

Blog Tuần 1 – Module 3

Khám Phá Pandas, Trực Quan Hóa Dữ Liệu và ETL Pipeline

Từ nền tảng xử lý dữ liệu đến ứng dụng thực tiễn

Tác giả: GRID034

Tuần thứ nhất của Module 3 đưa ta vào một hành trình kiến thức đa dạng, kết hợp giữa lý thuyết nền tảng và ứng dụng thực tiễn. Nội dung Blog không chỉ giới thiệu các công cụ và kỹ thuật quan trọng trong phân tích dữ liệu, mà còn cung cấp các ví dụ minh họa trực quan và các tình huống thực tế giúp bạn áp dụng ngay vào công việc hoặc học tập. Từ việc làm chủ thư viện **Pandas**, lựa chọn biểu đồ trực quan phù hợp, xây dựng **ETL Pipeline** hiệu quả, cho tới kết hợp sức mạnh của **Python** và **Excel** – mọi phần đều được trình bày rõ ràng, dễ hiểu và bám sát thực tế.

Các chủ đề nổi bật bao gồm:

1. Pandas – Từ lý thuyết đến ứng dụng thực tế

- Giới thiệu Series và DataFrame, hai cấu trúc dữ liệu quan trọng.
- Thao tác cơ bản: đọc/ghi dữ liệu, xem thông tin, chọn lọc, thêm/xóa, sắp xếp, lọc, nhóm dữ liệu.
- Ứng dụng thực tế: phân tích dữ liệu Pokemon, xử lý dữ liệu thời gian, resampling và nhận diện seasonality.

2. Trực quan hóa dữ liệu với Python

- Giải thích khái niệm, vai trò, và nguyên tắc chọn biểu đồ phù hợp.
- Các thư viện phổ biến: Matplotlib, Seaborn, Plotly.
- Trường hợp nghiên cứu: Bộ dữ liệu ETTh, Student Performance, và Iris với nhiều loại biểu đồ (line chart, box plot, bar chart, donut chart, heatmap, scatter plot, bubble chart, pairplot, 3D visualization, word cloud).

3. ETL Pipeline – Trái tim của dữ liệu thông minh

- Quy trình Extract – Transform – Load.

- Kỹ thuật trích xuất dữ liệu (full, incremental, CDC), xử lý lỗi như composite key collision.
- Làm sạch và biến đổi dữ liệu, tạo cột mới, chuẩn hóa và tích hợp.
- Tải dữ liệu vào Data Warehouse hoặc Data Mart, xử lý thay đổi theo thời gian với **SCD Type 1 & 2**.

4. Kết hợp Python và Excel trong phân tích dữ liệu

- So sánh ưu – nhược điểm giữa Excel và Python.
- Python trực tiếp trong Excel (hàm =PY()), áp dụng xử lý dữ liệu và vẽ biểu đồ ngay trong bảng tính.
- Lập trình Python bên ngoài để tự động hóa Excel: tạo file từ CSV, thêm định dạng, biểu đồ, và xử lý hàng loạt.

Giá trị nhận được sau khi đọc Blog

- Hiểu và sử dụng thành thạo Pandas để xử lý dữ liệu.
- Lựa chọn biểu đồ trực quan phù hợp để truyền tải thông tin.
- Xây dựng và tối ưu hóa ETL Pipeline cho phân tích dữ liệu chuyên sâu.
- Kết hợp hiệu quả Python và Excel để tận dụng sức mạnh của cả hai công cụ.

Pandas Cơ Bản – Từ Lý Thuyết Đến Ứng Dụng Thực Tế

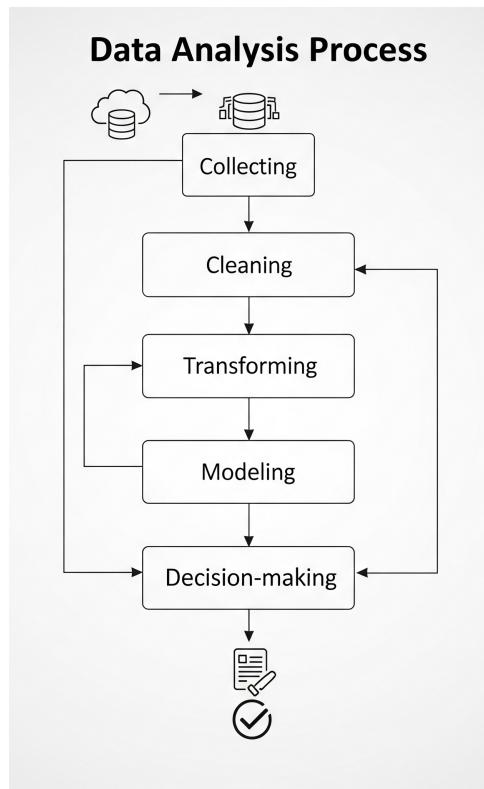
Dàm Nguyễn Khánh

1. Giới thiệu về Pandas

Pandas là một thư viện mã nguồn mở của Python, xây dựng dựa trên **NumPy**, cung cấp các cấu trúc dữ liệu mạnh mẽ và công cụ phân tích dữ liệu thuận tiện. Nó được dùng rộng rãi trong khoa học dữ liệu (*Data Science*), phân tích dữ liệu (*Data Analysis*) và xử lý dữ liệu thời gian thực.

Quy trình phân tích dữ liệu thường gồm:

1. Thu thập dữ liệu (*Collecting*)
2. Làm sạch dữ liệu (*Cleaning*)
3. Chuyển đổi dữ liệu (*Transforming*)
4. Phân tích và mô hình hóa (*Modeling*)
5. Kết luận & ra quyết định (*Decision-making*)



Hình 1: Sơ đồ quy trình phân tích dữ liệu gồm 5 bước: Thu thập, Làm sạch, Chuyển đổi, Mô hình hóa, và Ra quyết định.

2. Cấu trúc dữ liệu trong Pandas

2.1 Series

- **Series** là mảng một chiều (1D array) có nhãn (*index*) cho từng phần tử.
- Có thể chứa bất kỳ kiểu dữ liệu: số, chuỗi, giá trị boolean, object...
- Tương tự như một cột trong bảng Excel.

Bảng Series: 1 cột với index và value

Index	Value
a	10
b	20
c	30
d	40

Hình 2: Bảng Series: 1 cột với index và value.

2.2 DataFrame

- **DataFrame** là cấu trúc hai chiều (2D table) gồm nhiều Series ghép lại.
- Mỗi cột có thể chứa kiểu dữ liệu khác nhau.
- Tương tự như bảng Excel hoặc SQL table.

Bảng DataFrame: Nhiều cột (Name, Gender, Age) và chỉ số hàng

Index	Name	Gender	Age
0	Alice	Female	23
1	Bob	Male	30
2	Charlie	Male	25
3	David	Male	28

Hình 3: Bảng DataFrame: Nhiều cột (Name, Gender, Age) và chỉ số hàng.

3. Các thao tác cơ bản với DataFrame

3.1 Đọc và ghi dữ liệu

```

1 import pandas as pd      # 1. Import thư viện pandas và đặt bí danh (alias) là 'pd'
2     # - pandas: thư viện xử lý dữ liệu dạng bảng (DataFrame, Series)
3     # - alias 'pd' giúp viết code ngắn gọn hơn
4
5 # Đọc CSV
6 df = pd.read_csv("pokemon.csv") # 2. Đọc dữ liệu từ file CSV và lưu vào biến df (DataFrame)
7     # - "pokemon.csv": đường dẫn hoặc tên file cần đọc
8     # - pd.read_csv: hàm của pandas để đọc file CSV
9     # - df (DataFrame): cấu trúc dữ liệu 2 chiều, giống bảng Excel
10
11 # Ghi CSV
12 df.to_csv("output.csv", index=False) # 3. Ghi dữ liệu từ df ra file CSV mới
13     # - "output.csv": tên file xuất ra
14     # - index=False: không ghi cột chỉ số (index) vào file CSV
15     # - Nếu không có index=False, pandas sẽ tự thêm cột index vào file
16
17 # Ghi Excel
18 df.to_excel("output.xlsx", index=False) # 4. Ghi dữ liệu ra file Excel (.xlsx)
19     # - Yêu cầu cài thêm thư viện hỗ trợ (ví dụ: openpyxl hoặc xlsxwriter)
20     # - index=False: không xuất cột index

```

Giải thích thêm:

- pd.read_csv() là hàm phổ biến nhất trong Pandas để đọc dữ liệu từ tệp CSV.
- Khi lưu dữ liệu bằng to_csv() hoặc to_excel(), tham số index=False thường được dùng để tránh xuất cột chỉ số vì nhiều trường hợp không cần thiết cho phân tích.
- Các hàm này có thể nhận nhiều tham số bổ sung, ví dụ:
 - sep=";": dùng khi file CSV phân cách bằng dấu chấm phẩy.
 - encoding="utf-8" hoặc "utf-8-sig": tránh lỗi font tiếng Việt.
 - sheet_name="Data" (với Excel): đặt tên sheet khi xuất file.

3.2 Xem thông tin dữ liệu

```

1 df.head(5)    # 1. Hiển thị 5 dòng đầu tiên của DataFrame 'df'
2     # - Mặc định: df.head() sẽ hiển thị 5 dòng nếu không truyền tham số
3     # - Tham số: số nguyên n => df.head(n) hiển thị n dòng đầu
4
5 df.tail(5)    # 2. Hiển thị 5 dòng cuối cùng của DataFrame 'df'
6     # - Mặc định: df.tail() cũng hiển thị 5 dòng nếu không truyền tham số
7     # - Thường dùng để kiểm tra dữ liệu cuối cùng hoặc đảm bảo việc đọc file đầy đủ

```

```

8
9 df.describe() # 3. Thống kê nhanh các giá trị số trong DataFrame
10    # - Bao gồm: count (số lượng), mean (trung bình), std (độ lệch chuẩn),
11    # min, max, và các phân vị (25%, 50%, 75%)
12    # - Chỉ áp dụng mặc định cho cột dạng số, nếu muốn áp dụng cho mọi cột:
13    # df.describe(include='all')
14
15 df.info() # 4. Thông tin tổng quát về DataFrame
16    # - Số dòng, số cột, tên cột
17    # - Kiểu dữ liệu của từng cột (int64, float64, object...)
18    # - Số giá trị không null của mỗi cột
19    # - Dùng để kiểm tra dữ liệu bị thiếu (missing values)

```

Giải thích thêm:

- df.head() và df.tail() rất hữu ích khi dữ liệu có hàng nghìn dòng, giúp xem nhanh một phần dữ liệu mà không in toàn bộ.
- df.describe() hỗ trợ cả dữ liệu dạng ngày tháng nếu dùng include=[np.datetime64].
- df.info() là công cụ chẩn đoán nhanh cấu trúc dữ liệu, đặc biệt quan trọng trước khi bắt đầu tiền xử lý.

3.3 Lựa chọn cột và hàng

```

1 df['Name'] # 1. Chọn một cột duy nhất "Name"
2     # - Kết quả: một Pandas Series
3     # - Cú pháp: df['tên_cột']
4
5 df[['Name', 'Type 1']] # 2. Chọn nhiều cột "Name" và "Type 1"
6     # - Kết quả: một Pandas DataFrame
7     # - Cú pháp: df[['cột1', 'cột2', ...]]
8     # - Lưu ý: cần đặt danh sách tên cột trong một list []
9
10 df.iloc[0:5] # 3. Chọn các hàng theo vị trí index (dạng số nguyên)
11     # - .iloc: truy cập dữ liệu dựa trên chỉ số hàng/cột
12     # - 0:5 nghĩa là chọn từ hàng 0 đến hàng 4 (Python cắt trước giới hạn cuối)
13     # - Mặc định lấy tất cả cột
14
15 df.loc[0:5, 'Name'] # 4. Chọn dữ liệu theo nhãn (label) của index
16     # - .loc: truy cập theo tên (label) của index và tên cột
17     # - 0:5 ở đây bao gồm cả hàng 5 (khác với .iloc)
18     # - 'Name': chỉ lấy cột "Name"
19     # - Kết quả: Series gồm các giá trị ở cột Name từ hàng 0 tới hàng 5

```

Giải thích thêm:

- Khi dùng df['Name'] kết quả là một **Series**, còn df[['Name']] vẫn là **DataFrame**.

- **.iloc** (integer-location) chỉ dùng chỉ số nguyên (0, 1, 2, ...).
- **.loc** (label-location) cho phép truy cập theo tên hàng và tên cột.
- Với `.loc[start:end]` thì cả end vẫn được bao gồm, khác với `.iloc[start:end]`.

3.4 Thêm và xóa cột

```

1 df.drop('Type 2', axis=1, inplace=True)
2 # 1. Xóa cột "Type 2"
3 # - df.drop(): hàm dùng để xóa hàng hoặc cột khỏi DataFrame
4 # - 'Type 2': tên cột cần xóa
5 # - axis=1: chỉ định xóa theo chiều cột (axis=0 là chiều hàng)
6 # - inplace=True: thay đổi trực tiếp trên DataFrame gốc (nếu False sẽ trả về bản sao)
7
8 df['Power'] = df['Attack'] + df['Defense']
9 # 2. Thêm cột mới "Power"
10 # - df['Power']: tên cột mới, nếu chưa tồn tại sẽ được tạo
11 # - df['Attack'] + df['Defense']: phép cộng từng phần tử (element-wise) của hai cột
12 # - Kết quả: cột Power chứa tổng Attack và Defense cho từng hàng

```

Giải thích thêm:

- axis trong Pandas:
 - axis=0: thao tác theo chiều dọc (hàng).
 - axis=1: thao tác theo chiều ngang (cột).
- Khi thêm cột mới, giá trị có thể được tính từ các cột khác hoặc gán cố định.
- Nên cẩn thận với `inplace=True` vì sẽ thay đổi trực tiếp dữ liệu gốc, không thể hoàn tác nếu chưa lưu.

3.5 Sắp xếp và lọc dữ liệu

```

1 df.sort_values('Attack', ascending=False)
2 # 1. Sắp xếp DataFrame theo cột "Attack" giảm dần
3 # - df.sort_values(): hàm sắp xếp theo giá trị của một hoặc nhiều cột
4 # - 'Attack': tên cột dùng để sắp xếp
5 # - ascending=False: sắp xếp giảm dần (True: tăng dần)
6 # - Có thể sắp xếp nhiều cột:
7 #   df.sort_values(['Type 1', 'Attack'], ascending=[True, False])
8
9 df[df['Type 1'] == 'Fire']
10 # 2. Lọc dữ liệu theo điều kiện logic
11 # - df['Type 1'] == 'Fire': tạo một Series kiểu boolean (True/False)
12 # - df[...]: chỉ giữ lại những hàng mà điều kiện trong [...] là True
13 # - Kết quả: DataFrame chỉ chứa các Pokémon hệ "Fire"

```

```

14
15 df[df['Name'].str.contains('^Pi[a-zA-Z]*')]
16 # 3. Lọc dữ liệu bằng biểu thức chính quy (Regex)
17 #   - df['Name'].str.contains(): kiểm tra chuỗi trong cột Name có khớp pattern không
18 #   - '^Pi[a-zA-Z]*':
19 #     ^ : bắt đầu chuỗi
20 #     Pi : ký tự 'Pi'
21 #     [a-zA-Z]* : 0 hoặc nhiều ký tự thường từ a đến z
22 #   - Kết quả: các Pokémon có tên bắt đầu bằng "Pi", ví dụ: Pikachu, Pidgey
23 #   - Thêm tham số case=False nếu muốn không phân biệt hoa thường

```

Giải thích thêm:

- Sắp xếp dữ liệu giúp tìm giá trị lớn nhất/nhỏ nhất nhanh chóng.
- Lọc điều kiện trong Pandas thường kết hợp nhiều điều kiện:

```
1 df[(df['Type 1'] == 'Fire') & (df['Attack'] > 70)]
```

- Regex cho phép tìm kiếm linh hoạt, rất hữu ích khi dữ liệu chứa tên hoặc chuỗi phức tạp.

