ĐẠI HỌC HUẾ

KHOA KỸ THUẬT VÀ CÔNG NGHỆ

BÁO CÁO KẾT QUẢ
HỌC KÌ I, NĂM HỌC 2022-2023
HỌC PHẦN:
HỌC MÁY 1

Giảng viên hướng dẫn: TS. Hoàng Hữu Trung

Lớp: Khoa học dữ liệu và Trí tuệ nhân tạo K3

Nhóm sinh viên thực hiện: Nhóm 5

Thừa Thiên Huế, ngày 17 tháng 12 năm 2023

Bài 1: Linear Regression

Phần 1: Giới thiệu dữ liệu

Chúng tôi đã chọn bộ dữ liệu có tên là "Salary Dataset" để thực hiện một phân tích về mối quan hệ giữa năm kinh nghiệm và mức lương. Điều này nhằm mục đích hiểu rõ hơn về cách mức lương biến đổi theo thời gian làm việc và đồng thời có thể đưa ra các nhận định hoặc xu hướng quan trọng.

Chúng tôi chọn Google Colab để phân tích bộ dữ liệu "Salary Dataset". Với môi trường làm việc trực tuyến thuận tiện, tích hợp Python và Jupyter, khả năng xử lý mạnh mẽ, và miễn phí, Colab giúp chúng tôi tiếp cận, xử lý dữ liệu một cách hiệu quả và tiết kiệm tài nguyên.

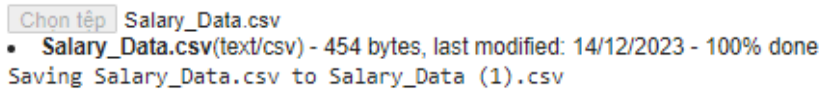
Nguồn: Salary_datasets

Link mã nguồn Python: Python_Linear regression

Phần 2: Thực hiện bài toán Linear Regression

Đầu tiên, tải lên dữ liệu từ máy tính cá nhân lên máy chủ Colab.

```
from google.colab import files
uploaded = files.upload()
```



Import các thư viện cần thiết cho việc thực hiện phân tích và mô hình hóa dữ liệu bằng Linear Regression.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

Sử dụng *df.info()* là một bước quan trọng để đạt được cái nhìn tổng quan về dữ liệu trong DataFrame bao gồm: Tổng số dòng và cột, tên và kiểu dữ liệu của mỗi cột, số lượng giá trị không rỗng, tổng số bộ nhớ chiếm dụng.

```
df = pd.read_csv('Salary_Data.csv')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
memory usage: 608.0 bytes
```

Sử dụng *df.head()* hiển thị một số dòng đầu tiên của DataFrame để có cái nhìn nhanh chóng về cấu trúc và nội dung của dữ liệu.

```
df.head()
```

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

Sử dụng *df* in ra toàn bộ DataFrame để có cái nhìn chi tiết.

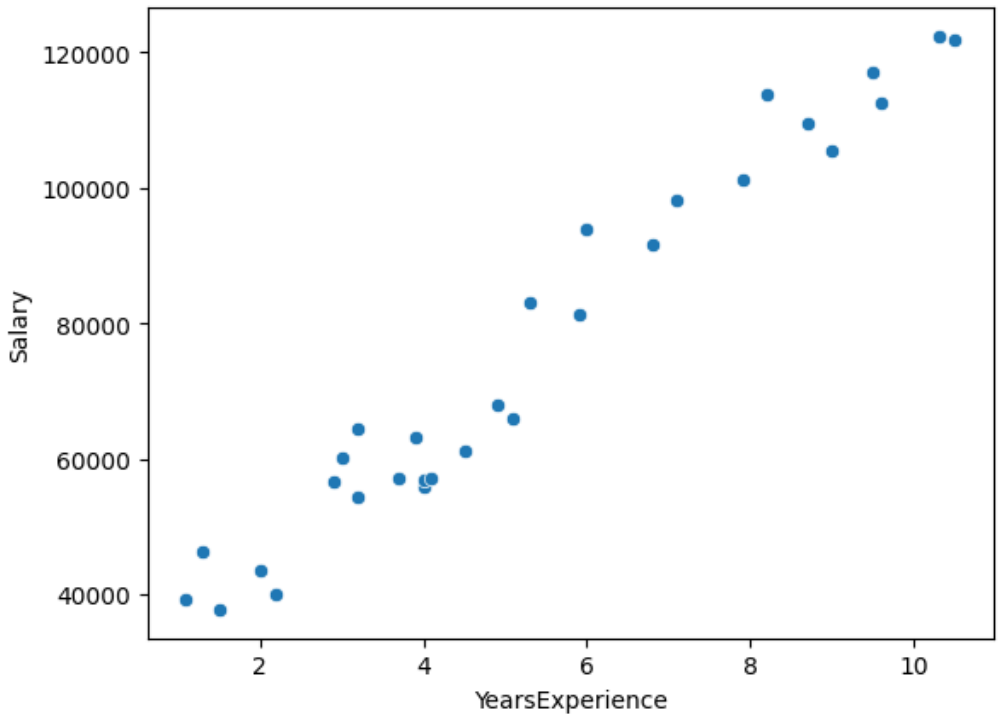
```
df
```

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0
8	3.2	64445.0
9	3.7	57189.0
10	3.9	63218.0
11	4.0	55794.0
12	4.0	56957.0
13	4.1	57081.0
14	4.5	61111.0
15	4.9	67938.0
16	5.1	66029.0
17	5.3	83088.0
18	5.9	81363.0
19	6.0	93940.0
20	6.8	91738.0
21	7.1	98273.0
22	7.9	101302.0
23	8.2	113812.0
24	8.7	109431.0
25	9.0	105582.0
26	9.5	116969.0
27	9.6	112635.0
28	10.3	122391.0
29	10.5	121872.0

Tạo một biểu đồ phân tán.

```
sns.scatterplot(data=df,x='YearsExperience',y='Salary')
```

Kết quả:



Theo như biểu đồ, những người có nhiều năm kinh nghiệm thường có mức lương cao hơn, cũng có một số người có ít năm kinh nghiệm nhưng mức lương cao.

