Data Mining Project2

H24116049 **莊秉宸**

A. 分類問題發想

水質汙染監測。特徵為該汙染物是否超標（0、1）& 是否經過水質淨化法（0、1），輸出為水質是否遭到汙染（0、1）。

B. Feature Name

- lead 鉛
- chromium 鉻
- mercury 汞
- arsenic 砷
- copper 銅
- nickel 鎳
- zinc 鋅
- aluminium 鋁
- perchlorate 高氯酸鹽
- chloramine 氯
- radium 鐳
- silver 銀
- uranium 鈾
- selenium 硒
- cadmium 鎘
- ammonia 銨

- biological adsorption 經過生物吸附法
- phytoremediation 經過植物整治法

C. Absolutely Right Rule

- **水質受汙染（output = 1）**
  - Perchlorate（高氯酸鹽） = 1
  - Aluminium（鋁） + Cadmium（鎘） + Chloramine（氯） >= 2

- **水質無汙染（output = 0）**
  - Biological adsorption（生物吸附法） = 1
  - Phytoremediation（植物整治法） = 1
    - Perchlorate + Aluminium + Cadmium + chloramine <= 3

D. Absolute Right Rule Explanation

- **水質受汙染（output = 1）**

高氯酸鹽的超標指數較高，若超標，將會被認定為水汙染；而鋁、鎘、氯均為嚴重汙染物，設置為三項中超過兩項指數超標，將被視為水汙染。

- **水質無汙染（output = 0）**

生物吸附法是當今處理水汙染常利用的方法，能有效去除重金屬離子；而植物整治法利用植物對某些污染物的吸收，達到淨化水質的作用。由於一種植物只對特定幾種汙染物有良好效用，因此設定為**高氯酸鹽、鋁、鎘、氯四種少於三項超標**時，被認定為無汙染，反之則依舊為水汙染。

註：即使水質受汙染的條件達成，若有效經過水質淨化法，將被視為無汙染。

E. 資料量設定

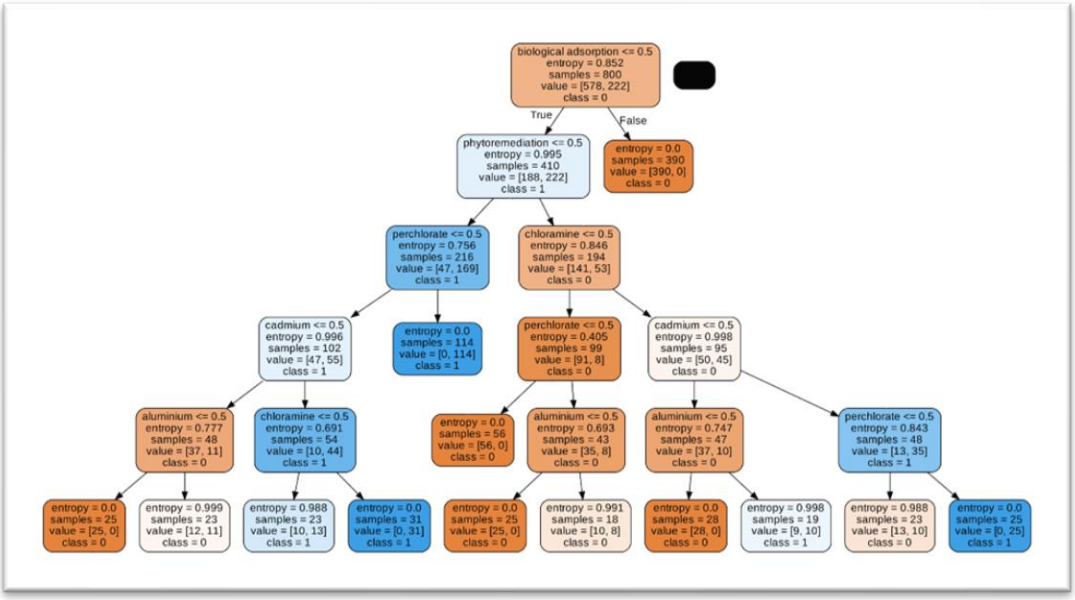
- **Data without noise**
  Training data (800 筆) + Testing data (200 筆) = 1000 筆
  資料均按照 absolute right rule 生成
- **Data with noise**
  Noise data (200 筆) + Training data (800 筆) + Testing data (200 筆) = 1200 筆