3.6 Nhóm dữ liệu (GroupBy)

```

1 df.groupby('Type 1')['Attack'].mean()
2 # 1. df.groupby('Type 1'):
3 #   - Nhóm các hàng trong DataFrame theo giá trị của cột "Type 1".
4 #   - Mỗi nhóm sẽ chứa tất cả Pokémon có cùng hệ (ví dụ: Fire, Water, Grass...).
5 #
6 # 2. ['Attack']:
7 #   - Sau khi nhóm, chỉ chọn cột "Attack" để tính toán.
8 #
9 # 3. .mean():
10 #   - Tính giá trị trung bình (mean) của Attack cho từng nhóm.
11 #
12 # Kết quả: Một Pandas Series với:
13 #   - Index: tên từng nhóm (các giá trị trong "Type 1")
14 #   - Value: Attack trung bình của nhóm đó

```

Ví dụ:

```

1 Type 1
2 Bug      55.23
3 Electric 76.50
4 Fire     82.33
5 Grass    65.40

```

```
6 Water    70.12  
7 Name: Attack, dtype: float64
```

Giải thích thêm:

- groupby có thể áp dụng nhiều hàm thống kê khác nhau:

```
1 df.groupby('Type 1')['Attack'].max() # Lấy giá trị Attack lớn nhất  
2 df.groupby('Type 1')['Attack'].min() # Lấy giá trị Attack nhỏ nhất  
3 df.groupby('Type 1')['Attack'].sum() # Tổng Attack theo nhóm
```

- Có thể nhóm theo nhiều cột:

```
1 df.groupby(['Type 1', 'Type 2'])['Attack'].mean()
```

- Nếu muốn kết quả là DataFrame thay vì Series, dùng reset_index():

```
1 df.groupby('Type 1')['Attack'].mean().reset_index()
```

4. Ứng dụng thực tế

4.1 Phân tích dữ liệu Pokémon

- Đọc file CSV Pokémon.
- Xem các chỉ số quan trọng (HP, Attack, Defense...).
- Lọc các Pokémon Legendary.
- Sắp xếp theo tổng chỉ số (Total).

4.2 Phân tích dữ liệu thời gian (Time Series)

Dataset: Daily Minimum Temperatures.

1. Đọc dữ liệu thời gian.
2. Đặt cột Date làm index.
3. Trích xuất **năm, tháng, ngày trong tuần**.
4. Lọc dữ liệu theo khoảng thời gian.
5. Trực quan hóa nhiệt độ theo thời gian.

4.3 Resampling & Seasonality

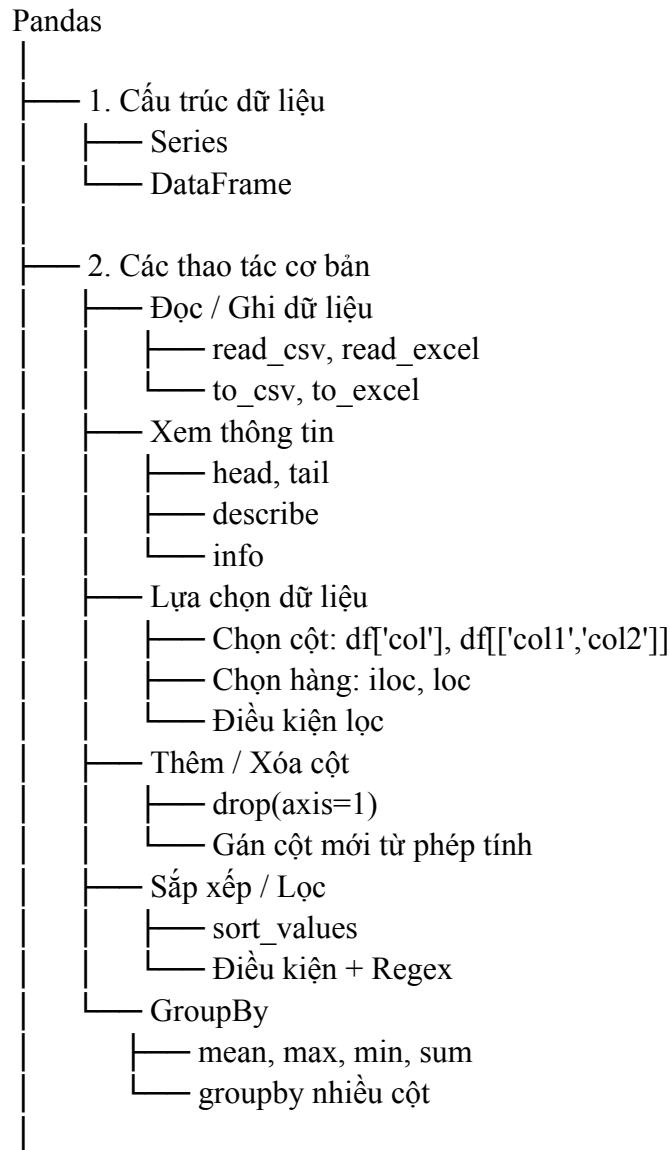
- **Resampling:** Chuyển đổi tần suất dữ liệu (ngày → tuần hoặc tháng).
- **Downsampling:** Giảm tần suất (ví dụ: trung bình nhiệt độ theo tháng).
- **Upsampling:** Tăng tần suất (thêm ngày bị thiếu, dùng ffill() để điền).

5. Kết luận

Pandas là thư viện nền tảng cho mọi dự án phân tích dữ liệu trong Python. Hiểu rõ Series, DataFrame, và các thao tác cơ bản sẽ giúp bạn:

- Làm sạch dữ liệu nhanh chóng.
- Chuẩn bị dữ liệu cho Machine Learning.
- Trực quan hóa và báo cáo hiệu quả.

Mindmap tổng kết kiến thức Pandas



- └── 3. Ứng dụng
 - └── Phân tích Pokémon Dataset
 - └── Lọc Legendary
 - └── Thống kê Attack, Defense
 - └── Phân tích Time Series
 - └── Chuyển đổi index thành DateTime
 - └── Trích xuất ngày, tháng, năm
 - └── Resampling
 - └── Seasonality
- └── 4. Kết luận
 - └── Pandas là nền tảng xử lý dữ liệu trong Python
 - └── Thành thạo các thao tác cơ bản trước khi ML
 - └── Kết hợp với NumPy, Matplotlib để mạnh hơn

Trực quan hóa và phân tích dữ liệu với Pandas

Dao Lam Hoang

Giới thiệu

Pandas là một thư viện Python mạnh mẽ và phổ biến trong lĩnh vực phân tích dữ liệu, cung cấp các cấu trúc dữ liệu linh hoạt như Series và DataFrame. Với Pandas, người dùng có thể thao tác, làm sạch, và biến đổi dữ liệu một cách nhanh chóng, đồng thời kết hợp với các thư viện trực quan hóa như Matplotlib hay Seaborn để phân tích và truyền tải thông tin trực quan hơn.

Tài liệu này tập trung vào việc minh họa các thao tác cơ bản và quan trọng nhất với Pandas, bao gồm:

- Làm việc với Series và các hàm thường dùng.
- Quản lý dữ liệu thiếu và xử lý dữ liệu dạng bảng với DataFrame.
- Thực hiện các thao tác nối, lọc, sắp xếp và truy xuất dữ liệu.
- Trực quan hóa dữ liệu để hỗ trợ quá trình phân tích.

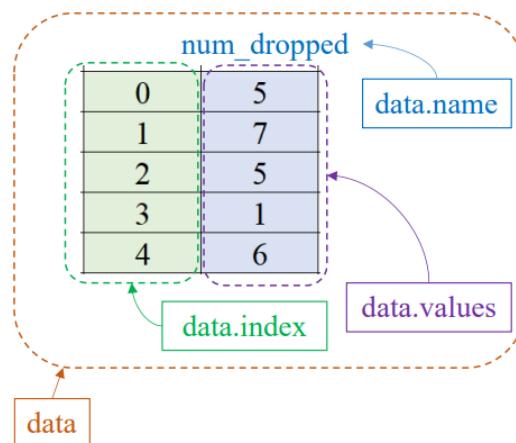
Qua các ví dụ cụ thể và hình ảnh minh họa, người đọc có thể dễ dàng nắm bắt cách ứng dụng Pandas trong các tình huống thực tế.

1. Pandas Series

1.1 Bắt đầu với Series

Tạo một Series

```
1 import pandas as pd
2
3 data = pd.Series([5, 7, 5, 1, 6],
4                  name='num_dropped')
```



Lấy hàng từ Series

Xoá hàng

Chèn hàng

```

1 # Lấy từ dòng 2 đến 3 (bao gồm 3)
2 result = data.loc[2:3]
3
4 # Lấy dòng 2 (không bao gồm dòng 3)
5 result = data[2]
6
7 # Lấy những giá trị giữa 3 và 6
8 result = data[data.between(3,6)]
9
10 # Lấy những giá trị lớn hơn 5
11 result = data[data > 5]

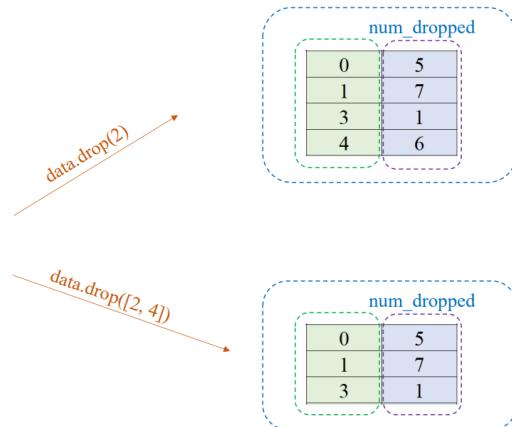
```

$\text{result} = \text{data.loc}[2:3]$ $\text{result} = \begin{array}{ c c }\hline 2 & 5 \\ \hline 3 & 1 \\ \hline\end{array}$	$\text{result} = \text{data[2:3]}$ $\text{result} = \begin{array}{ c c }\hline 2 & 5 \\ \hline\end{array}$
$\text{result} = \text{data[data.between}(3,6)]$ $\text{result} = \begin{array}{ c c }\hline 0 & 5 \\ \hline 2 & 5 \\ \hline 4 & 6 \\ \hline\end{array}$	$\text{result} = \text{data[data}>5]$ $\text{result} = \begin{array}{ c c }\hline 1 & 7 \\ \hline 4 & 6 \\ \hline\end{array}$

```

1 # Xoá phần tử có chỉ số 2
2 result = data.drop(2)
3
4 # Xoá phần tử có chỉ số 2 và 4
5 result = data.drop([2, 4])

```



1.2 Một số hàm thường dùng trong Series

Hàm thống kê

```

1 data.min()    # -> 1
2 data.max()    # -> 7
3 data.sum()    # -> 24
4 data.mean()   # -> 4.8
5 data.std()    # -> 2.28
6 data.var()    # -> 5.2
7 data.idxmax() # -> 1
8 data.argmax() # -> 1

```

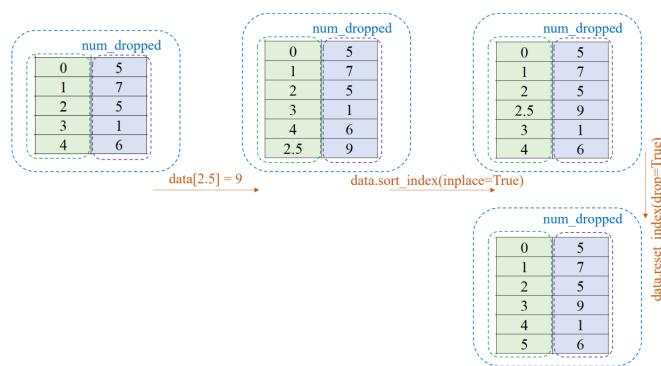
Cộng hai Series

Xử lý giá trị thiếu

```

1 # Thêm giá trị 9 vào chỉ số 2.5
2 data[2.5] = 9
3
4 # Sắp xếp lại chỉ số
5 data.sort_index(inplace=True)
6
7 # Đặt lại chỉ số mới
8 data.reset_index(drop=True, inplace=True)

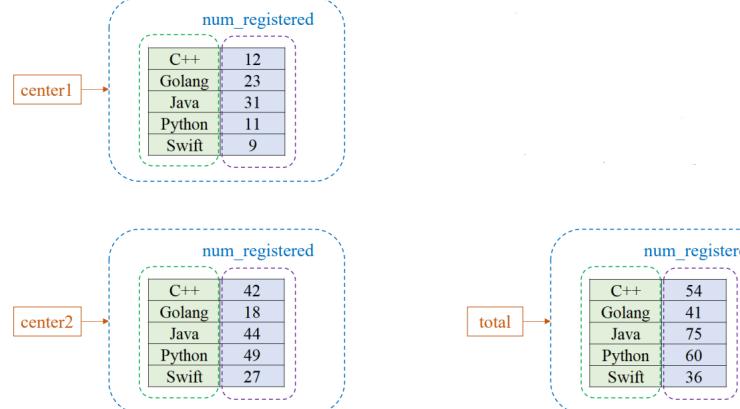
```



```

1 center1 = pd.Series([12, 23, 31, 11, 9],
2                     index=['C++', 'Golang', 'Java',
3                           'Python', 'Swift'],
4                     name='num_registered')
5
6 center2 = pd.Series([42, 18, 44, 49, 27],
7                     index=['C++', 'Golang', 'Java',
8                           'Python', 'Swift'],
9                     name='num_registered')
10
11 total = center1 + center2

```



2. DataFrame

Một số thao tác cơ bản

Tạo DataFrame từ file

```

1 df = pd.read_csv('advertising.csv')
2 print(df)

```

Sắp xếp dữ liệu

```

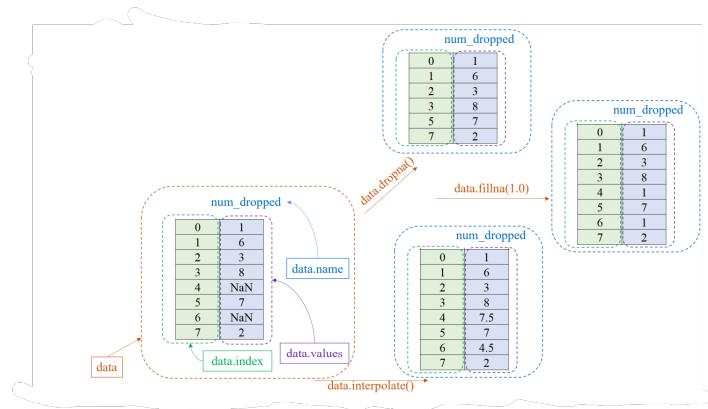
1 # Sắp xếp theo cột Sales
2 df.sort_values('Sales')
3
4 # Sắp xếp theo nhiều cột
5 df.sort_values(['Radio', 'Newspaper', 'Sales'])

```

```

1 import numpy as np
2
3 data = pd.Series([1, 6, 3, 8, np.nan, 7, np.nan,
4                   2], name='num_dropped')
5
6 # Loại bỏ giá trị NaN
7 data.dropna()
8
9 # Thay thế NaN bằng 1.0
10 data.fillna(1.0)
11
12 # Nối suy giá trị
13 data.interpolate()

```



Xoá dòng trong DataFrame

```

1 # Xoá dòng có chỉ số 1
2 df.drop(1, axis=0)
3
4 # Xoá nhiều dòng
5 df.drop([1, 2, 3], axis=0)

```

Lấy dữ liệu từ DataFrame

```

1 # Lấy cột
2 df['Radio']
3 df[['Radio', 'Sales']]

```

Sử dụng loc và iloc

- **loc:** Dùng để truy cập theo nhãn (label).
- **iloc:** Dùng để truy cập theo vị trí (index).

