KNN 與他的快樂夥伴們

李承諺 (B123040032) 高紹綸 (B123040012)

賴建愷 (B123040014) 徐子皓 (B123040036) 陳彥維 (B123040039)

摘要—本報告通過對比支持向量機 (SVC)、決策樹、隨機森林、梯度提升樹、多層感知器 (Multi-layer Perceptron)、線性判別分析、二次判別分析、高斯貝氏分類、邏輯回歸及 KNN 分類演算法等不同分類算法的性能，來分析各算法在不同指標（如 accuracy、precision、recall 和 F1 值）下的表現。本報告特別著重於 KNN 分類演算法，分析其在不同 K 值下的表現，並實作其他常見資料分類演算法以進行比較。

I. INTRODUCTION

在現代機器學習中，分類算法在許多領域中有著廣泛應用，例如影像識別、語音識別、醫療診斷等。不同分類演算法各有優劣，選擇合適的模型並根據問題特性進行調整，是機器學習應用中的重要課題。本報告將對多個常見的分類算法進行性能對比，並深入探討 KNN 算法，分析其在不同參數下的效果。

II. 分類算法概述

在本實驗中，我們比較了以下幾種常見的分類算法：

- **SVC:** 透過構造最大間隔超平面進行分類。
- **Decision Tree:** 基於特徵的分割來建立分類規則。
- **Random Forest:** 一種集成學習方法，利用多個決策樹進行分類。
- **Gradient Boosting:** 基於弱分類器組合的一種強大集成方法。
- **MLP:** 由多層神經元組成的深度學習方法。
- **Linear Discriminant Analysis:** 基於最大化類別區別度的線性分類方法。
- **Quadratic Discriminant Analysis:** 與 LDA 類似，但假設每個類別的分佈是二次型的。
- **Gaussian Naive Bayes:** 基於貝氏定理，假設每個特徵相互獨立。
- **Logistic Regression:** 用於二分類問題，基於邏輯函數。
- **K Nearest Neighbor(KNN):** 基於距離度量進行分類，簡單且有效。

III. KNN 演算法的詳細分析

KNN (K Nearest Neighbor) 是一種屬於監督式學習的演算法。對於每一筆新的資料，根據它與訓練資料中所有樣本的距離，找出距離最近的 k 個鄰居，再根據這 k 個鄰居中出現最多的類別，來判斷新資料屬於哪一類。

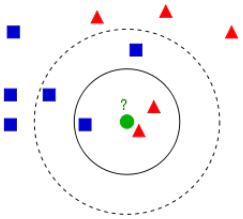


圖 1: KNN 示意圖 (內圈 k = 3, 外圈 k = 5)

A. 傳統 *KNN* 演算法

計算距離的方法有很多種，本報告採用最常見的歐式距離 (Euclidean Distance)

$$Distance(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{1}$$

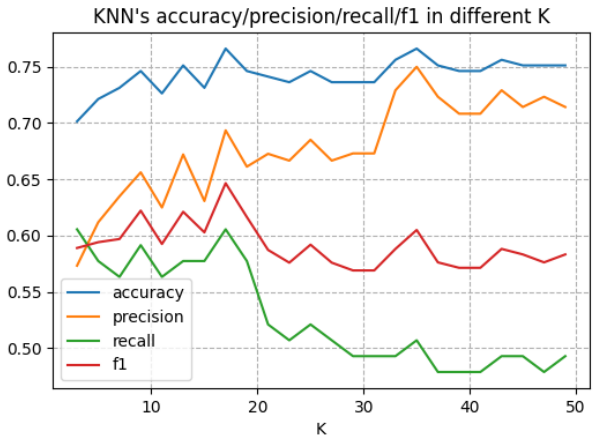
我們發現原本的 KNN 會需要一筆一筆地迴圈比對每筆測試資料與每筆訓練資料的距離，當訓練集數量是 n，測試集數量是 m 的情況，整個時間複雜度會是 $O(mn \log n)$，當資料量很大的時候效率可能會不佳，於是我們額外實作了向量化版本的 KNN。

B. 向量化 *KNN*

傳統 KNN 演算法在每筆測試資料中，需逐筆與所有訓練資料計算距離，計算成本為 $O(mn \log n)$，其中 m 為測試資料數量，n 為訓練資料數量。為了提升效

Algorithm 1 傳統 KNN 分類演算法

Input: train_set, test_set, k
predictions ← [ ]
for 每一筆測試資料 x_{test} **do**
  計算 $dis_x \leftarrow \text{EuclideanDistance}(x_{\text{test}}, \text{train\_set})$
  k_indices ← 最近的 k 筆訓練資料
  result ← k_indices 的投票
  將 *result* 加入 predictions
end for
return predictions