F. Decision Tree Classifier

- **資料無 noise**
  我利用 sklearn.tree 的 DecisionTreeClassifier 作為分類器，使用 entropy 當作 criterion 。首先，我使用不同的 max_depth 建構了 6 個決策樹模型。考量到模型的準確度和規則的可讀性，我使用 max_depth = 5 的模型進行分析，圖二為使用 graphviz 套件生成的模型圖片（DecisionTree_no_noise.pdf ）:

```
max_depth = 1
Acc: 0.765,     Test_Acc: 0.780
================================
max_depth = 2
Acc: 0.875,     Test_Acc: 0.855
================================
max_depth = 3
Acc: 0.875,     Test_Acc: 0.855
================================
max_depth = 4
Acc: 0.935,     Test_Acc: 0.920
================================
max_depth = 5
Acc: 0.940,     Test_Acc: 0.955
================================
max_depth = 6
Acc: 1.000,     Test_Acc: 1.000
================================
```

圖一、不同 max_depth 情況的分類器模型（無 noise）



圖二、max_depth = 5 的分類器圖片

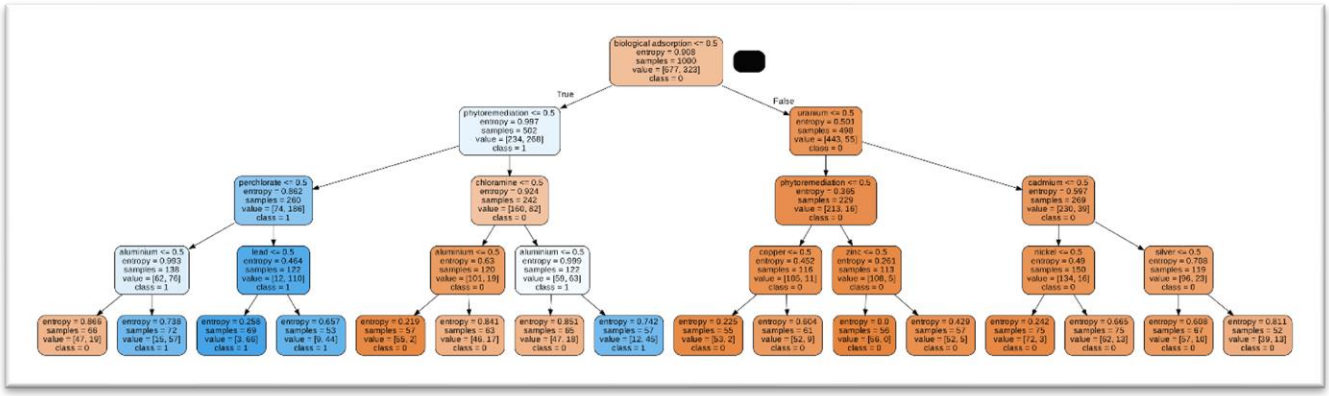
- **資料含有 Noise**

  我在資料中添加了 200 筆的 Noise data，並針對不同的 max_depth 建構模型。

```
max_depth = 1
Acc: 0.711,     Test_Acc: 0.765
================================
max_depth = 2
Acc: 0.789,     Test_Acc: 0.855
================================
max_depth = 3
Acc: 0.793,     Test_Acc: 0.875
================================
max_depth = 4
Acc: 0.850,     Test_Acc: 0.950
================================
max_depth = 5
Acc: 0.859,     Test_Acc: 0.915
================================
max_depth = 6
Acc: 0.900,     Test_Acc: 0.965
================================
max_depth = 7
Acc: 0.917,     Test_Acc: 0.920
================================
max_depth = 8
Acc: 0.944,     Test_Acc: 0.870
================================
```

圖三、不同 max_depth 情況的分類器模型（有 noise）

考慮模型的可觀察性和準確性，由於在 max_depth = 4 的模型在 test data 有不錯的表現（0.95），因此將其作為觀察對象（附圖：DecisionTree_Noise.pdf）。



圖四、max_depth = 4 的分類器圖片

G. 決策樹模型結果分析

在資料無 noise 的模型中（圖二），能最先發現模型一次只會判定一個特徵，進行二元分類，這也顯示了決策樹**「快速（Yes/No Question 計算量小）」**、**「對於不複雜的資料具有可觀察性（樹狀結構的二元分類）」**的特性。

