ThucHanh1

January 16, 2025

1 LAB: Dữ liệu với NumPy, Pandas và Matplotlib

1.0.1 Mục tiêu

- Làm quen với NumPy: Tạo mảng, thao tác cơ bản và nâng cao với mảng.
- Thực hành xử lý dữ liệu với Pandas: Đọc, phân tích và làm sạch dữ liệu.
- Trực quan hóa dữ liệu bằng Matplotlib: Biểu đồ cơ bản và nâng cao.

1.1 Phần 1: NumPy cơ bản

1.1.1 Bài tập 1: Tạo mảng và thao tác cơ bản

1 Tạo một mảng NumPy với các giá trị từ 1 đến 20.
2 Tìm tổng, giá trị lớn nhất, nhỏ nhất và trung bình của mảng.
3 Tạo một mảng 2D (3x5) chứa các số ngẫu nhiên từ 0 đến 100.
4 Lấy hàng thứ 2 và cột thứ 3 của mảng 2D.

[18]:
```
import numpy as np

# 1. Tạo một mảng NumPy với các giá trị từ 1 đến 20
array_1 = np.array([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20])
print("Mảng từ 1 đến 20:", array_1)

# 2. Tìm tổng, giá trị lớn nhất, nhỏ nhất và trung bình của mảng
print("Tổng:", array_1.sum())
print("Giá trị lớn nhất:", array_1.max() )
print("Giá trị nhỏ nhất:", array_1.min() )
print("Trung bình:", array_1.mean() )

# 3. Tạo một mảng 2D (3x5) chứa các số ngẫu nhiên từ 0 đến 100
array_2d = np.random.randint(0, 101, (3, 5))
print("Mảng 2D:", array_2d)

# 4. Lấy hàng thứ 2 và cột thứ 3 của mảng 2D
print("Hàng thứ 2:",array_2d[1] )
print("Cột thứ 3:", array_2d[:, 2])
```

```
Mảng từ 1 đến 20: [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
Tổng: 210
Giá trị lớn nhất: 20
```

```
Giá trị nhỏ nhất: 1
Trung bình: 10.5
Mảng 2D: [[78 61 46 95 17]
 [94 98 75 43 38]
 [32 69  1 20 16]]
Hàng thứ 2: [94 98 75 43 38]
Cột thứ 3: [46 75  1]
```

1.1.2 Bài tập 2: Các thao tác nâng cao

1 Tạo một mảng NumPy chứa 20 giá trị ngẫu nhiên từ 0 đến 1.
2 Chuẩn hóa mảng này (đưa các giá trị về khoảng [0, 1]).
3 Tính tích vô hướng (dot product) của hai mảng 1D: [1, 2, 3] và [4, 5,6].
4 Tạo một ma trận 5x5 và tính định thức (determinant) và nghịch đảo của ma trận.

[19]:
```python
# 1. Tạo một mảng NumPy chứa 20 giá trị ngẫu nhiên từ 0 đến 1
random_array = np.random.rand(20)
print("Mảng ngẫu nhiên từ 0 đến 1:", random_array)

# 2. Chuẩn hóa mảng này (đưa các giá trị về khoảng [0, 1])
normalized_array = (random_array - np.min(random_array)) / (np.
 ↪max(random_array) - np.min(random_array))
print("Mảng sau khi chuẩn hóa:", normalized_array)

# 3. Tính tích vô hướng (dot product) của hai mảng 1D
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
dot_product = np.dot(a, b)
print("Tích vô hướng của a và b:", dot_product)

# 4. Tạo một ma trận 5x5 và tính định thức, nghịch đảo
matrix = np.random.randint(1, 10, (5, 5))
print("Ma trận:", matrix)
determinant = np.linalg.det(matrix)
print("Định thức của ma trận:", determinant)
if determinant != 0:
    inverse_matrix = np.linalg.inv(matrix)
    print("Ma trận nghịch đảo:", inverse_matrix)
else:
    print("Ma trận không khả nghịch (định thức = 0).")
```

```
Mảng ngẫu nhiên từ 0 đến 1: [0.09282865 0.28300215 0.94062949 0.99455055
0.86685588 0.97084791
 0.05169948 0.60227588 0.95102834 0.2505001  0.16798217 0.74009096
 0.97364741 0.89579714 0.917467   0.57793476 0.85814184 0.29822784
 0.64723155 0.72064011]
Mảng sau khi chuẩn hóa: [0.04362212 0.2453226  0.94281063 1.         0.86456539
0.97486068
```

```
 0.         0.58394842 0.95383979 0.2108505  0.12333091 0.73011688
 0.97782986 0.89526086 0.9182442  0.55813192 0.85532316 0.26147115
 0.63162899 0.70948705]
Tích vô hướng của a và b: 32
Ma trận: [[9 4 6 1 2]
 [2 9 7 8 7]
 [9 5 1 4 8]
 [5 4 8 8 5]
 [3 4 8 3 6]]
Định thức của ma trận: -16444.000000000007
Ma trận nghịch đảo: [[ 0.07577232 -0.04950134  0.04366334  0.04050109
-0.05947458]
 [ 0.10313792  0.18864023 -0.0312576  -0.15434201 -0.08416444]
 [ 0.0356361  -0.03210898 -0.07978594  0.02627098  0.11007054]
 [-0.05558258  0.00741912  0.00328387  0.17574799 -0.14096327]
 [-0.12636828 -0.06190708  0.10374605 -0.04025784  0.17623449]]
```