率，可利用 NumPy 的向量化操作，批次計算所有測試資料與訓練資料的歐式距離，大幅減少迴圈與排序的次數，總複雜度降為 $O(mn)$。

對於測試資料點 $\boldsymbol{x} \in \mathbb{R}^d$ 和訓練資料點 $\boldsymbol{y} \in \mathbb{R}^d$，歐式距離為：

$$\|\boldsymbol{x} - \boldsymbol{y}\|^2 = \sum_{i=1}^{d} (x_i - y_i)^2 \tag{2}$$

將其展開後可寫為：

$$\|\boldsymbol{x} - \boldsymbol{y}\|^2 = \|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2 - 2\boldsymbol{x} \cdot \boldsymbol{y} \tag{3}$$

這使得我們可以一次計算所有測試與訓練樣本間的距離矩陣，而不需要使用兩層迴圈。

C. k 值對 KNN 性能的影響

k 值是 KNN 中的一個重要參數，它決定了分類決策中考慮的鄰近樣本數量。較小的 K 值會使模型對噪聲敏感，而較大的 k 值則會使模型變得過於平滑，忽略一些細節。為了選擇最佳的 K 值，我們進行了不同 k 值下的測試，嘗試分析了其對準確率與其他評價指標的影響。

實驗結果表現在 $k = 17$ 時，Accuracy 和其他評估值有著較好的表現，然而整體而言 k 值似乎對 Accuracy 沒有顯著的影響 (都落在 73%-76%)，推測其資料本身特徵的區分力不強，或者我們沒有做標準化/正規化的緣故。另外需注意，K 值應選為奇數，以避免在二元分類（如本次是否罹患糖尿病）中出現投票相同的情況。

IV. 其他資料分類演算法

A. Gaussian Naive Bayes 分類器實作

Naive Bayes 是一種基於貝氏定理（Bayes' Theorem）的簡單機率分類器，假設所有特徵之間是條件獨立的。對於連續型資料，常使用高斯分布（Gaussian distribution）來建模每個特徵在各類別下的分布。

根據貝氏定理，一筆資料 $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$ 屬於類別 c 的後驗機率為：

$$P(c \mid \boldsymbol{x}) \propto P(c) \prod_{i=1}^{d} P(x_i \mid c) \tag{4}$$

若每個特徵符合高斯分布，其條件機率密度函數為：

$$P(x_i \mid c) = \frac{1}{\sqrt{2\pi\sigma_{c,i}^2}} \exp\left(-\frac{(x_i - \mu_{c,i})^2}{2\sigma_{c,i}^2}\right) \tag{5}$$

為了避免在乘法過程中數值過小導致 `underflow`，我們將其轉換為對數形式，加總後比較大小：

$$\log P(c \mid \boldsymbol{x}) = \log P(c) + \sum_{i=1}^{d} \log P(x_i \mid c) \tag{6}$$

整體流程分為訓練與預測兩階段。在訓練階段，針對每個類別計算其先驗機率與特徵的平均與標準差。在預測階段，對每筆測試資料計算其屬於各個類別的對數機率，並選擇最大者為預測類別。

Algorithm 2 Gaussian Naive Bayes

Input: 訓練資料 $X_{\text{train}}, y_{\text{train}}$，測試資料 X_{test}
stats ← ComputeStats($X_{\text{train}}, y_{\text{train}}$) // 包含每類別的 μ 和 σ
priors ← ComputePriors(y_{train})
predictions ← [ ]
for 每筆測試資料 $\boldsymbol{x}$ **do**
  posteriors ← ComputePosteriors($\boldsymbol{x}$, stats, priors)
  result ← $\arg\max$(posteriors)
  將 result 加入 predictions
end for
return predictions

B. Logistic Regression 實作

Logistic Regression 是一種常用的二元分類演算法，通過學習樣本的特徵來預測屬於某個類別的機率，並將預測結果經過 sigmoid 函數轉換為機率值。

