資料探勘-期中報告

第 12 組

B104020024 蔡尚宸

B103025027 林瀚坪

B103040015 陳承新

B103040061 段向生

B103040063 蕭維亨

2023.04.06

一、 摘要

本報告將從 KNN 開始，介紹三種資料分類演算法；再來會尋找文獻查看相關的研究與應用；接著會分享我們設計的程式以及程式跑出的結果。最後我們會對得出的結果進行分析。

二、 前言簡介

首先介紹 KNN 演算法。

KNN 演算法，也就是 K-近鄰演算法。它是一種監督式學習演算法，主要可以用於分類和回歸問題。從我們實作的過程中，我們觀察到:

- 它的優點是簡單易懂且較易於程式實作，並且通用性比較高，基本上大部分的數據都有辦法應用。
- 它的缺點是會受到 noisy values 跟 outliers 比較明顯的影響，所以如果想有更高的準確率，可以考慮先進行資料前處理後，再開始分析。

總體而言，我們認為學習 KNN 是學習資料分類演算法很好的開始。

第二個要介紹 Random Forest 演算法。

Random Forest 是一種集成學習算法，使用多個決策樹對數據進行建模，然後將它們的預測結果合併來得出最終的預測結果。從我們實作的過程中以及查詢到的資料，我們發現到:

- 它的程式部分比 KNN 稍微複雜一點，不過若是搭配合適套件，依然可以成功實作出來。
- 理論上，使用 Random Forest 方法去處理大型的數據集的話，會有較快的速度。並且它對於 noisy values 跟 outliers 有比較高的容忍性。
- 但也因為 Random Forest 的演算方式是由多個決策樹組成的，所以某種程度上來說，比較難以解釋其內部運作方式。

第三種是 LVQ 演算法。

LVQ(Learning Vector Quantization)演算法是一種監督式學習算法，邏輯是透過不斷學習與即時的調整，讓訓練的模型與真實情況逐漸靠近。

根據我們查到的資料:

- 這個算法具有很好的可解釋性，可以較好理解模型的運作邏輯。
- 而其缺點在於，會需要對算法進行參數調整，才可以有更好的效果。

對於 LVQ，我們認為這是一個可以嘗試去學習的算法。

三、 相關研究

對於資料分類演算法的相關應用，我們發現在各個行業領域中，資料分類演算法都有廣泛的應用。以下是一個關於另一種演算法 SVM 的相關具體的例子：

資料來源: 林昕潔, 柯皓仁, & 楊維邦. (2005). *以 SVM 與詮釋資料設計書籍分類系統*.

“在當代社會中，因為書籍種類繁多，書店或圖書館需要花費大量的時間和人力進行書籍分類，以方便使用者尋找自己需要的書籍。”在經過研讀「以 SVM 與詮釋資料設計書籍分類系統 」後，該研究所提出的一套書籍自動分類系統，可以有效地節省人力成本，提高書籍分類的效率。

該研究使用 SVM 分類器產生分類模型；其次，分析統計書籍詮釋資料，並且單獨運用這些資訊進行書籍分類；最後，以線性組合將 SVM 分類和詮釋資料的分類結果加以合併，完成書籍分類的工作。(在實驗中，使用

了博客來網路書店的書籍資訊。）

以該研究的結果為例，從每個類別選取 50 個、100 個與 150 個特徵進行 SVM 分類對分類結果正確性的影響，由實驗結果顯示，選取 150 個特徵有利於 SVM 分類。

資料與圖片來源: 林昕潔, 柯皓仁, & 楊維邦. (2005). *以 SVM 與詮釋資料設計書籍分類系統*.

