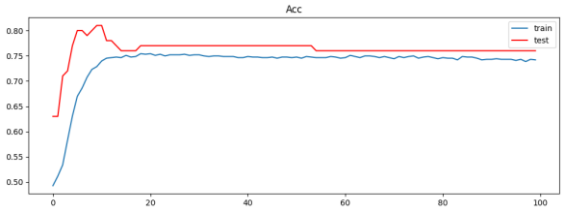

圖 4-15 Train、Test 的 Accuracy 變化(實驗 B)

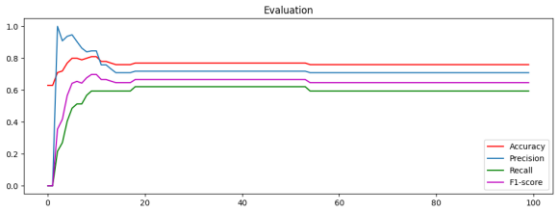

圖 4-16 Accuracy、Precision、Recall、F1-score 變化(實驗 B)

```
==================== Epoch 9 ====================
Train Acc: 0.728587 Train Loss: 0.610348
Test Acc: 0.810000 Test Loss: 0.635566
Test Precision: 0.846154
Test Recall: 0.594595
Test F1-score: 0.698413
```

圖 4-17 實驗 B 準確率最佳的 epoch

實驗 A	Accuracy	Precision	Recall	F1-score
Threshold = 2, K = 25	0.830845	0.813559	0.676056	0.738461
Threshold = 2 K = 17	0.820895	0.830188	0.619718	0.709677
Threshold = 3 K = 9	0.810945	0.698795	0.816901	0.753246

Threshold = 3, K = 23	0.771144	0.619047	0.915492	0.738636
MLP	0.800995	0.738462	0.676056	0.705882

表 4-3 實驗 A 總表

實驗 B, KNN	Accuracy	Precision	Recall	F1-score
Threshold = 2, K = 61	0.790000	0.900000	0.486486	0.631578
Threshold = 2, K = 13	0.780000	0.727272	0.648648	0.685714
Threshold = 3, K = 9	0.800000	0.697674	0.810810	0.750000
Threshold = 3, K = 35	0.780000	0.647058	0.891891	0.749999
MLP	0.810000	0.846154	0.594595	0.698413

表 4-4 實驗 B 總表

4.2 Logistic Regression(邏輯回歸)

使用 scikit-learn 的 Logistic Regression，分別對實驗 A 和實驗 B 做分析。

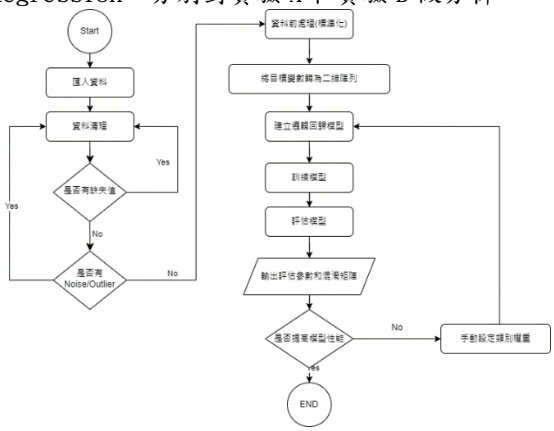

圖 4-18 Logistic Regression 程式流程圖

- **實驗結果:**

未調整參數和目標變數權重:

---實驗 A---

accuracy： 0.8059701492537313
recall： 0.5774647887323944
precision： 0.82
F1 分數： 0.6776859504132231

---實驗 B---

accuracy： 0.83
recall： 0.5945945945945946
precision： 0.9166666666666666
F1 分數： 0.7213114754098361

發現 accuracy 雖然高，但是 recall 很低，這個結果表示尿病患者誤判為無病的機率高，因此嘗試代價敏感學習，將誤判正樣本為負樣本的代價設置得更高。手動設定類別權重(目標變數) ： 正樣本和負樣本的比率調整為 2:1