1.2 Phần 2: Pandas cơ bản

1.2.1 Bài tập 3: Làm quen với DataFrame

1 Tạo một DataFrame chứa thông tin sau: | Name | Age | Score | |———|—|——-| | Alice | 23 | 85 | | Bob | 25 | 90 | | Charlie | 22 | 78 | | David | 24 | 92 | | Eva | 21 | 88 |

2 Tính giá trị trung bình của cột “Score”.
3 Lọc các hàng có “Score” lớn hơn 85.

[20]:
```python
import pandas as pd

# 1. Tạo DataFrame
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
    'Age': ['23', '25', '22', '24', '21'],
    'Score': ['85', '90', '78', '92', '88']
}
df = pd.DataFrame(data)
print("DataFrame:", df)

# 2. Tính giá trị trung bình của cột "Score"
df['Score'] = df['Score'].astype(float)  # Đảm bảo "Score" là số
average_score = df['Score'].mean()
print("\nGiá trị trung bình của cột Score:", average_score)

# 3. Lọc các hàng có "Score" lớn hơn 85
filtered_df = df[df['Score'] > 85]
print("Các hàng có Score > 85:", filtered_df)
```

```
DataFrame:       Name Age Score
0    Alice  23    85
```

```
1      Bob  25    90
2  Charlie  22    78
3    David  24    92
4      Eva  21    88

Giá trị trung bình của cột Score: 86.6
Các hàng có Score > 85:     Name Age  Score
1    Bob  25   90.0
3  David  24   92.0
4    Eva  21   88.0
```

1.2.2 Bài tập 4: Đọc và phân tích dữ liệu từ file

1 Tải file Iris.csv từ Kaggle Iris Dataset.
2 Đọc dữ liệu từ file CSV vào DataFrame.
3 Hiển thị thông tin cơ bản (tổng quan, kiểu dữ liệu, số lượng null).
4 Tính trung bình, lớn nhất, nhỏ nhất của cột sepal_length.

[21]:
```python
# Đọc file CSV
iris_df = pd.read_csv("Iris.csv", sep=",")
print("Thông tin tổng quan về dữ liệu:", iris_df.info())
print("Mô tả dữ liệu:", iris_df.describe())
print(iris_df.columns)

# Tính toán cơ bản
print("Trung bình sepal_length:", iris_df['SepalLengthCm'].mean())
print("Giá trị lớn nhất sepal_length:", iris_df['SepalLengthCm'].max())
print("Giá trị nhỏ nhất sepal_length:", iris_df['SepalLengthCm'].min())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
Thông tin tổng quan về dữ liệu: None
Mô tả dữ liệu:                Id  SepalLengthCm  SepalWidthCm  PetalLengthCm
PetalWidthCm
count  150.000000     150.000000    150.000000     150.000000    150.000000
mean    75.500000       5.843333      3.054000       3.758667      1.198667
std     43.445368       0.828066      0.433594       1.764420      0.763161
```

```
min      1.000000       4.300000      2.000000       1.000000      0.100000
25%     38.250000       5.100000      2.800000       1.600000      0.300000
50%     75.500000       5.800000      3.000000       4.350000      1.300000
75%    112.750000       6.400000      3.300000       5.100000      1.800000
max    150.000000       7.900000      4.400000       6.900000      2.500000
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
Trung bình sepal_length: 5.843333333333334
Giá trị lớn nhất sepal_length: 7.9
Giá trị nhỏ nhất sepal_length: 4.3
```