1) 數學模型: 假設有 n 個樣本，每個樣本有 d 個特徵，對應標籤 $y \in \{0, 1\}$。Logistic Regression 使用以下公式來計算預測結果：

$$z^{(i)} = \mathbf{w}^T \mathbf{x}^{(i)} + b \tag{7}$$

$$\hat{y}^{(i)} = \sigma(z^{(i)}) = \frac{1}{1 + e^{-z^{(i)}}} \tag{8}$$

其中，$\mathbf{w}$ 是權重，b 是 bias，σ 是 sigmoid 函數。

2) 損失函數與梯度下降: 損失函數使用交叉熵來衡量預測和實際標籤的差異，梯度下降用來最小化損失函數並更新參數：

$$J(\mathbf{w}, b) = -\frac{1}{n} \sum_{i=1}^{n} \left[y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \right] \tag{9}$$

梯度下降更新規則為：

$$\mathbf{w} := \mathbf{w} - \alpha \nabla_{\mathbf{w}} J(\mathbf{w}, b) \tag{10}$$

$$b := b - \alpha \frac{\partial J}{\partial b} \tag{11}$$

Algorithm 3 Logistic Regression

Input: 訓練資料 X, 標籤 y, 學習率 α, 迭代次數 MaxIter
初始化權重 $\mathbf{w}$ 和偏差 b
for $i = 1$ to MaxIter **do**
  $\hat{y} \leftarrow \sigma(X\mathbf{w} + b)$ // 計算預測機率
  $\mathbf{w}, b \leftarrow$ UpdateWeights($X, y, \hat{y}, \mathbf{w}, b, \alpha$)
end for
return 訓練好的 $\mathbf{w}$ 和 b

3) 模型預測: 訓練完成後，使用學得的 $\mathbf{w}$ 和 b 進行預測：

$$z^{(i)} = \mathbf{w}^T \mathbf{x}^{(i)} + b \tag{12}$$

$$\hat{y}^{(i)} = \sigma(z^{(i)}) \tag{13}$$

根據預測概率進行分類：

$$\hat{y}^{(i)} = \begin{cases} 1 & \text{if } \hat{y}^{(i)} \geq 0.5, \\ 0 & \text{if } \hat{y}^{(i)} < 0.5. \end{cases} \tag{14}$$

C. Decision Tree 實作

Decison Tree 是一個預測模型，會把每個特徵切成許多群，而當有資料進來時，能依據不同特徵而被 Decison Tree 歸類。常見的分裂準則是 Gini impurity 來建構二元決策樹，`sklearn` 也是採用相同處理方式，適用於連續特徵值得分裂問題。

Gini impurity 是一種常用的標準，用來衡量資料在某個節點上的「純度」。在一個節點裡有 c 個類別，p_i 為資料中第 i 類的出現機率。Gini impurity 衡量的則是如果我們隨機選一個樣本，它被錯分的機率有多高。

$$\text{Gini}(D) = 1 - \sum_{i=1}^{c} p_i^2 \tag{15}$$

對於特徵 X_j 和某一切點 t，我們將資料劃分為左右兩部分：

$$D_{\leq t} = \{x \mid x_j \leq t\}, \quad D_{>t} = \{x \mid x_j > t\} \tag{16}$$

切分後的加權 Gini impurity 為：

$$\text{Gini}_{\text{split}} = \frac{|D_{\leq t}|}{|D|} \cdot \text{Gini}(D_{\leq t}) + \frac{|D_{>t}|}{|D|} \cdot \text{Gini}(D_{>t}) \tag{17}$$

選擇能夠最小化 $\text{Gini}_{\text{split}}$ 的特徵與切點作為分裂依據。

Algorithm 4 建構 Decision Tree

Input: 資料集 D
if 所有 y 相同 **then**
  return 葉節點（標籤為 y）
end if
$(f^*, t^*) \leftarrow \textsc{FindBestSplit}(D)$
if 無法再分割 **then**
  return 葉節點（標籤為最多數類別）
end if
$D_L, D_R \leftarrow \textsc{SplitData}(D, f^*, t^*)$
左子樹 $\leftarrow \textsc{BuildTree}(D_L)$
右子樹 $\leftarrow \textsc{BuildTree}(D_R)$
return 節點（記錄特徵 f^* 與切點 t^*）

D. Random Forest 實作

在上一節的基礎上，透過集成學習（ensemble learning）結合多棵決策樹構成 Random Forest ，並加入隨機分配的訓練資料，以大幅增進最終的運算結果。。其核心概念在於將多個弱分類器（決策樹）組合，透過多數決提高整體預測的穩定性與準確性。

訓練森林時，對原始訓練資料集進行有放回地隨機抽樣，建立每棵樹所用的子資料集 (此為 bootstrap sampling)，而針對每棵樹，僅從所有特徵中隨機抽取部分特徵來避免所有樹都長得一樣。

