數據科學理論與方法

期末報告

Stroke Prediction Dataset

中風預測資料集的分析

應數三甲 10911107 高博彥

應數三甲 10911114 劉 芥

應數三甲 10911133 林育成

目錄

一、 研究動機

在台灣的十大死因中，腦血管疾病排名第四。每年大約有 3 至 5 萬人中風，根據統計數據，約有六個人中就有一人中風，發生率還在逐年增加。我們也經常在新聞上看到一些名人突然中風的報導，或者家中的長輩因中風而過世。中風可分為缺血性腦中風和出血性腦中風兩種，前者是由於腦部血管阻塞造成的血液供應不足，後者則是由於腦部血管出血導致的。出血性腦中風的死亡率高達 40%，因此中風後的死亡風險非常高。即使能夠倖存，也常常會有一些後遺症，例如肢體運動障礙、語言障礙等，對日後的生活影響相當嚴重。

因此，我們很好奇是什麼原因導致中風的發生。我們希望透過分析中風相關數據，更深入地了解中風的情況，以降低我們日後中風的風險。透過對中風的瞭解，我們或許可以採取預防措施，降低中風發生的可能性。

二、 研究方法

i. 資料集介紹與處理

在這次的期末報告選擇的資料集是來自「kaggle」這個網站，作者為fedesoriano，資料集名稱為「Stroke Prediction Dataset」。在作者對該資料集的說明中，提到了該資料集用於根據性別、年齡、各種疾病和吸煙狀況等輸入參數，預測患者是否可能罹患中風，資料集中的每一行都提供有關患者的相關參數。所以此資料集高度符合我們報告的主題。

在此資料集中，共有 12 個欄位，5110 筆原始資料，每筆資料都是以「人」為單位收集。欄位的說明如下：

1. id：每筆資料都有一組唯一的編號，屬於其他類型
2. gender：性別，值為"Male"、"Female"或"Other"，屬於範疇類型
3. age：該筆資料患者的年齡，浮點數資料，屬於數值類型
4. hypertension：若患者無高血壓則值為 0，否則為 1，屬於布林類型
5. heart_disease：若患者無心臟病則值為 0，否則為 1，屬於布林類型
6. ever_married：曾經結過婚為"Yes"，未結過婚為"No"，屬於布林類型
7. work_type：工作類型，小孩為"children"、政府部門工作為"Govt_job"、無工作為"Never_worked"、自由業為"Self-employed"、不願透漏為"Private"，屬於範疇類型
8. Residence_type：居住類型，鄉村為"Rural"、城市為"Urban"，屬於範疇類型
9. avg_glucose_level：血糖值，浮點數資料，屬於數值類型
10. bmi：身高體重指數(body mass index)，浮點數資料，屬於數值類型，其中"N/A"值表示 missing data
11. smoking_status：從未吸菸為"never smoked"、曾經吸菸為"formerly smoked"、有吸菸為"smokes"、未知為"Unknown"(表示該患者資料不可用)，屬於範疇類型
12. stroke：class 欄位，如果患者有中風則為 1，否則為 0，屬於布林類型

在進行分析前，需要將上述資料先做資料前處理，但因為資料的完整度高，只需去除無用欄位(id)、將非數值資料編碼、數值資料轉型、處理 bmi 的 missing data 與 smoking_status 欄位為"Unknown"的不可用資料，處理完後就拿到乾淨、可以切分成訓練、測試的資料。