1.3 Phần 3: Làm sạch dữ liệu

1.3.1 Bài tập 5: Xử lý dữ liệu thiếu

1 Tạo một DataFrame chứa các giá trị sau:

Name	Age	City	Salary
Alice	23	New York	60000
Bob	NaN	Boston	52000
Charlie	25	NaN	NaN
David	24	Chicago	58000
Eva	22	Boston	NaN

2 Điền giá trị thiếu trong cột Age bằng giá trị trung bình.
3 Xóa các hàng có nhiều hơn 1 giá trị thiếu.
4 Điền giá trị thiếu trong cột Salary bằng 50000.

[24]:
```python
# Tạo DataFrame chứa dữ liệu thiếu
data_with_missing = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
    'Age': [23, np.nan, 25, 24, 22],
    'City': ['New York', 'Boston', np.nan, 'Chicago', 'Boston'],
    'Salary': [60000, 52000, np.nan, 58000, np.nan]
}

df_missing = pd.DataFrame(data_with_missing)
print("Dữ liệu ban đầu:", df_missing)

# 1. Điền giá trị thiếu trong cột Age bằng giá trị trung bình
age_mean = df_missing['Age'].mean()  # Sửa df -> df_missing
df_missing['Age'].fillna(age_mean, inplace=True)  # Sửa df -> df_missing
print("\nDataFrame sau khi điền giá trị thiếu trong cột 'Age':")
print(df_missing)

# 2. Xóa các hàng có nhiều hơn 1 giá trị thiếu
```

```python
df_cleaned = df_missing.dropna(thresh=len(df_missing.columns)-1)  # Sửa df -> df_missing
print("\nDataFrame sau khi xóa các hàng có nhiều hơn 1 giá trị thiếu:")
print(df_cleaned)

# 3. Điền giá trị thiếu trong cột Salary bằng 50000
df_cleaned['Salary'].fillna(50000, inplace=True)
print("\nDataFrame sau khi điền giá trị thiếu trong cột 'Salary':")
print(df_cleaned)
```

```
Dữ liệu ban đầu:       Name   Age      City   Salary
0    Alice  23.0  New York  60000.0
1      Bob   NaN    Boston  52000.0
2  Charlie  25.0       NaN      NaN
3    David  24.0   Chicago  58000.0
4      Eva  22.0    Boston      NaN

DataFrame sau khi điền giá trị thiếu trong cột 'Age':
      Name   Age      City   Salary
0    Alice  23.0  New York  60000.0
1      Bob  23.5    Boston  52000.0
2  Charlie  25.0       NaN      NaN
3    David  24.0   Chicago  58000.0
4      Eva  22.0    Boston      NaN

DataFrame sau khi xóa các hàng có nhiều hơn 1 giá trị thiếu:
    Name   Age      City   Salary
0  Alice  23.0  New York  60000.0
1    Bob  23.5    Boston  52000.0
3  David  24.0   Chicago  58000.0
4    Eva  22.0    Boston      NaN

DataFrame sau khi điền giá trị thiếu trong cột 'Salary':
    Name   Age      City   Salary
0  Alice  23.0  New York  60000.0
1    Bob  23.5    Boston  52000.0
3  David  24.0   Chicago  58000.0
4    Eva  22.0    Boston  50000.0
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_23224\481741929.py:14:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
```

```
instead, to perform the operation inplace on the original object.


  df_missing['Age'].fillna(age_mean, inplace=True)  # Sửa df -> df_missing
C:\Users\Admin\AppData\Local\Temp\ipykernel_23224\481741929.py:24:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df_cleaned['Salary'].fillna(50000, inplace=True)
C:\Users\Admin\AppData\Local\Temp\ipykernel_23224\481741929.py:24:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_cleaned['Salary'].fillna(50000, inplace=True)
```