Lọc lấy cột đầu tiên trong df (biến độc lập)

```
x = df.iloc[:,0]
x.head()
```

```
0    1.1
1    1.3
2    1.5
3    2.0
4    2.2
Name: YearsExperience, dtype: float64
```

Lọc lấy cột cuối cùng trong df (biến phụ thuộc)

```
y = df.iloc[:,-1]
y.head()
```

```
0    39343.0
1    46205.0
2    37731.0
3    43525.0
4    39891.0
Name: Salary, dtype: float64
```

Chia tập dữ liệu thành tập huấn luyện và tập kiểm tra.

```
x_train,x_test,y_train,y_test = train_test_split(x,y, test_size = 0.1 , random_state=2)
```

x_train là biến đại diện cho tập dữ liệu huấn luyện. *x_train* chứa giá trị của các biến độc lập được sử dụng để "đào tạo" mô hình, giúp nó học từ dữ liệu và có khả năng dự đoán tốt trên dữ liệu mới.

```
x_train
```

```
9      3.7
21     7.1
19     6.0
23     8.2
6      3.0
3      2.0
20     6.8
5      2.9
27     9.6
12     4.0
4      2.2
10     3.9
16     5.1
28    10.3
25     9.0
17     5.3
2      1.5
7      3.2
26     9.5
24     8.7
18     5.9
11     4.0
22     7.9
29    10.5
13     4.1
15     4.9
8      3.2
Name: YearsExperience, dtype: float64
```

x_test là biến được sử dụng để đại diện cho tập dữ liệu kiểm tra. Nó chứa giá trị của các biến độc lập mà mô hình chưa bao giờ nhìn thấy trong quá trình đào tạo. *x_test* giúp đánh giá hiệu suất của mô hình trên dữ liệu mới và kiểm tra khả năng tổng quát hóa của nó.

```
x_test
```

```
1     1.3
0     1.1
14    4.5
Name: YearsExperience, dtype: float64
```

y_train là biến được sử dụng để đại diện cho tập dữ liệu kết quả huấn luyện. *y_train* chứa giá trị của biến phụ thuộc (kết quả) tương ứng với *x_train* - tức là giá trị mà mô hình cần học cách dự đoán từ các biến độc lập.

```
y_train
```

```
9      57189.0
21     98273.0
19     93940.0
23    113812.0
6      60150.0
3      43525.0
20     91738.0
5      56642.0
27    112635.0
12     56957.0
4      39891.0
10     63218.0
16     66029.0
28    122391.0
25    105582.0
17     83088.0
2      37731.0
7      54445.0
26    116969.0
24    109431.0
18     81363.0
11     55794.0
22    101302.0
29    121872.0
13     57081.0
15     67938.0
8      64445.0
Name: Salary, dtype: float64
```

y_test là biến được sử dụng để đại diện cho tập dữ liệu kết quả kiểm tra. *y_test* chứa giá trị của biến phụ thuộc (kết quả) tương ứng với **x_test**. Điều này giúp đánh giá khả năng dự đoán của mô hình trên dữ liệu mới, chưa được mô hình nhìn thấy trong quá trình đào tạo.

```
y_test
```

```
1     46205.0
0     39343.0
14    61111.0
Name: Salary, dtype: float64
```

Khởi tạo một mô hình tuyến tính và huấn luyện mô hình Linear Regression trên tập dữ liệu huấn luyện.

```
lr = LinearRegression()
```

```
lr.fit(x_train.values.reshape(-1,1),y_train.values.reshape(-1,1))
```

```
LinearRegression
LinearRegression()
```

Dự đoán giá trị của biến phụ thuộc (kết quả) trên tập dữ liệu kiểm tra *(x_test)*.

```
lr.predict(x_test.values.reshape(-1,1))
```

```
array([[37007.26618797],
       [35075.00926647],
       [67923.37693209]])
```

*a = lr.coef_* được sử dụng để lấy giá trị hệ số (coefficient) từ mô hình Linear Regression đã được đào tạo *(lr)*.

*b = lr.intercept_* được sử dụng để lấy giá trị của hệ số chệch (intercept) từ mô hình Linear Regression đã được đào tạo *(lr)*. Hệ số chệch là giá trị dự đoán của biến phụ thuộc khi tất cả các biến độc lập đều bằng 0.

```
a = lr.coef_
a
```

```
array([[9661.28460754]])
```

```
b = lr.intercept_
b
```

```
array([24447.59619818])
```

Thay số cho ra kết quả.

y = ax + b

```
9661.28460754 * 1.3 + 24447.59619818
```

```
37007.266187982
```

Hiển thị các thống kê mô tả của DataFrame bao gồm số lượng, mean, std, min, max và các phần vị của các biến trong DataFrame.

```
print(df.describe())
```

```
       YearsExperience         Salary
count        30.000000      30.000000
mean          5.313333   76003.000000
std           2.837888   27414.429785
min           1.100000   37731.000000
25%           3.200000   56720.750000
50%           4.700000   65237.000000
75%           7.700000  100544.750000
max          10.500000  122391.000000
```

Thực hiện kiểm định Shapiro-Wilk để kiểm tra tính chuẩn của phân phối của biến 'Salary'. Mục đích của kiểm định Shapiro-Wilk trong trường hợp này là kiểm tra xem biến 'Salary' có phân phối gần với phân phối chuẩn hay không, điều này có thể ảnh hưởng đến sự hiệu quả của các phương pháp thống kê dựa trên giả định phân phối chuẩn.