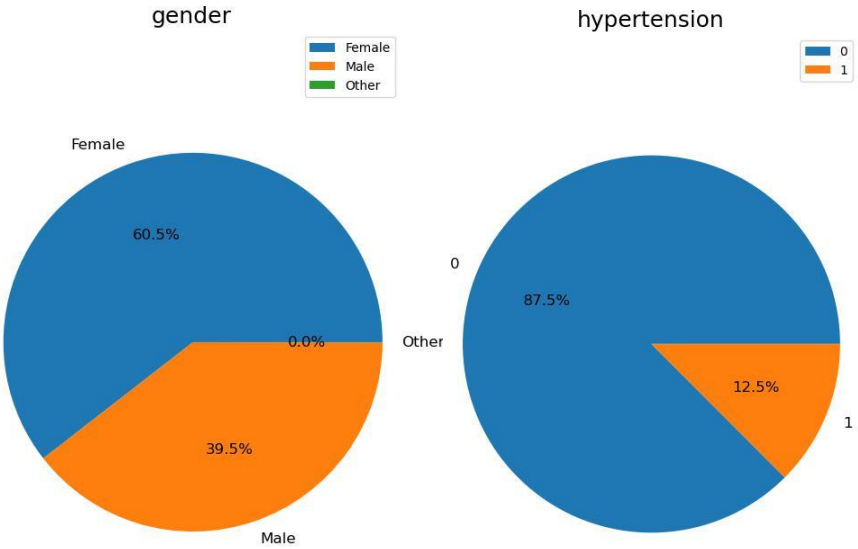
第一步先去除無意義欄位，即 id 欄位，只取剩下的 11 個欄位進行分析。第二步將非數值資料(gender、ever_married、work_type、Residence_type、smoking_status 欄位)做編碼，採 label encoding。age 與 avg_glucose_level 欄位資料從字串轉型為浮點數。Hypertension、heart_disease、stroke 欄位資料從字串轉型為整數。第三步要處理 missing data，bmi 的"N/A"值採取用中位數填入的方式，而 smoking_status 欄位為"Unknown"的不可用資料採 Complete case analysis，刪除該筆資料。

最終可以得到：資料集取用欄位數:11、可用資料筆數:3566、不可用資料筆數(smoking_status:Unknown):1544、bmi 值無遺失的資料筆數:3426。

ii. 資料集的統計分析

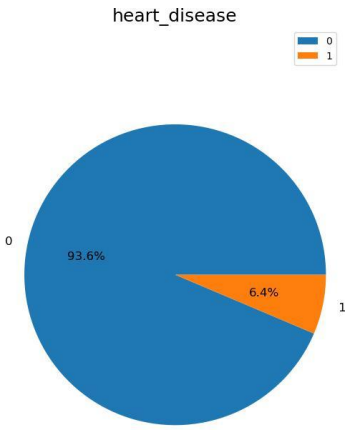
將資料處理完後，我們可以求出非連續數值欄位的數值出現次數、比例、該欄位的 entropy 值：

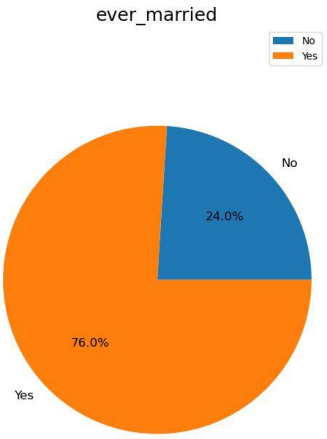
1. gender 欄位中各值出現次數: {'Female': 2158, 'Male': 1407, 'Other': 1}
   gender 欄位中各值出現比例: {'Female': 0.6051598429613012, 'Male': 0.39455973079080203, 'Other': 0.0002804262478968031 4}
   gender 欄位的 entropy:0.9711896669803112



2. hypertension 欄位中各值出現次數: {'0': 3120, '1': 446}
   hypertension 欄位中各值出現比例: {'0': 0.8749298934380259, '1': 0.1250701065619742}
   hypertension 欄位的 entropy:0.5437612247917808

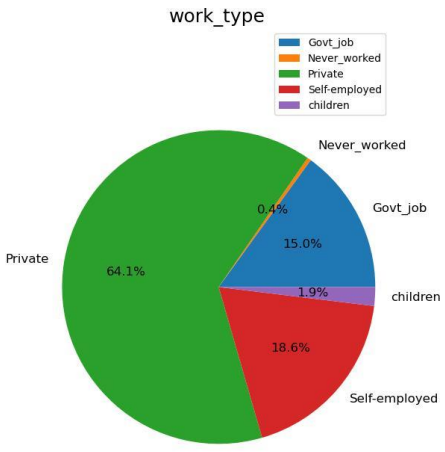
3. heart_disease 欄位中各值出現次數: {'0': 3338, '1': 228}
   heart_disease 欄位中各值出現比例: {'0': 0.9360628154795289, '1': 0.06393718452047112}
   heart_disease 欄位的 entropy:0.3428797407876658

