 Giảng viên: VÕ XUÂN THỂ: voxuanthephd@gmail.com (công việc) tranhailua@yahoo.com (cá nhân)

G?.stt

HỌ TÊN: **MSSV:** **KÝ TÊN:**

TRƯỜNG: **HCMUTE**

MÔN: Lập trình Python

NGÀY: ..../...../..... (BUỔI HỌC SỐ ?)

Riêng buổi học Phòng máy ghi thêm, SỐ MÁY: PHÒNG MÁY:

==========

Buổi 4_... = Bài tập 3: EDA [Exploratory Data Analysis] (Python EDA)

LẬP TRÌNH PYTHON PHÂN TÍCH DỮ LIỆU THĂM DÒ

1. GIỚI THIỆU CHỦ ĐỀ

1.1. Giới thiệu bài toán (Yêu cầu đặt ra)

- Xét Bài toán phân tích dữ liệu trời mưa trong quá khứ tại một số bang của Úc, thăm dò (dự đoán) khả năng có mưa hay không trong thời gian tới.
- thông qua phân tích dữ liệu quá khứ

1.2. Tài liệu và nguồn thực nghiệm

+ Dữ liệu quá khứ: weatherAUS.csv {sv tự search trên Internet và tải file .csv}

Vd: https://rdrr.io/cran/rattle.data/man/weatherAUS.html

A CSV version of this dataset is available as https://rattle.togaware.com/weatherAUS.csv.

191 K quan sát thời tiết tại nhiều địa điểm ở Úc

Gồm 23 thuộc tính input để đoán thuộc tính thứ 24: target variable = "RainTomorrow"[Y|N]

1.3. Tiến trình thực hiện

- Tiến trình phân tích dữ liệu thăm dò (EDA) thông thường được thực hiện qua 3 GĐ chính:

GIAI ĐOẠN 1: NẠP DỮ LIỆU GỐC (PRIMARY INPUT DATA LOAD): Bước 1_Bước 2

GIAI ĐOẠN 2: TIỀN XỬ LÝ (PRE-PROCESSING): Bước 3_Bước 7

GIAI ĐOẠN 3: PHÂN TÍCH DỮ LIỆU THĂM DÒ (EDA: Exploratory Data Analysis): Bước 8_

2. CÁC BƯỚC THỰC HIỆN

GIAI ĐOẠN 1: NẠP DỮ LIỆU GỐC (PRIMARY INPUT DATA LOAD)

Bước 1: Nạp các thư viện cần thiết

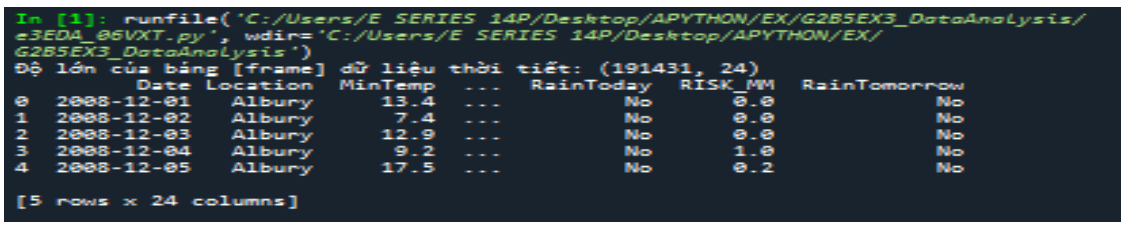
```
import numpy as np #Numeric Python: Thư viện về Đại số tuyến tính tính
import pandas as pd #Python Analytic on Data System: For data processing (Thư viện xử lý dữ liệu)
from scipy import stats # thư viện cung cấp các công cụ thống kê [statistics]
                        sub-lib của science python [các công cụ khoa học]
from sklearn import preprocessing #Thư viện tiền xử lý DL (XL ngoại lệ: Isolated)
from sklearn.feature_selection import SelectKBest, chi2 #Nạp hàm Thư viện phân tích dữ liệu thăm dò
```