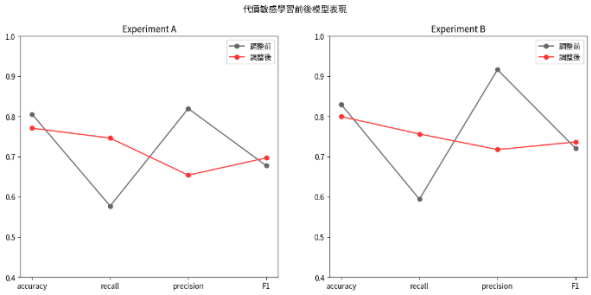

圖 4-19 設定權重後的模型比較

1. 實驗 A 和實驗 B 的 Accuracy 下降 ：

經過將目標變數正樣本權重加重後，模型更傾向於將邊界情況劃歸為正樣本。這種策略雖然可以提高正樣本的 recall（減少將真正樣本誤判為負樣本的情況），但同時可能會導致更多的假陽性（將實際上是負樣本-無糖尿病的情況，誤判為正樣本-有糖尿病）。

2. 準確率的下降:

當模型開始產生更多的假陽性時，雖然它可能正確地識別了更多的真正樣本，但同時也錯誤地將更多的負樣本標記為正樣本。如果整體增加的假陽性數目超過了由於更好的召回率帶來的真正增加數目，那麼模型正確預測的總比例（即準確率）可能會下降。

4.3 Decision Tree(決策樹)

使用 scikit-learn 的 Decision Tree 和 Random Forest 模型，分別對實驗 A 和實驗 B 做分析，另外還使用了 GridSearchCV 和 RandomizedSearchCV 找出最適模型參數。

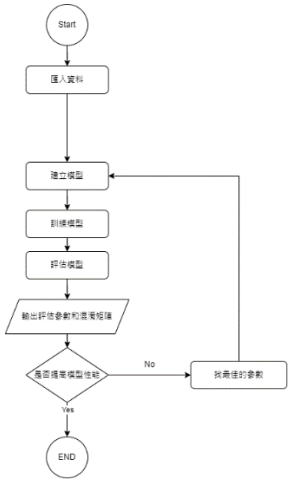

圖 4-20 Decision Tree 程式流程圖

- **實驗結果:**
  透過 GridSearchCV 找出最適模型參數，有助於防止 overfitting

---實驗 A---
最佳參數:
{'max_depth': 5, 'min_samples_leaf': 17, 'min_samples_split': 2}
最佳交叉驗證得分: 0.7583605030274803
Accuracy: 0.7611940298507462
Recall: 0.7450704225352113
Precision: 0.7393650793650793
F1 Score: 0.7418664383561644
---實驗 B---
最佳參數:
{'max_depth':7,'min_samples_leaf': 16,'min_samples_split': 2}
最佳交叉驗證得分: 0.7409942767366176
Accuracy: 0.76
Recall: 0.7425997425997426
Precision: 0.7425997425997426
F1 Score: 0.7425997425997426

4.4 Random Forest(隨機森林)

使用 RandomizedSearchCV 尋找最佳參數(考慮到參數範圍很廣，直接使用 GridSearchCV 可能會非常耗時，因為它會試圖對每一個可能的參數組合進行評估，因此使用 RandomizedSearchCV)

---實驗 A---
最佳參數:
{'n_estimators': 79, 'min_samples_split': 29,'min_samples_leaf': 30, 'max_features':'sqrt', 'max_depth': 22, 'bootstrap': True}
Accuracy: 0.7661691542288557
Recall: 0.7073672806067173
Precision: 0.7606209150326797
F1 Score: 0.7194821391454108
---實驗 B---
最佳參數:
{'n_estimators': 324, samples_split': 8, 'min_samples_leaf': 2, 'max_features': 'auto','max_depth': 38, 'bootstrap': True}
Accuracy: 0.79
Recall: 0.7496782496782497
Precision: 0.7886904761904762
F1 Score: 0.7606837606837606