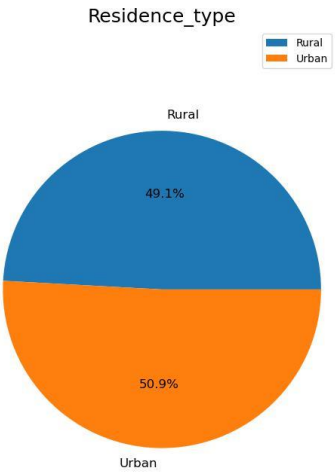




4. ever_married 欄位中各值出現次數: {'No': 856, 'Yes': 2710}
   ever_married 欄位中各值出現比例: {'No': 0.24004486819966347, 'Yes': 0.7599551318003365}
   ever_married 欄位的 entropy:0.7951148856695476

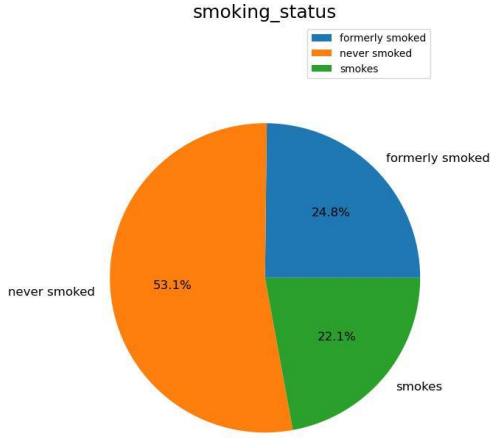
5. work_type 欄位中各值出現次數: {'Govt_job': 535, 'Never_worked': 14, 'Private': 2285, 'Self-employed': 663, 'children': 69}
   work_type 欄位中各值出現比例: {'Govt_job': 0.1500280426247897, 'Never_worked': 0.003925967470555244, 'Private': 0.6407739764441952, 'Self-employed': 0.18592260235558047, 'children': 0.019349411104879418}
   work_type 欄位的 entropy:1.4148139775588675



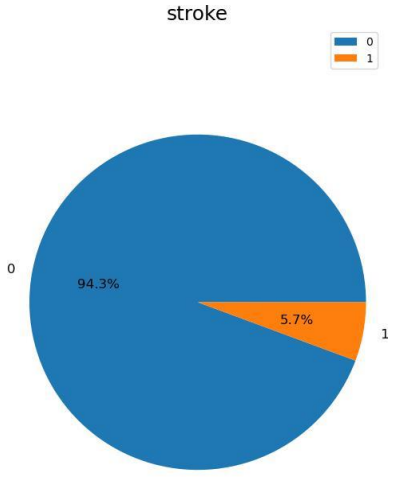


6. Residence_type 欄位中各值出現次數: {'Rural': 1752, 'Urban': 1814}
   Residence_type 欄位中各值出現比例: {'Rural': 0.4913067863151991, 'Urban': 0.5086932136848009}
   Residence_type 欄位的 entropy:0.9997819344169199

7. smoking_status 欄位中各值出現次數: {'formerly smoked': 885, 'never smoked': 1892, 'smokes': 789}
   smoking_status 欄位中各值出現比例: {'formerly smoked': 0.2481772293886708, 'never smoked': 0.5305664610207516, 'smokes': 0.22125630959057768}
   smoking_status 欄位的 entropy:1.4656217476425153