Features	偵探/懸疑小說		科幻/奇幻小說		愛情文藝小說	
	Accuracy	F-measure	Accuracy	F-measure	Accuracy	F-measure
50	89.3%	82.2%	85.3%	75.1%	89.1%	82.5%
100	90.7%	84.9%	86.3%	77.3%	89.6%	83.6%
150	91.1%	85.6%	88.1%	80.8%	89.8%	83.9%

從以上文獻，我們了解到選取不同特徵屬性的數量，可以有一定程度的影響分類的正確性。因此在往後進行分類演算時，可以將這個因素一起考量。

四、 程式設計步驟

關於本次的實作部份，我們選擇使用 Python 來進行實作 KNN 及 Random Forest。

- KNN 的實作部分，以下為程式步驟:

1. 導入數據，在 Python 中，我們使用 Pandas 庫來處理數據。

```
train_data_a = pd.read_csv("C://Users//josep//Des
test_data_a = pd.read_csv("C://Users//josep//Desk
train_data_b = pd.read_csv("C://Users//josep//Des
test_data_b = pd.read_csv("C://Users//josep//Desk
```

2. 將數據特徵分類。

```
feature = ["Pregnancies", "Glucose", "BloodPressure", "SkinThickness", "Insul
#將數據轉換成list
list_train_a = [list(train_data_a[x]) for x in feature]
list_test_a = [list(test_data_a[x]) for x in feature]
list_train_b = [list(train_data_b[x]) for x in feature]
list_test_b = [list(test_data_b[x]) for x in feature]
```

3. 將所有特徵的值歸一化、使用的是 MinMax 演算法

MinMax 歸一化：將數據線性映射到[0, 1]區間內。對於每個特徵，找到該特徵在數據集中的最小值和最大值，然後對該特徵進行線性映射。具體而言，假設 x 是某個特徵的原始值，xmin 和 xmax 分別是該特徵在數據集中的最小值和最大值，那麼該特徵的新值可以計算為：(x-xmin)/(xmax-xmin)。

```
def auto_norm_data(target, max_v, min_v):       # (舊的值-最
    return (target - min_v) / (max_v - min_v)
```

4. 切割數據，方便之後的計算。

```
#切割數據
single_train_a = [[list_train_a[x][y] for x in range(8)] for y
single_test_a = [[list_test_a[x][y] for x in range(8)] for y in
single_train_b = [[list_train_b[x][y] for x in range(8)] for y
single_test_b = [[list_test_b[x][y] for x in range(8)] for y in
```

5. 之後計算測試資料和每一筆要訓練資料的距離，使用的是歐基里德距離 (Euclidean distance)。

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

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

```
def dist(train, test):
    sum_dist = 0
    for i in range(len(test)):
        sum_dist +=  pow((train[i] - test[i]),2)
    return sqrt(sum_dist)
```

6. 尋找 K 個最近距離及紀錄下來，並按標簽數量投票分類。

```
#尋找K個最近距離
idx_list = min_idx(K_VALUE, arr_dist)
#記錄最近的K個標簽
outcome_list = [train_data_a["Outcome"][idx_list[x]] for x
#按標簽數量投票分類
final_list_a = final_list_a + [classify(outcome_list)]
```

7. 最後將結果進行比對。

```
#計算準確率
for i in range(len(single_tes
    if final_list_a[i] == tes
        current_a = current_a
for i in range(len(single_tes
    if final_list_b[i] == tes
        current_b = current_b
print("Accuracy of A : %.2f%%
print("Accuracy of B : %.2f%%
```

- Random Forest

1. 首先，我們載入了需要使用的套件：csv、sklearn.ensemble 中的 RandomForestClassifier 和 matplotlib.pyplot。

   -csv: Python 內建的 CSV 檔案讀取套件，用來讀取 CSV 檔案。

   -RandomForestClassifier 會提供 Random Forest 的演算法等。

   -matplotlib.pyplot: 用來繪製數據圖表，可以顯示在瀏覽器中或者保存為圖像文件。

```
import csv
from sklearn.ensemble impor
import matplotlib.pyplot as
```