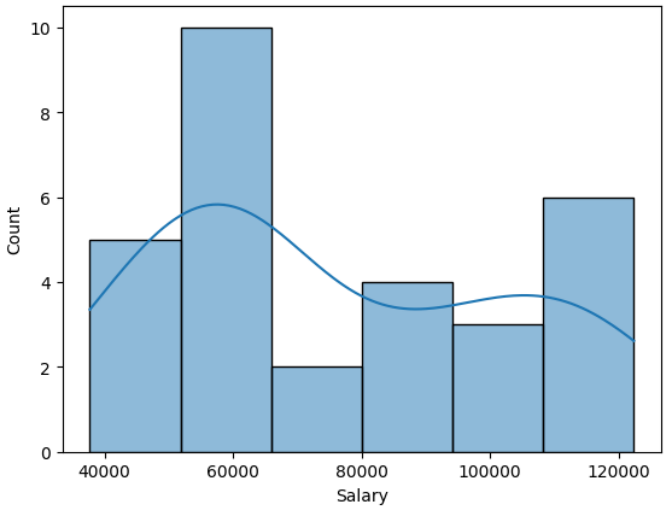
```
from scipy.stats import shapiro
stat, p_value = shapiro(df['Salary'])
print(f'Statistics={stat:.3f}, p-value={p_value:.3f}')
```

```
Statistics=0.910, p-value=0.015
```

Tạo biểu đồ histogram của biến 'Salary' và cùng lúc vẽ đường phân phối ước lượng (Kernel Density Estimate - KDE) lên trên histogram.

```
sns.histplot(df['Salary'], kde=True)
```

Kết quả:

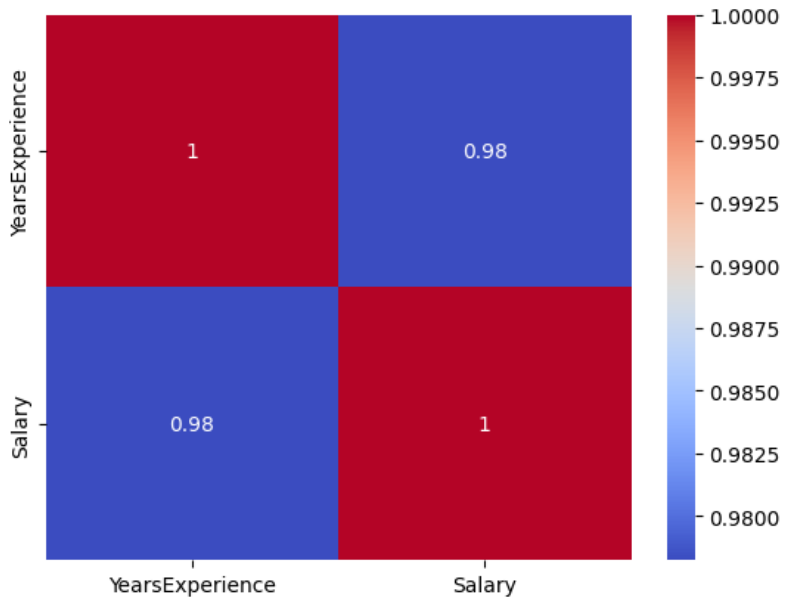


Biểu đồ này cho thấy phân bố tần số của mức lương trong dữ liệu. Đường KDE cho thấy rằng mức lương có xu hướng tập trung ở khoảng 50.000 đến 100.000 đô la.

Tạo ma trận tương quan và tạo biểu đồ heatmap của ma trận tương quan đó.

```
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
```

Kết quả:



Biểu đồ này cho thấy mối quan hệ giữa các biến trong DataFrame. Các giá trị dương cho thấy mối quan hệ thuận, trong khi các giá trị âm cho thấy mối quan hệ nghịch. Cường độ của mối quan hệ được biểu thị bằng độ đậm của màu.

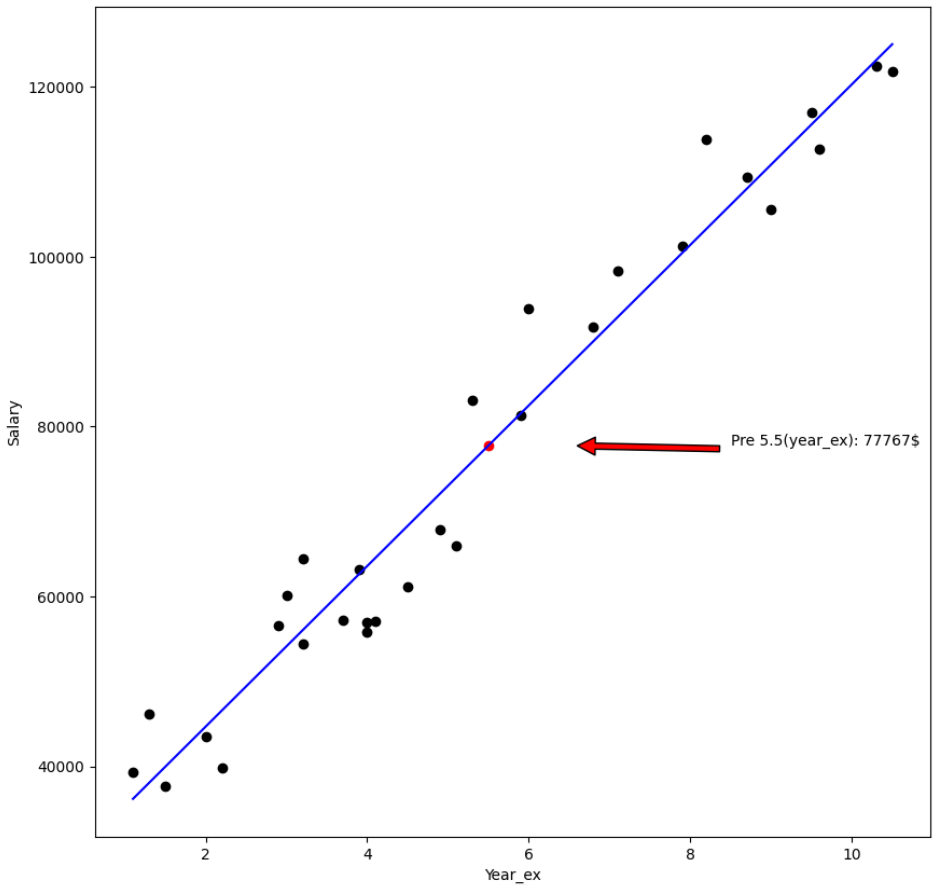
Dự đoán mức lương cho năm kinh nghiệm việc làm:

```
data = np.array(df)
years_experience = data[:,0]
salary = data[:,1]
regr = LinearRegression()
regr.fit(years_experience.reshape(-1,1),salary)

year_prediction = 5.5
need_prediction = regr.predict(np.array([[year_prediction]]))

plt.figure(figsize=(10,10))
plt.xlabel('Year_ex')
plt.ylabel('Salary')
plt.scatter(years_experience,salary,color='black')
plt.plot(years_experience,regr.predict(years_experience.reshape(-1,1)),color='blue')
plt.scatter([[year_prediction]],[need_prediction],color="red")
plt.annotate(f'Pre {year_prediction}(year_ex):{need_prediction[0]: .0f}$',xy=[6.5,need_prediction],
             xytext=[8.5,need_prediction],arrowprops=dict(facecolor='red',shrink=0.05))
```