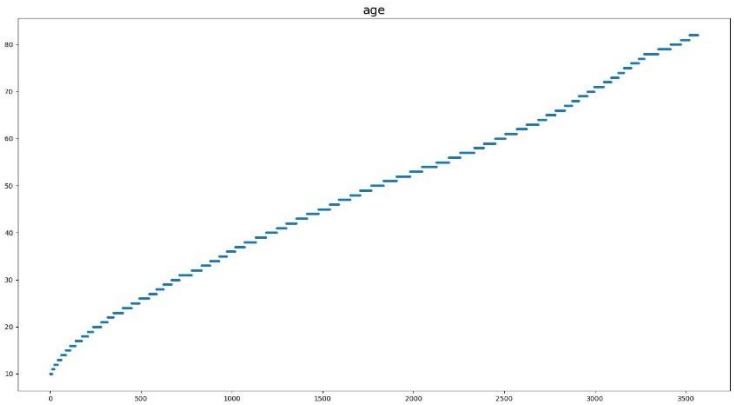


8. stroke 欄位中各值出現次數: {'0': 3364, '1': 202}
   stroke 欄位中各值出現比例: {'0': 0.9433538979248458, '1': 0.056646102075154234}
   stroke 欄位的 entropy:0.3139847468133943

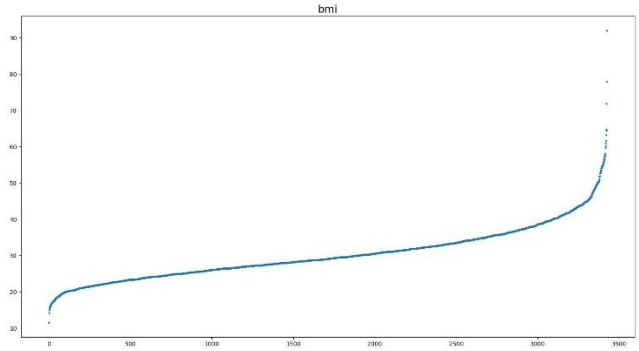


另外，我們可以求出連續數值欄位的最大值、最小值、平均值、中位數、變異數與標準差：

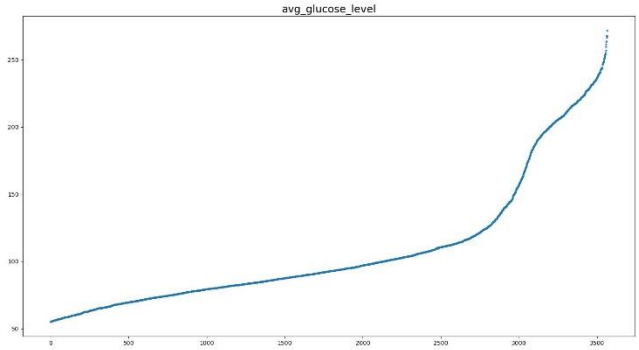
1. age 欄位中最大值:82.0、最小值:10.0、平均值:48.85389792484577、中位數:50.0、變異數:356.24204230542625、標準差:18.874375282520646



2. avg_glucose_level 欄位中最大值:271.74、最小值:55.12、平均值:108.98201065619742、中位數:92.65、變異數:2332.136941117418、標準差:48.29220373018214



3. bmi 欄位中最大值:92.0、最小值:11.5、平均值:30.290046701692937、中位數:29.1、變異數:53.2310103919789、標準差:7.295958497139282



iii. 分析(預測)方法介紹

在這次報告中，用此資料集訓練的模型有 2 個，分別是 KNN 與 Decision Tree，都是上課介紹過的。

KNN 是一個在機器學習方法中常見的模型。KNN 模型訓練步驟就是紀錄下所有資料，並決定一個 k 值即可。由於這種記下所有資料來進行分類的性質，KNN 被稱為是 memory-based learning，或稱為 instance-based learning。訓練好之後，在使用訓練時未出現過的新資料測試 KNN，或實際使用 KNN 進行辨識的時候，就會找出最接近新資料的 k 個已知的舊資料出來，根據這 k 個點屬於哪種，來決定新資料是屬於哪一種。

而 Decision Tree 是應用之前學過的 entropy 來實作。一開始在根節點，對於整個資料的不確定性是最大的，這時做一個條件判斷。如果這個條件判斷的答案可以讓整個資料的不確定性降低的程度最大，當做完這個條件判斷，走到下一個節點之後，這時的資料分佈就是目前為止不確定性最低的結果。在每個節點上要進行判斷時所的問題，應該要選擇可以讓整個資料分佈的 entropy 降低最多的問題。重複上面的步驟。在下一個節點上，也選擇一個可以讓整個資料分佈 entropy 降低最多的問題作為此節點要做的判斷。

iv. 分析(預測)方法過程

將資料集中的資料處理完後，就可以將處理完後的資料切分成訓練與測試用的資料。在分訓練與測試的資料時要小心"Data Snooping"的問題，可能處理資料的時候有做標準化或正規化等等有關統計量的操作，這些操作千萬不能在切分資料前就先做，否則測試出來的準確度會不準確，但在這次的報告中，皆無此種操作，所以不會有"Data Snooping"的問題。