細看規則，若經過生物吸收法，就會被視為水質無汙染。我們能看到模型會優先判定是否經過水質淨化（biological adsorption、phytoremediation），再考慮汙染物是否超標，代表在特徵的 Entropy 中，「是否經過水質淨化」具有較高的資訊量，也和我們預設的規則相符。

在深度為 4、5 層時，模型正在判斷**鋁、鎘、氯三者兩者超標則為水汙染**的特徵，只是因為深度不夠，使模型無法符合預設的規則；而當 max_depth 為 6 時，便能完美的符合此規則（其餘規則也與 absolute right rules 完全相符），使訓練、測試資料的準確度達到 100%。

在資料含有 noise 的模型中（圖四），能發現由於隨機測資的緣故，使得樹狀圖新增了一些新規則，變得很胖，體現決策樹「對於雜訊敏感」的特性，在深度高的情形下，也常出現 overfitting 的狀況（如圖三中 max_depth = 8 的情形）。

當我們仔細觀察樹的規則，不難發現雖然**經過生物吸附法**（第一層為 False）有眾多分支，最終都會導向水質無汙染（class = 0）。而細看左側模型的規則，也能發現和圖二模型（無雜訊資料的模型）的相似之處。經過推想，雖然模型在深度高時會因為分支太細而出現過度擬合，不過深度較淺的樹分支也含有有價值的特徵，因此能藉由 Post-Pruning 修剪分支，改善 overfitting 的問題。

H. KNN（K-Nearest Neighbor）

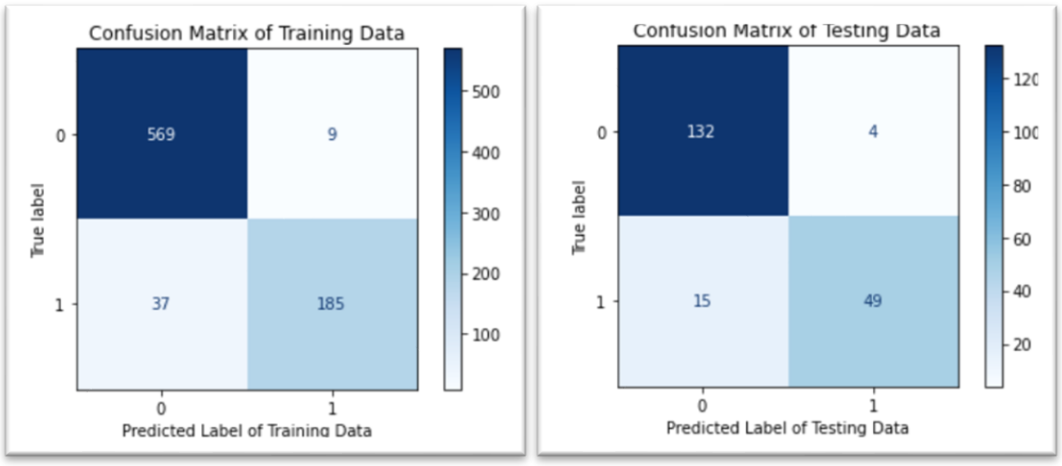
- **資料無 Noise**

  此部分我使用 sklearn.neighbors 中的 KNeighborsClassifier 當作分類器。KNN 尋找鄰近的點（n_neighbors 為 hyperparameter），並利用這些點的類別總數，決定該資料的分類。我針對不同的 n_neighbors 建立不同模型，並觀察準確率和混淆矩陣進行分析。

```
n_neighbors = 1
Acc: 1.000,     Test_Acc: 0.830
================================
n_neighbors = 2
Acc: 0.905,     Test_Acc: 0.820
================================
n_neighbors = 3
Acc: 0.940,     Test_Acc: 0.835
================================
n_neighbors = 4
Acc: 0.896,     Test_Acc: 0.860
================================
n_neighbors = 5
Acc: 0.926,     Test_Acc: 0.900
================================
n_neighbors = 6
Acc: 0.907,     Test_Acc: 0.865
================================
n_neighbors = 7
Acc: 0.927,     Test_Acc: 0.895
================================
```