Kết quả:



Biểu đồ này được sử dụng để hiển thị mối quan hệ giữa kinh nghiệm làm việc và mức lương. Đường màu xanh dương cho thấy đường hồi quy tuyến tính giữa hai biến. Mũi tên đỏ cho thấy mức lương dự đoán (77767$) cho năm kinh nghiệm làm việc là 5,5 năm.

Bài 2: Logistic Regression

Phần 1: Giới thiệu

1.1 Tập dữ liệu

Tập dữ liệu nhóm chọn có tên **weatherAUS.csv**.

Nguồn: Rain in Australia (kaggle.com)

Link mã nguồn Python: Python_Logistic_regression

1.1.1 Ngữ cảnh

Dự đoán mưa ngày hôm sau bằng cách đào tạo các mô hình phân loại trên biến mục tiêu RainTomorrow.

1.1.2 Nội dung

Bộ dữ liệu này chứa khoảng 10 năm quan sát thời tiết hàng ngày từ nhiều địa điểm trên khắp nước Úc.

RainTomorrow là biến mục tiêu để dự đoán. Nó có nghĩa là - trời mưa vào ngày hôm sau, Có hay Không? Cột này là Có nếu mưa cho ngày hôm đó là 1mm trở lên.

1.2 Thành phần dữ liệu

Bộ dữ liệu này chứa khoảng 10 năm quan sát thời tiết hàng ngày từ nhiều trạm thời tiết Úc.

RainTomorrow là biến mục tiêu để dự đoán. Nó có nghĩa là - trời mưa vào ngày hôm sau, Có hay Không?

Cột này là Có nếu mưa cho ngày hôm đó là 1mm trở lên.

Tên cột	Giải thích
Date	Ngày quan sát
Location	Tên chung của vị trí trạm thời tiết
MinTemp	Nhiệt độ tối thiểu (°C)
MaxTemp	Nhiệt độ tối đa (°C)
Rainfall	Lượng mưa ghi nhận trong ngày (mm)
Evaporation	Sự bốc hơi của chảo loại A (mm) trong 24 giờ đến 9 giờ sáng
Sunshine	Số giờ nắng chói chang trong ngày (giờ)
WindGustDir	Hướng gió giật mạnh nhất trong 24 giờ đến nửa đêm
WindGustSpeed	Tốc độ (km/h) gió giật mạnh nhất trong 24 giờ đến nửa đêm
WindDir9am	Hướng gió lúc 9 giờ sáng
WindDir3pm	Hướng gió lúc 3 giờ chiều

WindSpeed9am	Tốc độ gió (km/giờ) trung bình trên 10 phút trước 9 giờ sáng
WindSpeed3pm	Tốc độ gió (km/giờ) trung bình trong 10 phút trước 3 giờ chiều
Humidity9am	Độ ẩm (%) lúc 9 giờ sáng
Humidity3pm	Độ ẩm (%) lúc 3 giờ chiều
Pressure9am	Áp suất khí quyển (hpa) giảm xuống mực nước biển trung bình lúc 9 giờ sáng
Pressure3pm	Áp suất khí quyển (hpa) giảm xuống mực nước biển trung bình lúc 3 giờ chiều
Cloud9am	Một phần bầu trời bị mây che khuất lúc 9 giờ sáng. Điều này được đo bằng "oktas", là đơn vị của phần tám. Nó ghi lại bao nhiêu
Cloud3pm	Phần bầu trời bị mây che khuất (trong "oktas": phần tám) lúc 3 giờ chiều. Xem Cload9am để biết mô tả về các giá trị
Temp9am	Nhiệt độ ((°C) lúc 9 giờ sáng
Temp3pm	Nhiệt độ ((°C) lúc 3 giờ chiều
RainToday	Boolean: 1 nếu lượng mưa (mm) trong 24 giờ đến 9 giờ sáng vượt quá 1mm, nếu không thì 0
RainTomorrow	Lượng mưa ngày hôm sau tính bằng mm. Được sử dụng để tạo biến phản hồi RainTomorrow. Một loại thước đo "rủi ro".