切分資料是使用 sklearn.model_selection 中的 train_test_split()函式，參數 test_size 設為 0.2，random_state 設為 50，並將資料切成 x_train、x_test、y_train、y_test(訓練與測試的資料比例為 8:2)。

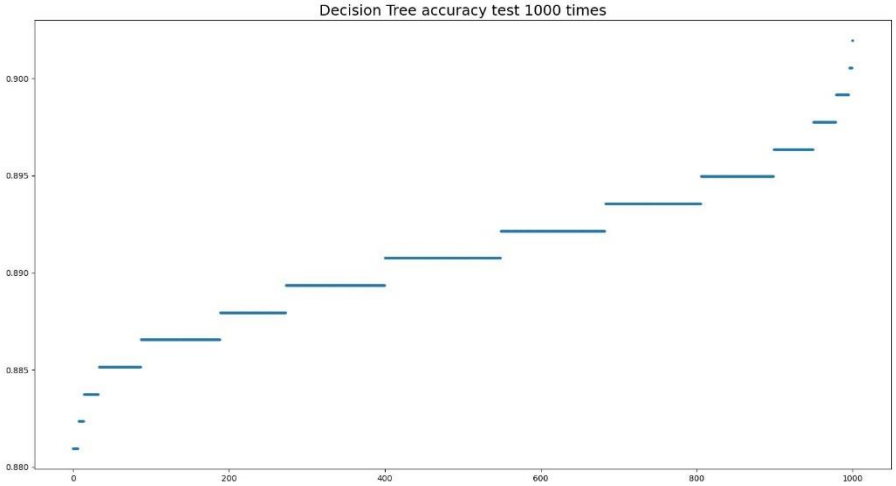
訓練 Decision Tree 模型是使用 sklearn.tree 中的 DecisionTreeClassifier，參數 criterion 設為"entropy"，代表以 entropy 的下降程度作為節點分裂的判斷標準，訓練完後用測試資料計算準確度並印出，然後將訓練出來的樹存成圖檔呈現。

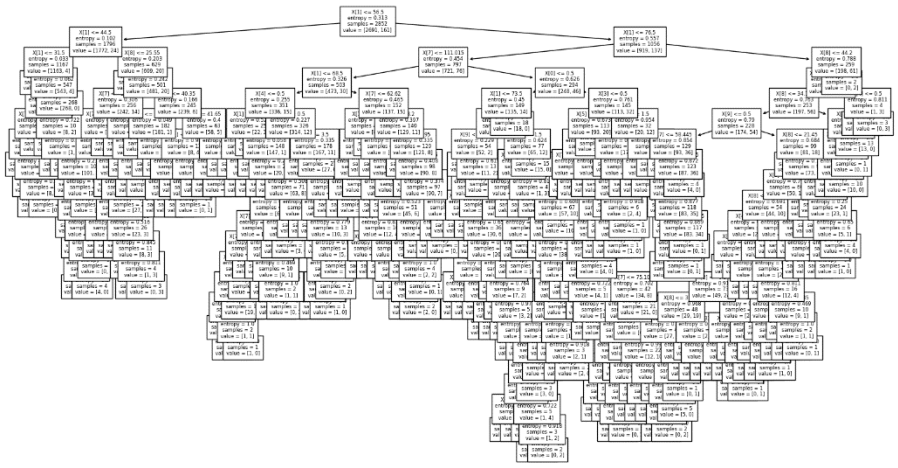
而訓練 KNN 模型是使用 sklearn.neighbors 中的 KNeighborsClassifier(k)，其中參數 k 是 KNN 中的 k 值，過程中會測試 k=1 到 15 的各 k 值準確度，並以圖表呈現。