- **比較決策樹和隨機森林:**
  根據結果可以得知隨機森林較決策樹有更好的預測表現

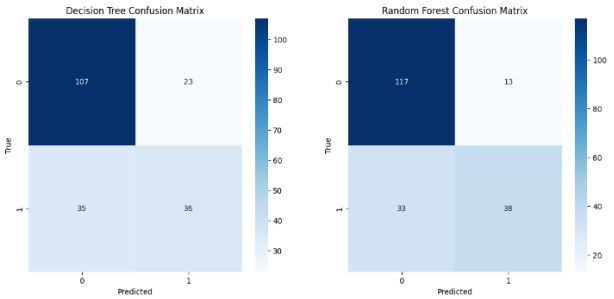

圖 4-21 實驗 A 混淆矩陣比對

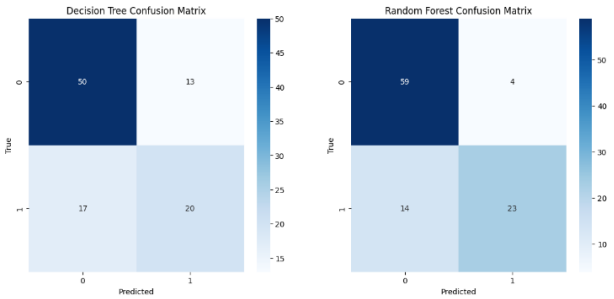

圖 4-22 實驗 B 混淆矩陣比對

4.5 PCA、K-means 和 Logistic Regression

第一步驟我們使用主成分分析（PCA）減少數據中的特徵數量，只保留重要的特徵以減少不必要的資訊干擾模型性能。PCA 受尺度的影響較大，因此我們將數據進行特徵標準化，讓每個特徵的平均為 0，變異數為 1。

以下我們用視覺化的方式，觀察 PCA 的 n_components 數量改變所帶來的模型性能變化（左圖為 Accuracy;右圖為 F1 score ），因為本次作業有 7 個欄位，因此 n_componentsgk4 設定 1 到 7，另外還可以將 n_components 設為浮點數（0.95、0.90），讓 PCA 選擇足夠多的主成分來解釋指定比例的變異性。

根據圖中的資料，我們選擇將 PCA 的的 n_components 數量設置為 0.95，繼續進行下一步驟。

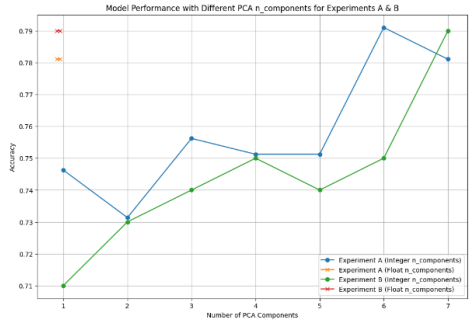

圖 4-23 Accuracy、PCA 數比對

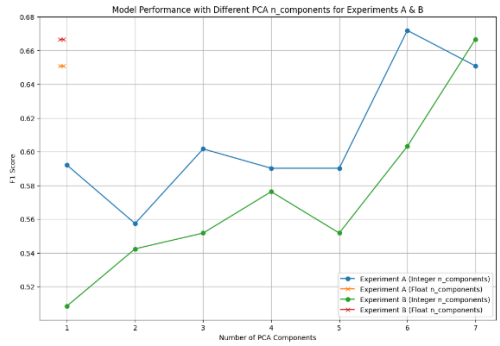

圖 4-24 F1 Score、PCA 數比對

下一步驟，我們使用 K-means 聚類演算法，將 PCA 降維後的數據分成幾個群組（clusters）。clusters 的過程是將每筆資料點指派給最近的聚類中心，目的是讓每個群組內的資料點相互之間盡可能相似，而不同群組之間的數據點則盡可能不同，這個動作可以視為一種分群。而我們可以去計算 K-means 的 cluster 數量是否影響模型性能，因此我們設定 cluster 數量為 1 到 20，同樣做視覺化觀察(PCA 設定為 n_components=0.95)，根據下圖可知，實驗 A 的 cluster=4；實驗 B 的 cluster=15 會有最高的 f1 分數，因此將此參數帶入模型。

圖 4-25 F1 Score、Cluster 數比對

---實驗 A---
'Accuracy': 0.8059701492537313,
'Recall': 0.6338028169014085,
'Precision': 0.7758620689655172,
'F1 Score': 0.6976744186046512
---實驗 B---
'Accuracy': 0.8,
'Recall': 0.5675675675675675,
'Precision': 0.84,
'F1 Score': 0.6774193548387097

然而根據結果，相關研究論文所提出的結合了 PCA、K-means 和邏輯迴歸的模型方法，在 A 資料集上的表現不如單獨使用 KNN 或邏輯迴歸的模型，但在 B 實驗中的效能表現不錯。