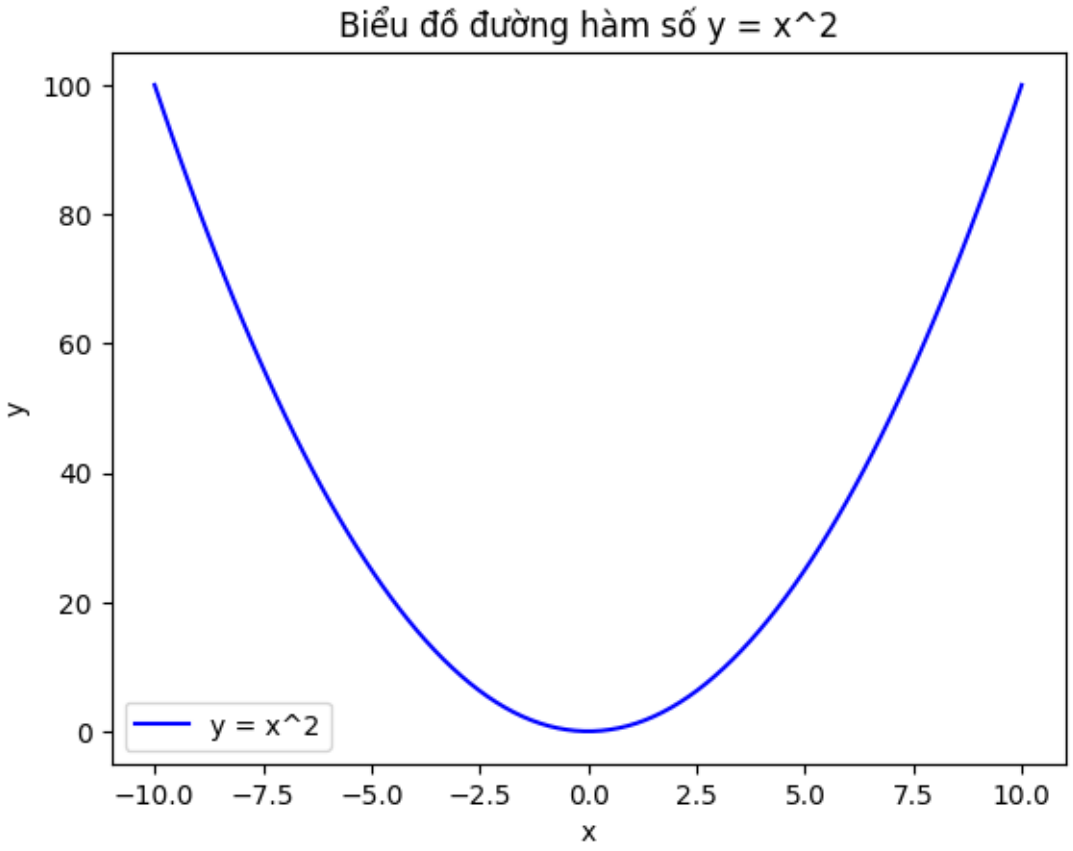
1.4 Phần 4: Trực quan hóa dữ liệu với Matplotlib