圖五、不同 n_neighbors 情況的分類器模型（無 noise）

綜合觀察訓練和測試資料的準確度，n_neighbors = 5 的模型準確度達到 0.9，因此，我將此模型作為分析對象，生成混淆矩陣。



圖六、n_neighbors = 5 模型的混淆矩陣

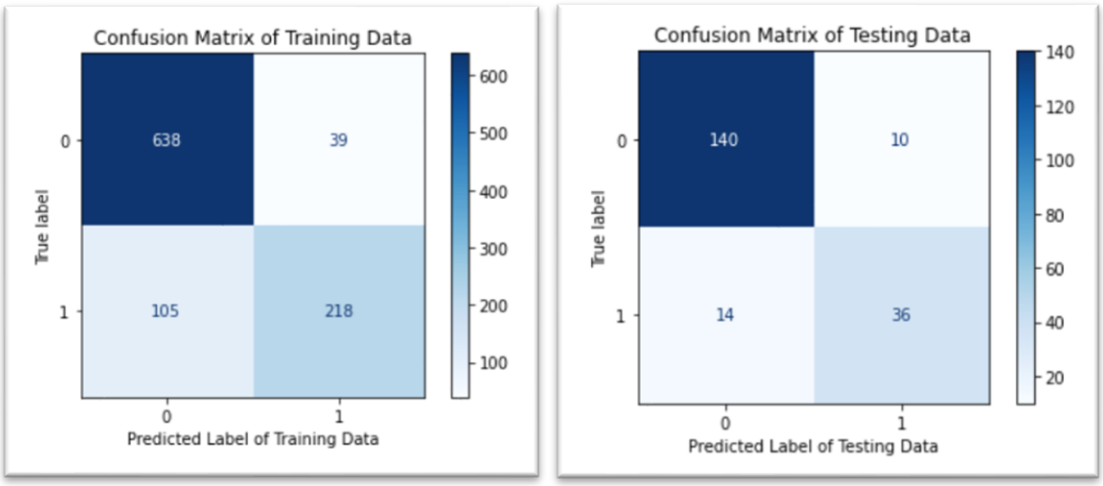
- **資料含有 Noise**

  此資料含有 200 筆的 Noise Data，我針對不同的 n_neighbors 生成 KNN 模型，觀察其在訓練資料和測試資料的準確度。

```
n_neighbors = 1
Acc: 0.999,     Test_Acc: 0.720
================================
n_neighbors = 2
Acc: 0.867,     Test_Acc: 0.840
================================
n_neighbors = 3
Acc: 0.869,     Test_Acc: 0.815
================================
n_neighbors = 4
Acc: 0.844,     Test_Acc: 0.830
================================
n_neighbors = 5
Acc: 0.863,     Test_Acc: 0.830
================================
n_neighbors = 6
Acc: 0.843,     Test_Acc: 0.855
================================
n_neighbors = 7
Acc: 0.856,     Test_Acc: 0.880
================================
```

圖七、不同 n_neighbors 情況的分類器模型（有 noise）

根據圖七，我們能發現 n_neighbors = 7 時有不錯的表現，達到 0.88 的準確度，因此對其生成混淆矩陣。



圖八、n_neighbors = 7 模型的混淆矩陣

I. KNN 模型結果分析

首先能從兩種資料的模型準確度（圖五、圖七）中發現，當 n_neighbors = 1 時，訓練資料準確率為 100%，但測試資料的表現卻不好，出現 overfitting 的問題。這是因為在計算訓練資料的準確率時，會抽取資料中的點當作 test point，尋找鄰居進行分類，而當尋訪鄰居數為 1 時，距離最近的點為它本身，因此 Error Rate 為 0、準確率為 100%，出現嚴重的過度擬和問題。