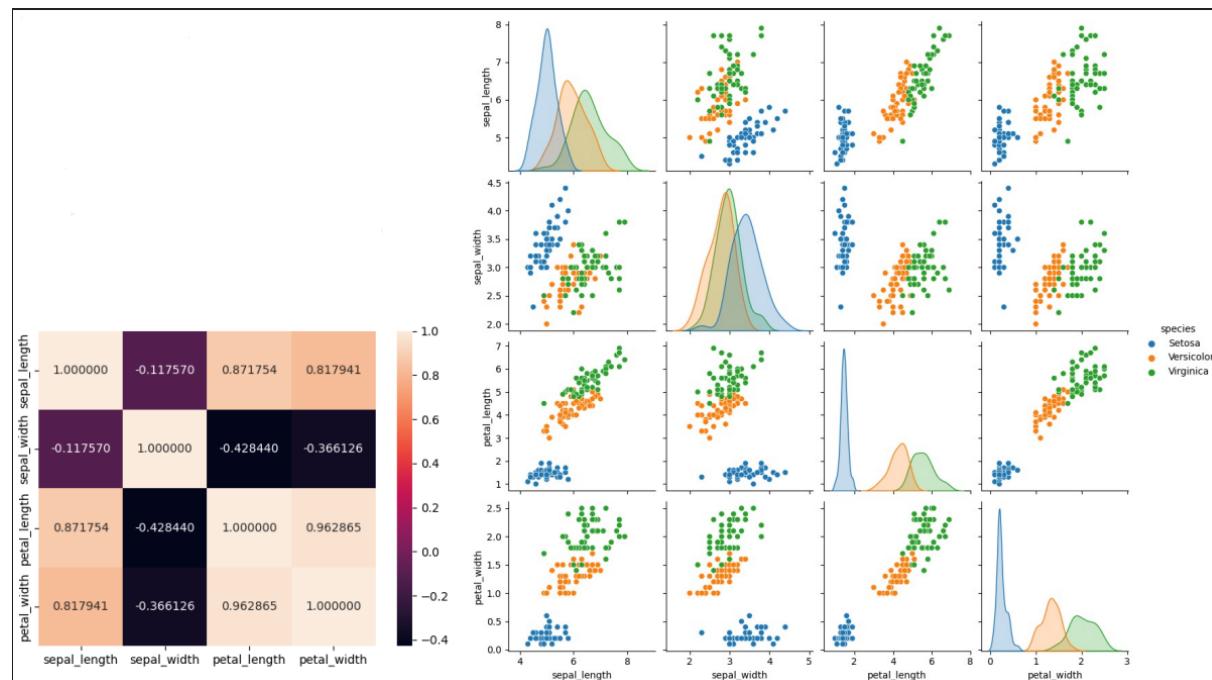
```
1 df.loc[['E', 'G'], :]
```

Nối nhiều DataFrame

```
1 df_1 = pd.read_csv('ads_1.csv')
2 df_2 = pd.read_csv('ads_2.csv')
3
4 df_3 = pd.concat([df_1, df_2], ignore_index=True)
```

3. Trực quan hóa dữ liệu

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 data = pd.read_csv('iris.csv')
```



Kết luận

Qua các ví dụ và hướng dẫn trong tài liệu, ta có thể thấy rằng Pandas đóng vai trò quan trọng trong quy trình phân tích dữ liệu — từ khâu tiền xử lý, quản lý dữ liệu, cho đến trực quan hóa. Sự kết hợp giữa cú pháp dễ đọc, tốc độ xử lý nhanh và khả năng tương tác tốt với các thư viện Python khác giúp Pandas trở thành công cụ không thể thiếu đối với nhà phân tích dữ liệu và nhà khoa học dữ liệu.

Việc nắm vững các thao tác cơ bản như tạo Series, DataFrame, xử lý dữ liệu thiếu, sắp xếp và lọc dữ liệu sẽ tạo nền tảng vững chắc để bạn tiếp tục khai thác các tính năng nâng cao như groupby, merge, hoặc áp dụng các kỹ thuật phân tích thống kê và học máy trên dữ liệu đã chuẩn bị.

Phân Tích và Trực Quan Hóa Dữ Liệu Nâng Cao với Python

Bùi Đức Xuân

Tóm tắt nội dung

Trực quan hóa dữ liệu là một kỹ năng thiết yếu trong kỹ nguyên dữ liệu, giúp chúng ta biến các bộ dữ liệu phức tạp thành những hình ảnh trực quan dễ hiểu. Bài viết này sẽ đi sâu vào các kiến thức và ví dụ code thực tế từ slide bài giảng về "Advanced Data Visualization", hướng dẫn bạn cách chọn biểu đồ phù hợp và sử dụng các thư viện Python mạnh mẽ như Matplotlib, Seaborn, và Plotly để tạo ra các biểu đồ ấn tượng.

1. Trực Quan Hóa Dữ Liệu Là Gì?

Trực quan hóa dữ liệu là thực hành chuyển đổi thông tin thành ngữ cảnh trực quan, chẳng hạn như bản đồ hoặc biểu đồ, để giúp não bộ con người dễ hiểu và rút ra thông tin chi tiết từ dữ liệu hơn. Nó giúp dịch các bộ dữ liệu phức tạp thành các định dạng trực quan mà não bộ con người dễ dàng nắm bắt. Trực quan hóa dữ liệu là một công cụ thiết yếu cho bất kỳ tổ chức dựa trên dữ liệu nào, cho phép người dùng hiểu các bộ dữ liệu phức tạp và truyền đạt thông tin chi tiết đến các bên liên quan một cách hiệu quả.

Trực quan hóa dữ liệu có thể có nhiều hình thức khác nhau và có thể sử dụng nhiều loại dữ liệu khác nhau.

2. Trực Quan Hóa Dữ Liệu VỚI Python

Python cung cấp các thư viện mạnh mẽ để trực quan hóa dữ liệu. Các thư viện phổ biến bao gồm:

- **Matplotlib**: Cung cấp khả năng tùy chỉnh rộng rãi nhưng yêu cầu nhiều mã hơn.
- **Seaborn**: Đơn giản hóa các biểu đồ thống kê với các chủ đề tích hợp.
- **Plotly**: Xuất sắc trong việc tạo các trực quan hóa động và tương tác.

3. Cách Chọn Biểu Đồ Phù Hợp

Với rất nhiều loại biểu đồ để lựa chọn, việc quyết định sử dụng loại nào cho một bộ dữ liệu cụ thể có thể là một thách thức. Quá trình này bao gồm 5 bước chính:

- **Bước 1: Xác định loại dữ liệu** Dữ liệu có thể được phân loại thành bốn loại: định lượng (quantitative), phân loại (categorical), thời gian (temporal), hoặc không gian (spatial).
 - Dữ liệu định lượng (Quantitative data): Đề cập đến các giá trị số.
 - Dữ liệu phân loại (Categorical data): Đề cập đến các giá trị không phải số.
 - Dữ liệu thời gian (Temporal data): Đề cập đến dữ liệu dựa trên thời gian.
 - Dữ liệu không gian (Spatial data): Đề cập đến dữ liệu dựa trên vị trí.
- **Bước 2: Xác định mối quan hệ giữa các biến** Bạn muốn hiển thị mối quan hệ nào trong trực quan hóa của mình? So sánh, phân phối, hay mối quan hệ?

- Biểu đồ so sánh (comparison chart): Hữu ích để hiển thị sự khác biệt giữa hai hoặc nhiều điểm dữ liệu, như biểu đồ cột hoặc biểu đồ thanh.
 - Biểu đồ phân phối (distribution chart): Hữu ích để hiển thị cách dữ liệu được phân tán, như biểu đồ tần suất (histogram) hoặc biểu đồ hộp (box plot).
 - Biểu đồ mối quan hệ (relationship chart): Hữu ích để hiển thị mối quan hệ giữa hai hoặc nhiều biến, như biểu đồ phân tán (scatter plot) hoặc biểu đồ bong bóng (bubble chart).
- **Bước 3: Xác định mục đích của trực quan hóa** Bạn muốn truyền tải thông điệp gì qua trực quan hóa dữ liệu của mình? Xu hướng, so sánh, hay phân phối?
 - Để hiển thị xu hướng theo thời gian, biểu đồ đường (line chart) hoặc biểu đồ vùng (area chart) có thể phù hợp hơn.
 - Để so sánh các điểm dữ liệu, biểu đồ cột (bar chart) hoặc biểu đồ thanh (column chart) có thể là lựa chọn tốt hơn.
 - Để hiển thị phân phối, biểu đồ tần suất (histogram) hoặc biểu đồ hộp (box plot) có thể hữu ích hơn.
 - **Bước 4: Xác định đối tượng mục tiêu** Xem xét đối tượng mục tiêu của bạn. Họ có hiểu các biểu đồ phức tạp hay cần một cách trình bày đơn giản hơn?
 - **Bước 5: Chọn loại biểu đồ phù hợp** Không có loại biểu đồ nào là giải pháp "một kích cỡ cho tất cả". Điều cần thiết là thử nghiệm với các loại biểu đồ khác nhau để tìm ra loại phù hợp nhất cho dữ liệu của bạn.

4. Các Trường Hợp Nghiên Cứu và Mã Tương Ứng

Chúng ta sẽ đi sâu vào các loại biểu đồ thông dụng thông qua ba trường hợp nghiên cứu: Bộ dữ liệu ETTh, Bộ dữ liệu hiệu suất sinh viên, và Bộ dữ liệu Iris.

4.1 Trường Hợp Nghiên Cứu: Bộ Dữ Liệu ETTh (Electricity Transformer Dataset)

Bộ dữ liệu này có thể được sử dụng để xác định thời điểm OT cao nhất, hiểu mối tương quan giữa các biến, hiểu phân phối dữ liệu, và phân tích tải trung bình.

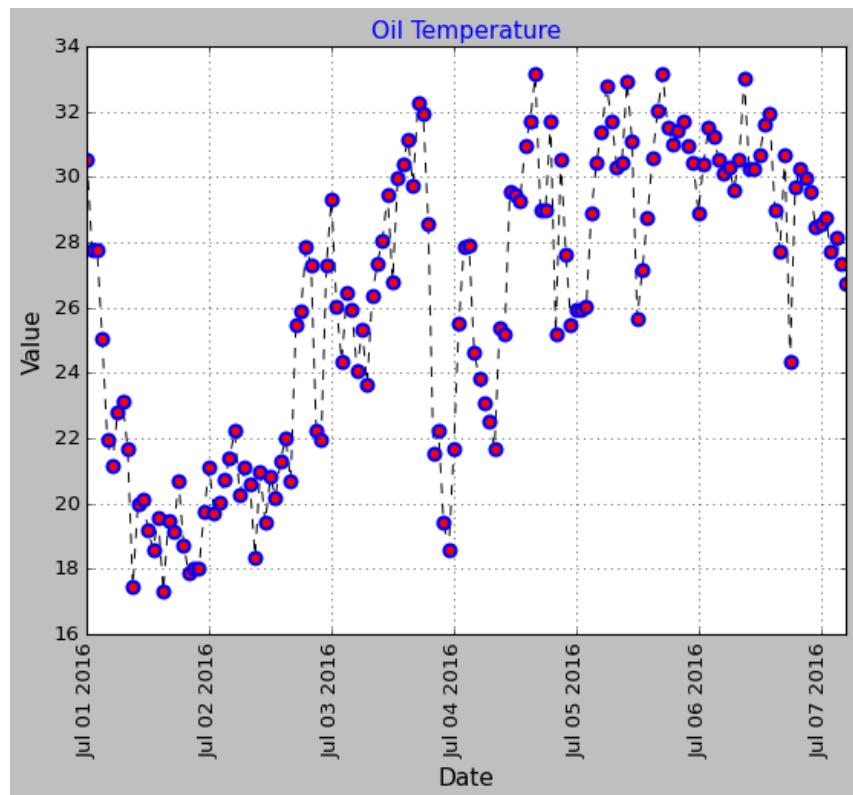
4.1.1 Biểu Đồ Đường (Line Chart)

Biểu đồ đường là một trong những biểu đồ phổ biến nhất được sử dụng để quan sát một hoặc nhiều biến với sự thay đổi của một biến khác, thường được sử dụng trong phân tích xu hướng và phân tích chuỗi thời gian.

```
1 import matplotlib.pyplot as plt
2 import pandas as pd
3
4 plt.plot(df.date[0:150], df.OT[0:150], marker = 'o', color = 'black',
5           linewidth = 0.9, linestyle = '--',
6           markeredgecolor = 'blue',
7           markeredgewidth = '2.0',
```

```
8     markerfacecolor = 'red', markersize = 7.0)
9 plt.title('Oil Temperature', color = 'Blue', size = 14)
10 plt.xlabel('Date', size = 14)
11 plt.ylabel('Value', size = 14)
12 plt.style.use('fivethirtyeight')
13 plt.grid(True)
14 plt.xticks(rotation = 90)
15 plt.show()
```

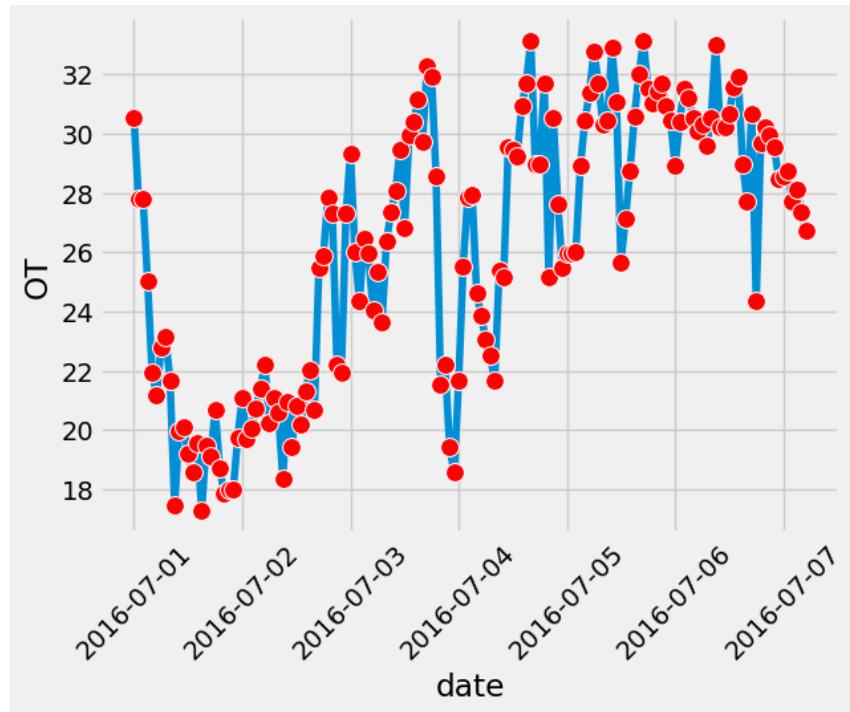
Listing 1: Biểu đồ đường đơn với Matplotlib



Hình 4: Biểu đồ đường đơn với Matplotlib

```
1 import seaborn as sns
2 sns.lineplot(data=df, x=df.date[0:150], y=df.OT[0:150], marker='o',
3               markersize=10, markerfacecolor='red')
4 plt.xticks(rotation=45)
```

Listing 2: Biểu đồ đường đơn với Seaborn



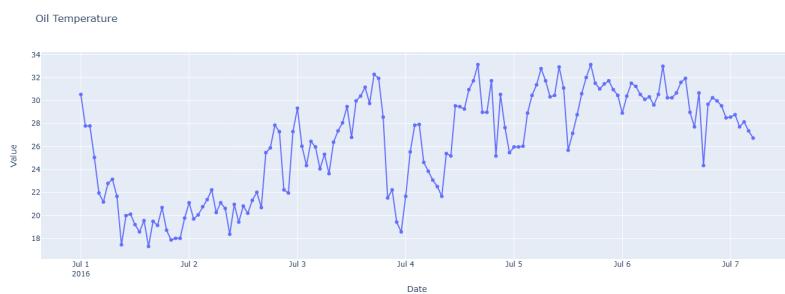
Hình 5: Biểu đồ đường đơn với Seaborn

```

1 import plotly.express as px
2 fig = px.line(df, x=df.date[0:150], y=df.OT[0:150],
3                 title='Oil Temperature', markers=True)
4 fig.update_layout(
5     xaxis_title="Date", yaxis_title="Value"
6 )
7 fig.show()

```

Listing 3: Biểu đồ đường đơn với Plotly



Hình 6: Biểu đồ đường đơn với Plotly

4.1.2 Biểu Đồ Hộp (Box Plots)

Biểu đồ hộp và râu (box and whisker plots) mô tả sự phân phối của dữ liệu, các giá trị ngoại lai (outliers) và giá trị trung vị (median). Nó giúp xác định sự tồn tại của các giá trị ngoại lai trong bộ dữ liệu.