1.3 Ảnh về tập dữ liệu

Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday	RainTomorrow
12/1/2008	Albury	13.4	22.9	0.6	NA	NA	W	44	W	WNW	20	24	71	22	1007.7	1007.1	8	NA	16.9	21.8	No	No
12/2/2008	Albury	7.4	25.1	0	NA	NA	WNW	44	NNW	WSW	4	22	44	25	1010.6	1007.8	NA	NA	17.2	24.3	No	No
12/3/2008	Albury	12.9	25.7	0	NA	NA	WSW	46	W	WSW	19	26	38	30	1007.6	1008.7	NA	2	21	23.2	No	No
12/4/2008	Albury	9.2	28	0	NA	NA	NE	24	SE	E	11	9	45	16	1017.6	1012.8	NA	NA	18.1	26.5	No	No
12/5/2008	Albury	17.5	32.3	1	NA	NA	W	41	ENE	NW	7	20	82	33	1010.8	1006	7	8	17.8	29.7	No	No
12/6/2008	Albury	14.6	29.7	0.2	NA	NA	WNW	56	W	W	19	24	55	23	1009.2	1005.4	NA	NA	20.6	28.9	No	No
12/7/2008	Albury	14.3	25	0	NA	NA	W	50	SW	W	20	24	49	19	1009.6	1008.2	1	NA	18.1	24.6	No	No
12/8/2008	Albury	7.7	26.7	0	NA	NA	W	35	SSE	W	6	17	48	19	1013.4	1010.1	NA	NA	16.3	25.5	No	No
12/9/2008	Albury	9.7	31.9	0	NA	NA	NNW	80	SE	NW	7	28	42	9	1008.9	1003.6	NA	NA	18.3	30.2	No	Yes
12/10/2008	Albury	13.1	30.1	1.4	NA	NA	W	28	S	SSE	15	11	58	27	1007	1005.7	NA	NA	20.1	28.2	Yes	No
12/11/2008	Albury	13.4	30.4	0	NA	NA	N	30	SSE	ESE	17	6	48	22	1011.8	1008.7	NA	NA	20.4	28.8	No	Yes
12/12/2008	Albury	15.9	21.7	2.2	NA	NA	NNE	31	NE	ENE	15	13	89	91	1010.5	1004.2	8	8	15.9	17	Yes	Yes
12/13/2008	Albury	15.9	18.6	15.6	NA	NA	W	61	NNW	NNW	28	28	76	93	994.3	993	8	8	17.4	15.8	Yes	Yes
12/14/2008	Albury	12.6	21	3.6	NA	NA	SW	44	W	SSW	24	20	65	43	1001.2	1001.8	NA	7	15.8	19.8	Yes	No
12/15/2008	Albury	8.4	24.6	0	NA	NA	NA	NA	S	WNW	4	30	57	32	1009.7	1008.7	NA	NA	15.9	23.5	No	NA
12/16/2008	Albury	9.8	27.7	NA	NA	NA	WNW	50	NA	WNW	NA	22	50	28	1013.4	1010.3	0	NA	17.3	26.2	NA	No
12/17/2008	Albury	14.1	20.9	0	NA	NA	ENE	22	SSW	E	11	9	69	82	1012.2	1010.4	8	1	17.2	18.1	No	Yes
12/18/2008	Albury	13.5	22.9	16.8	NA	NA	W	63	N	WNW	6	20	80	65	1005.8	1002.2	8	1	18	21.5	Yes	Yes
12/19/2008	Albury	11.2	22.5	10.6	NA	NA	SSE	43	WSW	SW	24	17	47	32	1009.4	1009.7	NA	2	15.5	21	Yes	No
12/20/2008	Albury	9.8	25.6	0	NA	NA	SSE	26	SE	NNW	17	6	45	26	1019.2	1017.1	NA	NA	15.8	23.2	No	No
12/21/2008	Albury	11.5	29.3	0	NA	NA	S	24	SE	SE	9	9	56	28	1019.3	1014.8	NA	NA	19.1	27.3	No	No
12/22/2008	Albury	17.1	33	0	NA	NA	NE	43	NE	N	17	22	38	28	1013.6	1008.1	NA	1	24.5	31.6	No	No
12/23/2008	Albury	20.5	31.8	0	NA	NA	WNW	41	W	W	19	20	54	24	1007.8	1005.7	NA	NA	23.8	30.8	No	No
12/24/2008	Albury	15.3	30.9	0	NA	NA	N	33	ESE	NW	6	13	55	23	1011	1008.2	5	NA	20.9	29	No	No
12/25/2008	Albury	12.6	32.4	0	NA	NA	W	43	E	W	4	19	49	17	1012.9	1010.1	NA	NA	21.5	31.2	No	No
12/26/2008	Albury	16.2	33.9	0	NA	NA	WSW	35	SE	WSW	9	13	45	19	1010.9	1007.6	NA	1	23.2	33	No	No
12/27/2008	Albury	16.9	33	0	NA	NA	WSW	57	NA	W	0	26	41	28	1006.8	1003.6	NA	1	26.6	31.2	No	No
12/28/2008	Albury	20.1	32.7	0	NA	NA	WNW	48	N	WNW	13	30	56	15	1005.2	1001.7	NA	NA	24.6	32.1	No	No
12/29/2008	Albury	19.7	27.2	0	NA	NA	WNW	46	NW	WSW	19	30	49	22	1004.8	1004.2	NA	NA	21.6	26.1	No	Yes
12/30/2008	Albury	12.5	24.2	1.2	NA	NA	WNW	50	WSW	SW	11	22	78	70	1005.6	1003.4	8	8	12.5	18.2	Yes	No
12/31/2008	Albury	12	24.4	0.8	NA	NA	W	39	WNW	WNW	17	17	48	28	1006.1	1005.1	1	NA	16.9	22.7	No	No
1/1/2009	Albury	11.3	26.5	0	NA	NA	WNW	56	W	WNW	19	31	46	26	1004.5	1003.2	NA	NA	19.7	25.7	No	No
1/2/2009	Albury	9.6	23.9	0	NA	NA	W	41	WSW	SSW	19	11	44	22	1014.4	1013.1	NA	NA	14.9	22.1	No	No
1/3/2009	Albury	10.5	28.8	0	NA	NA	SSE	26	SSE	E	11	7	43	22	1018.7	1014.8	NA	NA	17.1	26.5	No	No
1/4/2009	Albury	12.3	34.6	0	NA	NA	WNW	37	SSE	NW	6	17	41	12	1015.1	1010.3	NA	NA	20.7	33.9	No	No
1/5/2009	Albury	12.9	35.8	0	NA	NA	WNW	41	ENE	NW	6	26	41	9	1012.6	1009.2	NA	NA	22.4	34.4	No	No

PHẦN 2: Chuẩn bị dữ liệu

Nhập thư viện để chuẩn bị cho các bước xử lý dữ liệu sau này.

```
import warnings
import numpy as np
import pandas as pd
from scipy import stats
from sklearn import preprocessing
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

Dùng hàm warnings để tắt tất cả các cảnh báo trong quá trình chạy mã.