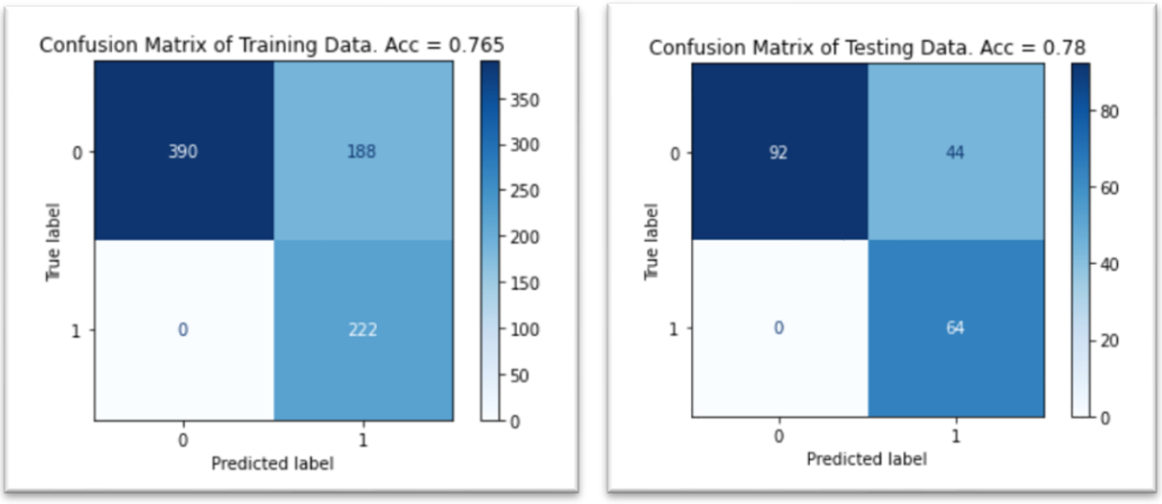
此外，由兩種資料的模型混淆矩陣（圖六、圖八）中能觀察到資料無 noise 的 True label 中，輸出為 0 的數量與 1 的數量約為 3:2 （資料有 noise 中，輸出 0 跟 1 比例約為 2:1）。在這種資料數不平均的情況下，由於類別為 0 的的資料較多，尋找最近的鄰居時，會找到較多的 0 類別，這使得 KNN 的投票系統中，模型會比較傾向輸出為 0。觀察混淆矩陣中 FN 和 FP 的數值中，能發現模型在預測錯誤時，FN（True 預測成 False）占了大多數，也證實了上面的說法。

而對比資料無 noise 和資料有 noise 的模型時，能發現雜訊資料對於訓練產生了影響，在 n_neighbors 較小時，若 Test point 附近含有雜訊資料，預測就較容易出錯導致準確率下降，隨著增加近鄰的數量，能平均掉噪音造成的誤差，使準確率上升。

J. 貝氏分類器（Naive Bayes classifier）

- **資料無 Noise**

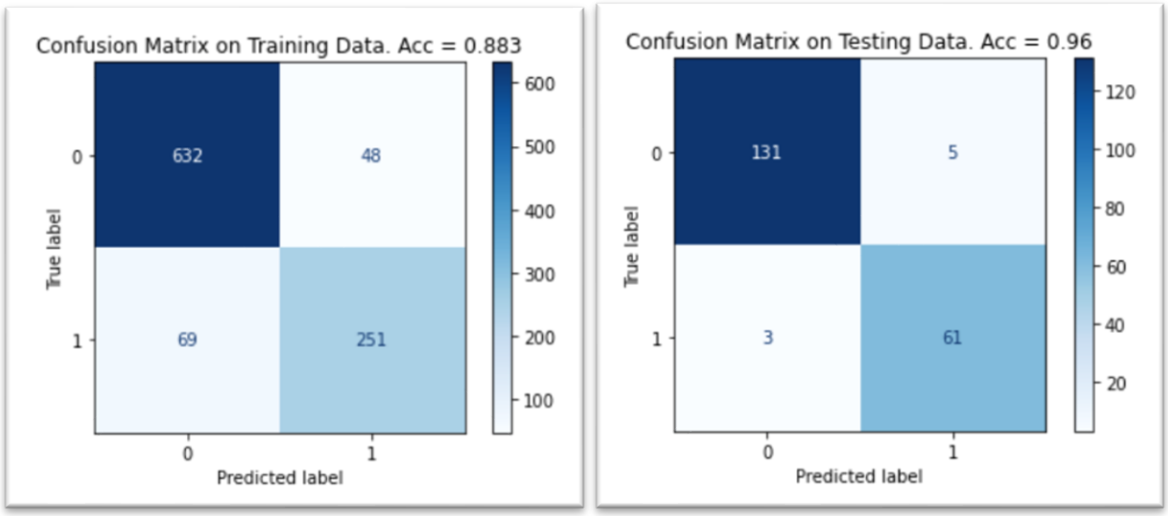
  我使用 sklearn.naive_bayes 的 GaussianNB 作為分類器，計算模型於 Training Set 和 Testing Set 的準確度（分別為 0.765 和 0.78），並生成混淆矩陣。