Bước 2: Tải tập dữ liệu

```
# Bước 2: Tải tập dữ liệu: Load the data set (Nạp tập dữ liệu)
# ./sttHoTen_weatherAUS.csv
df = pd.read_csv('./sttHoTen_weatherAUS.csv')
# Display the shape of the data set (xem lượng dòng & cột dữ liệu của tập DL gốc)
print('Độ lớn của bảng [frame] dữ liệu thời tiết:',df.shape)
#Display data (Hiển thị dữ liệu dạng mảng 5 dòng đầu của tập DL gốc)
print(df[0:5])
```

KẾT QUẢ:

```
# Display the shape of the data set (xem lượng dòng & cột dữ liệu của tập DL gốc)
print('Độ lớn của bảng [frame] dữ liệu thời tiết:',df.shape)
----Tập dữ liệu gốc gồm có 191 431 dòng , 24 cột

Độ lớn của bảng [frame] dữ liệu thời tiết: (191431, 24)

========================

#Display data (Hiển thị dữ liệu dạng mảng 5 dòng đầu của tập DL gốc)
print(df[0:5])

-----------

     Date        Location  MinTemp  ...  RainToday  RISK_MM  RainTomorrow

0  2008-12-01   Albury     13.4     ...         No      0.0            No

1  2008-12-02   Albury      7.4     ...         No      0.0            No

2  2008-12-03   Albury     12.9     ...         No      0.0            No

3  2008-12-04   Albury      9.2     ...         No      1.0            No

4  2008-12-05   Albury     17.5     ...         No      0.2            No

[5 rows x 24 columns]
```



GIAI ĐOẠN 2: TIỀN XỬ LÝ (PRE-PROCESSING)

Bước 3: Xử lý CỘT dữ liệu NULL quá nhiều OR không có giá trị phân tích

```
# Checking for null values (Kiểm tra giá trị null = đếm số dòng có dữ liệu ứng từng thuộc# tính)
print(df.count().sort_values()) #df.count(): đếm số lượng dòng có dữ liệu của df,
                                .sort_values() sx tăng dần
```

KẾT QUẢ:

```
Sunshine..... 86442 -----Cột(Ngày nắng) có 86 442 dòng có dữ liệu =>191431-86442= 104 989 dòng NULL (55%)
Evaporation..... 94845 -----Cột(Độ ẩm)
Cloud3pm         105192
Cloud9am..... 111072 -----Cột(Có mây 9h sáng)     ……..80 359 dòng NULL (42%)
Pressure9am      170333
Pressure3pm      170342
WindDir9am       176693
WindGustDir      176954
WindGustSpeed    177055
WindDir3pm       183702
Humidity3pm      183959
Temp3pm          184959
WindSpeed3pm     185025
RISK_MM          186393
RainTomorrow     186393
RainToday        186394
Rainfall         186394
Humidity9am      187603
WindSpeed9am     188080
Temp9am          188599
MinTemp          188671
MaxTemp          188876
Location         191431
Date             191431
dtype: int64 ..... -----Kiee6u3 dữ liệu gốc là Integer 64 bit
```

Nhận xét: kết quả cho thấy bốn cột đầu tiên có hơn 40% giá trị null (80 _ 100 nghìn / gần 191 nghìn records), do đó, tốt nhất nên loại các cột này [TIỀN XỬ LÝ]

tiền xử lý cột dữ liệu cần loại các cột dữ liệu [biến|| thuộc tính] có dữ liệu không đáng kể.

vì nó chỉ làm tăng tính toán không giá trị.

+ BỎ “biến” 'vị trí'[Location] và 'ngày'[Date] vì không có ý nghĩa trong dự đoán thời tiết.