三、 研究結論

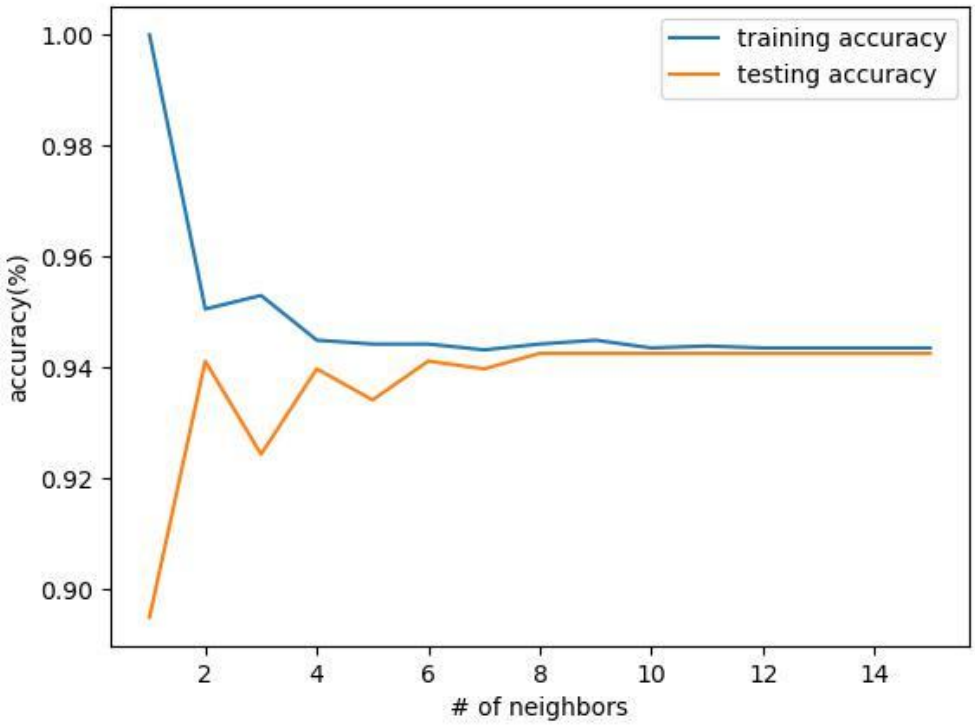
i. 實驗結果與討論

訓練了幾次 Decision Tree 模型並測試準確度後，發現模型的準確度大約落在 87.8%到 91%之間，我推測可能是因為在挑選節點分裂標準時，有多個決策的 entropy 下降程度是差不多的，但他們若在更下層的地方當作節點分裂標準，entropy 下降程度會有落差，造成準確度也會有落差。下圖為 Decision Tree 模型分別訓練 1000 次的準確度分布與最高準確度的樹(最高準確度為 90.196%)。





下圖為 KNN 模型，當為 k=1 到 k=15 的準確度折線圖，可以看出基本上在 k>6 時，訓練的準確度與測試的準確度都趨於收斂，都在 94%左右，比 Decision Tree 模型最高的 90.056%略高一截。而 k=1 時的訓練準確度一定是 100%，由 KNN 模型的定義就可以得知。



ii. 結論

由以上實驗結果得知，如果要從上面 2 個模型選其中一個來猜測病患是否中風，則使用 KNN 模型會比 Decision Tree 模型來的更好一些(準確度更高)。

在本次的報告中，對於資料集中的統計分析並不是很完善，未來可以再更詳細的分析與討論各個欄位之間的相關性、各欄位與是否中風的相關性(有無線性相關)、或是可以多訓練不同種的模型來比較準確度，例如 Naïve Bayes 模型、回歸分析模型等等。