5. 結論

F1-score 為 Precision 和 Recall 值的調和平均數，他同時考慮了 Precisn 以及 Recall 兩個變數，因此我們取各個模型最佳的 F1-score 值做比較，結果如下表：

實驗 A				
Model	Accuracy	Recall	Precision	F1 - Score
KNN	0.81	0.81	0.69	0.75
Logistic Regression	0.77	0.77	0.64	0.70
Decission Tree	0.76	0.74	0.73	0.74
Random Forest	0.76	0.70	0.76	0.71
PCA + K-means	0.71	0.78	0.57	0.66
PCA + K-means + Logistic Regression	0.8	0.63	0.77	0.69
MLP	0.80	0.67	0.73	0.70

表 5-1 模型結果比較(實驗 A)

實驗 B				
Model	Accuracy	Recall	Precision	F1 - Score
KNN	0.80	0.81	0.69	0.75
Logistic Regression	0.8	0.75	0.71	0.73
Decission Tree	0.76	0.74	0.74	0.74
Random Forest	0.79	0.78	0.74	0.76
PCA + K-means	0.73	0.72	0.61	0.66
PCA + K-means + Logistic Regression	0.8	0.56	0.84	0.67
MLP	0.81	0.59	0.84	0.69

表 5-2 模型結果比較(實驗 B)

在實驗 A 中，KNN 的表現特為突出，Accuracy、Recall 和 F1-score 最高的都是 KNN，PCA + K-means + Logistic Regression 的方式可以得到最高的 Precision 值。

在實驗 B 中，各項指標的冠軍就較為分散，Accuracy 和 Precision 值最高的是 MLP，Recall 值最高的是 KNN，F1-score 最高的則是 Random forest。

從上述實驗中，我們第一個可以觀察到，在實驗 A 中 KNN 表現亮眼，實驗 B 中大家的表現就較為平均。我們認為這個現象可能的原因是，實驗 A 的 train data 數量相對於實驗 B 來說少很多，所以其他機器學習以及深度學習的模型在實驗 A 中可能就比較難訓練起來，所以不用訓練的 KNN 模型，表現就相對就好。

第二個可以注意到的是，KNN 的 Recall 值在兩個實驗中都是最高的，可能的原因是，我們在 KNN 的參數調整中，有特別調整 Threshold，因此在 true data 的判定上也較為敏感，所以 Recall 值上相對比其他來的高。

所以從上述可以得知，在訓練資料不多時，我們可以使用 KNN 的方式來預測，相對於模型訓練的方式可以得到更好的準確率。在訓練資料較多的情況下我們就可以嘗試其他需要訓練的模型，並且根據所需情況選擇以及調整模型，以此來達到最高的效益。

參考文獻

[1] Changsheng Zhu, Christian Uwa Idemudia, Wenfang Feng.(2019) "Improved logistic regression model for diabetes prediction by integrating PCA and K-means techniques."Informatics in Medicine Unlocked.

[2] 什麼是邏輯迴歸？ - 邏輯迴歸介紹 - AWS (amazon.com)

[3] 決策樹 Decision Tree | Medium

[4] [Day 14] 多棵決策樹更厲害：隨機森林 (Random forest) - iT 邦幫忙::一起幫忙解決難題，拯救 IT 人的一天 (ithome.com.tw)

[5] 【Python Advanced】Pandas 套件必學資料處理函數介紹與應用！| by NTU Data Analytics Club | NTU Data Analytics Club | Medium

[6] [Machine Learning] kNN 分類演算法. 最近在學 Machine Learning～ 因為要學的東西太多了… | by 林罡北 | Medium

[7] Python 機器學習-分類模型的 5 個評估指標 | Medium

[8] [Day 27] 從零開始學 Python - 科學繪圖 Matplotlib：畫著你，畫不出你的骨骼 - iT 邦幫忙::一起幫忙解決難題，拯救 IT 人的一天 (ithome.com.tw)

[9] 機器學習中的距離 - HackMD

[10] Pandas 學習筆記 常用的統計函數. 與 Numpy 相同 | by Chung-chi Huang | hccuse - 隨手筆記 | Medium

[11] 【python】sklearn 中 PCA 的使用方法 from sklearn.decomposition import pca-CSDN 博客