+ BỎ “biến” 'RISK_MM' (lượng mưa vào ngày hôm sau) vì ta muốn dự đoán 'Rain Tomorrow' nên RISK_MM ko có tác dụng trong trường hợp này (có thể làm sai lệch kết quả)

KẾT LUẬN: BỎ 7 CỘT: Sunshine, Evaporation, Cloud3pm, Cloud9am, 'vị trí'[Location], 'ngày'[Date], ‘RISK_MM'

```
df = df.drop(columns=['Sunshine','Evaporation','Cloud3pm','Cloud9am','Location','Date','RISK_MM'],axis=1)
print(df.shape) # kiểm tra lại số lượng cột & dòng của df sau khi XL NULL cột
```

⇨ Độ lớn của bảng [frame] dữ liệu thời tiết: **(191431, 24)** => (191431, 14) => (191431, 11)**=>(191431, 17)**
⇨ SAU KHI XỬ LÝ CỘT DL NULL = CÒN LẠI 17 CỘT

```
Sunshine          86442
Evaporation       94845
Cloud3pm         105192
Cloud9am         111072
Pressure9am      170333
Pressure3pm      170342
WindDir9am       176693
WindGustDir      176954
WindGustSpeed    177055
WindDir3pm       183702
Humidity3pm      183959
Temp3pm          184959
WindSpeed3pm     185025
RISK_MM          186393
RainTomorrow     186393
RainToday        186394
Rainfall         186394
Humidity9am      187603
WindSpeed9am     188080
Temp9am          188599
MinTemp          188671
MaxTemp          188876
Location         191431
Date             191431
dtype: int64
(191431, 17)
```

Bước 4: Xử lý DÒNG dữ liệu NULL

Xóa tất cả các dòng có giá trị null trong tập FRAME dữ liệu.

```
# Bước 4: Xử lý DÒNG dữ liệu NULL
# Removing null values (Xóa tất cả các dòng có giá trị null trong tập FRAME dữ liệu.)
df = df.dropna(how='any')
print(df.shape)# kiểm tra lại số lượng cột & dòng của df sau khi XL NULL các dòng DL
```

KẾT QUẢ: (147486, 17)

⇨ Độ lớn của bảng [frame] dữ liệu thời tiết: **(191431, 17) =>(147486, 17)**
SAU KHI XỬ LÝ DÒNG DL NULL = CÒN LẠI 147 486 dòng DL (vẫn 17 CỘT)

⇨ Số lượng dòng DL trống (không có số liệu: NA = .dropna(…)) là: **43 945 (23%)**

Bước 5: Xử lý loại bỏ các giá trị ngoại lệ (cá biệt): isolated

+ kiểm tra tập dữ liệu có bất kỳ ngoại lệ nào không.

+ Một ngoại lệ là một điểm dữ liệu khác biệt đáng kể so với các quan sát khác.

+ Các ngoại lệ thường xảy ra do tính toán sai (or do thiết bị hư hỏng,..) trong khi thu thập dữ liệu.

```
# Thư viện xử lý DL ngoại lệ

from scipy import stats

#kiểm tra tập dữ liệu có bất kỳ ngoại lệ nào không
z = np.abs(stats.zscore(df._get_numeric_data())) # Dò tìm và lấy các giá trị cá biệt trong tập
dữ liệu gốc thông qua điểm z (z_score)
print('MA TRAN Z-SCORE\n')
print(z) # in ra tập (ma trận) các giá trị z-score từ tập dữ liệu gốc
df= df[(z < 3).all(axis=1)] # kiểm tra và chỉ giữ lại trong df các giá trị số liệu tương ứng với
z-score < 3  # {loại các giá trị >= 3} vì các giá trị z-score >=3 tướng ứng với số liệu quá khác
biệt so với các số liệu còn lại (“cá biệt” = “ngoại lệ” = isolated}
print(df.shape) # xác định số dòng & cột dữ liệu sau khu xử lý các giá trị cá biệt
```

- **Giải thích:** `z = np.abs(stats.zscore(df._get_numeric_data()))`

[1] Lấy giá trị dữ liệu số để tính toán thống kê

`df._get_numeric_data()` =lấy các giá trị số từ df (Dataset Frame nêu trên = tập dữ liệu "gốc" thực nghiệm)
=> không sử dụng các giá trị chữ : VD `Cột "WindGustDir", "WindDir9am", . . .`
`Vì giá trị các cột này (hướng gió), có dạng chữ, như:`
`NNW = hướng gió  N (North)  S (South) W (West)  E (East) => không tính thống kê (có thể số hóa)`