```

1 import matplotlib.pyplot as plt

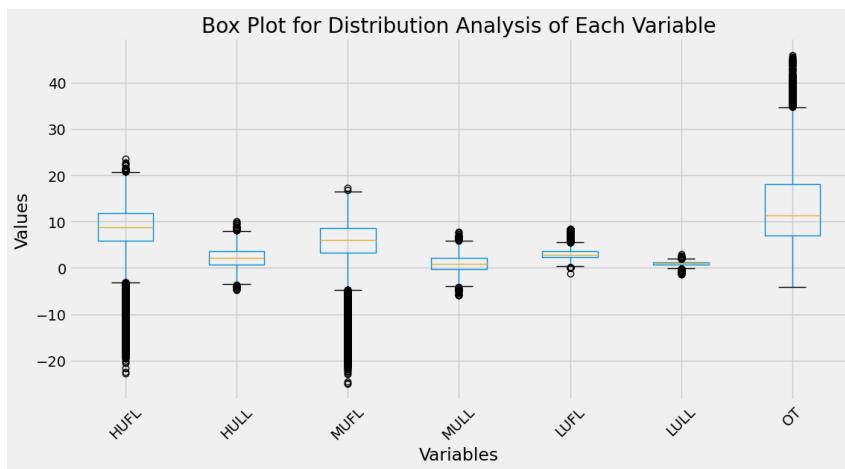
```

```

2
3 # Creating Box Plots for Distribution Analysis of each variable
4 plt.figure(figsize=(12, 6))
5 df.boxplot(column=[ 'HUFL' , 'HULL' , 'MUFL' , 'MULL' , 'LUFL' , 'LULL' , 'OT' ])
6 plt.xlabel( 'Variables' )
7 plt.ylabel( 'Values' )
8 plt.title( 'Box Plot for Distribution Analysis of Each Variable' )
9 plt.xticks(rotation=45)
10 plt.show()

```

Listing 4: Biểu đồ Hộp với Matplotlib



Hình 7: Biểu đồ hộp với Matplotlib

4.1.3 Biểu Đồ Cột (Bar Chart)

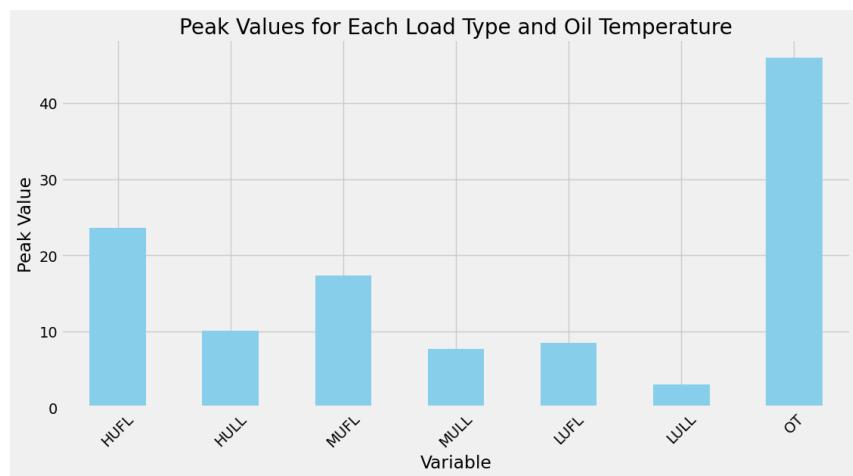
Biểu đồ cột sử dụng các thanh để hiển thị sự thay đổi giá trị của một biến cụ thể so với biến khác. Loại biểu đồ này thường được sử dụng với dữ liệu rời rạc hoặc phân loại.

```

1 import matplotlib.pyplot as plt
2
3 # Identifying peak values and their corresponding times for each column
4 peak_values = df.max()
5 peak_times = df.idxmax()
6
7 # Creating a DataFrame to display peak values and their times
8 peak_analysis = pd.DataFrame({ 'Peak Value': peak_values , 'Peak Time': peak_times })
9
10 # Dropping the 'date' column as it's not a load or temperature type
11 peak_analysis = peak_analysis.drop('date')
12
13 # Plotting the peak values for visual representation
14 plt.figure(figsize=(12, 6))
15 peak_analysis[ 'Peak Value' ].plot(kind='bar' , color='skyblue')
16 plt.title( 'Peak Values for Each Load Type and Oil Temperature' )
17 plt.xlabel( 'Variable' )
18 plt.ylabel( 'Peak Value' )
19 plt.xticks(rotation=45)
20 plt.show()

```

Listing 5: Biểu đồ Cột với Matplotlib



Hình 8: Biểu đồ cột với Matplotlib

4.1.4 Biểu Đồ Donut (Donut Chart)

Biểu đồ Donut là một biến thể của biểu đồ tròn, thường được sử dụng để hiển thị thành phần part-to-whole (phần-trong-tổng thể) giống như biểu đồ tròn, nhưng có một lỗ ở giữa.

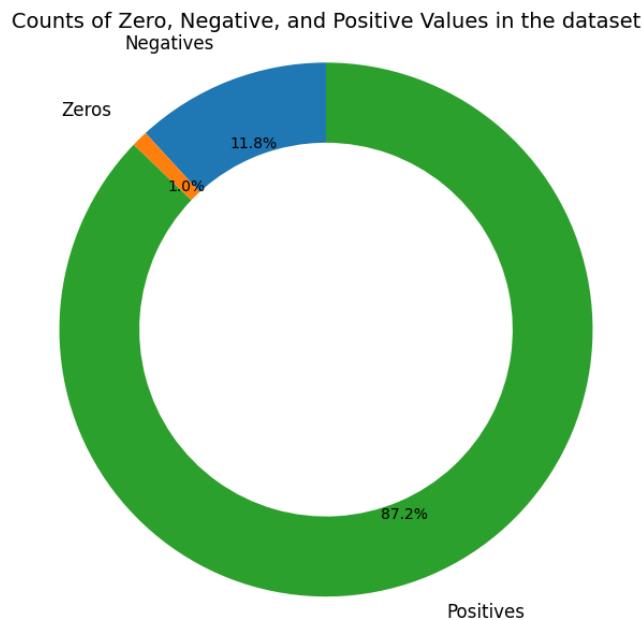
```
1 import matplotlib.pyplot as plt
2 import pandas as pd
3 import numpy as np
4
5
6 counts = {
7     col: [
8         np.sum(df[col] < 0),
9         np.sum(df[col] == 0),
10        np.sum(df[col] > 0)
11    ]
12   for col in df.columns[1:]
13 }
14
15 counts_df = pd.DataFrame(counts, index=['Negatives', 'Zeros', 'Positives'])
16 .T
17 counts_for_donut = counts_df.sum()
18
19 # Plot
20 fig, ax = plt.subplots(figsize=(8, 6))
21
22 colors = ['#1f77b4', '#ff7f0e', '#2ca02c'] # blue, orange, green
23 wedges, texts, autotexts = ax.pie(
24     counts_for_donut,
25     labels=counts_for_donut.index,
26     autopct='%.1f%%',
27     startangle=90,
28     wedgeprops=dict(width=0.3),
29     colors=colors,
30     pctdistance=0.75,
31     labeldistance=1.15
32 )
33
34 # Set font size
35 for text in texts:
```

```

35     text.set_fontsize(12)
36 for autotext in autotexts:
37     autotext.set_fontsize(10)
38
39 # Donut hole
40 centre_circle = plt.Circle((0, 0), 0.65, fc='white')
41 fig.gca().add_artist(centre_circle)
42
43 ax.axis('equal')
44 plt.title('Counts of Zero, Negative, and Positive Values in the dataset',
45            fontsize=14)
46 plt.tight_layout()
47 plt.show()

```

Listing 6: Biểu đồ Donut với Matplotlib



Hình 9: Biểu đồ Donut với Matplotlib

4.1.5 Biểu Đồ Tương Quan (Correlation Chart)

- Bản đồ nhiệt (Heatmap):** Là một biểu đồ hai chiều, chủ yếu được sử dụng để phân tích mối tương quan giữa các trường khác nhau trong một bộ dữ liệu.
- Biểu đồ phân tán (Scatter Plot):** Được sử dụng để nghiên cứu mối tương quan giữa hai biến.

```

1 import matplotlib.pyplot as plt
2 import pandas as pd
3 import numpy as np
4 corr_matrix = df.corr(numeric_only=True)
5 plt.figure(figsize=(10, 8))
6 sns.heatmap(corr_matrix, annot=True, cmap='viridis', linewidths=0.5)

```

Listing 7: Biểu đồ Heatmap với Seaborn



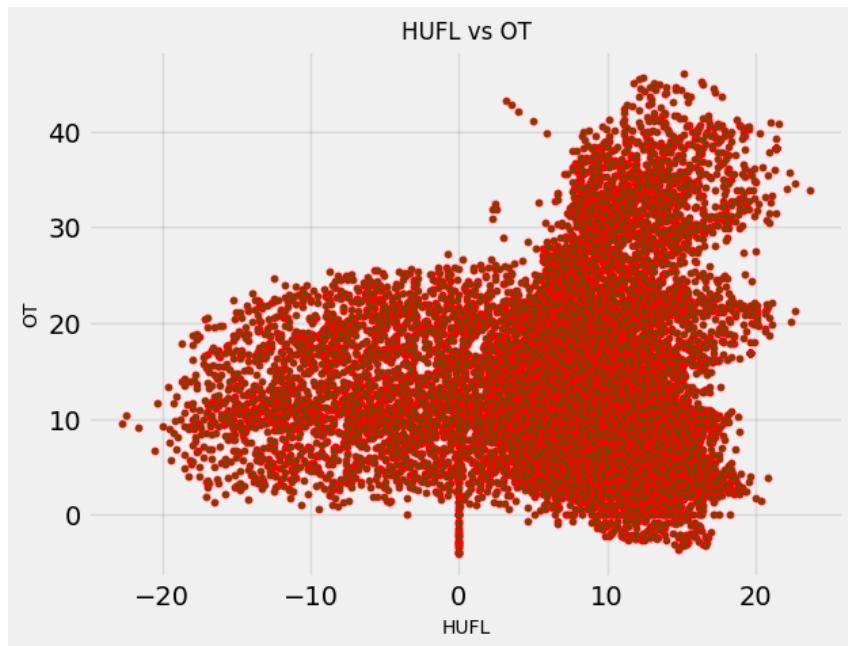
Hình 10: Heatmap với Seaborn

```

1 import matplotlib.pyplot as plt
2 import pandas as pd
3 import numpy as np
4 plt.scatter(df.HUFL, df.OT, marker = "o",
5             color = "green", linewidths = 1, edgecolors = "red", s = 10)
6 plt.style.use('fivethirtyeight')
7 plt.xlabel("HUFL", size = 10, color = "black")
8 plt.ylabel("OT", size = 10, color = "black")
9 plt.title("HUFL vs OT", size =12, color = "black")
10 plt.xticks(color = "black")
11 plt.yticks(color = "black")
12 plt.grid(color = "grey", alpha = 0.2)
13 plt.show()

```

Listing 8: Biểu đồ Scatter với Matplotlib



Hình 11: Scatterplot với Matplotlib

4.2 Trường Hợp Nghiên Cứu: Bộ Dữ Liệu Hiệu Suất Học Sinh (Student Performance Dataset)

4.2.1 Biểu Đồ Cột (Bar Chart)

Biểu đồ cột được sử dụng để hiển thị sự thay đổi giá trị của một biến cụ thể so với biến khác.

```

1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 grouped_data = df_st.groupby(['race/ethnicity', 'gender']).size().unstack()
5 grouped_data
6
7 fig = plt.figure(figsize=(4, 5))
8 plt.style.use('fivethirtyeight')
9 plt.bar(
10     x=grouped_data.index,
11     height=grouped_data["female"],
12     label="Female",
13     color='orange',      # Set the color of the bars
14     edgecolor='black',   # Set the color of the bar edges
15     linewidth=1.5,       # Set the width of the bar edges
16     alpha=0.8            # Set the transparency of the bars
17 )
18
19 plt.bar(
20     x=grouped_data.index,
21     height=grouped_data["male"],
22     bottom=grouped_data["female"],
23     label="Male",
24     color='blue',        # Set the color of the bars
25     edgecolor='black',   # Set the color of the bar edges
26     linewidth=1.5,       # Set the width of the bar edges
27     alpha=0.8            # Set the transparency of the bars

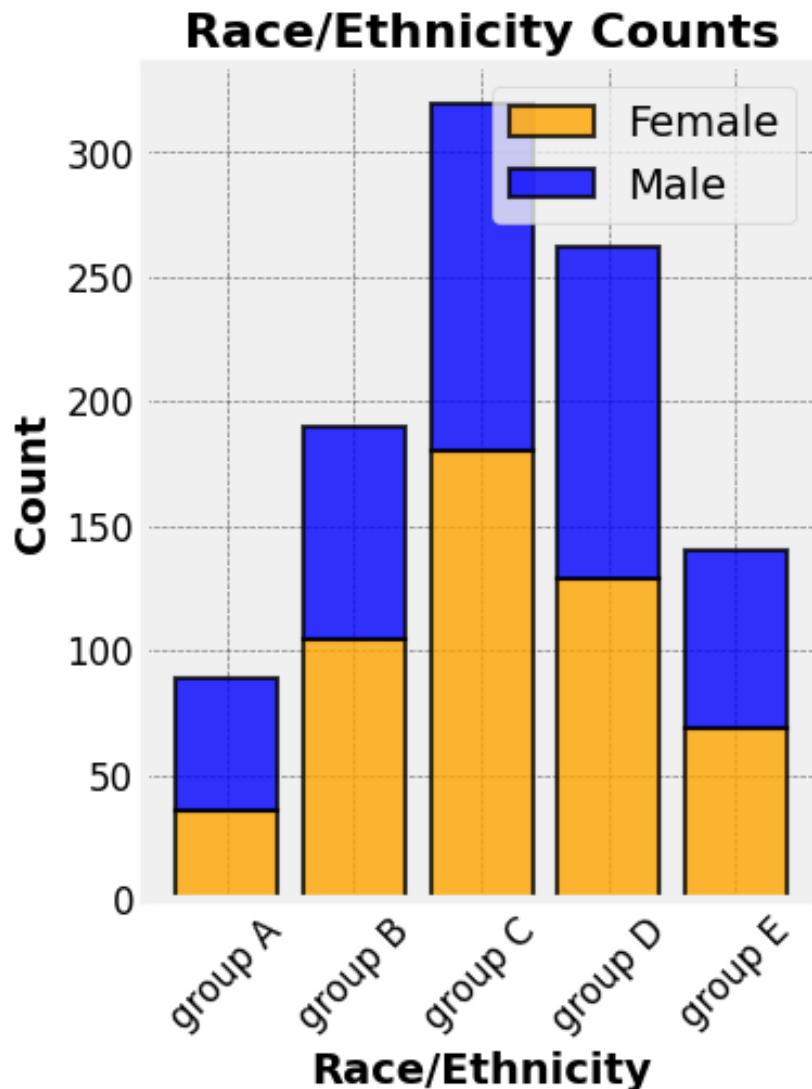
```

```

28 )
29
30 plt.xlabel('Race/Ethnicity', fontsize=14, fontweight='bold') # Set the x-
31 axis label with font size and style
32 plt.ylabel('Count', fontsize=14, fontweight='bold') # Set the y-
33 axis label with font size and style
34 plt.title('Race/Ethnicity Counts', fontsize=16, fontweight='bold') # Set
35 the chart title with font size and style
36 plt.xticks(fontsize=12, rotation = 45) # Set the font size of the x-axis
37 tick labels
38 plt.yticks(fontsize=12) # Set the font size of the y-axis tick labels
39 plt.grid(True, linestyle='--', linewidth=0.5, alpha=0.5, color = "black")# Display gridlines with a dashed style and reduced opacity
40 plt.legend()
41
42 plt.show()

```

Listing 9: Điểm Toán theo Trình độ học vấn của Phụ huynh



Hình 12: Bar Chart với Matplotlib (StudentsPerformance dataset)

4.2.2 Biểu Đồ Đếm (Count Plot)

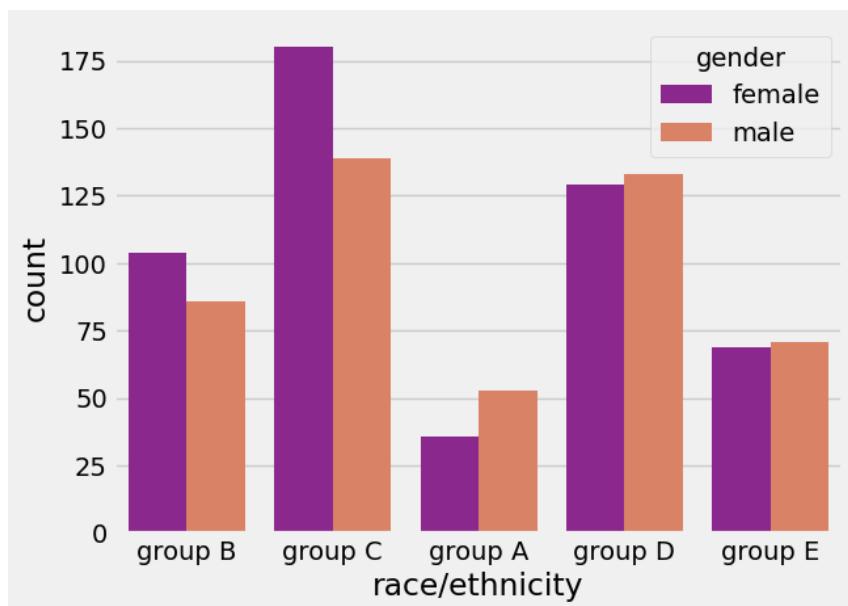
Biểu đồ đếm được sử dụng để hiển thị số lượng quan sát của một biến phân loại trong một bộ dữ liệu.

```

1 import pandas as pd
2 import seaborn as sns
3 import matplotlib.pyplot as plt
4 # Seaborn
5 sns.countplot(data = df_st, x = "race / ethnicity", hue = "gender",
6 palette = "plasma")

```

Listing 10: Số lượng Học sinh theo Giới tính



Hình 13: Count Chart với Seaborn (StudentsPerformance dataset)

4.3 Trường Hợp Nghiên Cứu: Bộ Dữ Liệu Iris

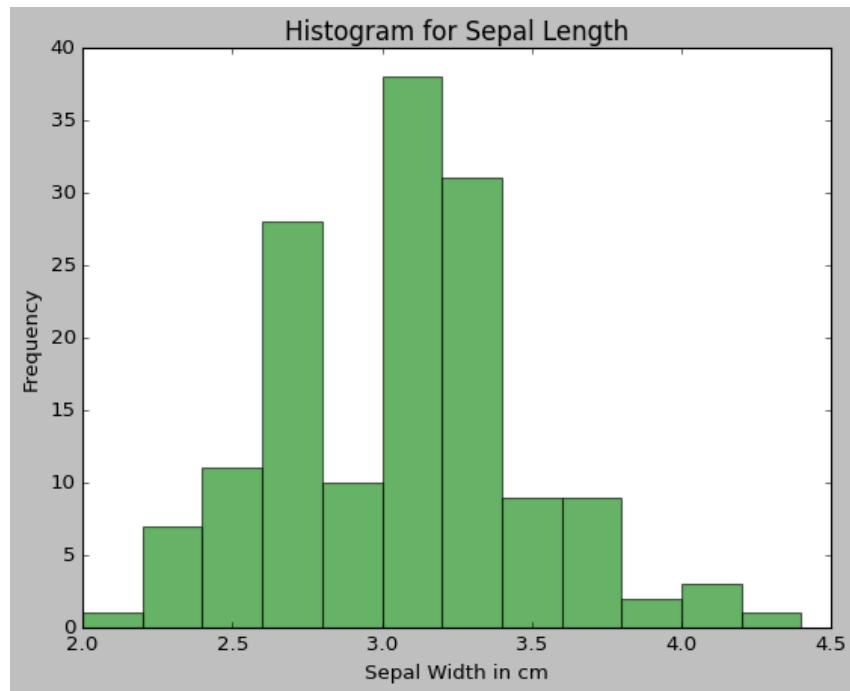
4.3.1 Biểu Đồ Tần Suất (Histogram)

Biểu đồ tần suất được sử dụng để quan sát phân phối tần suất của một biến.

```

1 import seaborn as sns
2 import matplotlib.pyplot as plt
3 iris = pd.read_csv("/content/Iris.csv")
4
5 # If bins of specific width are needed
6 bin_width = 0.2
7 bins = int((iris.SepalWidthCm.max() - iris.SepalWidthCm.min()) / bin_width)
8 plt.style.use('classic')
9 plt.hist(iris.SepalWidthCm, bins = bins, color = "green", alpha = 0.6,
10         orientation = "vertical", rwidth = 1)
11 plt.xlabel("Sepal Width in cm", size = 12, color = "black")
12 plt.ylabel("Frequency", size = 12, color = "black")
13 plt.title("Histogram for Sepal Length", size = 15, color = "black")
14 plt.xticks(color = "black")
15 plt.yticks(color = "black")

```



Hình 14: Histogram với Matplotlib (Iris dataset)

16 plt.show()

Listing 11: Phân phối Chiều dài Đài hoa

4.3.2 Biểu đồ bong bóng(Bubble Chart)

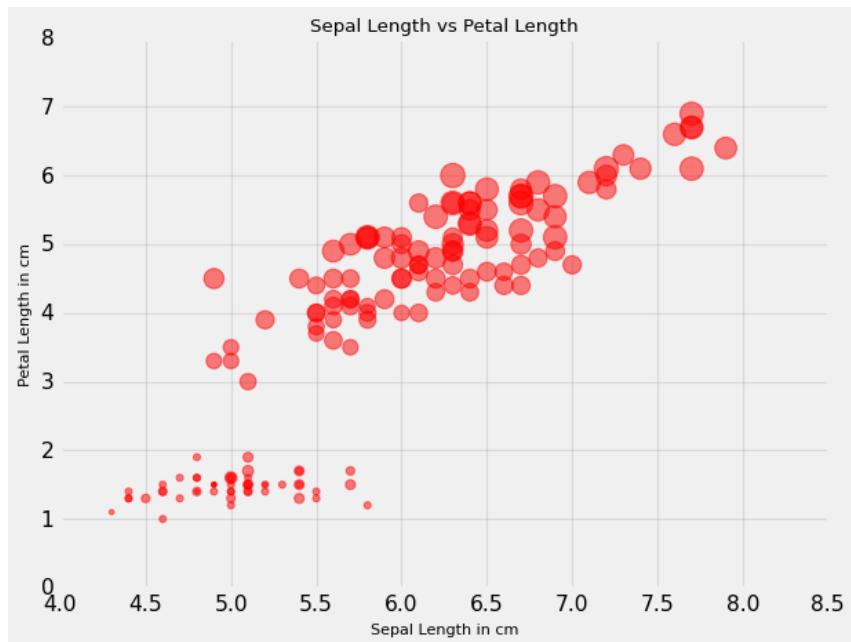
Biểu đồ Bubble mở rộng scatter plot bằng cách thêm một chiều thông qua kích thước điểm, giúp hiển thị quan hệ giữa ba biến và rất hữu ích trong trực quan hóa đa biến.

```

1 import matplotlib.pyplot as plt
2 from sklearn.preprocessing import MinMaxScaler
3
4 sizes = iris['PetalWidthCm']/iris['PetalWidthCm'].max()
5
6 # Here, the size of the bubble will show the Petal Width
7
8 plt.figure()
9 plt.scatter(iris.SepalLengthCm, iris.PetalLengthCm, marker = "o",
10             color = "red", linewidths = 1, edgecolors = "red",
11             s= sizes*250, alpha = 0.5)
12 plt.style.use('fivethirtyeight')
13 plt.xlabel("Sepal Length in cm", size = 10, color = "black")
14 plt.ylabel("Petal Length in cm", size = 10, color = "black")
15 plt.title("Sepal Length vs Petal Length", size =12, color = "black")
16 plt.xticks(color = "black")
17 plt.yticks(color = "black")
18 plt.grid(color = "grey", alpha = 0.2)
19 plt.show()

```

Listing 12: Bubble Chart



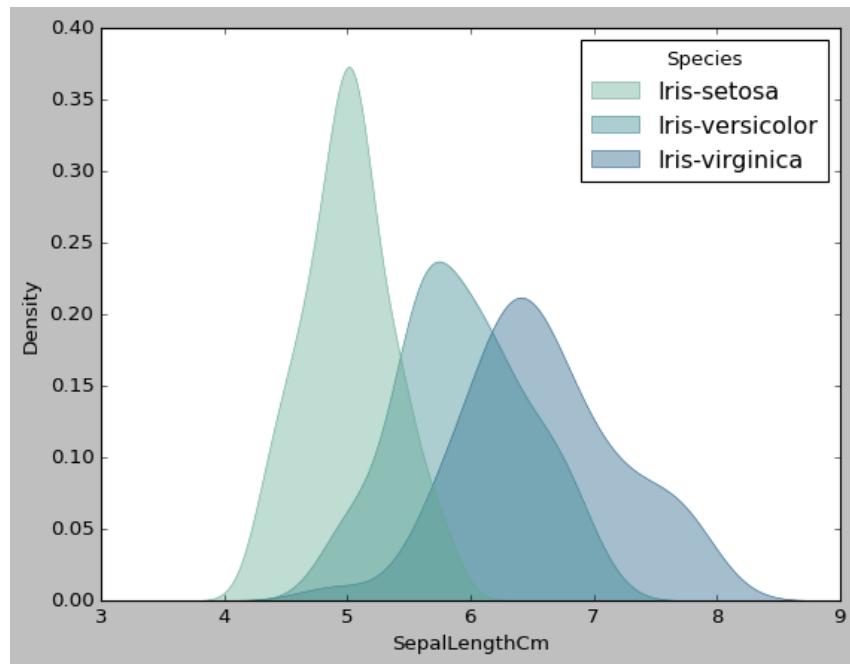
Hình 15: Bubble Chart với Matplotlib (Iris dataset)

4.3.3 Biểu đồ mật độ xác suất(KDE Plot)

Biểu đồ KDE ước lượng mật độ phân bố xác suất liên tục một cách mượt mà hơn histogram, giúp dễ dàng so sánh nhiều nhóm trên cùng một biểu đồ.

```
1 import matplotlib.pyplot as plt
2 import seaborn as sns
3 sns.kdeplot(x=iris["SepalLengthCm"], hue = iris["Species"],
4               linewidth = 0.5, fill = True, multiple = "layer", cbar = True,
5               palette = "crest", alpha = 0.4)
```

Listing 13: Trực quan hóa 3D của Bộ dữ liệu Iris



Hình 16: KDE Plot với Seaborn (Iris dataset)

4.3.4 Distribution Plot(DisPlot)

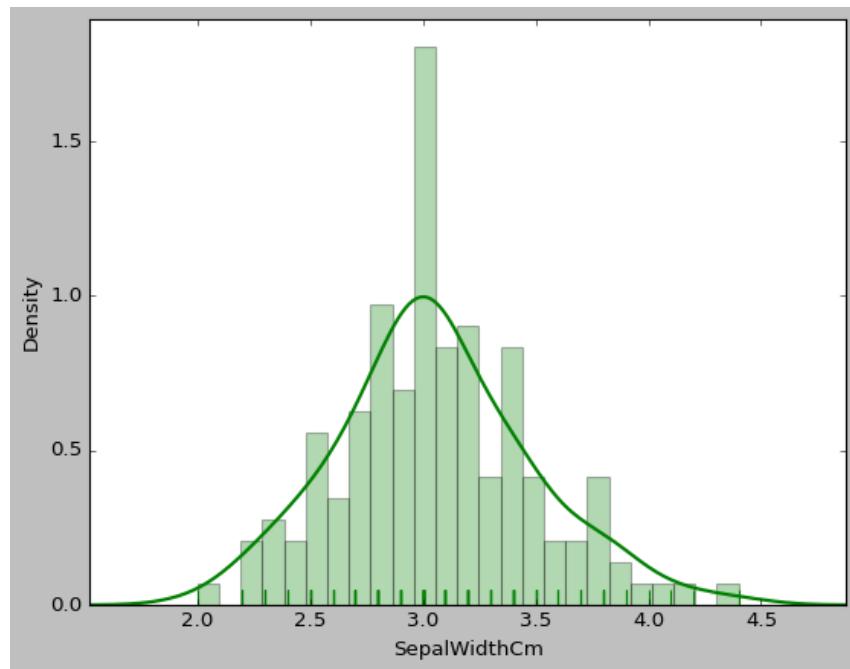
Biểu đồ DisPlot kết hợp giữa histogram và KDE, giúp thể hiện đồng thời tần suất và mật độ xác suất của một biến liên tục.

```

1 import matplotlib.pyplot as plt
2 import seaborn as sns
3 sns.distplot(iris["SepalWidthCm"], bins = 25, kde = True,
4             rug = True, color = "green", hist_kws = {"alpha":0.3},
5             kde_kws = {"linewidth":2})

```

Listing 14: Trực quan hóa 3D của Bộ dữ liệu Iris



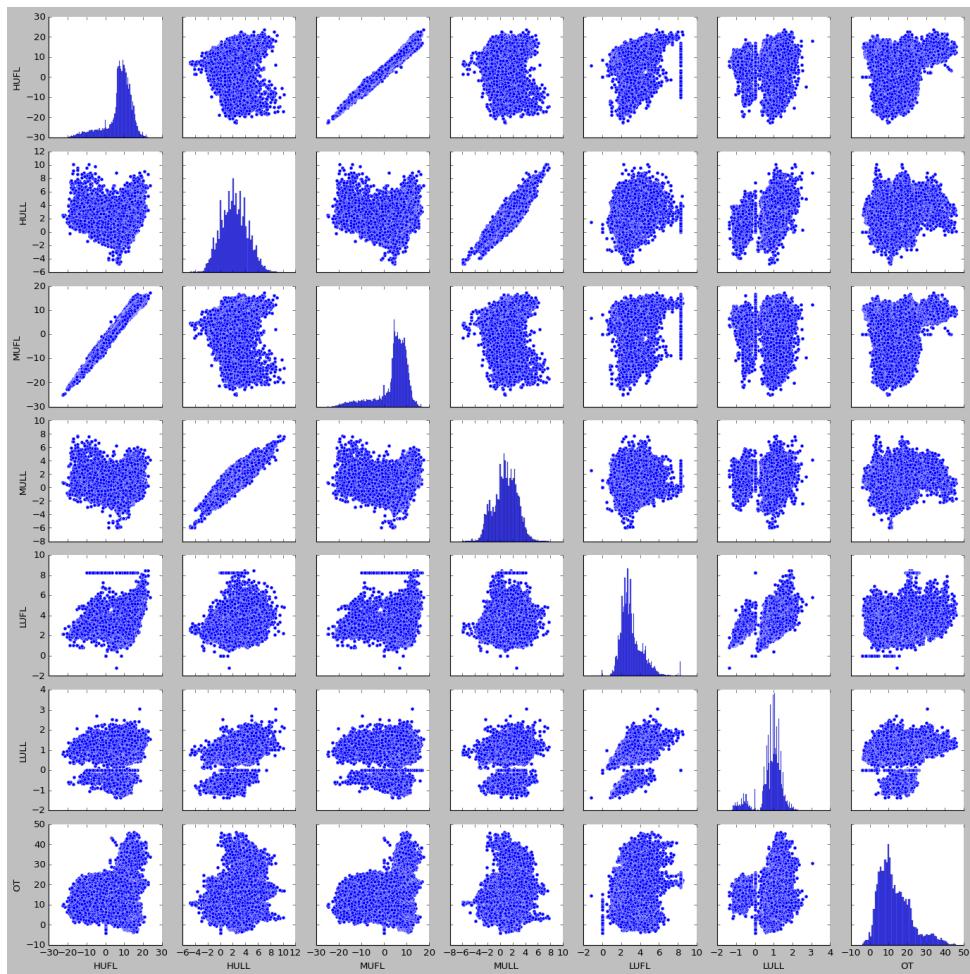
Hình 17: DisPlot với Seaborn (Iris dataset)

4.3.5 Biểu đồ cặp(Pairplot)

Biểu đồ pairplot tự động hiển thị phân tán cho mọi cặp biến và phân phối đơn biến trên đường chéo, giúp khám phá nhanh mối quan hệ giữa các biến trong tập dữ liệu.

```
1 import matplotlib.pyplot as plt
2 import seaborn as sns
3 sns.pairplot(df)
```

Listing 15: Trực quan hóa 3D của Bộ dữ liệu Iris



Hình 18: Pairplot với Seaborn (Iris dataset)

4.3.6 Trục Quan Hóa 3D (3D Visualization)

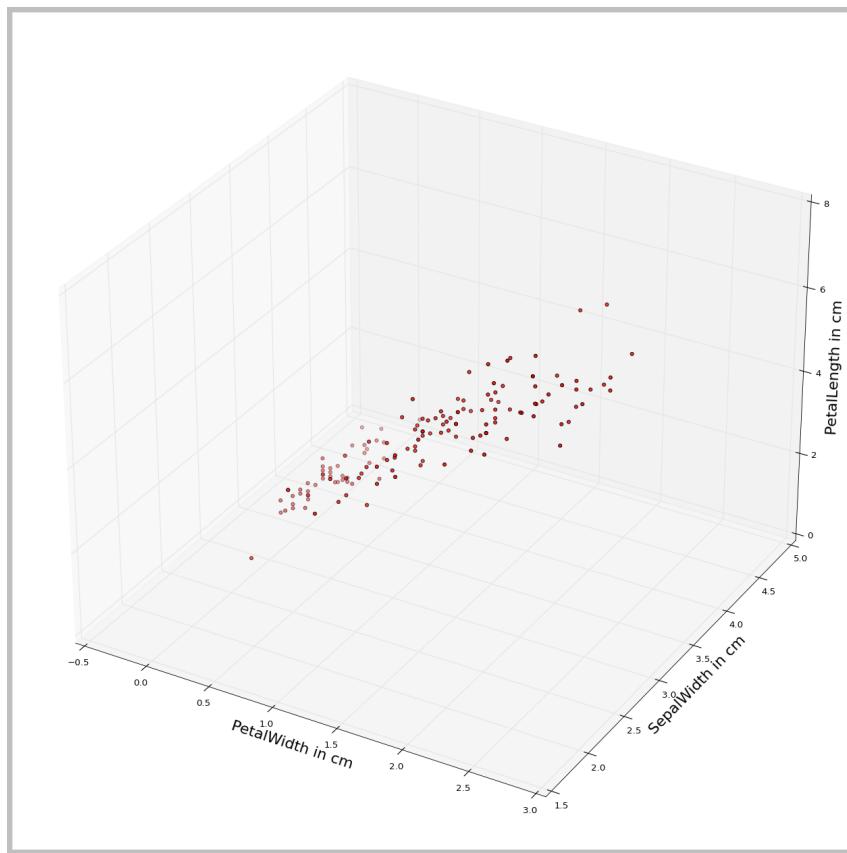
Biểu đồ 3D Scatter hiển thị mối quan hệ giữa ba biến định lượng trong không gian ba chiều, rất hữu ích để kiểm tra khả năng phân tách trực quan khi dữ liệu có nhiều biến.

```

1 import matplotlib.pyplot as plt
2 from mpl_toolkits.mplot3d import Axes3D
3 from mpl_toolkits.mplot3d import Axes3D
4
5 fig = plt.figure(figsize = (20,20))
6 plt.style.use("classic")
7 ax = fig.add_subplot(111, projection='3d')
8 ax.scatter(iris["PetalWidthCm"], iris["SepalWidthCm"], iris["PetalLengthCm"],
9           c = "red")
10 ax.set_xlabel("PetalWidth in cm", fontsize = 20)
11 ax.set_ylabel("SepalWidth in cm", fontsize = 20)
12 ax.set_zlabel("PetalLength in cm", fontsize = 20)

```

Listing 16: Trục quan hóa 3D của Bộ dữ liệu Iris



Hình 19: 3D Visualization với Matplotlib (Iris dataset)

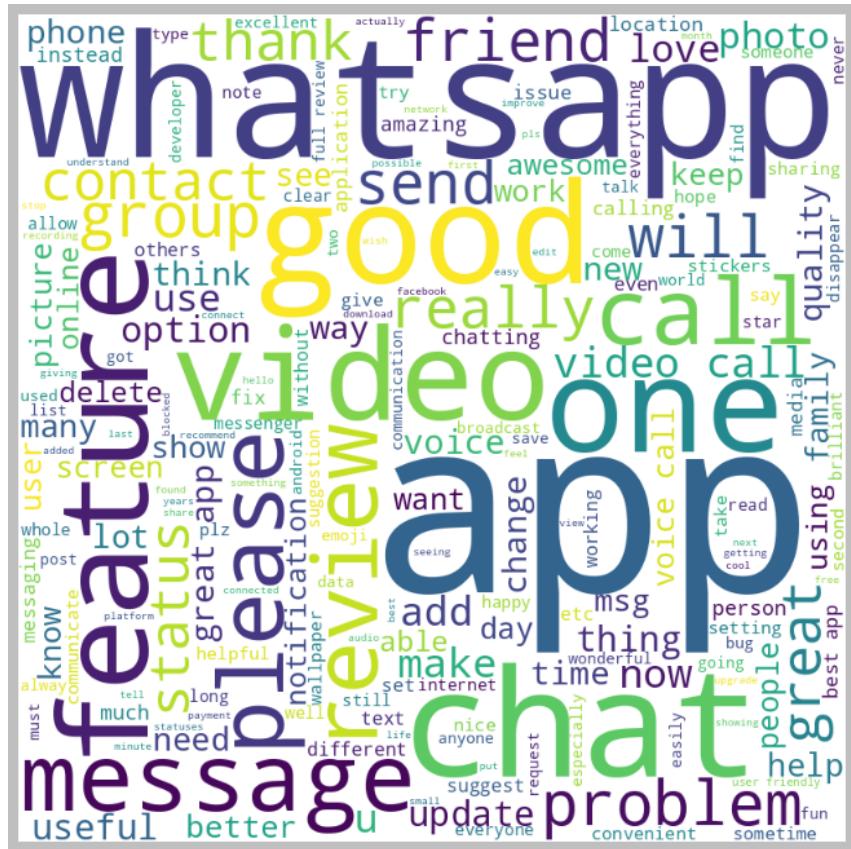
5. Các Biểu Đồ Khác: Word Cloud

Biểu đồ Word Cloud cung cấp thông tin về ngữ cảnh của dữ liệu. Kích thước của các từ khác nhau thể hiện tần suất hoặc tầm quan trọng của những từ đó.

```
1 from wordcloud import WordCloud, STOPWORDS
2 import matplotlib.pyplot as plt
3 import pandas as pd
4
5 stopwords = set(STOPWORDS)
6 words = ''
7 for review in df_fs.Review:
8     tokens = str(review).split()
9     tokens = [i.lower() for i in tokens]
10    words += ' '.join(tokens) + ' '
11
12 wordcloud = WordCloud(width = 800, height = 800,
13                         background_color ='white',
14                         stopwords = stopwords,
15                         min_font_size = 10).generate(words)
16
17 # plot the WordCloud image
18 plt.figure(figsize = (8, 8), facecolor = None)
19 plt.imshow(wordcloud)
20 plt.axis("off")
21 plt.tight_layout(pad = 0)
```

23 | plt.show()

Listing 17: Word Cloud: Đánh giá 5 sao cho ứng dụng



Hình 20: WordCloud (five star reviews dataset)

6. Mẹo Chọn Biểu Đồ Phù Hợp (Tổng hợp)

Để chọn đúng biểu đồ, hãy xem xét mục đích của trực quan hóa:

- **Hiển thị sự thay đổi theo thời gian:** Biểu đồ đường (Line Chart), Biểu đồ vùng (Area Chart).
 - **Hiển thị thành phần part-to-whole:** Biểu đồ tròn (Pie Chart), Biểu đồ Donut (Donut Chart).
 - **Cách dữ liệu được phân phối:** Biểu đồ tần suất (Histogram), Biểu đồ hộp (Box Plot).
 - **So sánh giá trị giữa các nhóm:** Biểu đồ cột (Bar Chart), Biểu đồ thanh (Column Chart).
 - **Quan sát mối quan hệ giữa các biến:** Biểu đồ phân tán (Scatter Plot), Biểu đồ bong bóng (Bubble Chart), Bản đồ nhiệt (Heatmap).
 - **Trình bày tính năng nổi bật từ văn bản:** Biểu đồ Word Cloud.

Kết luận

Qua các trường hợp nghiên cứu và biểu đồ đã trình bày, chúng ta thấy rằng việc chọn đúng biểu đồ và thư viện trực quan hóa là yếu tố then chốt để hiểu rõ dữ liệu và truyền tải thông tin một cách hiệu quả. Python cung cấp công cụ đa dạng, từ các thư viện cơ bản như Matplotlib đến các thư viện hiện đại như Plotly và Seaborn, giúp người dùng tạo nên những trực quan hóa đẹp mắt và giàu thông tin.

Tìm hiểu về ETL Pipeline: Từ dữ liệu thô đến Thông tin giá trị

Vũ Thái Sơn

Hành trình khám phá cách dữ liệu được xử lý và biến đổi thành thông tin hữu ích trong kỷ nguyên AI

1. Giới thiệu: Khám phá ETL Pipeline - Nền tảng của Dữ liệu thông minh

Bạn có bao giờ tự hỏi làm thế nào các công ty lớn có thể tổng hợp và phân tích hàng núi dữ liệu từ đủ mọi nguồn để đưa ra những quyết định kinh doanh "đắt giá"? Bí mật nằm ở một kỹ thuật quan trọng trong ngành dữ liệu: **ETL Pipeline** (Extract, Transform, Load).

Hãy tưởng tượng một công ty bán lẻ với các bộ phận khác nhau như Marketing, Vận hành, Tài chính, và Bán hàng, mỗi bộ phận lại lưu trữ dữ liệu riêng trên các hệ thống khác nhau như MySQL, Supabase hay file Excel. Khi các nhà quản lý cần cái nhìn tổng thể về hiệu suất kinh doanh (ví dụ: xu hướng hành vi khách hàng trong 2 năm qua, tổng doanh thu tháng trước), họ phải đối mặt với nhiều thách thức:

- **Tốn thời gian và công sức:** Tổng hợp dữ liệu từ nhiều nguồn khác nhau là một quá trình thủ công, mất rất nhiều thời gian và nguồn lực.
- **Dễ sai sót và không nhất quán:** Sự tham gia của con người trong quá trình tổng hợp thủ công dễ dẫn đến sai sót và dữ liệu không đồng bộ.
- **Khó có báo cáo tổng hợp:** Mỗi nguồn dữ liệu có định dạng và thuật ngữ khác nhau, gây khó khăn trong việc tạo ra một báo cáo tích hợp, toàn diện.
- **Thông tin không đáng tin cậy:** Nguy cơ số liệu không khớp hoặc thông tin sai lệch có thể xảy ra, làm giảm độ tin cậy của các quyết định dựa trên dữ liệu.

Để giải quyết những vấn đề này và cho phép đưa ra các quyết định dựa trên dữ liệu (**Data-driven decision making**), chúng ta cần một giải pháp tập trung, đáng tin cậy và có khả năng mở rộng để sử dụng dữ liệu hiệu quả. Đây chính là lúc ETL Pipeline phát huy vai trò của mình.

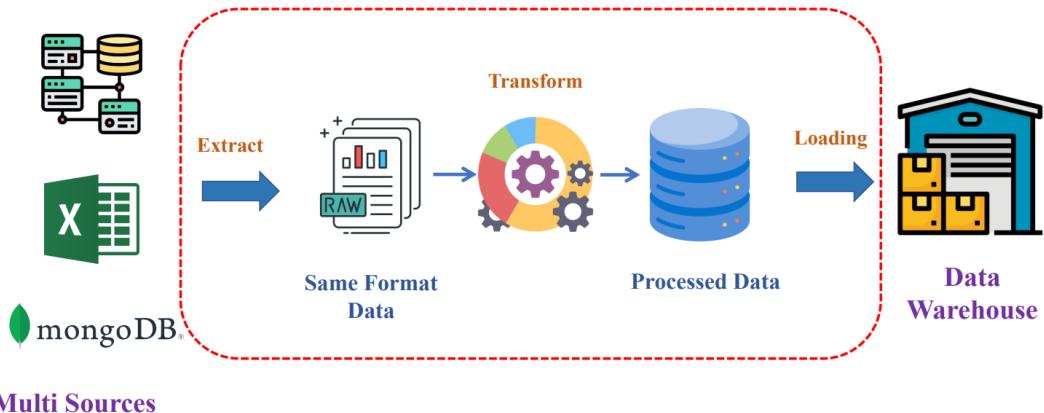
1.1 ETL Pipeline là gì?

ETL là viết tắt của ba giai đoạn chính trong quá trình di chuyển và xử lý dữ liệu:

- **Extract (Trích xuất):** Thu thập dữ liệu thô từ các nguồn khác nhau.
- **Transform (Chuyển đổi):** Làm sạch, chuẩn hóa, và biến đổi dữ liệu để phù hợp với mục đích phân tích.
- **Load (Tải):** Chuyển dữ liệu đã được xử lý vào một kho lưu trữ cuối cùng, sẵn sàng cho việc phân tích và báo cáo.

Hãy hình dung ETL Pipeline như một "nhà bếp" trong một nhà hàng lớn, nơi dữ liệu thô được biến thành món ăn ngon, sẵn sàng phục vụ thực khách (người dùng phân tích).

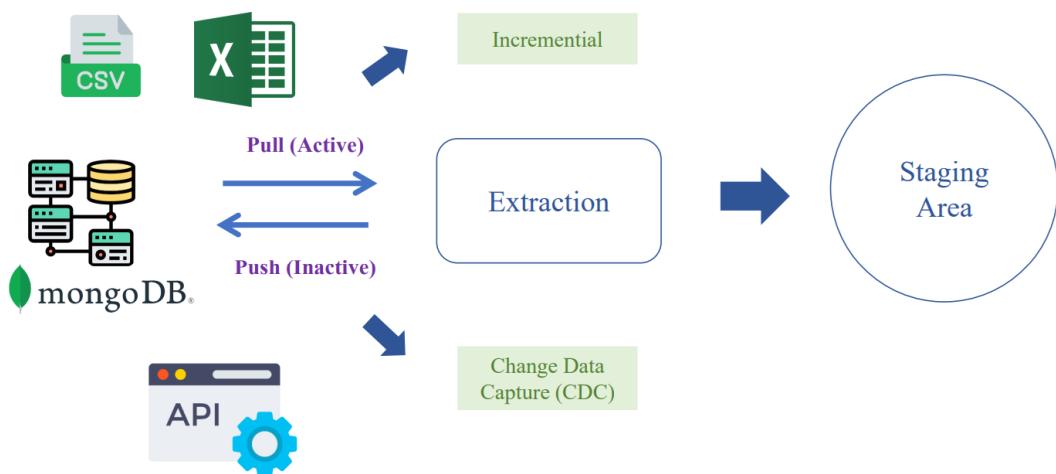
❖ ETL process



Hình 21: Tổng quan quy trình ETL.

2. Data Extraction - "Thu thập nguyên liệu"

Giai đoạn **Data Extraction** là bước đầu tiên và quan trọng nhất, nơi chúng ta "thu thập nguyên liệu" (dữ liệu thô) từ mọi ngóc ngách khác nhau. Dữ liệu này có thể đến từ các cơ sở dữ liệu (ví dụ: MySQL, MongoDB, Supabase), các tệp tin (ví dụ: CSV, Excel), hoặc thông qua các API của các ứng dụng. Mục tiêu là đưa tất cả dữ liệu thô này về một khu vực lưu trữ tạm thời, được gọi là **Staging Area**, trước khi đi vào các bước xử lý sâu hơn.



Hình 22: Quá trình Data Extraction.

2.1 Các phương pháp trích xuất dữ liệu

Có hai cách tiếp cận chính để lấy dữ liệu:

- **Pull (Kéo):** Hệ thống của chúng ta chủ động gửi yêu cầu để lấy dữ liệu từ nguồn. Đây là phương pháp phổ biến khi làm việc với các cơ sở dữ liệu.
- **Push (Đẩy):** Nguồn dữ liệu tự động gửi dữ liệu về cho hệ thống của chúng ta khi có sự kiện hoặc theo lịch trình định sẵn.

2.2 Các kiểu trích xuất dữ liệu

Để tối ưu hiệu suất và tài nguyên, chúng ta cần cân nhắc kiểu trích xuất:

- **Full Extraction (Trích xuất toàn bộ):** Lấy toàn bộ dữ liệu từ nguồn mỗi lần chạy. Phương pháp này đơn giản nhưng tốn kém tài nguyên nếu dữ liệu lớn.
- **Incremental Extraction (Trích xuất tăng dần):** Chỉ lấy những dữ liệu mới được thêm vào hoặc đã thay đổi kể từ lần trích xuất cuối cùng. Để thực hiện điều này, chúng ta thường sử dụng kỹ thuật **Change Data Capture (CDC)**. CDC giúp theo dõi và ghi lại các thay đổi trong dữ liệu nguồn.

2.3 Vấn đề thường gặp trong quá trình trích xuất: Xung đột khóa tổ hợp

Khi thu thập dữ liệu từ nhiều nguồn khác nhau, một vấn đề phổ biến là **Composite Key Collisions** (xung đột khóa tổ hợp). Điều này xảy ra khi các bản ghi từ các nguồn khác nhau có cùng một khóa nhưng lại đại diện cho các thực thể khác nhau, hoặc cùng một thực thể nhưng có sự mâu thuẫn trong dữ liệu.

Ví dụ, hai cửa hàng khác nhau có thể cùng có một giao dịch với cùng một transac_id (ID giao dịch). Nếu chỉ dựa vào transac_id, chúng ta sẽ không thể phân biệt được giao dịch nào thuộc về cửa hàng nào.

```
-- Data from Store A
"store_id", "transac_id", "amount", "cost"
"st00a", "tran001", "1", "2"
"st00a", "tran002", "3", "2.3"
"st00a", "tran003", "2", "1.9"

-- Data from Store B
"store_id", "transac_id", "amount", "cost"
"st00b", "tran001", "1", "2"
"st00b", "tran002", "3", "2.3"
"st00b", "tran003", "2", "1.9"
```

Hình 23: Example of composite key collision from different sources.

Khi gộp hai bảng này lại mà không xử lý, chúng ta sẽ có các bản ghi trùng lặp transac_id như sau:

```
"transac_id", "store_id", "amount", "cost"
"tran001", "st00a", "1", "2"
"tran001", "st00b", "3", "2.3" -- Note: "amount" value may differ if it's a
different transaction
```

Hình 24: Table after merging without handling collisions.

Để giải quyết, chúng ta cần xem xét (store_id, transac_id) như một khóa duy nhất. Các vấn đề này cần được xử lý ở bước Transform hoặc trong chiến lược tải dữ liệu.

2.4 Case Study: Trích xuất dữ liệu bán rượu Iowa Liquor Sale

Chúng ta sẽ minh họa quá trình trích xuất bằng bộ dữ liệu "Iowa Liquor Sales". Bộ dữ liệu này chứa thông tin chi tiết về việc mua rượu của các cửa hàng được cấp phép tại Iowa, rất hữu ích cho việc phân tích xu hướng bán hàng.

Bộ dữ liệu này có khoảng 1 triệu bản ghi và 24 cột khác nhau, bao gồm các thông tin về hóa đơn, ngày, cửa hàng, sản phẩm, giá cả, số lượng bán, và thông tin địa lý.

Trong thực tế, bộ dữ liệu này có thể được cung cấp dưới dạng nhiều file CSV nhỏ. Để xử lý hiệu quả lượng dữ liệu lớn từ nhiều file, chúng ta thường sử dụng phương pháp **Batch Processing** (xử lý theo lô). Dữ liệu từ các file CSV sẽ được đọc từng phần (ví dụ: 100.000 dòng hoặc 10.000 dòng một lần) và sau đó được chuyển vào một cơ sở dữ liệu tạm thời như SQLite, đóng vai trò là Staging Area.

```
1 DROP TABLE IF EXISTS Staging_Sales;
2 CREATE TABLE Staging_Sales (
3     invoice_line_no TEXT,
4     date TEXT,
5     store TEXT,
6     name TEXT,
7     address TEXT,
8     city TEXT,
9     zipcode TEXT,
10    store_location TEXT,
11    county_number TEXT,
12    county TEXT,
13    category TEXT,
14    category_name TEXT,
15    vendor_no TEXT,
16    vendor_name TEXT,
17    itemno TEXT,
18    im_desc TEXT,
19    pack TEXT,
20    bottle_volume_ml TEXT,
21    state_bottle_cost TEXT,
22    state_bottle_retail TEXT,
23    sale_bottles TEXT,
24    sale_dollars TEXT,
25    sale_liters TEXT,
26    sale_gallons TEXT,
27    file_name TEXT NOT NULL,
28    processed_timestamp DATETIME NOT NULL,
29    record_source TEXT NOT NULL -- Add record_source field for more
30    detailed tracking
31 );
```

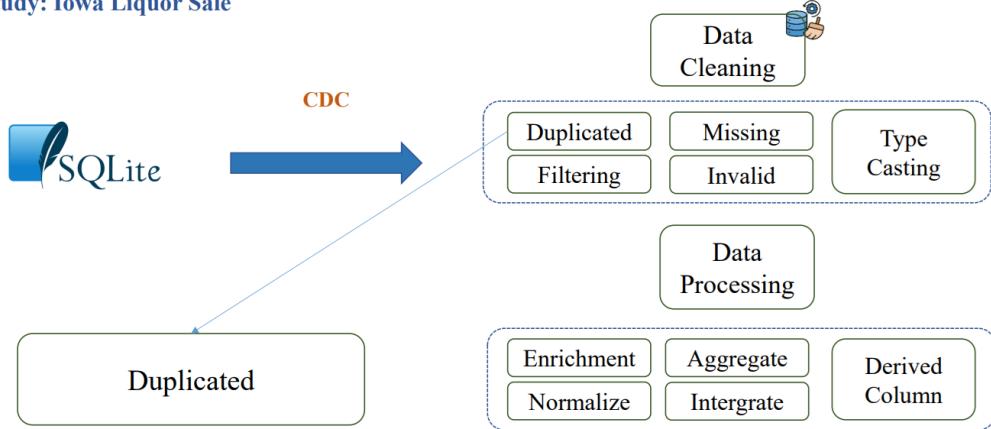
Hình 25: Example of creating Staging_Sales table in SQLite.

3. Data Transformation - "Sơ chế và chế biến nguyên liệu"

Sau khi "thu thập nguyên liệu" và đưa vào khu vực lưu trữ tạm thời (**Staging Area**), bước tiếp theo là **Data Transformation**. Đây là giai đoạn quan trọng nhất của ETL, nơi dữ liệu thô được

”sơ chế” và ”chế biến” thành dạng sạch sẽ, có cấu trúc và sẵn sàng cho việc phân tích. Giống như việc bạn rửa sạch, gọt vỏ, cắt thái, và tẩm ướp gia vị cho rau củ và thịt cá để chuẩn bị cho món ăn ngon.

Case Study: Iowa Liquor Sale



Mục tiêu chính của Data Transformation là đảm bảo chất lượng dữ liệu cao, đồng nhất và phù hợp với các mục đích phân tích sau này.

3.1 Các hoạt động chính trong Data Transformation

1. **Data Cleaning (Làm sạch dữ liệu):** Đây là công đoạn đầu tiên và thiết yếu, giống như việc loại bỏ những phần rau củ bị hỏng, úa.

- **Xử lý dữ liệu trùng lặp (Duplicated data):**

- **Vấn đề:** Trong quá trình trích xuất, đặc biệt khi lấy dữ liệu từ nhiều nguồn hoặc theo thời gian, có thể xuất hiện các bản ghi hoàn toàn giống nhau hoặc các bản ghi chỉ khác nhau ở một vài thông tin nhỏ (ví dụ, thời gian cập nhật của trường store_location).
- **Giải pháp:**
 - Loại bỏ trùng lặp hoàn toàn: Nếu các bản ghi giống hệt nhau, chúng ta sẽ giữ lại một bản và xóa các bản còn lại.
 - Xử lý bản ghi cập nhật: Trong trường hợp dữ liệu có cùng khóa chính (ví dụ invoice_line_no và store) nhưng một số trường bị thay đổi (ví dụ, store_location được cập nhật), chúng ta cần xác định đâu là bản ghi mới nhất hoặc chính xác nhất để giữ lại. Một chiến lược phổ biến là giữ bản ghi đầu tiên hoặc cuối cùng khi loại bỏ trùng lặp dựa trên một tập hợp các cột xác định.

- **Xử lý dữ liệu thiếu (Missing data - Null values):**

- **Vấn đề:** Nhiều cột dữ liệu có thể bị thiếu thông tin (ví dụ, address, city, zipcode, store_location, county_number, county, category, category_name, state_bottle_cost, state_bottle_retail, sale_bottles, sale_dollars, sale_liters, sale_gallons trong bộ dữ liệu Iowa Liquor Sale có thể có giá trị null).

- **Giải pháp:**

- Điền giá trị mặc định: Đối với các trường như địa chỉ, bạn có thể điền "Unknown" (Không xác định) thay vì để trống.
- Xóa bỏ bản ghi: Nếu một số cột quan trọng (ví dụ: các cột liên quan đến doanh số như state_bottle_cost, state_bottle_retail, sale_bottles, sale_dollars, sale_liters, sale_gallons) bị thiếu, và việc điền dữ liệu không khả thi hoặc có thể làm sai lệch kết quả, chúng ta có thể xóa các bản ghi đó.
- **Xử lý dữ liệu không hợp lệ (Invalid data - Outliers, incorrect formats):**
 - **Vấn đề:** Dữ liệu có thể không tuân theo các quy tắc nghiệp vụ hoặc có định dạng sai (ví dụ, giá trị âm cho số lượng bán, hoặc định dạng ngày tháng không đúng). **Giải pháp:**
 - Lọc (Filtering): Loại bỏ các bản ghi không hợp lệ theo logic nghiệp vụ (ví dụ: loại bỏ các giao dịch có số lượng hoặc giá trị doanh thu < 1, vì chúng không có ý nghĩa kinh doanh).
 - Chuyển đổi kiểu dữ liệu (Type Casting): Đảm bảo các cột có đúng kiểu dữ liệu. Ví dụ, cột ngày (date) phải là kiểu ngày tháng, và các cột số (state_bottle_cost, state_bottle_retail, sale_bottles, sale_dollars, sale_liters, sale_gallons) phải là kiểu số để có thể thực hiện tính toán. Các giá trị không thể chuyển đổi sẽ được chuyển thành NaN (Not a Number), sau đó có thể xử lý như dữ liệu thiếu.

2. Data Processing (Xử lý dữ liệu): Sau khi làm sạch, dữ liệu sẽ được "chế biến" để tạo ra thông tin hữu ích hơn.

- **Tạo cột dẫn xuất (Derived Column):** Tính toán các cột mới từ các cột hiện có dựa trên logic nghiệp vụ. Ví dụ, từ sale_dollars và state_bottle_cost, bạn có thể tính profit (lợi nhuận), revenue (doanh thu), cost (chi phí), profit_margin (tỷ suất lợi nhuận), average_bottle_price (giá trung bình mỗi chai), volume_per_bottle_sold (thể tích mỗi chai đã bán).

```

1 df['revenue'] = df['sale_dollars']
2 df['cost'] = df['state_bottle_cost'] * df['sale_bottles']
3 df['profit'] = (df['state_bottle_retail'] - df['state_bottle_cost']) * df['sale_bottles']
4 df['total_bottles_sold'] = df['sale_bottles']
5 df['total_volume_sold_in_liters'] = df['sale_liters']
6 df['profit_margin'] = (df['profit'] / df['revenue'] * 100).
    round(2).where(df['revenue'] > 0, 0)
7 df['average_bottle_price'] = (df['sale_dollars'] / df['sale_bottles']).round(2).where(df['sale_bottles'] > 0, 0)
8 df['volume_per_bottle_sold'] = (df['sale_liters'] / df['sale_bottles']).round(2).where(df['sale_bottles'] > 0, 0)
9 derived_columns = ['revenue', 'cost', 'profit', 'total_bottles_sold', 'total_volume_sold_in_liters', 'profit_margin', 'average_bottle_price', 'volume_per_bottle_sold']
10 df_derived = df[derived_columns]
11 df_derived.head()

```

Listing 18: Example of creating derived columns in Python

- **Tổng hợp (Aggregate):** Gom nhóm dữ liệu và thực hiện các phép tính tổng hợp như tổng, trung bình, đếm, v.v. Ví dụ: Tổng doanh thu theo từng tháng, tổng số lượng sản phẩm bán ra của từng cửa hàng.
- **Chuẩn hóa (Normalize):** Tổ chức lại dữ liệu để giảm thiểu sự trùng lặp và cải thiện tính toàn vẹn của dữ liệu (ví dụ: tách các bảng lớn thành các bảng nhỏ hơn, liên kết với nhau bằng khóa).
- **Tích hợp (Integrate):** Kết hợp dữ liệu từ các nguồn khác nhau thành một cấu trúc thống nhất.

Sau bước Transformation này, dữ liệu của chúng ta đã trở nên gọn gàng, có ý nghĩa và sẵn sàng để được "tải" vào kho dữ liệu cuối cùng.

4. Data Loading - "Phục vụ món ăn"

Sau khi dữ liệu đã được Extract và Transform, bước cuối cùng là **Data Loading**. Đây là giai đoạn chuyển dữ liệu đã được xử lý vào nơi lưu trữ cuối cùng, thường là một **Data Warehouse** hoặc **Data Mart**. Việc này giống như đưa "món ăn đã chế biến sẵn" vào "khu vực trưng bày" của nhà hàng, nơi khách hàng (các nhà phân tích dữ liệu, hệ thống báo cáo) có thể dễ dàng tiếp cận và thưởng thức.

4.1 Data Warehouse và Data Mart: Kho chứa thông tin

Để tối ưu cho việc phân tích, dữ liệu trong Data Warehouse thường được tổ chức theo các mô hình chuyên biệt, với hai loại bảng chính:

1. Fact Tables (Bảng dữ kiện):

- Lưu trữ các dữ liệu định lượng, có thể đo lường được, liên quan đến các sự kiện kinh doanh. Đây là các "con số" chính mà bạn muốn phân tích.
- **Đặc điểm chính:** Chứa các số liệu đo lường được và bao gồm các khóa ngoại (Foreign Keys) liên kết đến các bảng chiều (Dimension Tables).
- **Ví dụ trong Iowa Liquor Sale:** Bảng sales_fact sẽ chứa các thông tin như revenue, profit, cost, total_bottles_sold, total_volume_sold_in_liters, profit_margin, average_bottle_price, volume_per_bottle_sold. Nó cũng sẽ có các khóa ngoại như date_key, store_key, item_key, vendor_key để liên kết đến các bảng chiều tương ứng.

2. Dimension Tables (Bảng chiều):

- Lưu trữ các thuộc tính mô tả, cung cấp ngữ cảnh cho các dữ kiện. Chúng trả lời các câu hỏi "ai, cái gì, ở đâu, khi nào".
- **Đặc điểm chính:** Chứa các cột mô tả, có một khóa chính (Primary Key), và thường có ít dòng hơn các bảng dữ kiện.
- **Ví dụ trong Iowa Liquor Sale:**
 - date_dim (chiều thời gian): Chứa date_key, date, year, month, day, quarter, weekday.
 - store_dim (chiều cửa hàng): Chứa store_key, store_id, address, city, zipcode, store_location, county_number, county.

- item_dim (chiều sản phẩm): Chứa item_key, itemno, im_desc, category, category_name, pack, bottle_volume_ml, state_bottle_cost, state_bottle_retail.
- vendor_dim (chiều nhà cung cấp): Chứa vendor_key, vendor_no, vendor_name.

4.2 Xử lý thay đổi dữ liệu theo thời gian: Slowly Changing Dimension (SCD)

Một vấn đề quan trọng khi tải dữ liệu vào Dimension Tables là làm thế nào để xử lý khi thông tin mô tả bị thay đổi. Ví dụ, một cửa hàng thay đổi địa chỉ. Có nhiều loại SCD, nhưng phổ biến nhất là:

1. SCD Type 1 - Overwrite (Ghi đè):

- Khi có thay đổi, bạn cập nhật trực tiếp bản ghi cũ bằng thông tin mới.
- **Ưu điểm:** Dễ thực hiện, không tốn không gian lưu trữ. **Nhược điểm:** Mất đi lịch sử dữ liệu. Bạn sẽ không biết địa chỉ cũ của cửa hàng là gì.

2. SCD Type 2 - Add New Row (Thêm hàng mới):

- Đây là phương pháp phổ biến và được khuyến nghị khi cần giữ lại lịch sử dữ liệu. Khi có thay đổi, bạn không cập nhật bản ghi cũ mà thêm một bản ghi mới với thông tin cập nhật. Bản ghi cũ sẽ được đánh dấu là không còn hoạt động.
- Để làm được điều này, các bảng chiều cần có thêm các cột theo dõi trạng thái lịch sử như start_date, end_date, và is_active.
- **Ưu điểm:** Giữ lại toàn bộ lịch sử thay đổi của dữ liệu, rất hữu ích cho các phân tích xu hướng hoặc so sánh theo thời gian. **Nhược điểm:** Tốn thêm không gian lưu trữ do có nhiều bản ghi cho cùng một thực thể.

• Quy trình tải SCD Type 2:

```

1 def process_scd_type2(df, dim_table, key_col, attributes, conn):
2     # Create a copy to avoid SettingWithCopyWarning
3     df = df.copy()
4     if current_dim.empty:
5         # Initial load: all records are new
6         df = df.assign(
7             start_date=datetime.now().date(),
8             end_date=None,
9             is_active=True
10        )
11     df.to_sql(dim_table, conn, if_exists='append', index=False)
12     return

```

Listing 19: Handling SCD Type 2 for initial load

```

1 # Find new and changed records
2 merged = df.merge(current_dim, on=key_col, how='outer', suffixes=('_new', '_current'), indicator=True)
3
4 # New records

```