四、 附件(程式碼)

```
import csv
import math
import statistics
import numpy as np
import seaborn as sns
import pandas as pd                # 資料處理套件
import matplotlib.pyplot as plt    # 資料視覺化套件
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier

def print_cloumn_stv(cloumn_name,cloumn_data):
    print(f"\n{cloumn_name}欄位中最大值:{max(cloumn_data)}"
          +f"\n{cloumn_name}欄位中最小值:{min(cloumn_data)}"
          +f"\n{cloumn_name}e 欄位平均值:{statistics.mean(cloumn_data)}"
          +f"\n{cloumn_name}欄位中位數:{statistics.median(cloumn_data)}"
          +f"\n{cloumn_name}欄位變異
數:{statistics.variance(cloumn_data)}"
          +f"\n{cloumn_name}欄位標準差:{statistics.stdev(cloumn_data)}")

def convalue_plt_scatter(cloumn_no):
    plt.figure(figsize=(19,10))
    plt.scatter([j for j in range(len(age_cloumn))],age_cloumn,s=5)
    plt.title(head[cloumn_no], {"fontsize" : 18})  # 設定標題及其文字大小
    plt.savefig(f"scatter chart cloumn
{cloumn_no}_{head[cloumn_no]}.jpg",pad_inches=0.0)
    plt.close()

try:
    #開檔
    f=open("healthcare-dataset-stroke-data.csv","r")
    rd=csv.reader(f,delimiter=",")
    #不取欄位名稱
    head=next(rd)
```

```
    data=[]
    for row in rd:
        data.append(row)

    #去除不可用資料(id 欄位、smoking_status:Unknown)
    cleared_data=[]
    for i in range(len(data)):
        if data[i][10]!="Unknown":
            cleared_data.append(data[i][1:len(data[0])])


    #取得 bmi data 的中位數、非數值欄位分布狀況
    D=[ [] for i in range(len(cleared_data[0]))]
    bmi_data=[]
    for i in range(len(cleared_data)):
        if cleared_data[i][8]!="N/A":
            bmi_data.append(float(cleared_data[i][8]))
        for j in range(len(cleared_data[0])):
            D[j].append(cleared_data[i][j])

    bmi_mean=float(statistics.median(bmi_data))


    for i in [0,2,3,4,5,6,9,10]:
        value, counts = np.unique(D[i], return_counts=True)
        print(f"{head[i+1]}欄位中各值出現次數:", dict(zip(value,
counts)))
        print(f"{head[i+1]}欄位中各值出現比例:", dict(zip(value,
counts/len(D[i]))))
        #根據分佈計算此欄位的 entropy
        probs=counts/len(D[i])  # numpy 陣列可以直接除以一個數值
        entropy=sum(-p*math.log(p,2) if p>0 else 0 for p in probs)
        print(f"{head[i+1]}欄位的 entropy:{entropy}\n")

        plt.figure(figsize=(6,9))               # 顯示圖框架大小
        plt.pie(counts,                         # 數值
                labels = value,                 # 標籤
```

```
                autopct = "%1.1f%%",            # 將數值百分比並留到小數點
一位
                pctdistance = 0.6,              # 數字距圓心的距離
                textprops = {"fontsize" : 12})  # 文字大小

        plt.axis('equal')                       # 使圓餅圖比例相等
        plt.title(head[i+1], {"fontsize" : 18})  # 設定標題及其文字大小
        plt.legend(loc = "best")                # 設定圖例及其位置為最佳

        #plt.show()
        plt.savefig(f"Pie chart cloumn {i+1}_{head[i+1]}.jpg",     # 儲存
圖檔
                    bbox_inches='tight',              # 去除座標軸占用的空
間
                    pad_inches=0.0)                   # 去除所有白邊
        plt.close()


    bmi_count=0   #bmi missing data 的格數
    #將數值資料從 string 轉型成 float or int、編碼
    age_cloumn=[]
    avg_glucose_level=[]
    encoding=[{"Female":0,"Male":1,"Other":3},{},{},{}
              ,{"No":0,"Yes":1},{"Never_worked":0,"children":1,"Private"
:2,"Self-employed":3,"Govt_job":4}
              ,{"Rural":0,"Urban":1},{},{},{"never smoked":0,"formerly
smoked":1,"smokes":2},{}]
    for i in range(len(cleared_data)):
        for j in [0,4,5,6,9]:
            cleared_data[i][j]=encoding[j][cleared_data[i][j]]