圖九、貝氏分類器的準確度和混淆矩陣（資料無 Noise）

- **資料含有 Noise**

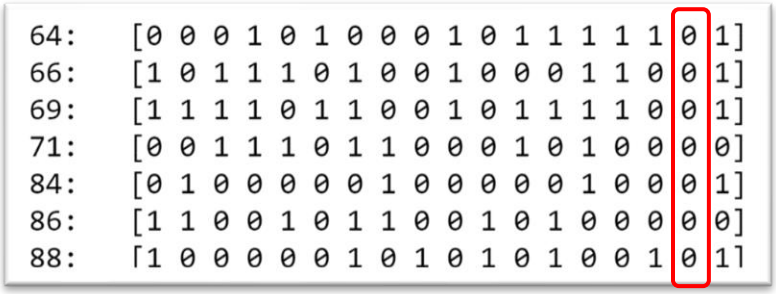
  此部分資料含有 200 筆雜訊，同樣的我計算模型於訓練和測試資料的準確率（分別為 0.883 和 0.96），並生成混淆矩陣。



圖十、貝氏分類器的準確度和混淆矩陣（資料有 Noise）

K. 貝氏分類器預測分析

貝氏分類器假設每項 Feature 均為獨立，擁有相同的權重，藉由計算每項特徵的條件機率後乘積，進行分類。由於我預設的 absolute right rule 並非各自獨立，因此能看到資料無 Noise 的模型準確率並不高。觀察混淆矩陣中 FN 和 FP 的值能發現 FN 為 0，猜錯的均為 FP。我將 FP 的測資單獨輸出進行觀察，能發現這些資料中，**是否經過生物吸附法**欄位均為 0（如圖十一）。



圖十一、預測為 FP 的測資（第 16 行均為 0）

根據我生成的規則，若經過水質淨化法，即判定為 0（水質無汙染）。而在查看未經過水質淨化法的資料時，發現大部分輸出值為 1。這導致模型在計算條件機率時，P(未經過水質淨化法 | 1) 的值相當大，生成了若**未經過生物吸附，為水質汙染**的規則，造成 FP 值很大，FN 值為 0。

此外我也發現了很有趣的現象，具有噪音的資料集，模型卻比無噪音來的優秀，甚至在測試集上達到了 0.96 的準確率。經過思考，我認為 200 筆的噪音資料有效的平衡了上述**未經過生物吸附即為水質汙染**的特徵，在條件機率下降的情況下，模型的判斷反而被導正。從混淆矩陣也能發現 FP、FN 的值不再有極端的差距，在訓練和測試資料上也具有較高的準確率。

由於貝氏分類器會將所有特徵列入計算，在建立模型時，我原本猜想準確率會不高，然而效果並不差。這是因為資料隨機生成的情況下，不重要的特徵與輸出的條件機率會較為相仿，最後還是由重要的特徵進行決策。這也說明了貝氏分類器對 Noise Feature 較不敏感。

L. Project 小結

這份作業我總共使用了三種模型（決策樹、KNN、貝氏分類器）進行預測分析。三種模型中，決策樹總體擁有較好的表現，兩種資料的測試集準確率均有 0.95 以上，其可觀察性也有效的幫助我了解分類規則；KNN 在兩種資料的準確率為 0.9、0.88，效果較為平庸；而貝氏分類器因為假設獨立的特性，於無噪音資料的效果並不理想（未達 0.8），雖然模型在噪音測試資料上有優異的表現（0.96），但在噪音訓練集上的準確率並不突出（0.88）。

經過三種模型的實作，我也對模型的特性有了更深的了解，藉由實際資料的測試，檢驗出課堂上提到的模型屬性，讓我有種融會貫通的感覺。這次添加的雜訊為訓練資料的隨機噪音，很開心能在看到雜訊對模型的影響後，對其進行有脈絡性的解釋，未來會嘗試添加特徵雜訊、測試資料雜訊等等，幫助自己更透徹了解模型特性，也更貼近實際任務會遇到的問題。