2. 接著，我們使用 csv 套件讀取我們需要的資料，讀取後，我們將這些資料存儲在 testfile、trainfile、test2 和 train2 等變數中。

```
csvfile = open(r"C:\Users\User\Desktop\schc
testfile = list(csv.reader(csvfile))
csvfile.close()

csvfile = open(r"C:\Users\User\Desktop\schc
trainfile = list(csv.reader(csvfile))
csvfile.close()

csvfile = open(r"C:\Users\User\Desktop\schc
test2 = list(csv.reader(csvfile))
csvfile.close()

csvfile = open(r"C:\Users\User\Desktop\schc
train2 = list(csv.reader(csvfile))
csvfile.close()
```

3. 接下來，我們從 trainfile 和 testfile 中特徵和標籤將其分別存儲在 x_train、x_test、y_train 和 y_test 中。

```
x_train = [i[:-1] for i in trainfile[1:]]
x_test = [i[:-1] for i in testfile[1:]]
y_train = [i[-1] for i in trainfile[1:]]
y_test = [[i[-1]] for i in testfile[1:]]
```

4. 之後，我們建立一個名為 randomForestModel 的 RandomForestClassifier 物件，並使用 n_estimators=100 設置隨機森林中樹木的數量。接著，我們使用 randomForestModel.fit()方法來訓練模型，傳入的參數為 x_train 和 y_train。

   - randomForestModel.fit() 是用來訓練隨機森林模型的方法。它的作用是根據給定的訓練資料（通常包括特徵和標籤）來訓練一個隨機森林模型。

```
randomForestModel = RandomForestClassifier(n_es
randomForestModel.fit(x_train2, y_train2)
```

5. 之後使用 randomForesModel.predict() 方法可以對新的數據進行預測，以預測它屬於哪個類別，並將預測結果存儲在 predict 中。

```
predict = randomForestModel.predict(x_
```

6. 在模型訓練完成後，我們使用 score()方法來計算模型的精度，並將結果輸出到屏幕上。

五、 得出結果

我們得到以下結果:

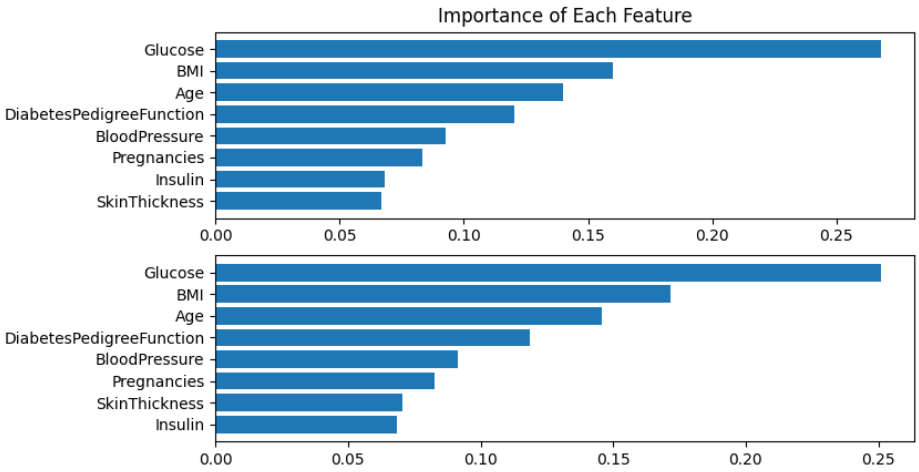
KNN:A 實驗的正確率為 76.62%。

B 實驗的正確率為 75.00%。

Random Forest:

A 實驗的正確率為 78.1%。

B 實驗的正確率為 82.00%。



我們觀察到，兩種演算法皆有可以接受的正確率。

並且使用 Random Forest 的時候可以觀察各 features 對訓練的重要性。

六、 參考文獻。

林昕潔, 柯皓仁, & 楊維邦. (2005). *以 SVM 與詮釋資料設計書籍分類系統* (Doctoral dissertation).

Han, J., Pei, J., & Tong, H. (2022). *Data mining: concepts and techniques*. Morgan kaufmann.