        for j in [2,3,10]:
            cleared_data[i][j]=int(cleared_data[i][j])
        for j in [1,7]:
            cleared_data[i][j]=float(cleared_data[i][j])
            if j==1:
                age_cloumn.append(cleared_data[i][j])
            if j==7:
                avg_glucose_level.append(cleared_data[i][j])
```

```
        if cleared_data[i][8]!="N/A":
            cleared_data[i][8]=float(cleared_data[i][8])
            bmi_count=bmi_count+1
        else:
            #將 bmi missing data 填入 bmi data 的中位數
            cleared_data[i][8]=bmi_mean

    age_cloumn.sort()
    avg_glucose_level.sort()
    bmi_data.sort()

    print_cloumn_stv("age",age_cloumn)
    print_cloumn_stv("avg_glucose_level",avg_glucose_level)
    print_cloumn_stv("bmi",bmi_data)

    convalue_plt_scatter(2)
    convalue_plt_scatter(8)
    convalue_plt_scatter(9)

    print(f"\n 資料集總欄位數:{len(head)}\n"
        +f"資料集取用欄位數:{len(cleared_data[0])}\n"
        +f"原始資料筆數:{len(data)}\n"
        +f"可用資料筆數:{len(cleared_data)}\n"
        +f"不可用資料筆數(smoking_status:Unknown):{len(data)-
len(cleared_data)}\n"
        +f"bmi 值無遺失的資料筆數:{bmi_count}")

    #建立 Decision Tree 模型並回報模型的準確度
#****************將資料切成訓練/測試的地方(以下)***************
    #將 data、target 分開
    data_information=[]
    data_class=[]
    for i in range(len(cleared_data)):
        data_information.append(cleared_data[i][0:-1])
        data_class.append(cleared_data[i][len(cleared_data[i])-1])
    #切分訓練、測試資料
    x_train,x_test,y_train,y_test=train_test_split(data_information,data
_class,test_size=0.2,random_state=50)
```

```
#****************將資料切成訓練/測試的地方(以上)***************

#****************產生模型與訓練模型的的地方(以下)***************
    train_accuracy=[]
    test_accuracy=[]
    accuracy_plt=[]
    for i in range(1001):
        dt=DecisionTreeClassifier(criterion="entropy")

        dt.fit(x_train,y_train)
        #******計算(呼叫套件提供方法)準確度******
        #print(f"{i}. Accuracy={dt.score(x_test,y_test)}")
        accuracy_plt.append(dt.score(x_test,y_test))
        if dt.score(x_test,y_test)>0.9:
            #畫數
            plt.figure(figsize=(19,10))
            tree.plot_tree(dt,fontsize=6)
            plt.savefig(f'tree_high_dpi{math.ceil(dt.score(x_test,y_test
)*1000000)}', dpi=100)
            plt.close()

    accuracy_plt.sort()
    print("最高準確度:"+str(max(accuracy_plt)))
    print("最低準確度:"+str(min(accuracy_plt)))
    plt.figure(figsize=(19,10))
    plt.scatter([j for j in range(len(accuracy_plt))],accuracy_plt,s=5)
    plt.title("Decision Tree accuracy test 1000 times", {"fontsize" :
18})  # 設定標題及其文字大小
    plt.savefig(f"Decision Tree accuracy test 1000
times(scatter).jpg",pad_inches=0.0)
    plt.close()

#****************產生模型與訓練模型的的地方(以上)***************

    #------------KNN 模型------------
    k_range=range(1,16) # 測試 k=1 到 10
    for k in k_range:
        knn=KNeighborsClassifier(k)
```

```
        knn.fit(x_train,y_train)
        train_accuracy.append(knn.score(x_train,y_train))
        test_accuracy.append(knn.score(x_test,y_test))

    plt.plot(k_range,train_accuracy,label="training accuracy")
    plt.plot(k_range,test_accuracy,label="testing accuracy")
    plt.xlabel("# of neighbors")
    plt.ylabel("accuracy(%)")
    plt.legend()
    plt.savefig("KNN.jpg")
    #plt.show()
    plt.close()

    #---------------------------------
except Exception as e:
    print(e)
```

五、 參考資料

Stroke Prediction Dataset | Kaggle
https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset

1112 數據科學理論與方法-蔣益庭(應數三乙) [MA309H] 上課的投影片