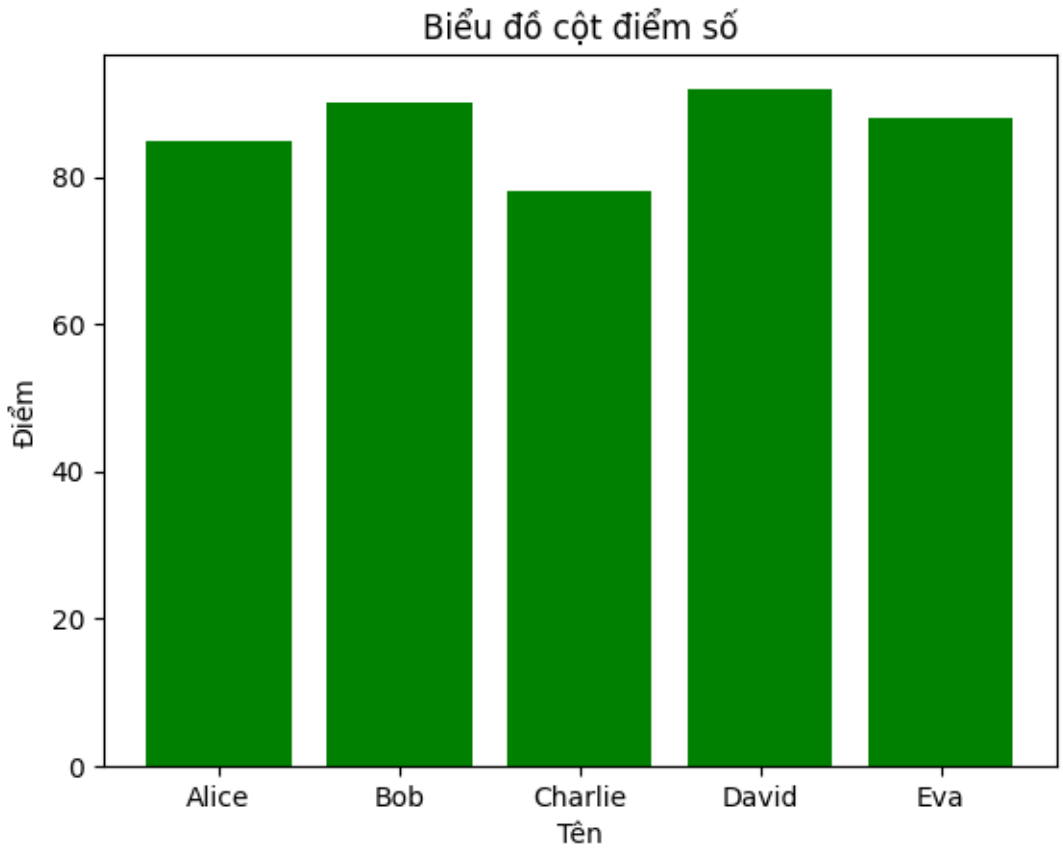
1.4.1 Bài tập 6: Biểu đồ cơ bản

1 Tạo một biểu đồ đường biểu diễn hàm số y = x ** 2 trên khoảng [-10,10]
2 Vẽ biểu đồ cột thể hiện điểm số (Score) của các sinh viên từ Bài tập 3.
3 Tạo một biểu đồ tròn (pie chart) thể hiện phần trăm mỗi loại hoa trong tập dữ liệu Iris.

[27]:
```python
import matplotlib.pyplot as plt
import numpy as np

# 1. Biểu đồ đường hàm số y = x^2
x = np.linspace(-10, 10, 100)
y = x ** 2

plt.plot(x, y, label='y = x^2', color='blue')
plt.title('Biểu đồ đường hàm số y = x^2')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()

# 2. Biểu đồ cột điểm số
```

```
names = ['Alice', 'Bob', 'Charlie', 'David', 'Eva']
scores = [85, 90, 78, 92, 88]

plt.bar(names, scores, color='green')
plt.title('Biểu đồ cột điểm số')
plt.xlabel('Tên')
plt.ylabel('Điểm')
plt.show()
```





1.4.2 Bài tập 7: Biểu đồ nâng cao

1 Vẽ biểu đồ phân tán (scatter plot) giữa sepal_length và sepal_width của tập dữ liệu Iris. Dùng màu sắc để phân biệt các loại hoa (species).
2 Thêm tiêu đề, nhãn trục và chú thích cho biểu đồ.

[31]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Đọc dữ liệu Iris
iris_df = pd.read_csv("Iris.csv")

# Kiểm tra lại tên cột để phù hợp với dữ liệu
iris_df.columns = ['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm',␣
 ↪'PetalWidthCm', 'Species']

# Vẽ biểu đồ phân tán (scatter plot)
plt.figure(figsize=(8, 6))
```

```
sns.scatterplot(data=iris_df, x='SepalLengthCm', y='SepalWidthCm',␣
 ↪hue='Species', palette='Set1')

# Thêm tiêu đề, nhãn trục và chú thích
plt.title('Biểu đồ phân tán giữa Sepal Length và Sepal Width', fontsize=14)
plt.xlabel('Sepal Length (cm)', fontsize=12)
plt.ylabel('Sepal Width (cm)', fontsize=12)
plt.legend(title='Loại hoa', loc='upper left')  # Chú thích

plt.show()
```