Algorithm 5 訓練 Random Forest

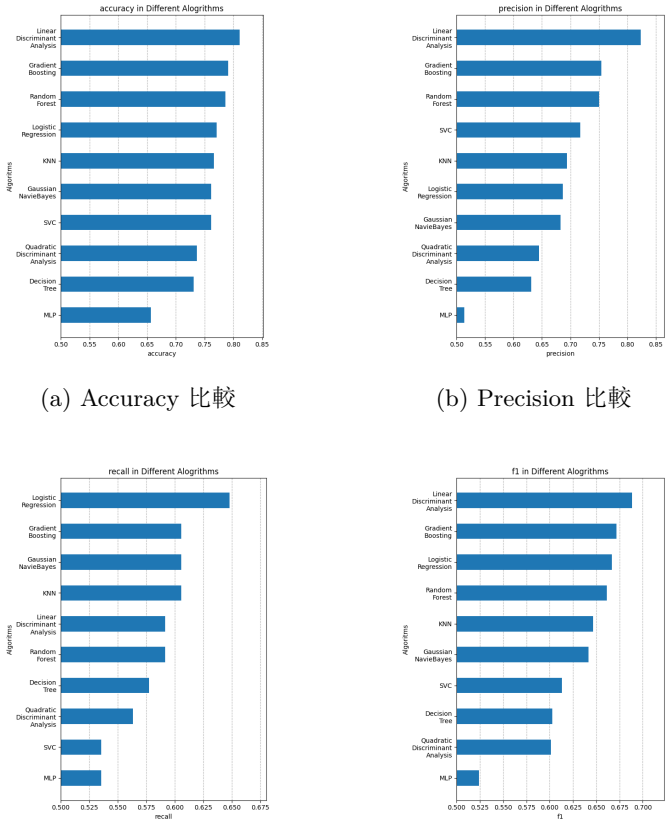
Input: 訓練資料 D，樹的數量 T
forest $\leftarrow [\,]$
for $i = 1$ to T **do**
  $D_i \leftarrow \textsc{BootstrapSample}(D)$
  $F_i \leftarrow \textsc{SelectRandomFeatures}(D_i)$
  $T_i \leftarrow \textsc{ConstructTree}(D_i, F_i)$
  forest $\leftarrow$ forest $\cup\{T_i\}$
end for
return forest

預測結果時，將測試資料分別送入每棵決策樹，收集每棵樹的預測結果，最終以多數決方式決定輸出類別。

$$\text{Predict}(x) = \arg\max_{c} \sum_{i=1}^{T} \mathbf{1}(T_i(x) = c) \tag{18}$$

V. 實驗結果與比較

在本實驗中，我們對比了多個分類算法的性能，並將結果與 KNN 進行了比較。實驗使用了多種評估指標，包括 accuracy、precision、recall 和 F1 值。以下是各算法在不同指標下的表現。



(a) Accuracy 比較　(b) Precision 比較

(c) Recall 比較　(d) F1-score 比較

图 2: 各分類算法在不同指標下的比較

實驗結果 (如圖 2 所示) 顯示，不同的演算法在各項指標上各有優劣。整體而言，Linear Discriminant Analysis、Gradient Boosting 和 Random Forest 的四個指標明顯優於其他分類演算法，展現了相對穩健且優越的綜合性能。而 MLP 在此資料集上的表現則相對基礎。

VI. 結論

本報告成功比較了 10 種常見的分類演算法，(包含 sklearn 的 SVC、Gradient Boosting、MLP、Linear

Discriminant、Quadratic Discriminant，和我們從頭實作的 KNN、Gaussian Naive Bayes、Logistic Regression、Decision Tree、Random Forest)。我們使用了 Accuracy、Precision、Recall 及 F1-score 作為評估指標，系統性地分析了這些演算法在特定資料集上的表現。

然而，本研究亦存在一些限制。首先，所有結論均基於單一資料集，其泛化能力至其他不同特性的數據集可能有所不同。其次，雖然我們探討了 KNN 的 k 值，但對於其他演算法的眾多超參數（如 SVC 的核函數與懲罰項、Random Forest 的樹數量與深度等）並未進行系統性的優化，這可能影響了部分模型的最佳性能展現。

總體而言，本報告提供了一個關於多種分類器性能的比較，並深入分析了 KNN 演算法的特性與 k 值影響。研究結果強調了根據具體應用場景和數據特性選擇合適模型及調整參數的重要性。未來的研究可考慮在更多元的數據集上進行驗證，並引入更全面的超參數優化技術，以及對計算成本進行更詳細的評估。