```
# Tắt tất cả các cảnh báo
warnings.filterwarnings("ignore")
```

Đọc file CSV và lưu DataFrame vào biến **df**. Sau đó in ra màn hình kích thước của DataFrame.

```
# Đọc tập dữ liệu từ file csv
df = pd.read_csv("./weatherAus.csv")
print("Kích thước dataframe dữ liệu thời tiết:"
, df.shape)
# Hiển thị tập dữ liệu
print(df.head())
```

Kết quả:

```
Kích thước dataframe dữ liệu thời tiết: (145460, 23)
         Date Location  MinTemp  MaxTemp  Rainfall  Evaporation  Sunshine  \
0  2008-12-01   Albury     13.4     22.9       0.6          NaN       NaN
1  2008-12-02   Albury      7.4     25.1       0.0          NaN       NaN
2  2008-12-03   Albury     12.9     25.7       0.0          NaN       NaN
3  2008-12-04   Albury      9.2     28.0       0.0          NaN       NaN
4  2008-12-05   Albury     17.5     32.3       1.0          NaN       NaN

  WindGustDir  WindGustSpeed WindDir9am  ... Humidity9am  Humidity3pm  \
0           W           44.0          W  ...        71.0         22.0
1         WNW           44.0        NNW  ...        44.0         25.0
2         WSW           46.0          W  ...        38.0         30.0
3          NE           24.0         SE  ...        45.0         16.0
4           W           41.0        ENE  ...        82.0         33.0

   Pressure9am  Pressure3pm  Cloud9am  Cloud3pm  Temp9am  Temp3pm  RainToday  \
0       1007.7       1007.1       8.0       NaN     16.9     21.8         No
1       1010.6       1007.8       NaN       NaN     17.2     24.3         No
2       1007.6       1008.7       NaN       2.0     21.0     23.2         No
3       1017.6       1012.8       NaN       NaN     18.1     26.5         No
4       1010.8       1006.0       7.0       8.0     17.8     29.7         No

   RainTomorrow
0            No
1            No
...
3            No
4            No

[5 rows x 23 columns]
```

In ra màn hình tên các cột.

```
# Tên các cột
col_names = df.columns
print(col_names)
```

Kết quả:

```
Index(['Date', 'Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation',
       'Sunshine', 'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
       'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
       'Temp3pm', 'RainToday', 'RainTomorrow'],
      dtype='object')
```

In ra các thống kê cơ bản như mean, std, min, max của mỗi cột dạng số trong DataFrame.

```
# Thống kê cơ bản của mỗi cột
print(df.describe())
```

Kết quả:

	MinTemp	MaxTemp	Rainfall	Evaporation	\
count	143975.000000	144199.000000	142199.000000	82670.000000	
mean	12.194034	23.221348	2.360918	5.468232	
std	6.398495	7.119049	8.478060	4.193704	
min	-8.500000	-4.800000	0.000000	0.000000	
25%	7.600000	17.900000	0.000000	2.600000	
50%	12.000000	22.600000	0.000000	4.800000	
75%	16.900000	28.200000	0.800000	7.400000	
max	33.900000	48.100000	371.000000	145.000000	

	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	\
count	75625.000000	135197.000000	143693.000000	142398.000000	
mean	7.611178	40.035230	14.043426	18.662657	
std	3.785483	13.607062	8.915375	8.809800	
min	0.000000	6.000000	0.000000	0.000000	
25%	4.800000	31.000000	7.000000	13.000000	
50%	8.400000	39.000000	13.000000	19.000000	
75%	10.600000	48.000000	19.000000	24.000000	
max	14.500000	135.000000	130.000000	87.000000	

	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	\
count	142806.000000	140953.000000	130395.00000	130432.000000	
mean	68.880831	51.539116	1017.64994	1015.255889	
std	19.029164	20.795902	7.10653	7.037414	
min	0.000000	0.000000	980.50000	977.100000	
...					
25%	1.000000	2.000000	12.300000	16.60000	
50%	5.000000	5.000000	16.700000	21.10000	
75%	7.000000	7.000000	21.600000	26.40000	
max	9.000000	9.000000	40.200000	46.70000	

PHẦN 3: Xử lí dữ liệu

2.1 Loại bỏ dữ liệu

In ra số lượng giá trị không rỗng (không phải là null) của từng cột, sắp xếp theo giá trị tăng dần.

```
# Kiểm tra giá trị Null
print(df.count().sort_values())
```

Kết quả:

```
Sunshine          75625
Evaporation       82670
Cloud3pm          86102
Cloud9am          89572
Pressure9am      130395
Pressure3pm      130432
WindDir9am       134894
WindGustDir      135134
WindGustSpeed    135197
Humidity3pm      140953
WindDir3pm       141232
Temp3pm          141851
RainTomorrow     142193
RainToday        142199
Rainfall         142199
WindSpeed3pm     142398
Humidity9am      142806
Temp9am          143693
WindSpeed9am     143693
MinTemp          143975
MaxTemp          144199
Location         145460
Date             145460
dtype: int64
```

Loại bỏ các cột không cần thiết từ DataFrame. Các cột bị loại bỏ là 'Sunshine', 'Evaporation', 'Cloud3pm', 'Cloud9am', và 'Date' và in ra kích thước mới của DataFrame.