```

5   new_records = merged[merged['_merge'] == 'left_only'][[key_col]
6     + attributes].copy()
7   if not new_records.empty:
8     new_records = new_records.assign(
9       start_date=datetime.now().date(),
10      end_date=None,
11      is_active=True
12    )
13   new_records.to_sql(dim_table, conn, if_exists='append',
14   index=False)
15
16 # Changed records
17 changed_records = merged[merged['_merge'] == 'both'].copy()
18 for attr in attributes:
19   changed_records = changed_records[changed_records[attr] !=
20   changed_records[f'{attr}_current']]
21
22 if not changed_records.empty:
23   # Expire old records
24   conn.execute(text("""
25     UPDATE {dim_table}
26     SET end_date = :end_date, is_active = 0
27     WHERE {key_col} = :key_val AND is_active = 1
28   """), [{ 'end_date': datetime.now().date(), 'key_val': row[key_col]} for _, row in changed_records.iterrows()])
29
30 # Insert new versions
31 new_versions = changed_records[[key_col] + attributes].copy()
32 new_versions = new_versions.assign(
33   start_date=datetime.now().date(),
34   end_date=None,
35   is_active=True
36 )
37 new_versions.to_sql(dim_table, conn, if_exists='append',
38 index=False)

```

Listing 20: Handling SCD Type 2 for subsequent loads

4.3 Sau khi dữ liệu được tải vào Data Warehouse: Bước tiếp theo

Dữ liệu đã được tải vào Data Warehouse hoặc Data Mart giờ đây đã sạch sẽ, có cấu trúc và sẵn sàng để khai thác giá trị. Từ đây, dữ liệu có thể được sử dụng cho các mục đích quan trọng như:

- **Phân tích dữ liệu (Data Analysis):** Sử dụng các công cụ và kỹ thuật như Machine Learning (ML) hoặc Trực quan hóa (Visualize) để khám phá các xu hướng, insight từ dữ liệu (ví dụ: biểu đồ lợi nhuận, số lượng chai bán theo tháng).
- **Báo cáo (Reporting):** Tạo ra các báo cáo định kỳ, Dashboard để theo dõi hiệu suất kinh doanh, giúp các nhà quản lý đưa ra quyết định nhanh chóng và chính xác.

Quá trình này được gọi là **Online Analytical Processing (OLAP)**, cho phép người dùng thực

hiện các truy vấn phức tạp và phân tích đa chiều trên dữ liệu.

5. Kết luận: ETL Pipeline - Trái tim của Dữ liệu thông minh

ETL Pipeline không chỉ là một chuỗi các bước kỹ thuật, mà nó là trái tim của bất kỳ hệ thống dữ liệu thông minh nào. Nắm vững ETL giúp bạn:

- **Hiểu sâu dữ liệu:** Không chỉ nhìn con số mà còn thấy câu chuyện phía sau.
- **Đưa ra quyết định thông minh:** Chọn đúng phương pháp xử lý dữ liệu cho từng tình huống.
- **Xây dựng hệ thống dữ liệu tốt hơn:** Nền tảng vững chắc dẫn đến các mô hình phân tích và báo cáo hiệu quả.

5.1 Bước tiếp theo

Để trở thành một chuyên gia trong lĩnh vực này, hãy:

1. Thực hành xây dựng các ETL Pipeline đơn giản với Python và các thư viện như Pandas, SQLAlchemy.
2. Tìm hiểu sâu hơn về các kiến trúc dữ liệu như Data Lake, Data Lakehouse.
3. Khám phá các công cụ ETL chuyên nghiệp trong công nghiệp (ví dụ: Apache Airflow, Talend, Microsoft SSIS).

Hãy nhớ: Mọi đột phá trong AI đều bắt đầu từ dữ liệu chất lượng cao, và ETL Pipeline chính là cầu nối để biến dữ liệu thô thành tài sản giá trị!

Kết Hợp Python và Excel Trong Phân Tích Dữ Liệu: Từ Cơ Bản Đến Ứng Dụng Nâng Cao

Đàm Nguyên Khánh

1. Mở đầu

Phân tích dữ liệu (*Data Analysis – DA*) đã trở thành kỹ năng quan trọng trong nhiều lĩnh vực — từ kinh doanh, tài chính đến y tế, giáo dục và sản xuất. Mục tiêu không chỉ là thu thập dữ liệu, mà còn biến dữ liệu thành thông tin có giá trị để hỗ trợ ra quyết định nhanh chóng và chính xác.

Hai công cụ phổ biến nhất hiện nay:

- **Excel**: Dễ dùng, trực quan, quen thuộc với người dùng văn phòng.
- **Python**: Linh hoạt, mạnh mẽ, phù hợp với xử lý dữ liệu lớn và tự động hóa.

Điểm đột phá là việc Microsoft tích hợp **Python trực tiếp vào Excel**, cho phép người dùng vừa tận dụng giao diện quen thuộc của Excel vừa khai thác sức mạnh lập trình.

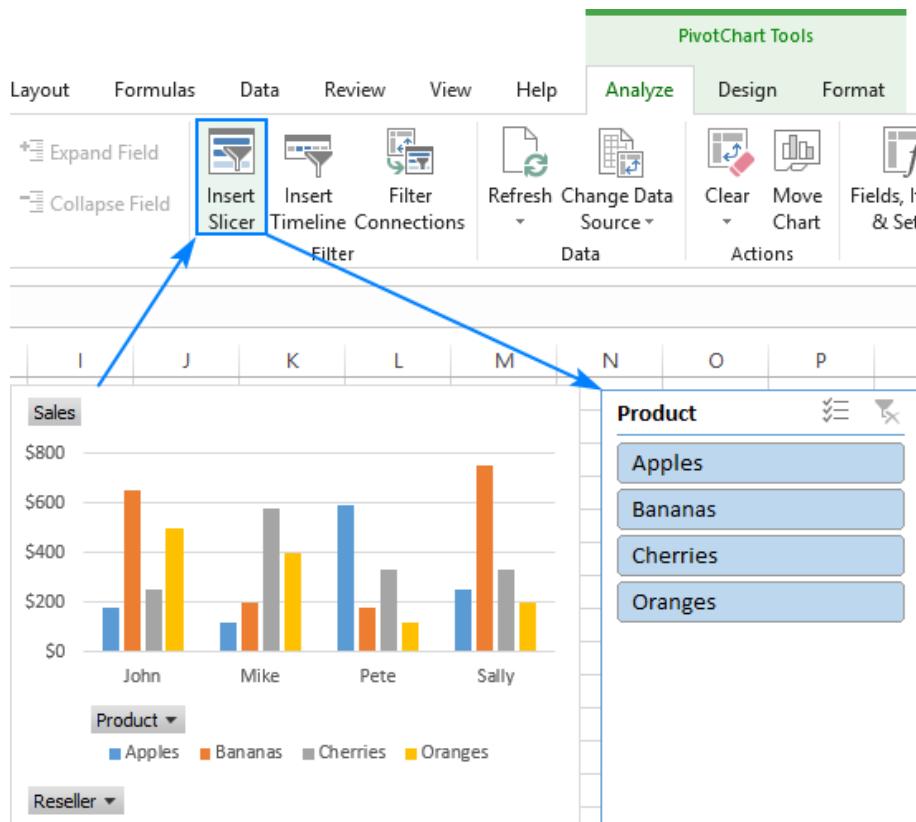
Bảng 1: So sánh nhanh Excel và Python trong phân tích dữ liệu

Tiêu chí	Excel	Python
Mục tiêu điển hình	Phân tích mô tả nhanh, báo cáo, nhập liệu thủ công	Tự động hóa, xử lý dữ liệu lớn, phân tích nâng cao/ML
Trải nghiệm người dùng	Giao diện trực quan, thao tác kéo-thả	Lập trình (script, notebook), cần kỹ năng code
Quy mô dữ liệu	Tốt với bảng nhỏ-trung bình; dễ châm khi rất lớn	Tốt với dữ liệu lớn (pandas, Dask, Spark)
Tự động hóa	Hạn chế (VBA/Power Query/Power Automate)	Mạnh (script, scheduler, CI/CD)
Tái lập/kết quả lặp lại	Dễ lặp do thao tác tay	Tái lập cao qua mã nguồn và môi trường
Biểu đồ	Nhanh, đẹp, đủ dùng cho báo cáo	Linh hoạt, tùy biến cao (matplotlib, plotly, seaborn)
Phân tích nâng cao	Hạn chế (thống kê cơ bản, add-in)	Đầy đủ: thống kê, ML, NLP, CV (scikit-learn, statsmodels, PyTorch)
Làm việc hàng loạt	Không tối ưu nhiều file/folder	Mạnh (batch, pipeline, luồng ETL)
Tích hợp dữ liệu	Chủ yếu file, ODBC, Power Query	Rộng: file, DB, API, message queue, cloud
Hợp tác	Phổ biến trong doanh nghiệp, dễ chia sẻ file	Quản lý phiên bản tốt (Git), hợp tác qua repo
Bảo trì	Khó kiểm soát thay đổi thủ công	Dễ bảo trì qua version control, test
Phụ thuộc môi trường	Cần giấy phép Microsoft 365/Excel	Cần Python + thư viện; chạy mọi nền tảng
Chi phí	Giấy phép phần mềm	Mã nguồn mở (phần lớn miễn phí)
Độ dốc học tập	Thấp-trung bình	Trung bình-cao (nhưng mở rộng mạnh)

2. Tổng quan về Excel và Python

2.1 Ưu điểm của Excel

- Công thức tính toán sẵn có (SUM, AVERAGE, IF, VLOOKUP).
- Tạo biểu đồ nhanh chóng (cột, tròn, đường, kết hợp...).
- Pivot Table mạnh mẽ để tổng hợp và phân tích dữ liệu.
- Phổ biến trong doanh nghiệp, dễ chia sẻ và chỉnh sửa.



Hình 26: Minh họa Pivot Table và biểu đồ trong Excel

2.2 Ưu điểm của Python

- Xử lý dữ liệu lớn hiệu quả với thư viện pandas.
- Tự động hóa toàn bộ quy trình phân tích.
- Hỗ trợ từ thống kê cơ bản đến Machine Learning (scikit-learn, statsmodels).
- Kết nối đa dạng nguồn dữ liệu: Excel, SQL, API, JSON, CSV,...



Hình 27: Hệ sinh thái thư viện Python cho phân tích dữ liệu

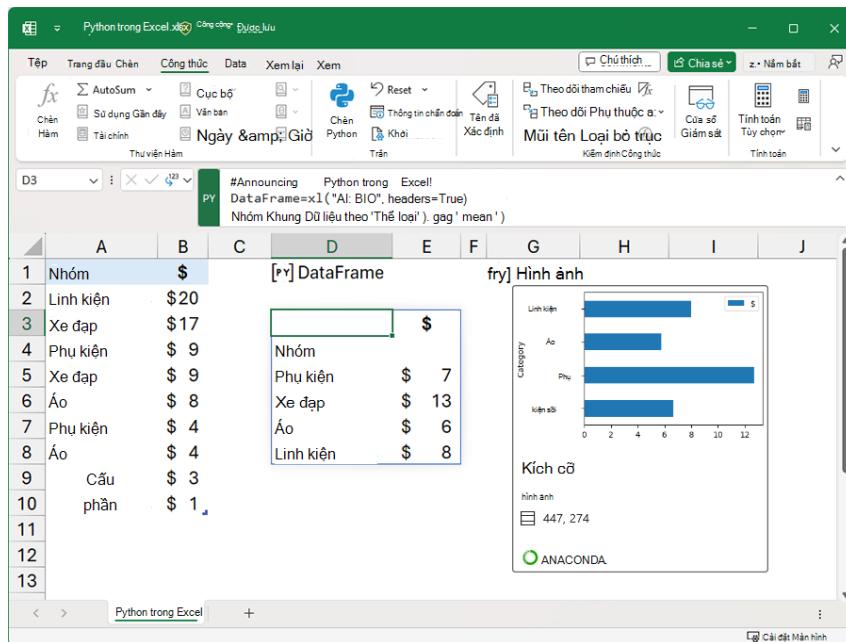
2.3 Hạn chế khi chỉ dùng một công cụ

Excel	Python
Khó xử lý dữ liệu lớn, hạn chế phân tích nâng cao	Cần kỹ năng lập trình
Khó tự động hóa quy trình phức tạp	Thiếu giao diện nhập liệu trực quan
Dễ treo khi dữ liệu vượt giới hạn	Cần môi trường lập trình, quản lý thư viện

3. Python trực tiếp trong Excel

Từ năm 2023, Excel 365 hỗ trợ hàm =PY() cho phép:

- Viết mã Python ngay trong ô Excel.
- Sử dụng thư viện như pandas, matplotlib.
- Kết hợp trực tiếp với dữ liệu bảng tính.



Hình 28: Giao diện Python trong Excel

3.1 Thực hành với Iris Dataset

- Tải dữ liệu Iris, import vào Excel, đặt tên bảng IrisDataset.
- Mô tả dữ liệu:

```
df = xl("IrisDataset[#All]", headers=True)
df.describe()
```

- Vẽ biểu đồ phân tán:

```
import matplotlib.pyplot as plt
plt.scatter(df["SepalLengthCm"], df["SepalWidthCm"])
plt.xlabel("SepalLengthCm")
plt.ylabel("SepalWidthCm")
plt.title("Sepal length and width analysis")
```

- Tạo ma trận tương quan:

```
df.corr(numeric_only=True)
```

4. Lập trình Python xử lý file Excel bên ngoài

4.1 Tạo file Excel từ CSV với openpyxl

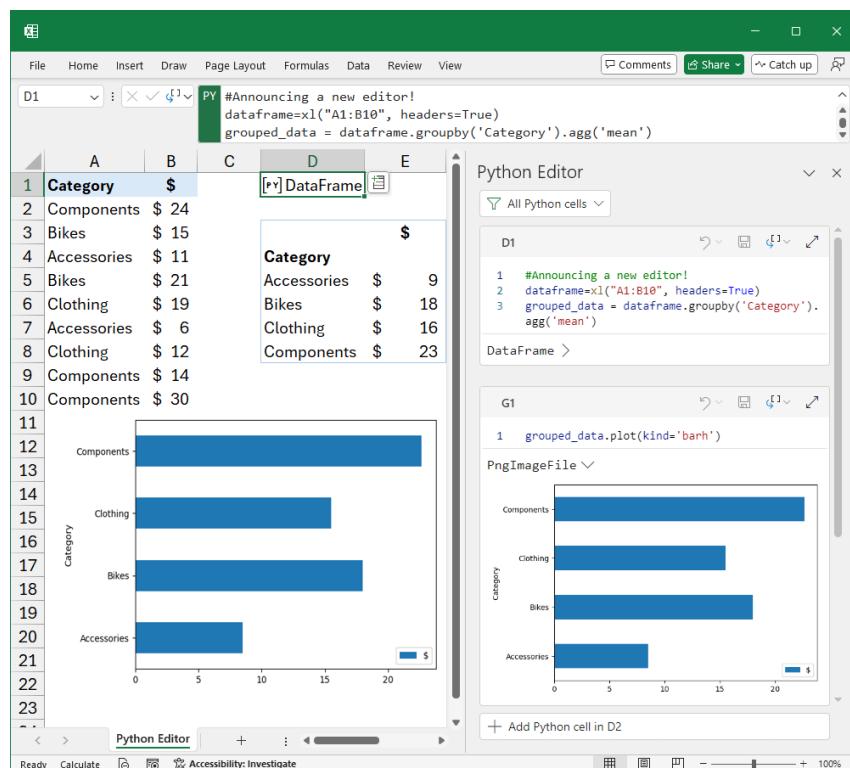
- Đọc CSV bằng pandas.
- Ghi dữ liệu vào Excel.
- Định dạng tiêu đề (màu nền, in đậm).

4.2 Xử lý dữ liệu có sẵn

- Thêm cột SOAmount Level, phân loại High/Medium.
- Tô màu theo điều kiện.
- Xóa dòng có giá trị thấp.

4.3 Biểu đồ và định dạng nâng cao

- Thêm dòng TOTAL và công thức SUM.
- Tạo biểu đồ cột (BarChart).
- Thêm viền và căn giữa dữ liệu.



Hình 29: File Excel cuối cùng với biểu đồ và định dạng

5. So sánh nhanh hai hướng tiếp cận

Tiêu chí	Python trong Excel	Python độc lập
Môi trường	Trong Excel	Script bên ngoài
Trực quan	Cao	Thấp hơn
Tự động hóa	Hạn chế	Mạnh mẽ
Xử lý hàng loạt	Không tối ưu	Rất tốt
Phụ thuộc	Cần Microsoft 365	Chỉ cần Python

6. Kết luận

Sự kết hợp Python + Excel tạo nên giải pháp linh hoạt:

- **Python trong Excel:** Phân tích nhanh, trực quan.
- **Python độc lập:** Tối ưu cho quy trình lớn, tự động hóa mạnh.

Ứng dụng đúng cách giúp tiết kiệm thời gian, giảm lỗi và mở rộng quy mô phân tích dữ liệu.