[2] Hàm tính giá trị thống kê : "điểm z" {z-score}

```
stats.zscore(…tập dữ liệu: DataSet) lấy từ thư viện con (Sub Lib) của thư viện Scipy
      from scipy import stats # sub thư viện thống kê ... statistics (scipy = science python)
```

+ Việc xác định giá trị cá biệt (Ngoại lệ) {Isolated} là nhờ vào chỉ số thống kê "điểm z" = Z score

`+ Zscore : là 1 chỉ số thống kê (Statistics), tính theo công thức`

```
   =(DataPoint-AVERAGE(DataSet))/STDEV(DataSet)
```

Minh họa:

DataSet = { 1 , 9 , 8, 6}

Average = (1+9+8+6)/4 = 6

Độ lệch (so với giá trị bình quân): -5 3 2 0

STDEV = độ lệch chuẩn (standard…) = …

Tính z-scores trong Python dùng scipy.stats.zscore, có cú pháp:

1. scipy.stats.**zscore**(a, axis=0, ddof=0, nan_policy='propagate')
2. Step 1: Import modules.
3. Step 2: Create an array of **values**.
4. Step 3: Calculate the **z-scores** for each value in the array.

- Giá trị cá biệt (Ngoại lệ) {Isolated} là giá trị mà độ khác biệt của nó là "quá xa" so với độ lệch chuẩn

[3] Lấy giá trị tuyệt đối của số liệu thống kê z-score: NumPy.ABS(..giá_trị..)

`np.abs(….) = Absolute = giá trị tuyệt đố (từ thư viện Numeric Python)`

[4] `z = np.abs(stats.zscore(…)) là ma trận các giá trị z-score tương ứng từng điểm dữ liệu (+)`

Kết quả

```
print(z) # in ra tập (ma trận) các giá trị z-score từ tập dữ liệu gốc
```

```
MA TRAN Z-SCORE
[[0.12286663 0.12399474 0.19709472 ... 1.16290217 0.09328701 0.06181431]
 [0.82989218 0.18891363 0.26781661 ... 1.06196028 0.04644989 0.30112384]
 [0.04347006 0.27425227 0.26781661 ... 0.93217786 0.54682025 0.14143106]
 ...
 [0.6786426  1.36943156 0.26781661 ... 1.1484819  1.17131513 1.38993829]
 [0.75803917 1.58277818 0.26781661 ... 1.6243508  1.26498936 1.68028881]
 [0.98034956 1.92413276 0.26781661 ... 1.72529269 1.51478731 2.14484964]
]
```

Số liệu z-scores và độ lớn dữ liệu (shape) sau khi xử lý loại các số liệu "cá biệt"

```
MA TRAN Z-SCORE

[[0.12286663 0.12399474 0.19709472 ... 1.16290217 0.09328701 0.06181431]
 [0.82989218 0.18891363 0.26781661 ... 1.06196028 0.04644989 0.30112384]
 [0.04347006 0.27425227 0.26781661 ... 0.93217786 0.54682025 0.14143106]
 ...
 [0.6786426  1.36943156 0.26781661 ... 1.1484819  1.17131513 1.38993829]
 [0.75803917 1.58277818 0.26781661 ... 1.6243508  1.26498936 1.68028881]
 [0.98034956 1.92413276 0.26781661 ... 1.72529269 1.51478731 2.14484964]]
(141174, 17)
```

```
df= df[(z < 3).all(axis=1)] # kiểm tra và chỉ giữ lại trong df các giá trị số liệu tưng ứng với z-score < 3  # {loại các giá trị >= 3} vì các giá trị z-score >=3 tướng ứng với số liệu quá khác biệt so với các số liệu còn lại ("cá biệt" = "ngoại lệ" = isolated}
```

GIẢI THÍCH: Việc chọn giá trị z-scores =3 để xác định các số liệu "cá biệt" dựa vào thực tế các giá trị trong ma trận z-score (nêu trên) = giá trị này "ướn" tương đối (có thể điều chỉnh)

```
print(df.shape) # xác định số dòng & cột dữ liệu sau khu xử lý các giá trị cá biệt
```

(147486, 17) => **(141174, 17);** đã LOẠI **6 312** dòng dữ liệu "cá biệt"

Bước 6: Thay thế các vị trí giá trị 0 và 1 bởi CÓ (Yes) và KHÔNG (No).

```
# Bước 6: Thay thế các vị trí giá trị  0 và 1 bởi CÓ (Yes) và KHÔNG (No).
#Thay thế yes (CÓ) and no (KO) vào vị trí giá trị 1 (Y) và 0 (N) tương ứng cột|biến RainToday và# RainTomorrow
df['RainToday'].replace({'KHONG': 'No', 'CO': 'Yes'},inplace = True)
df['RainTomorrow'].replace({'KHONG': 'No', 'CO': 'Yes'},inplace = True)
# chú ý = phải lấy & chạy lại từ DataSet gốc
```

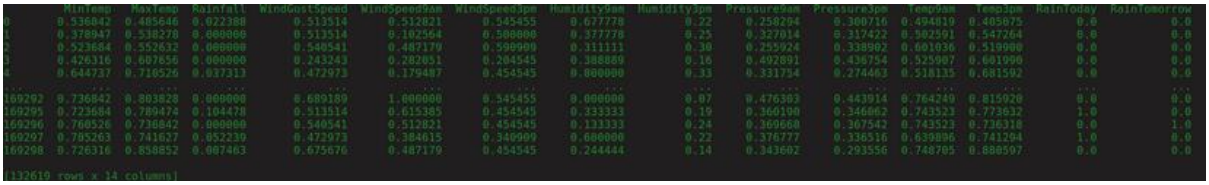
Bước 7: Chuẩn hóa (Rời rạc hóa) tập dữ liệu Input dùng ..MaxMin

+ Chuẩn hóa dữ liệu để tránh sai sót dự đoán kết quả.

Sử dụng hàm MinMaxScaler có trong thư viện sklearn.

```
# Thư viện chuẩn hóa DL

from sklearn import preprocessing


#Bước 7: Chuẩn hóa (Rời rạc hóa) tập dữ liệu Input dùng ..MaxMin
# CHUẨN HÓA DL
scaler = preprocessing.MinMaxScaler() #preprocessing là Sub-Library của thư viện sklearn
                                      => hàm .MinMaxScaler() Rời rạc hóa tập dữ liệu Input
scaler.fit(df)

df = pd.DataFrame(scaler.transform(df), index=df.index, columns=df.columns)
                    # Rời rạc hóa số liệu theo thang đo scaler
df.iloc[4:10]


print(df)
```

Kết quả



GIAI ĐOẠN 3: PHÂN TÍCH DỮ LIỆU THĂM DÒ : (EDA = Exploratory Data Analysis)

Sử dụng hàm `SelectKBest` có trong thư viện `sklearn`:

```
# Nạp hàm Thư viện phân tích dữ liệu thăm dò

from sklearn.feature_selection import SelectKBest, chi2
```

Bước 8: Nạp các thuộc tính quan trọng vào Dataset

```
#The important features are put in a data frame
df = df[['Humidity3pm','Rainfall','RainToday','RainTomorrow']]
```

Bước 9: thực hiện các tính toán trên mô hình phân tích

```
#To simplify computations we will use only one feature (Humidity3pm) to build the model
X = df
X = df[['Humidity3pm']]
y = df[['RainTomorrow']]
X = df.loc[:,df.columns!='RainTomorrow']
y = df[['RainTomorrow']]
selector = SelectKBest(chi2, k=3)
selector.fit(X, y)
X_new = selector.transform(X)
df(['Rainfall', 'Humidity3pm', 'RainToday'], dtype='object')


print(X.columns[selector.get_support(indices=True)])
```

Đầu ra cho chúng ta ba biến dự đoán quan trọng:

1. Rainfall
2. Humidity3pm
3. RainToday

```
Index(['Rainfall', 'Humidity3pm', 'RainToday'], dtype='object')

print(X.columns[selector.get_support(indices=True)])
```

để đơn giản hóa các tính toán, sẽ chỉ gán một trong các biến quan trọng này làm đầu vào.

```
# PHÂN TÍCH DỮ LIỆU THĂM DÒ : EDA

#The important features are put in a data frame

df = df[['Humidity3pm','Rainfall','RainToday','RainTomorrow']]

 #To simplify computations we will use only one feature (Humidity3pm) to build the model

X = df[['Humidity3pm']]

y = df[['RainTomorrow']]


# X = df.loc[:,df.columns!='RainTomorrow']

# y = df[['RainTomorrow']]

selector = SelectKBest(chi2, k=3)

selector.fit(X, y)

X_new = selector.transform(X)

Index(['Rainfall', 'Humidity3pm', 'RainToday'], dtype='object')

print(X.columns[selector.get_support(indices=True)])
```

Trong đoạn mã trên, x và y biểu thị đầu vào và đầu ra tương ứng.

```
# GIAI ĐOẠN 3: PHÂN TÍCH DỮ LIỆU THĂM DÒ : EDA
#Bước 8: Nạp các thuộc tính dùng để EDA: The important features are put in a data frame
df = df[['Humidity3pm','Rainfall','RainToday','RainTomorrow']]

#Bước 9: xử lý cột dữ liệu tham chiếu
#To simplify computations we will use only one feature (Humidity3pm) to build the model
X = df
x = df[['Humidity3pm']]
y = df[['RainTomorrow']]

x = df.loc[:,df.columns!='RainTomorrow']
y = df[['RainTomorrow']]

selector = SelectKBest(chi2, k=3)  #thư viện con from sklearn.feature_selection
selector.fit(X, y)
X_new = selector.transform(X)

#Bước 10: EDA
df(['Rainfall', 'Humidity3pm', 'RainToday'], dtype='object')
print(X.columns[selector.get_support(indices=True)])
```

Trong đoạn mã trên, `x` và `y` biểu thị đầu vào và đầu ra tương ứng.

3. FULL CODES (Tham khảo: sv nên cá nhân hóa thông tin bài làm)

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 25 19:12:17 2022

@author: VOXUAN
"""

# GIAI ĐOẠN 1: NẠP DỮ LIỆU GỐC (PRIMARY INPUT DATA LOAD)

# Bước 1: Nạp các thư viện cần thiết

import numpy as np #Numeric Python: Thư viện về Đại số tuyến tính tính

import pandas as pd #Python Analytic on Data System: For data processing (Thư viện xử lý dữ
liệu)

from scipy import stats # thư viện cung cấp các công cụ thống kê [statistics] sub-lib của
science python [các công cụ khoa học]

from sklearn import preprocessing # Thư viện tiền xử lý DL (XL ngoại lệ: Isolated)


from sklearn.feature_selection import SelectKBest, chi2 # Nạp hàm Thư viện phân tích dữ liệu
thăm dò

# Bước 2: Tải tập dữ liệu: Load the data set (Nạp tập dữ liệu)
# ./sttHoTen_weatherAUS.csv
df = pd.read_csv('./sttHoTen_weatherAUS.csv')
# Display the shape of the data set (xem lượng dòng & cột dữ liệu của tập DL gốc)
print('Độ lớn của bảng [frame] dữ liệu thời tiết:',df.shape)
#Display data (Hiển thị dữ liệu dạng mảng 5 dòng đầu của tập DL gốc)
print(df[0:5])

# GIAI ĐOẠN 2: TIỀN XỬ LÝ (PRE-PROCESSING)

# Bước 3: Xử lý CỘT dữ liệu NULL quá nhiều OR không có giá trị phân tích
# Checking for null values (Kiểm tra giá trị null = đếm số dòng có dữ liệu ứng từng thuộc# tính)
print(df.count().sort_values()) #df.count(): đếm số lượng dòng có dữ liệu của df, .sort_values()
sx tăng dần

df =
df.drop(columns=['Sunshine','Evaporation','Cloud3pm','Cloud9am','Location','Date','RISK_MM'],axi
s=1)
#df = df.drop(columns=['Sunshine','Evaporation','Cloud3pm','Cloud9am','Pressure9am',#
'Pressure3pm','WindDir3pm', 'WindDir9am', 'WindGustDir',#
'WindGustSpeed','Location','Date','RISK_MM'],axis=1)
print(df.shape) # kiểm tra lại số lượng cột & dòng của df sau khi XL NULL cột

# Bước 4: Xử lý DÒNG dữ liệu NULL
# Removing null values (Xóa tất cả các dòng có giá trị null trong tập FRAME dữ liệu.)
df = df.dropna(how='any')
print(df.shape) # kiểm tra lại số lượng cột & dòng của df sau khi XL NULL các dòng DL

# Bước 5: Xử lý loại bỏ các giá trị ngoại lệ (cá biệt): isolated
#kiểm tra tập dữ liệu có bất kỳ ngoại lệ nào không
z = np.abs(stats.zscore(df._get_numeric_data())) # Dò tìm và lấy các giá trị cá biệt trong tập
dữ liệu gốc thông qua điểm z (z_score)
```

```
print('MA TRAN Z-SCORE\n')
print(z) # in ra tập (ma trận) các giá trị z-score từ tập dữ liệu gốc
df= df[(z < 3).all(axis=1)] # kiểm tra và chỉ giữ lại trong df các giá trị số liệu tưng ứng với
z-score < 3  # {loại các giá trị >= 3} vì các giá trị z-score >=3 tướng ứng với số liệu quá khác
biệt so với các số liệu còn lại (“cá biệt” = “ngoại lệ” = isolated}
print(df.shape) # xác định số dòng & cột dữ liệu sau khu xử lý các giá trị cá biệt

# Bước 6: Thay thế các vị trí giá trị  0 và 1 bởi CÓ (Yes) và KHÔNG (No).

#Thay thế yes (CÓ) and no (KO) vào vị trí giá trị 1 (Y) và 0 (N) tương ứng cột|biến RainToday
và# RainTomorrow
df['RainToday'].replace({'KHONG': 'No', 'CO': 'Yes'},inplace = True)
df['RainTomorrow'].replace({'KHONG': 'No', 'CO': 'Yes'},inplace = True)

#Bước 7: Chuẩn hóa (Rời rạc hóa) tập dữ liệu Input dùng ..MaxMin
# CHUẨN HÓA DL
scaler = preprocessing.MinMaxScaler() #preprocessing là Sub-Library của thư viện sklearn=> hàm
.MinMaxScaler() Rời rạc hóa tập dữ liệu Input
scaler.fit(df)
df = pd.DataFrame(scaler.transform(df), index=df.index, columns=df.columns)                    #
Rời rạc hóa số liệu theo thang đo scaler
df.iloc[4:10]
print(df)


# GIAI ĐOẠN 3: PHÂN TÍCH DỮ LIỆU THĂM DÒ : EDA

# Bước 8: Nạp các thuộc tính quan trọng vào Dataset
#The important features are put in a data frame
df = df[['Humidity3pm','Rainfall','RainToday','RainTomorrow']]

# Bước 9: thực hiện các tính toán trên mô hình phân tích
#To simplify computations we will use only one feature (Humidity3pm) to build the model
X = df
X = df[['Humidity3pm']]
y = df[['RainTomorrow']]
X = df.loc[:,df.columns!='RainTomorrow']
y = df[['RainTomorrow']]
selector = SelectKBest(chi2, k=3)
selector.fit(X, y)
X_new = selector.transform(X)
df(['Rainfall', 'Humidity3pm', 'RainToday'], dtype='object')
print(X.columns[selector.get_support(indices=True)])
```

4. SV TỰ TÌM CÁC TẬP DỮ LIỆU PHÂN TÍCH EDA = CHỌN ĐỀ TÀI ĐAHP

Có thể Tham khảo = để chọn DataSet nguồn (chỉ là gợi ý để sv dễ tìm, sv có thể tìm nguồn khác)
https://github.com/gchoi/Dataset/commit/8e81d46872564218a78be95d53a9c6bd3b00b199