```
# Loại bỏ các cột không cần thiết
df = df.drop(columns=['Sunshine','Evaporation','Cloud3pm','Cloud9am',
'Date'],axis=1)
print(df.shape)
```

Loại bỏ các hàng có ít nhất một giá trị null từ DataFrame và in ra kích thước mới của DataFrame sau khi loại bỏ giá trị null.

```
# Loại bỏ các hàng chứa giá trị Null
df = df.dropna()
print(df.shape)
```

2.2 Loại bỏ các giá trị ngoại lệ

Sử dụng Z-score để xác định giá trị ngoại lệ và loại bỏ chúng khỏi DataFrame. In ra kích thước mới của DataFrame sau khi loại bỏ giá trị ngoại lệ.

```
# Loại bỏ các giá trị ngoại lệ
z = np.abs(stats.zscore(df._get_numeric_data()))
print(z)
df = df[(z < 3).all(axis=1)]
print(df.shape)
```

Kết quả:

```
         MinTemp   MaxTemp  Rainfall  WindGustSpeed  WindSpeed9am  \
0       0.133606  0.108909  0.226104       0.353152      0.692529
1       0.826105  0.209120  0.379755       0.353152      1.395172
2       0.053630  0.295856  0.379755       0.520274      0.562048
3       0.538192  0.628341  0.379755       1.318072      0.481803
4       0.789409  1.249943  0.123670       0.102468      1.003728
...          ...       ...       ...            ...           ...
145454  1.449917  0.267923  0.379755       0.733144      0.040123
145455  1.561884  0.036629  0.379755       0.733144      0.220840
145456  1.433922  0.238032  0.379755       1.485194      0.220840
145457  1.146009  0.469326  0.379755       0.231776      0.742765
145458  0.762124  0.483782  0.379755       0.983827      0.220840

        WindSpeed3pm  Humidity9am  Humidity3pm  Pressure9am  Pressure3pm  \
0           0.615619     0.195478     1.389100     1.493284     1.232228
1           0.366169     1.263051     1.242074     1.062586     1.127614
2           0.865070     1.587168     0.997030     1.508136     0.993110
3           1.255260     1.209031     1.683153     0.022970     0.380371
4           0.116718     0.789694     0.850004     1.032883     1.396622
...              ...          ...          ...          ...          ...
145454      0.756359     0.452757     1.144057     1.031498     0.874999
145455      1.005810     0.884914     1.291083     1.016646     0.740495
145456      1.255260     0.614816     1.438109     0.853278     0.561157
145457      1.255260     0.776874     1.291083     0.481987     0.217424
145458      1.504711     0.884914     1.291083     0.244360     0.172590
...
145458  0.363629  0.569886

[107868 rows x 12 columns]
(103113, 18)
```

2.3 Chuẩn hóa dữ liệu

Chuẩn hóa dữ liệu số bằng cách sử dụng hàm scale từ thư viện preprocessing của scikit-learn.

```
# Chuẩn hóa dữ liệu
numerical = [var for var in df.columns if df[var].dtype=="float64"]
for col in numerical:
    df[col] = preprocessing.scale(df[col])

print(df.head())
```

Kết quả:

```
  Location   MinTemp   MaxTemp  Rainfall WindGustDir  WindGustSpeed  \
0   Albury  0.145603 -0.120266 -0.136427           W       0.412970
1   Albury -0.812264  0.198699 -0.406190         WNW       0.412970
2   Albury  0.065781  0.285690 -0.406190         WSW       0.587351
3   Albury -0.524904  0.619154 -0.406190          NE      -1.330844
4   Albury  0.800146  1.242587  0.043415           W       0.151398

  WindDir9am WindDir3pm  WindSpeed9am  WindSpeed3pm  Humidity9am  Humidity3pm  \
0          W        WNW      0.731022      0.651862     0.220448    -1.375995
1        NNW        WSW     -1.400118      0.397380    -1.251106    -1.227261
2          W        WSW      0.597826      0.906343    -1.578118    -0.979372
3         SE          E     -0.467744     -1.256749    -1.196604    -1.673462
4        ENE         NW     -1.000529      0.142899     0.819970    -0.830638

   Pressure9am  Pressure3pm   Temp9am   Temp3pm RainToday RainTomorrow
0    -1.552929    -1.275167 -0.078542 -0.063572        No           No
1    -1.114503    -1.169077 -0.031111  0.307266        No           No
2    -1.568048    -1.032677  0.569679  0.144098        No           No
3    -0.056231    -0.411298  0.111181  0.633604        No           No
4    -1.084266    -1.441878  0.063750  1.108277        No           No
```

Sử dụng LabelEncoder để chuyển các giá trị "Yes" và "No" trong cột "RainToday" và "RainTomorrow" thành 1 và 0 tương ứng.

```
# Chuyển Yes/No -> 1/0
df["RainToday"] = LabelEncoder().fit_transform(df["RainToday"])
df["RainTomorrow"] = LabelEncoder().fit_transform(df["RainTomorrow"])

print(df.head())
```

Kết quả:

```
  Location   MinTemp   MaxTemp  Rainfall WindGustDir  WindGustSpeed  \
0   Albury  0.145603 -0.120266 -0.136427           W       0.412970
1   Albury -0.812264  0.198699 -0.406190         WNW       0.412970
2   Albury  0.065781  0.285690 -0.406190         WSW       0.587351
3   Albury -0.524904  0.619154 -0.406190          NE      -1.330844
4   Albury  0.800146  1.242587  0.043415           W       0.151398

  WindDir9am WindDir3pm  WindSpeed9am  WindSpeed3pm  Humidity9am  Humidity3pm  \
0          W        WNW      0.731022      0.651862     0.220448    -1.375995
1        NNW        WSW     -1.400118      0.397380    -1.251106    -1.227261
2          W        WSW      0.597826      0.906343    -1.578118    -0.979372
3         SE          E     -0.467744     -1.256749    -1.196604    -1.673462
4        ENE         NW     -1.000529      0.142899     0.819970    -0.830638

   Pressure9am  Pressure3pm   Temp9am   Temp3pm  RainToday  RainTomorrow
0    -1.552929    -1.275167 -0.078542 -0.063572          0             0
1    -1.114503    -1.169077 -0.031111  0.307266          0             0
2    -1.568048    -1.032677  0.569679  0.144098          0             0
3    -0.056231    -0.411298  0.111181  0.633604          0             0
4    -1.084266    -1.441878  0.063750  1.108277          0             0
```

2.4 Xử lý dữ liệu phân loại

In ra số lượng và tên của các biến phân loại trong DataFrame.

```
# Xử lý dữ liệu phân loại
categorical = [var for var in df.columns if df[var].dtype=='object']
print("Number of categorical variables: ", len(categorical))
print(categorical)
```

Kết quả:

```
Number of categorical variables:  4
['Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm']
```

In ra các giá trị duy nhất trong từng cột phân loại và tạo các biến giả tưởng cho các biến phân loại sử dụng phương pháp "one-hot encoding" bằng hàm get_dummies của pandas.

```
categorical_columns = ['Location','WindGustDir', 'WindDir3pm',
'WindDir9am']

# Hiển thị các giá trị duy nhất trong từng cột phân loại
for col in categorical_columns:
    print(np.unique(df[col]))

# Tạo các biến giả tưởng (dummy variables)
df = pd.get_dummies(df, columns=categorical_columns)
```

Kết quả:

```
['Adelaide' 'Albury' 'AliceSprings' 'BadgerysCreek' 'Ballarat' 'Bendigo'
 'Brisbane' 'Cairns' 'Canberra' 'Cobar' 'CoffsHarbour' 'Dartmoor' 'Darwin'
 'GoldCoast' 'Hobart' 'Katherine' 'Launceston' 'Melbourne'
 'MelbourneAirport' 'Mildura' 'Moree' 'MountGambier' 'Nhil' 'NorahHead'
 'NorfolkIsland' 'Nuriootpa' 'PearceRAAF' 'Perth' 'PerthAirport'
 'Portland' 'Richmond' 'Sale' 'Sydney' 'SydneyAirport' 'Townsville'
 'Tuggeranong' 'Uluru' 'WaggaWagga' 'Walpole' 'Watsonia' 'Williamtown'
 'Witchcliffe' 'Wollongong' 'Woomera']
['E' 'ENE' 'ESE' 'N' 'NE' 'NNE' 'NNW' 'NW' 'S' 'SE' 'SSE' 'SSW' 'SW' 'W'
 'WNW' 'WSW']
['E' 'ENE' 'ESE' 'N' 'NE' 'NNE' 'NNW' 'NW' 'S' 'SE' 'SSE' 'SSW' 'SW' 'W'
 'WNW' 'WSW']
['E' 'ENE' 'ESE' 'N' 'NE' 'NNE' 'NNW' 'NW' 'S' 'SE' 'SSE' 'SSW' 'SW' 'W'
 'WNW' 'WSW']
```

PHẦN 4: Thực hiện bài toán Logistic regression

3.1 Chuẩn bị dữ liệu

3.1.1 Tính năng mô hình

Đặt X là tất cả các đặc trưng và y là biến mục tiêu:

- X là DataFrame chứa tất cả các đặc trưng (các cột trừ cột "RainTomorrow");
- y là Series chứa biến mục tiêu "RainTomorrow".

```
# Đặt X là tất cả các đặc trưng
X = df.loc[:, df.columns != 'RainTomorrow']
# Đặt y là biến mục tiêu RainTomorrow
y = df.RainTomorrow
```

3.1.2 Chia dữ liệu thành tập huấn luyện và tập kiểm thử

Sử dụng hàm train_test_split từ scikit-learn để chia dữ liệu thành tập huấn luyện và tập kiểm thử.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=
0.25, random_state=42)
```

3.2 Khởi tạo và huấn luyện mô hình Logistic Regression

Sử dụng mô hình Logistic Regression và huấn luyện nó trên tập huấn luyện (X_train, y_train) và y_pred.

```
# Tách dữ liệu thành 5 phần và trộn để tránh thiên lệch
logReg = LogisticRegression()
logReg.fit(X_train, y_train)

# Dự đoán trên tập kiểm thử với mô hình đã được huấn luyện
y_pred = logReg.predict(X_test)
```

3.3 Đánh giá mô hình

Dùng mô hình đã huấn luyện để dự đoán trên tập kiểm thử và tính độ chính xác của mô hình.

```
# Đánh giá mô hình
accuracy_score = accuracy_score(y_test, y_pred)
print("Độ chính xác của tập dữ liệu: {:.2f}".format(accuracy_score))
```

Kết quả:

```
Độ chính xác của tập dữ liệu: 0.86
```

Chọn ngẫu nhiên 100 điểm từ tập kiểm thử để biểu diễn trên biểu đồ.

```
# Chọn ngẫu nhiên 100 điểm từ tập kiểm thử
random_indices = np.random.choice(len(X_test), size=100, replace=
False)
```

Vẽ biểu đồ scatter plot so sánh giữa kết quả thực tế và kết quả dự đoán.

```
# Biểu đồ điểm so sánh kết quả thực tế và dự đoán
plt.figure(figsize=(16, 5))
plt.scatter(range(100), y_test.iloc[random_indices], color="black"
, label="Thực tế")
plt.scatter(range(100), y_pred[random_indices], color="red", marker=
"x", label="Dự đoán")

plt.title(
"Biểu đồ so sánh Kết quả Thực tế và Dự đoán 'Ngày mai trời có mưa ha
y không?'"
, fontsize=15, fontweight="bold")
plt.xlabel("Số mẫu (mẫu)", fontsize=12)
plt.ylabel("Có mưa (1) / Không có mưa (0)", fontsize=12)
plt.legend(title="Kết quả", title_fontsize=12, fontsize=12)
plt.yticks([0, 1])
plt.show()
```

Kết quả: