Bài 11 - Visualization trong python

16 Sep 2019 - phamdinhkhanh

Menu

1. Giới thiệu về biểu đồ

Visualization hiểu một cách đơn giản là hình ảnh hóa dựa trên dữ liệu. Khái niệm của visualization rất ngắn gọn nhưng trên thực tế visualization lại là một mảng rất rộng và có thể coi là một lĩnh vực kết hợp của khoa học và nghệ thuật bởi nó vừa lên quan đến đồ họa (sử dụng hình học để diễn tả kết quả), vừa liên quan đến khoa học thống kê (sử dụng con số để nói lên vấn đề). Nhờ có visualization, chúng ta có thể dễ dàng đưa ra các so sánh trực quan, tính toán tỷ trọng, nhận biết trend, phát hiện outlier, nhận diện đặc điểm phân phối của biến tốt hơn. Từ đó hỗ trợ quá trình nắm thông tin và đưa ra quyết định tốt hơn. Trong các kĩ năng của data scientist thì visualization là một trong những kĩ năng cơ bản và quan trọng nhất. Thế nhưng nhiều data scientist lại chưa nhận diện được điều này và thường xem nhẹ vai trò của visualization. Trước đây tôi cũng đã từng mắc sai lầm như vậy. Qua kinh nghiệm nhiều năm xây dựng mô hình và phân tích kinh doanh đã giúp tôi nhìn nhận lại vai trò của visualization. Chính vì thế tôi quyết định tổng hợp bài viết này theo cách bao quát và sơ đẳng nhất về visualization trên python như một tài liệu sử dụng khi cần và đồng thời cũng là cách củng cố lại kiến thức.

Nhắc đến visualization chúng ta không thể không nói đến một số dạng biểu đồ cơ bản như: line, barchart, pie, area, boxplot.

Trong đó:

- line: Là biểu đồ đường kết nối các điểm thành 1 đường liền khúc.
- barchart: Biểu diễn giá trị của các nhóm dưới dạng cột.
- pie: Biểu đồ hình tròn biểu diễn phần trăm của các nhóm.
- area: Biểu đồ biểu diễn diện tích của các đường.
- boxplot: Biểu đồ biểu diễn các giá trị thống kê của một biến trên đồ thị bao gồm: Trung bình, Max, Min, các ngưỡng percent tile 25%, 50%, 75%.

Sau đây chúng ta sẽ học cách sử dụng các dạng biểu đồ này trên matplotlib.

Top

1.1. Biểu đồ line

Biểu đồ line là biểu đồ biểu diễn các giá trị dưới dạng những đường. Trên matplotlib. Line được vẽ thông qua `plt.plot()`. Sau đây ta cùng biểu diễn giá chứng khoán thông qua biểu đồ line.

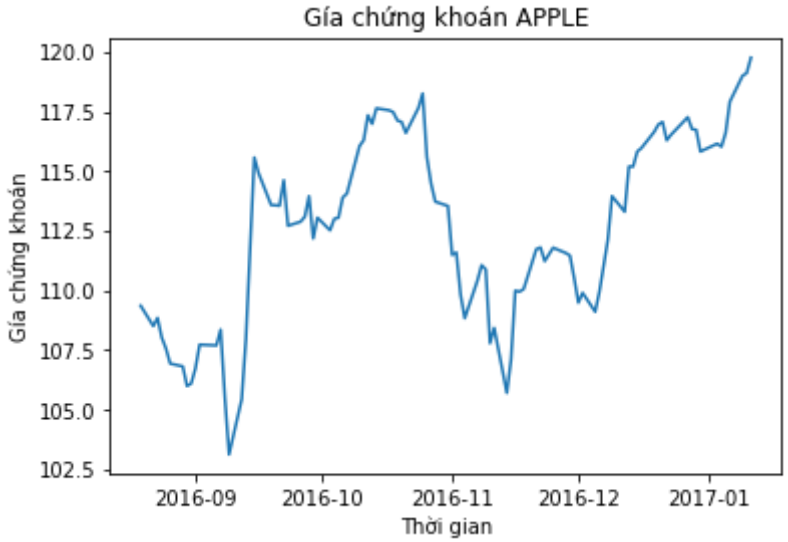
Lấy dữ liệu chứng khoán của apple

```
1     import matplotlib.pyplot as plt
2     import pandas as pd
3
4     import datetime
5     import pandas_datareader.data as web
6     from pandas import Series, DataFrame
7
8
9     start = datetime.datetime(2010, 1, 1)
10    end = datetime.datetime(2017, 1, 11)
11
12    df = web.DataReader("AAPL", 'yahoo', start, end)
13    df.tail()
```

	High	Low	Open	Close	Volume	Adj Close
Date						
2017-01-05	116.860001	115.809998	115.919998	116.610001	22193600.0	111.727715
2017-01-06	118.160004	116.470001	116.779999	117.910004	31751900.0	112.973305
2017-01-09	119.430000	117.940002	117.949997	118.989998	33561900.0	114.008080
2017-01-10	119.379997	118.300003	118.769997	119.110001	24462100.0	114.123047
2017-01-11	119.930000	118.599998	118.739998	119.750000	27588600.0	114.736275

Biểu diễn giá chứng khoán dưới dạng biểu đồ line

```
1     plt.plot(df['Close'].tail(100))
2     plt.ylabel('Gía chứng khoán')
3     plt.xlabel('Thời gian')
4     plt.title('Gía chứng khoán APPLE')
```



Thay đổi định dạng line

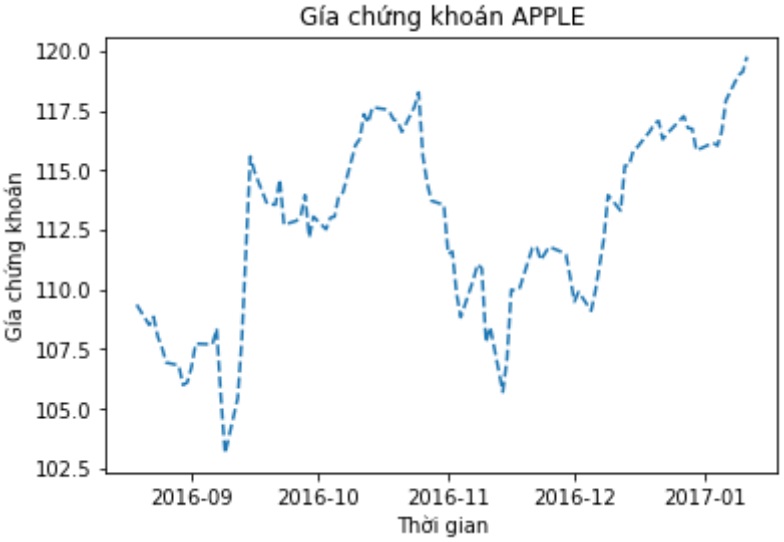
Nếu muốn thay đổi định dạng của line chúng ta sẽ sử dụng thêm 1 tham số khác là `linestype`. Một số line styles thông dụng: {'-', '–', '-.', ':', ''}

Top

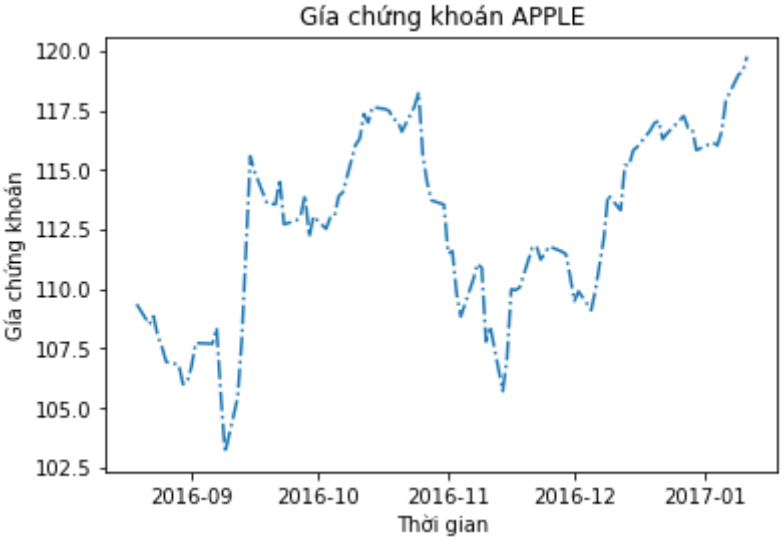
- `-` : Đường nét liền.
- `--` : Đường nét đứt dài.
- `-.` : Đường line nét đứt dài kết hợp với dấu chấm.
- `:` : Đường line gồm các dấu chấm.

Chẳng hạn để thay đổi line từ dạng đường nét liền sang nét đứt:

```
plt.plot(df['Close'].tail(100), linestyle = '--')
plt.ylabel('Gía chứng khoán')
plt.xlabel('Thời gian')
plt.title('Gía chứng khoán APPLE')
```

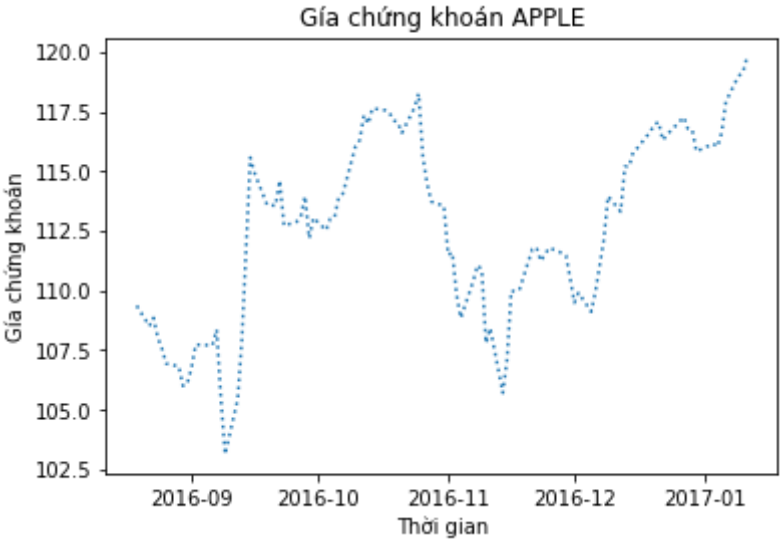



```
# Đường nét đứt có gạch nối
plt.plot(df['Close'].tail(100), linestyle = '-.')
plt.ylabel('Gía chứng khoán')
plt.xlabel('Thời gian')
plt.title('Gía chứng khoán APPLE')
```

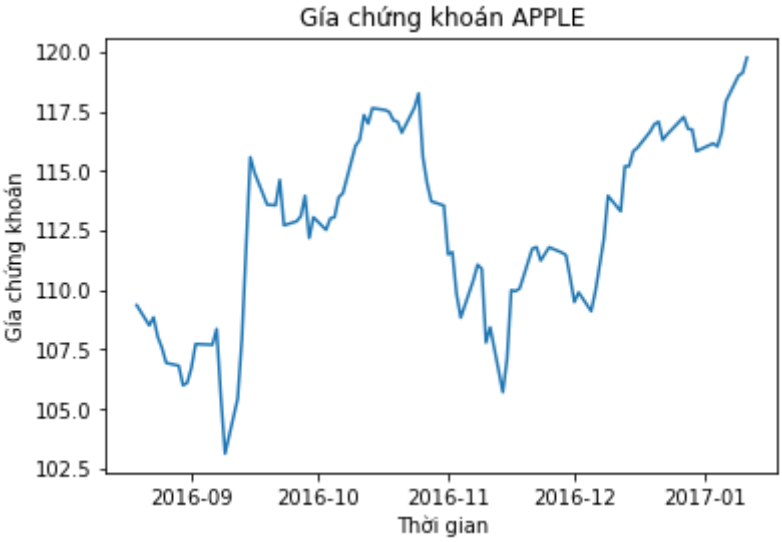


Top

```
# Đường nét chấm
plt.plot(df['Close'].tail(100), linestyle = ':')
plt.ylabel('Gía chứng khoán')
plt.xlabel('Thời gian')
plt.title('Gía chứng khoán APPLE')
```




```
plt.plot(df['Close'].tail(100), linestyle = '-')
plt.ylabel('Gía chứng khoán')
plt.xlabel('Thời gian')
plt.title('Gía chứng khoán APPLE')
```



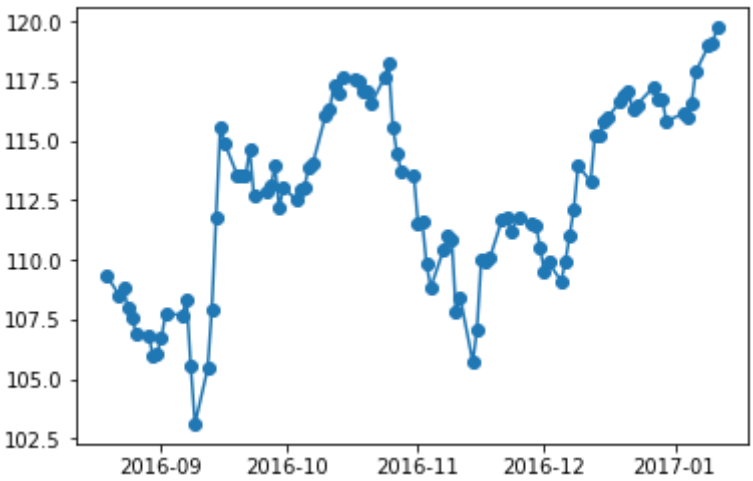
Kết hợp line và point

Bên cạnh line chúng ta còn có thể đánh dấu các điểm mút bằng các point. Hình dạng của point có thể là hình tròn, vuông hoặc tam giác và được khai báo thông qua tham số `marker` . Các giá trị của marker sẽ tương ứng như sau:

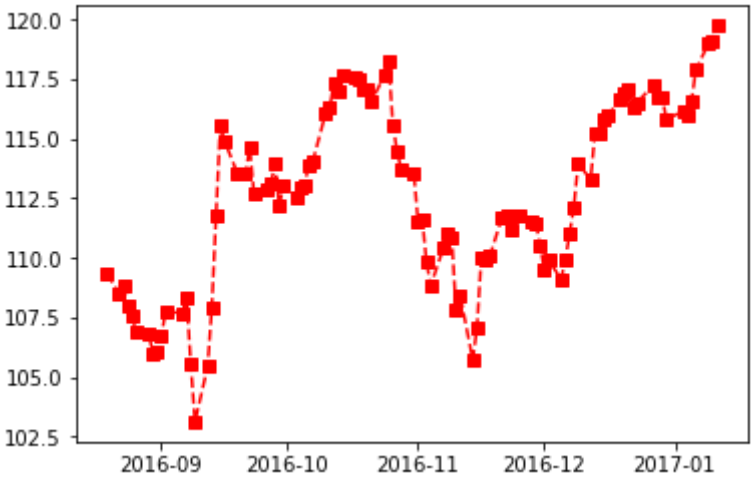
- `^` : Hình tam giác
- `o` : Hình tròn.
- `s` : Hình vuông (s tức là square).

Bên dưới là một số kết hợp của linstyle và marker. Top

```
1    plt.plot(df['Close'].tail(100), linestyle = '-', marker = 'o')
```

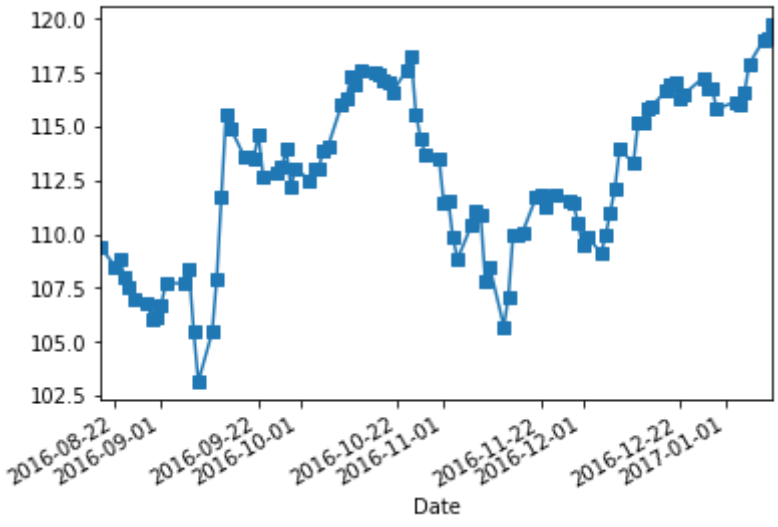



```
1    plt.plot(df['Close'].tail(100), linestyle = '--', marker = 's', color =
```



Hoặc chúng ta cũng có thể vẽ biểu đồ line từ pandas dataframe.

```
1    df['Close'].tail(100).plot(marker = 's')
```

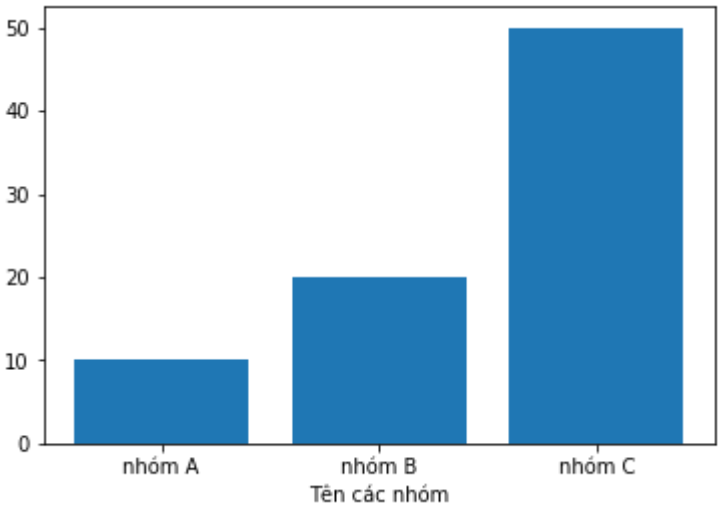


Top

1.2. Biểu đồ barchart

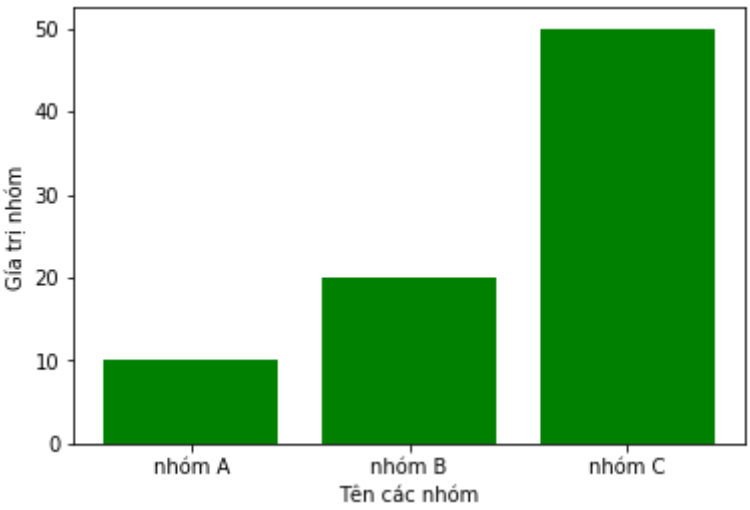
Biểu đồ barchart là dạng biểu đồ có thể coi là phổ biến nhất và được dùng chủ yếu trong trường hợp so sánh giá trị giữa các nhóm thông qua độ dài cột. Để biểu diễn biểu đồ barchart trong python chúng ta sử dụng hàm `plt.bar()`. Các tham số truyền vào bao gồm tên các nhóm (tham số `x`) và giá trị của các nhóm (tham số `height`).

```
1     plt.bar(x = ['nhóm A', 'nhóm B', 'nhóm C'], height = [10, 20, 50])
2     plt.xlabel('Tên các nhóm')
3     plt.ylabel('Gía trị nhóm')
```



Thay đổi màu sắc các nhóm.

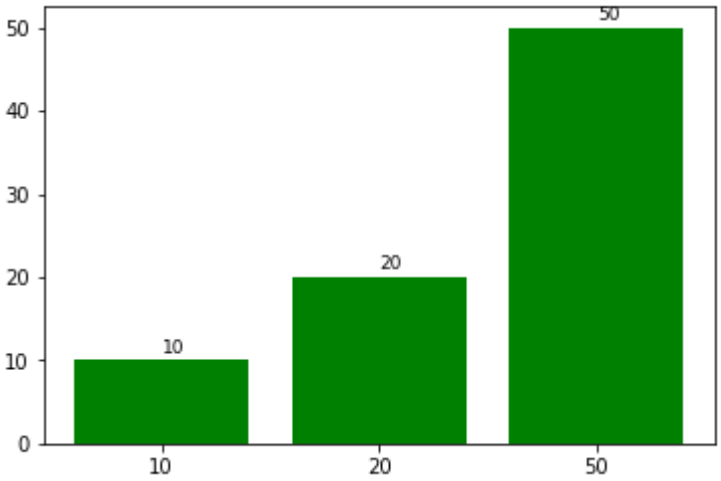
```
1     plt.bar(x = ['nhóm A', 'nhóm B', 'nhóm C'], height = [10, 20, 50], col
2     plt.xlabel('Tên các nhóm')
3     plt.ylabel('Gía trị nhóm')
```



Thêm nhãn giá trị cho các cột bằng tham số plt.text(). Trong đó tham số `x` và `y` của `plt.text()` qui định tọa độ điểm bắt đầu của rectangle chứa label tên của nhóm. `s` chứa tên labels của nhóm và `size` qui định kích thước của text.

Top

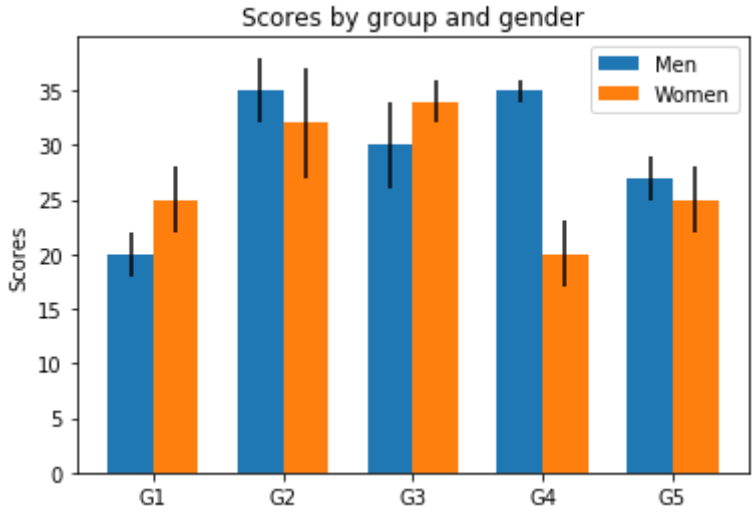
```
x_values = [0, 1, 2]
y_values = [10, 20, 50]
data_labels = ['10', '20', '50']
plt.bar(x = data_labels, height = y_values, color = 'green')

for i in range(len(data_labels)): # your number of bars
  plt.text(x = x_values[i], #takes your x values as horizontal positi
  y = y_values[i]+1, #takes your y values as vertical positioning arg
  s = data_labels[i], # the labels you want to add to the data
  size = 9)
```



Chúng ta cũng có thể vẽ biểu đồ của 2 biến trở lên là các barchart liền kề nhau.

```
import numpy as np

men_means, men_std = (20, 35, 30, 35, 27), (2, 3, 4, 1, 2)
women_means, women_std = (25, 32, 34, 20, 25), (3, 5, 2, 3, 3)

ind = np.arange(len(men_means))  # the x locations for the groups
width = 0.35  # the width of the bars

fig, ax = plt.subplots()
rects1 = ax.bar(ind - width/2, men_means, width, yerr=men_std,
                label='Men')
rects2 = ax.bar(ind + width/2, women_means, width, yerr=women_std,
                label='Women')

# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('Scores')
ax.set_title('Scores by group and gender')
ax.set_xticks(ind)
ax.set_xticklabels(('G1', 'G2', 'G3', 'G4', 'G5'))
ax.legend()

plt.show()
```

Top

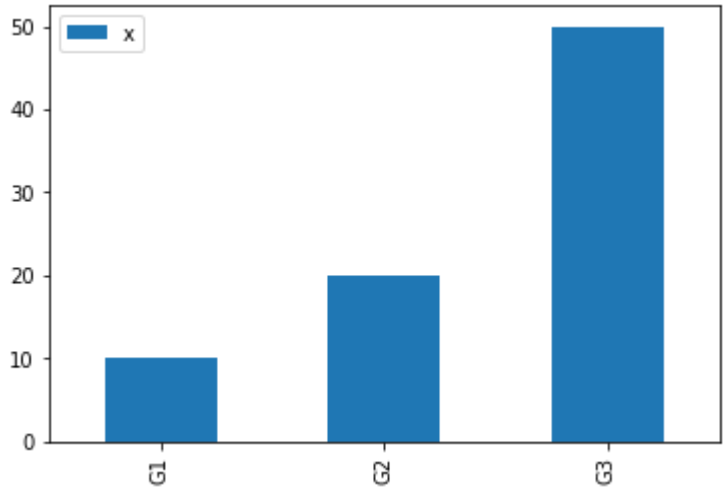


Ta cũng có thể biểu diễn biểu đồ thông qua dataframe.

```
df = pd.DataFrame({'x': [10, 20, 50]}, index = ['G1', 'G2', 'G3'])
df
```

	x
G1	10
G2	20
G3	50

```
df.plot.bar()
```



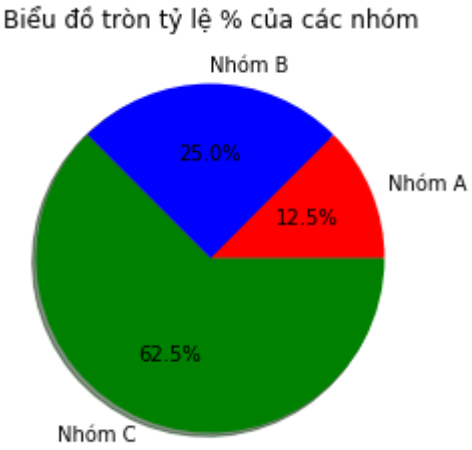
1.3. Biểu đồ tròn

Biểu đồ tròn được sử dụng để visualize tỷ lệ phần trăm các class. Ưu điểm của biểu đồ này là dễ dàng hình dung được giá trị % mà các class này đóng góp vào số tổng. Nhưng nhược điểm là không thể hiện số tuyệt đối.

Để tạo biểu đồ tròn trong mathplotlib.

Top

```
import numpy as np
plt.pie(x = np.array([10, 20, 50]), # giá trị của các nhóm
        labels = ['Nhóm A', 'Nhóm B', 'Nhóm C'], # Nhãn của các nhóm
        colors = ['red', 'blue', 'green'], # Màu sắc của các nhóm
        autopct = '%1.1f%%', # Format hiển thị giá trị %
        shadow = True
       )
plt.title('Biểu đồ tròn tỷ lệ % của các nhóm')
```

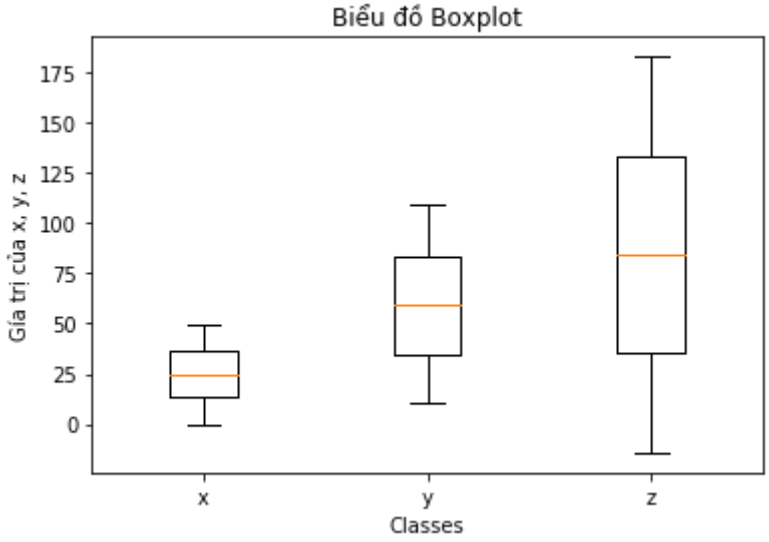


1.4. Biểu đồ boxplot

Biểu đồ boxplot sẽ cho ta biết đặc trưng về phân phối của 1 biến dựa trên các giá trị trung bình, min, max, các khoảng phân vị 25%, 50%, 75%. Đây là biểu đồ được sử dụng nhiều trong chứng khoán và thống kê học để so sánh các biến với nhau.

```
import numpy as np
x = np.random.randn(100) + np.arange(0, 100) * 0.5
y = np.random.randn(100) + np.arange(0, 100) * 1.0 + 10
z = np.random.randn(100) + np.arange(0, 100) * 2 - 15

plt.boxplot([x, y, z],
            labels = ['x', 'y', 'z'],
            showfliers = True)

plt.title('Biểu đồ Boxplot')
plt.xlabel('Classes')
plt.ylabel('Gía trị của x, y, z')
```

Top



1.5. Vẽ biểu đồ trên dataframe

Định dạng dataframe của pandas không chỉ hỗ trợ các truy vấn và thống kê dữ liệu có cấu trúc nhanh hơn mà còn support vẽ biểu đồ dưới dạng matplotlib-based. Sau đây chúng ta cùng sử dụng dataframe để vẽ các đồ thị cơ bản.

Để tìm hiểu kĩ hơn về thống kê và vẽ biểu đồ trên dataframe các bạn có thể tham khảo bài Giới thiệu pandas (https://www.kaggle.com/phamdinhkhanh/gi-i-thi-u-pandas).

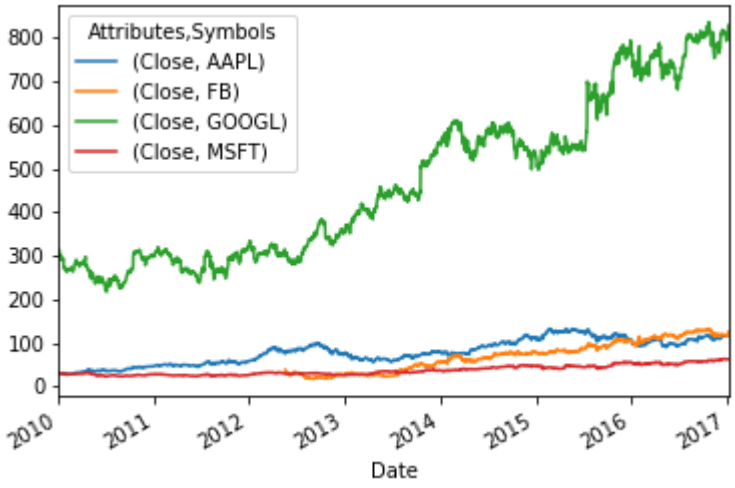
```
1     import pandas as pd
2     import datetime
3     import pandas_datareader.data as web
4     from pandas import Series, DataFrame
5
6
7     start = datetime.datetime(2010, 1, 1)
8     end = datetime.datetime(2017, 1, 11)
9
10    df = web.DataReader(["AAPL", "GOOGL", "MSFT", "FB"], 'yahoo', start, 
11    # Chỉ lấy giá close
12    df = df[['Close']]
13    df.tail()
```

Attributes	Close			
Symbols	AAPL	FB	GOOGL	MSFT
Date				
2017-01-05	116.610001	120.669998	813.020020	62.299999
2017-01-06	117.910004	123.410004	825.210022	62.840000
2017-01-09	118.989998	124.900002	827.179993	62.639999
2017-01-10	119.110001	124.349998	826.010010	62.619999
2017-01-11	119.750000	126.089996	829.859985	63.189999

Biểu đồ line

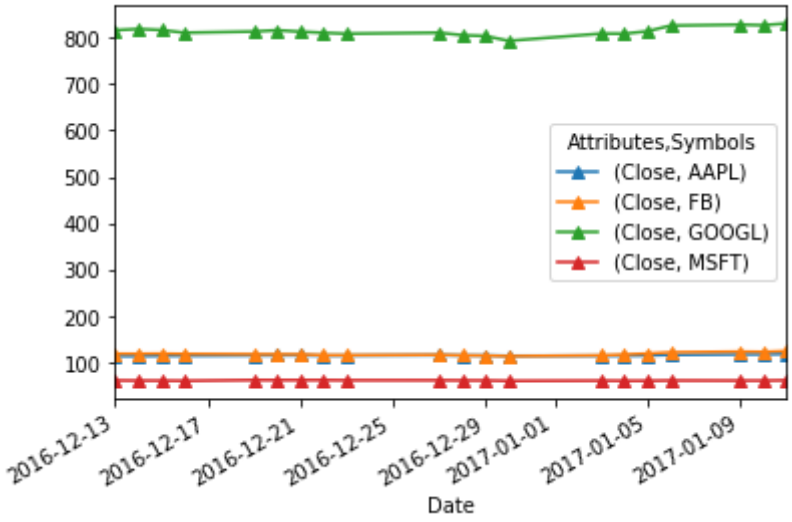
```
1     df.plot()
```

Top



Biểu đồ line kết hợp point

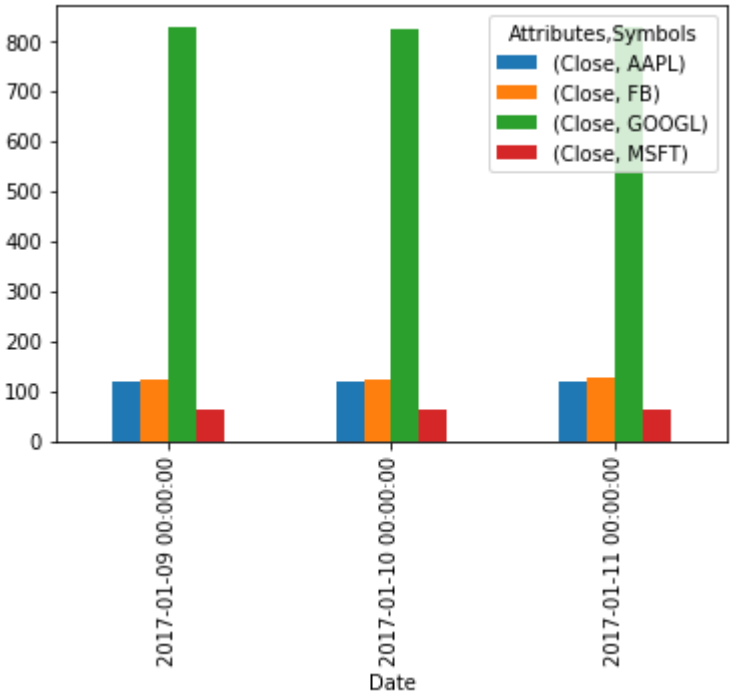
```
1     df.tail(20).plot(linestyle = '-', marker = '^')
```



Biểu đồ barchart

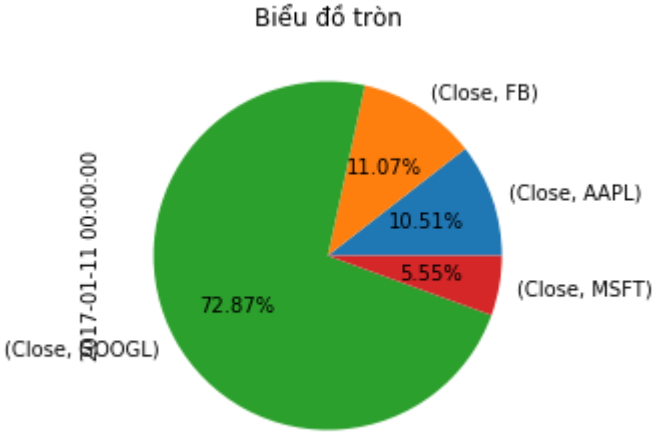
```
1     df.tail(3).plot.bar()
```

Top



Biểu đồ tròn

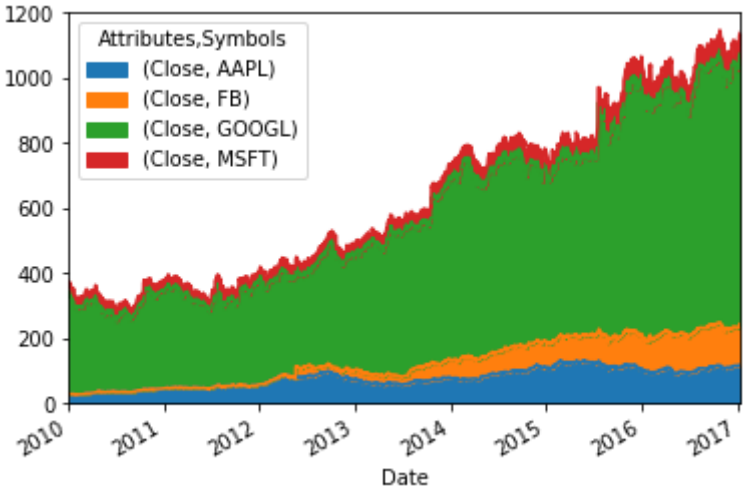
```
1     df.iloc[-1, :].plot.pie(autopct = '%.2f%%')
2     plt.title('Biểu đồ tròn')
```



Biểu đồ diện tích

```
1     df.plot.area()
```

Top



Vùng có diện tích càng lớn thì khoảng chênh lệch về giá theo thời gian của nó càng lớn và các vùng có diện tích nhỏ hơn cho thây các mã chứng khoán ít có sự chênh lệch về giá theo thời gian.

1.6. Biểu đồ heatmap.

Heatmap là biểu đồ sử dụng cường độ màu sắc để thể hiện độ lớn của giá trị. Khi đó các giá trị lớn sẽ được làm nổi bật bằng các vùng màu có cường độ ánh sáng mạnh và các giá trị nhỏ hơn sẽ được thể hiện bằng các mảng màu nhạt hơn. Các trường hợp thường sử dụng heatmap:

- Biểu đồ hệ số tương quan.
- Biểu đồ địa lý về cảnh báo thiên tai.
- Biểu đồ mật độ dân số.
- Biểu đồ crazy egg trong đo lường các component được sử dụng nhiều trong 1 website hoặc app. …

Trong machine learning ứng dụng lớn nhất của heatmap có lẽ là thể hiện các giá trị của hệ số tương quan. Ta sẽ cùng tìm hiểu cách vẽ biểu đồ heatmap biểu diễn hệ số tương quan.

```
1    # Tính correlation
2    df_cor = df.corr()
3    df_cor
```

	Attributes	Close			
	Symbols	**AAPL**	**FB**	**GOOGL**	**MSFT**
Attributes	**Symbols**				
Close	**AAPL**	1.000000	0.707744	0.795063	0.820183
	FB	0.707744	1.000000	0.954793	0.958231
	GOOGL	0.795063	0.954793	1.000000	0.960193
	MSFT	0.820183	0.958231	0.960193	1.000000

Để vẽ biểu đồ heatmap chúng ta có thể sử dụng hàm số `heatmap()` như bên dưới.

Top

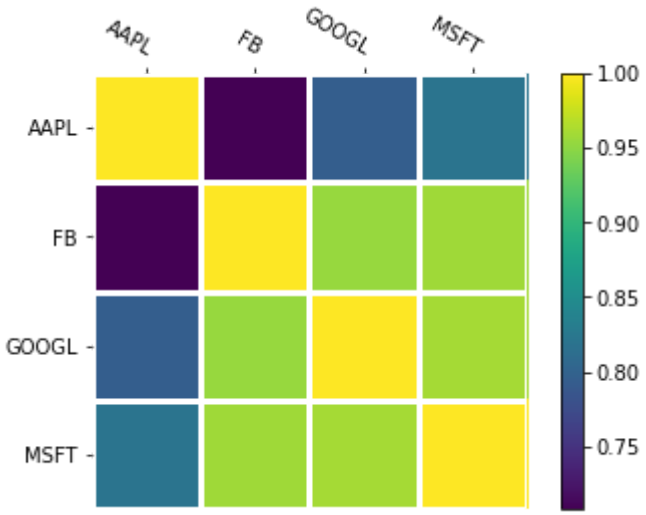
```
def heatmap(data, row_labels, col_labels, ax=None,
            cbar_kw={}, cbarlabel="", **kwargs):
    """
    Create a heatmap from a numpy array and two lists of labels.

    Parameters
    ----------
    data
        A 2D numpy array of shape (N, M).
    row_labels
        A list or array of length N with the labels for the rows.
    col_labels
        A list or array of length M with the labels for the columns.
    ax
        A `matplotlib.axes.Axes` instance to which the heatmap is plo
        not provided, use current axes or create a new one.  Optional
    cbar_kw
        A dictionary with arguments to `matplotlib.Figure.colorbar`.
    cbarlabel
        The label for the colorbar.  Optional.
    **kwargs
        All other arguments are forwarded to `imshow`.
    """

    if not ax:
        ax = plt.gca()

    # Plot the heatmap
    im = ax.imshow(data, **kwargs)

    # Create colorbar
    cbar = ax.figure.colorbar(im, ax=ax, **cbar_kw)
    cbar.ax.set_ylabel(cbarlabel, rotation=-90, va="bottom")

    # We want to show all ticks...
    ax.set_xticks(np.arange(data.shape[1]))
    ax.set_yticks(np.arange(data.shape[0]))
    # ... and label them with the respective list entries.
    ax.set_xticklabels(col_labels)
    ax.set_yticklabels(row_labels)

    # Let the horizontal axes labeling appear on top.
    ax.tick_params(top=True, bottom=False,
                   labeltop=True, labelbottom=False)

    # Rotate the tick labels and set their alignment.
    plt.setp(ax.get_xticklabels(), rotation=-30, ha="right",
             rotation_mode="anchor")

    # Turn spines off and create white grid.
    for edge, spine in ax.spines.items():
        spine.set_visible(False)

    ax.set_xticks(np.arange(data.shape[1]+1)-.5, minor=True)
    ax.set_yticks(np.arange(data.shape[0]+1)-.5, minor=True)
    ax.grid(which="minor", color="w", linestyle='-', linewidth=3)
    ax.tick_params(which="minor", bottom=False, left=False)
```

Top

```
58
59         return im, cbar
```

Hàm số sẽ có tác dụng thiết lập các bảng heatmap và labels của trục x, y trên đồ thị.

```
1     inds = ['AAPL', 'FB', 'GOOGL', 'MSFT']
2     fig, ax = plt.subplots()
3
4     im, cbar = heatmap(df_cor, row_labels = inds, col_labels = inds)
```



Chúng ta sẽ thêm titles giá trị các biến nằm trong `df_cor` vào các ô giá trị tương ứng thông qua hàm số `annotate_heatmap()`

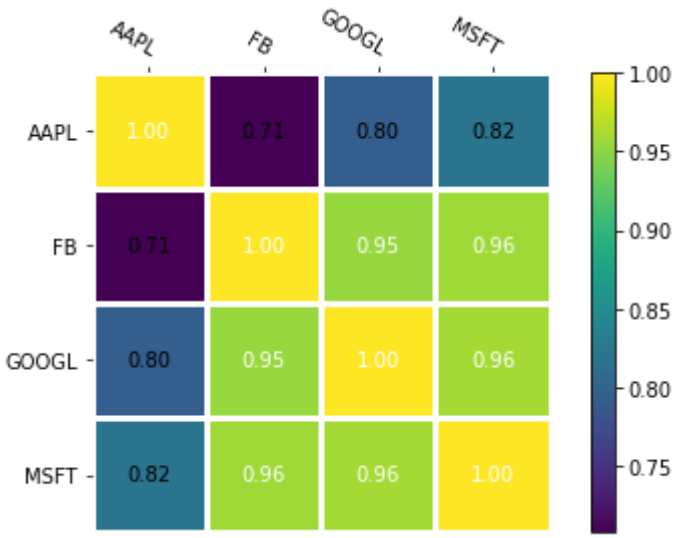
Top

```
import matplotlib

def annotate_heatmap(im, data=None, valfmt="{x:.2f}",
                     textcolors=["black", "white"],
                     threshold=None, **textkw):
    """
    A function to annotate a heatmap.

    Parameters
    ----------
    im
        The AxesImage to be labeled.
    data
        Data used to annotate.  If None, the image's data is used.  O
    valfmt
        The format of the annotations inside the heatmap.  This shoul
        use the string format method, e.g. "$ {x:.2f}", or be a
        `matplotlib.ticker.Formatter`.  Optional.
    textcolors
        A list or array of two color specifications.  The first is us
        values below a threshold, the second for those above.  Option
    threshold
        Value in data units according to which the colors from textco
        applied.  If None (the default) uses the middle of the colorm
        separation.  Optional.
    **kwargs
        All other arguments are forwarded to each call to `text` used
        the text labels.
    """

    if not isinstance(data, (list, np.ndarray)):
        data = im.get_array()

    # Normalize the threshold to the images color range.
    if threshold is not None:
        threshold = im.norm(threshold)
    else:
        threshold = im.norm(data.max())/2.

    # Set default alignment to center, but allow it to be
    # overwritten by textkw.
    kw = dict(horizontalalignment="center",
              verticalalignment="center")
    kw.update(textkw)

    # Get the formatter in case a string is supplied
    if isinstance(valfmt, str):
        valfmt = matplotlib.ticker.StrMethodFormatter(valfmt)

    # Loop over the data and create a `Text` for each "pixel".
    # Change the text's color depending on the data.
    texts = []
    for i in range(data.shape[0]):
        for j in range(data.shape[1]):
            kw.update(color=textcolors[int(im.norm(data[i, j]) > thre
            text = im.axes.text(j, i, valfmt(data[i, j], None), **kw)
            texts.append(text)
```

Top

```
58
59          return texts
```

```
1     inds = ['AAPL', 'FB', 'GOOGL', 'MSFT']
2     fig, ax = plt.subplots()
3     im, cbar = heatmap(df_cor, row_labels = inds, col_labels = inds)
4     texts = annotate_heatmap(im, valfmt="{x:.2f}")
5     fig.tight_layout()
6     plt.show()
```



Hoặc ta có thể visualize biểu đồ heatmap thông qua package `seaborn` .

```
1     import seaborn as sns
2
3     sns.heatmap(df_cor, annot=True)
```

2. Các biểu đồ biểu diễn phân phối.

2.1. Density plot

Top

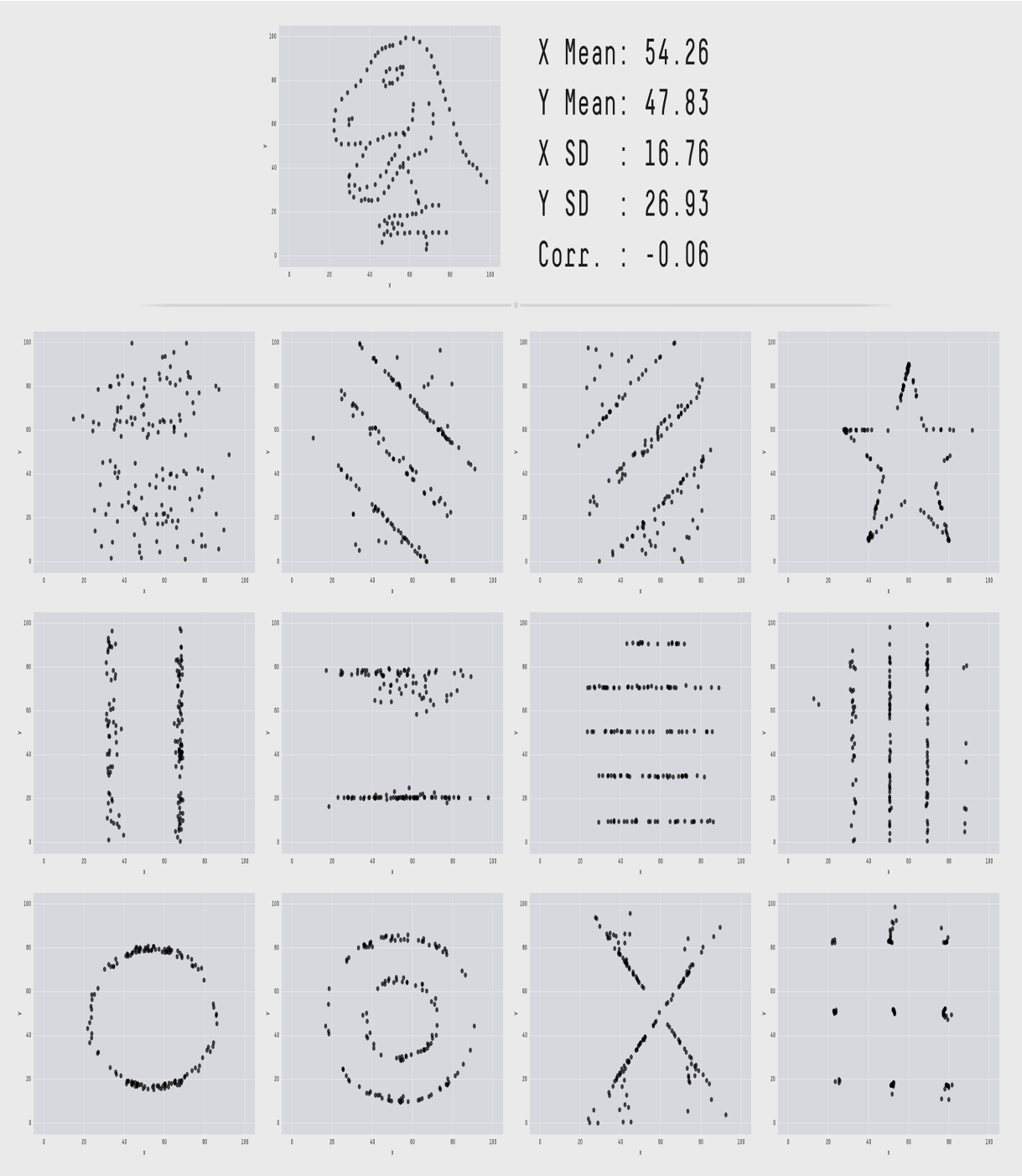
Mỗi một bộ dữ liệu đều có một đặt trưng riêng của nó. Để mô hình hóa những đặc trưng này, thống kê học sử dụng thống kê mô tả như tính mean, max, median, standard deviation, percentile. Để tính thống kê mô tả cho một dataset dạng pandas dataframe trong python đơn giản ta sử dụng hàm `describe()`.

```
from sklearn import datasets
iris = datasets.load_iris()

X = iris.data
y = iris.target

import pandas as pd
dataset = pd.DataFrame(data = X, columns = iris['feature_names'])
dataset['species'] = y
print('dataset.shape: ', dataset.shape)

dataset.describe()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

Tuy nhiên không phải lúc nào thống kê mô tả là duy nhất đối với một bộ dữ liệu. Một ví dụ tiêu biểu về phân phối hình `chú khủng long`.

Top

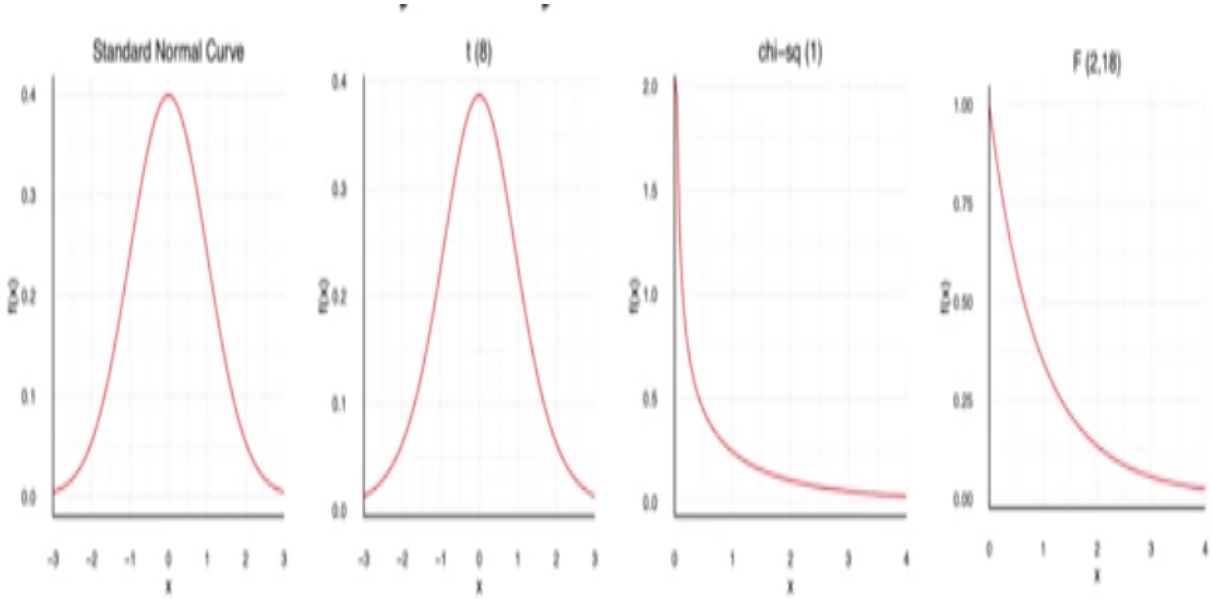


Hình 1: Đồ thị hình chú khủng long và các hình bên dưới có hình dạng hoàn toàn khác biệt nhau nhưng đều dựa trên 2 chuỗi X, Y có chung thống kê mô tả mean, phương sai và hệ số tương quan.

Do đó không nên hoàn toàn tin tưởng vào thống kê mô tả mà bên cạnh đó chúng ta cần visualize phân phối của dữ liệu.

Trong thống kê mỗi một bộ dữ liệu đều được đặc trưng bởi một hàm mật độ xác suất (pdf - probability density function). Các phân phối điển hình như `standard normal`, `T-student`, `poisson`, `fisher`, `chi-squared` đều được đặc trưng bởi những hình dạng đồ thị phân phối của hàm mật độ xác suất khác nhau.

Top



Hình 2: Đồ thị hàm mật độ xác suất của những phân phối xác suất `standard normal, T-student, poisson, fisher, chi-squared`.

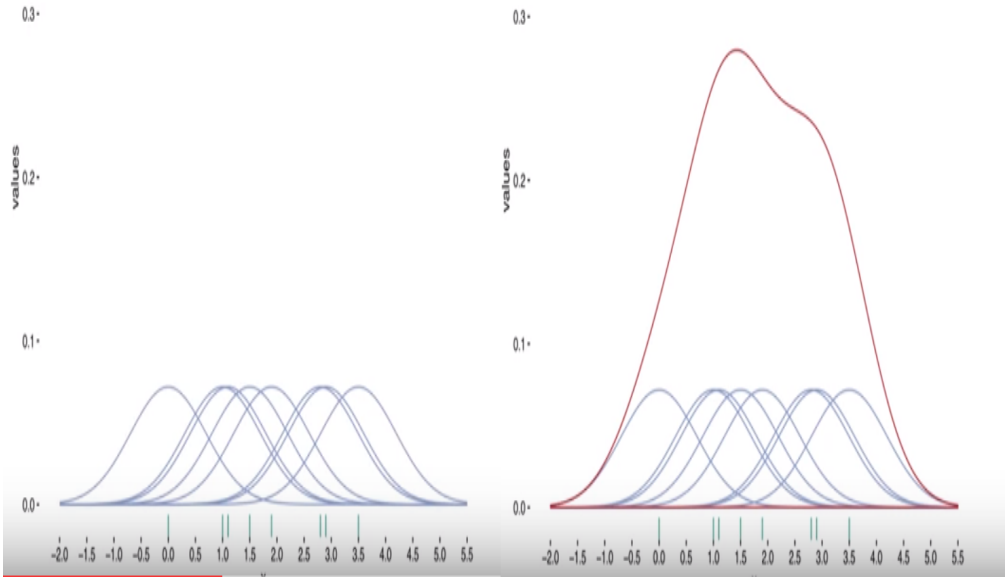
Về mặt lý thuyết (*theoreotical*) những phân phối này đều dựa trên những phương trình xác định.

Trong thực nghiệm (*empirical*) nhiều bộ dữ liệu cho thấy có hình dạng tương đồng với những phân phối này.

Để tìm ra một hình dạng tương đối cho hàm mật độ xác suất của một bộ dữ liệu chúng ta sẽ sử dụng phương pháp KDE (*kernel density estimate*)

KDE là gì?

Hãy tưởng tượng tại mỗi một quan sát ta có đường cong phân phối đặc trưng. Hàm kernel sẽ giúp xác định hình dạng của đường cong trong khi độ rộng của đường cong được xác định bởi `bandwidth - h`. Phương pháp KDE sẽ tính tổng của các đường cong chạy dọc theo trục x để hình thành nên đường cong mật độ xác suất tổng quát cho dữ liệu.



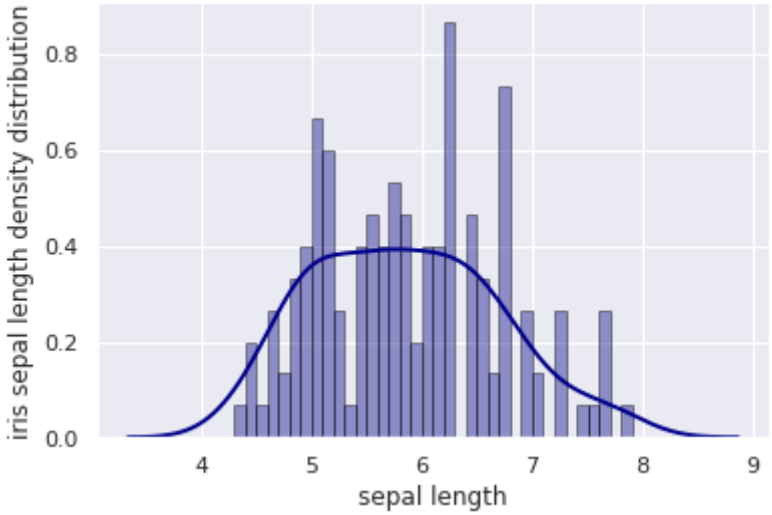
Top

Hình 3: Phương pháp KDE giúp xây dựng hình dạng phân phối của dữ liệu. Ở những nơi có nhiều điểm dữ liệu tập trung thì số lượng các đường cong chồng lấn lên nhau sẽ nhiều hơn và do đó khi tính tổng cộng dồn của nó ta sẽ thu được một giá trị lũy kế kernel density lớn hơn và trái lại với những nơi có nhiều ít điểm dữ liệu tập trung.

Ngoài ra hình dạng `bandwidth - h` sẽ giúp xác định mức độ khái quát hoặc chi tiết của đường cong. Nếu ta muốn đường cong smoothing hơn thì cần thiết lập h lớn hơn và đường cong mấp mô hơn thì h cần nhỏ hơn. Tuy nhiên bạn đọc cũng không cần quá quan tâm đến `bandwidth` vì cách tốt hơn là sử dụng giá trị mặc định được tính trong matplotlib.

Bên dưới ta sẽ thực hành vẽ hàm mật độ xác suất của độ dài các đài hoa thông qua hàm `distplot()` của package seaborn.

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.distplot(dataset['sepal length (cm)'],
             hist = True,
             bins=int(180/5),
             kde = True,
             color = 'darkblue',
             hist_kws={'edgecolor':'black'},
             kde_kws={'linewidth':2})
# Khai báo tiêu đề cho trục x
plt.xlabel('sepal length')
# Khai báo tiêu đề cho trục y
plt.ylabel('iris sepal length density distribution')
plt.show()
```



Tham số quan trọng nhất của hàm số là `kde = True` để xác nhận chúng ta sử dụng phương pháp KDE để tính toán đường cong hàm mật độ. Các tham số khác như `color` , `hist_kws` , `kde_kws` chỉ là những tham số râu ria qui định màu sắc, format, kích thước. Ngoài ra `hist = True` để thiết lập đồ thị histogram mà chúng ta sẽ tìm hiểu bên dưới.

Top

2.2 Histogram plot

Histogram là biểu đồ áp dụng trên một biến liên tục nhằm tìm ra phân phối tần suất trong những khoảng giá trị được xác định trước của một biến.

Có 2 cách tạo biểu đồ histogram theo các khoảng giá trị đó là:

- Phân chia các khoảng giá trị có độ dài bằng nhau và độ dài được tính toán từ số lượng bins khai báo.
- Tự định nghĩa các khoảng giá trị dựa trên bins_edge là các đầu mút của khoảng.

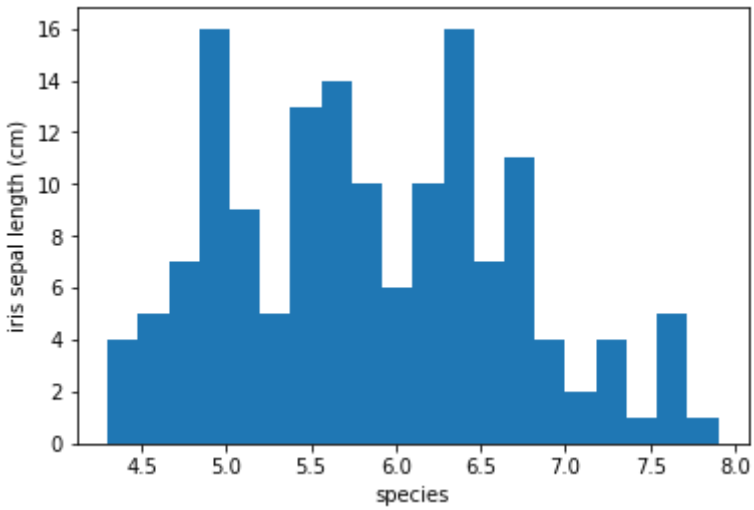
Biểu đồ histogram có thể được visualize qua package `mathplotlib` . Các biểu đồ của mathplotlib được thể được setup dưới nhiều `style` đồ họa khác nhau (thay đổi về theme, kiểu chữ, … nhưng về bản chất vẫn là các đối tượng của mathplotlib). Trong đó `seaborn` , một matplotlib-based package xuất sắc được phát triển bởi Michael Waskom là một trong những `style` được ưa chuộng nhất. Trong bài viết này chúng ta sẽ setup style của đồ thị dưới dạng `seaborn` .

Bên dưới là biểu đồ histogram của độ rộng đài hoa visualize theo 2 cách: Khai báo bins và khai báo bins edge.

Đồ thị histogram theo số lượng bins = 20

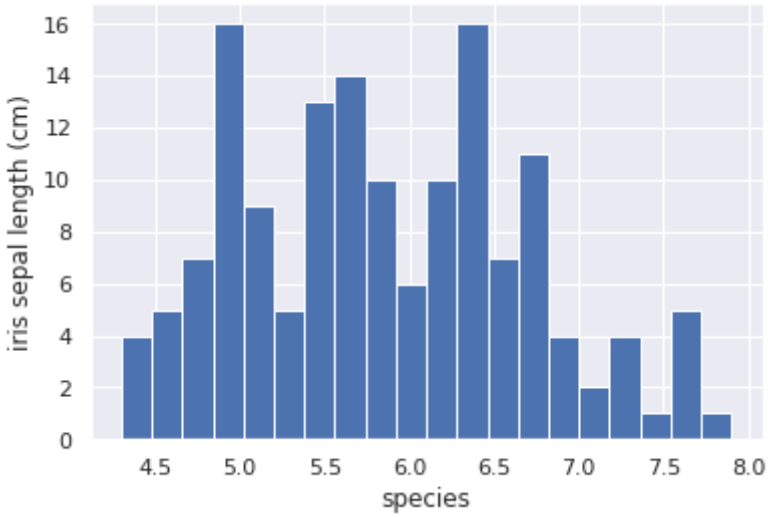
Nếu không set style hiển thị mặc định là seaborn đồ thị sẽ là:

```
1     import matplotlib.pyplot as plt
2
3     plt.hist(dataset['sepal length (cm)'], bins = 20)
4     # Khai báo tiêu đề cho trục x
5     plt.xlabel('species')
6     # Khai báo tiêu đề cho trục y
7     plt.ylabel('iris sepal length (cm)')
8     plt.show()
```



Top

```
import matplotlib.pyplot as plt
import seaborn as sns

# Setup style của matplotlib dưới dạng seaborn

sns.set()
plt.hist(dataset['sepal length (cm)'], bins = 20)
# Khai báo tiêu đề cho trục x
plt.xlabel('species')
# Khai báo tiêu đề cho trục y
plt.ylabel('iris sepal length (cm)')
plt.show()
```



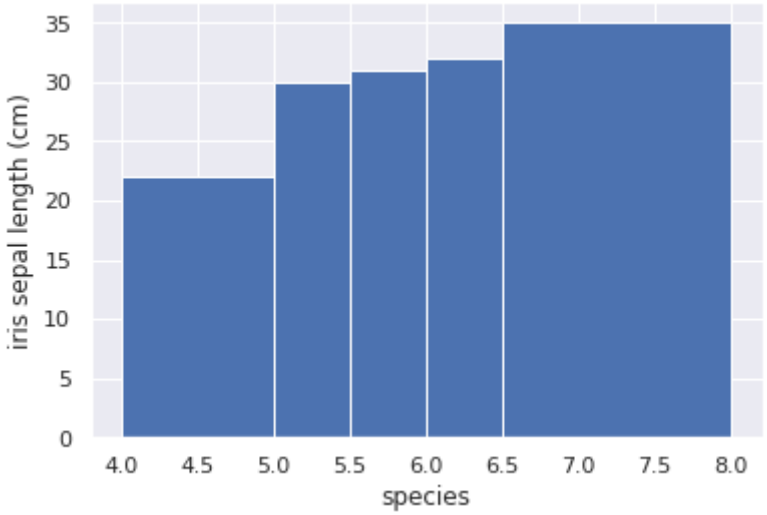
Ta thấy theme của đồ thị được chuyển sang màu xám nhạt và giữa các cột histogram có viền trắng phân chia nhìn rõ ràng hơn. Đây là những thay đổi về đồ họa rất nhỏ nhưng giúp đồ thị trở nên đẹp mắt hơn so với mặc định của mathplotlib.

Đồ thị histogram theo bin edges

Các bin edges được khai báo thông qua cũng cùng tham số `bins` , giá trị được truyền vào khi đó là 1 list các điểm đầu mút. Từ đó giúp đồ thị linh hoạt hơn khi có thể hiệu chỉnh độ dài các bins tùy thích.

```
import matplotlib.pyplot as plt
import seaborn as sns

bin_edges = [4, 5, 5.5, 6, 6.5, 8]
plt.hist(dataset['sepal length (cm)'], bins = bin_edges)
# Khai báo tiêu đề cho trục x
plt.xlabel('species')
# Khai báo tiêu đề cho trục y
plt.ylabel('iris sepal length (cm)')
plt.show()
```

Top



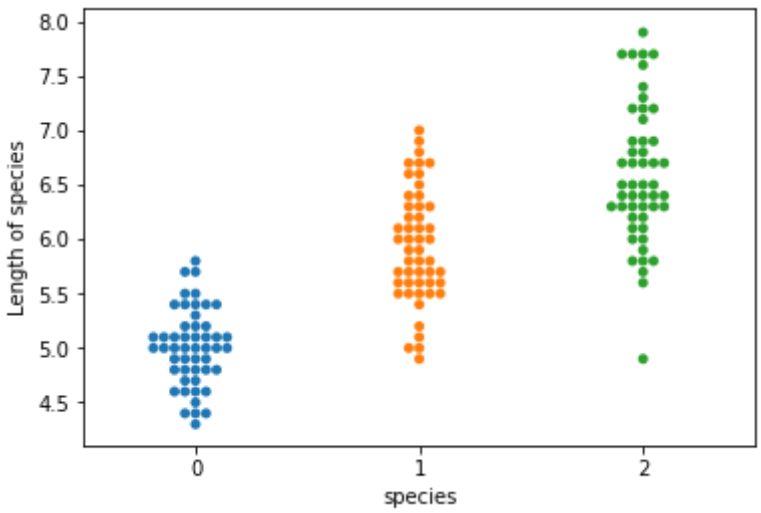
Ta thấy nhược điểm của histogram đó là đồ thị sẽ bị thay đổi tùy theo số lượng bins được thiết lập hoặc list các đầu mút range được khai báo. Do đó để nhận biết được hình dạng phân phối của dữ liệu, một biểu đồ khác thường được sử dụng thay thế đó chính là swarn plot.

2.3. Swarn plot

Swarn plot là biểu đồ point biểu diễn các giá trị dưới dạng các điểm. Các giá trị trên đồ thị bằng đúng với giá trị thật của quan sát. Do đó không xảy ra mất mát thông tin như histogram. Thông qua swarn plot ta có thể so sánh được phân phối của các class khác nhau trên cùng một đồ thị.

Hãy hình dung qua ví dụ cụ thể khi visualization dữ liệu iris theo chiều dài, rộng cánh hoa và đài hoa.

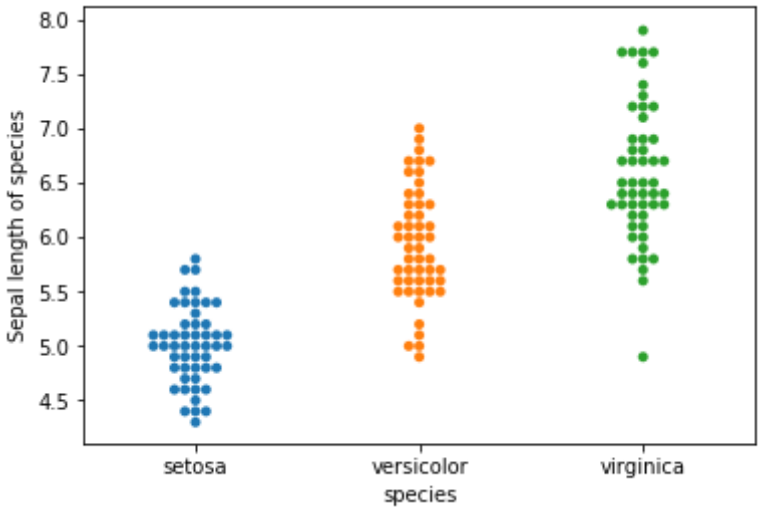
```
1    import seaborn as sn
2    import matplotlib.pyplot as plt
3
4    sn.swarmplot(x = 'species', y = 'sepal length (cm)', data = dataset)
5    plt.xlabel('species')
6    plt.ylabel('Length of species')
7    plt.show()
```



Muốn thay nhãn của các x = [0, 1, 2] sang target_names = [‘setosa’, ‘versicolor’, ‘virginica’] ta sử dụng hàm plt.xticks().

Top

```
import seaborn as sn
import matplotlib.pyplot as plt

sn.swarmplot(x = 'species', y = 'sepal length (cm)', data = dataset)
plt.xlabel('species')
# Thêm plt.xticks() để thay nhãn của x
plt.xticks(ticks = [0, 1, 2], labels = ['setosa', 'versicolor', 'virgi
plt.ylabel('Sepal length of species')
plt.show()
```



Từ biểu đồ ta nhận thấy độ dài đài hoa có sự khác biệt ở cả 3 giống hoa iris. Trung bình độ dài của đài hoa tăng dần từ setosa, versicolor đến virginica. Vì swarm là đồ thị giữ nguyên giá trị thực của trục y nên các điểm outliers được thể hiện đúng với thực tế trên từng class. Thông tin thể hiện trên biểu đồ swarm dường như là không có sự mất mát so với biểu đồ bins hoặc density.

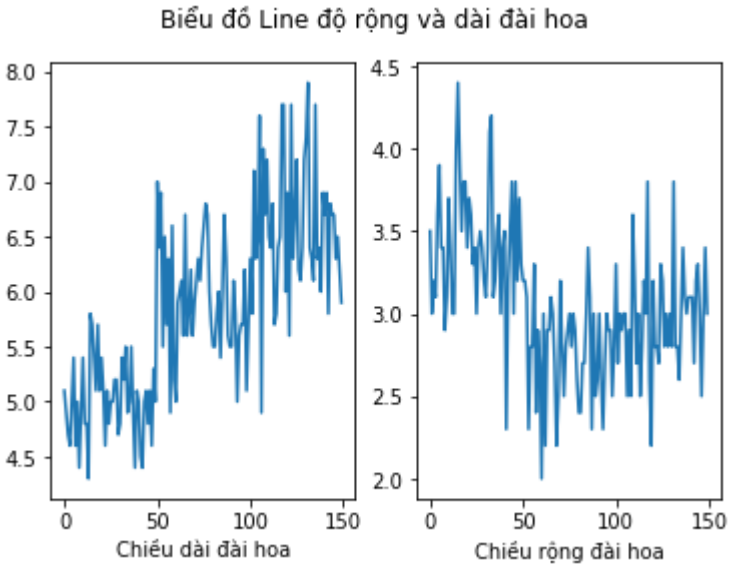
3. Vẽ nhiều biểu đồ trên cùng 1 biểu đồ.

Matplotlib cho phép chúng ta vẽ được nhiều biểu đồ trên cùng 1 đồ thị thông qua các subplots. Chúng ta có thể xác định vị trí của subplots dựa trên việc khai báo chỉ số dòng và chỉ số cột tương tự như khai báo phần tử của ma trận.

Chẳng hạn bên dưới trên cùng 1 biểu đồ chúng ta biểu diễn độ rộng và dài của đài hoa.

```
import matplotlib.pyplot as plt
fg, ax = plt.subplots(1, 2)

ax[0].plot(dataset.iloc[:, 0])
ax[0].set_xlabel('Chiều dài đài hoa')
ax[1].plot(dataset.iloc[:, 1])
ax[1].set_xlabel('Chiều rộng đài hoa')

fg.suptitle('Biểu đồ Line độ rộng và dài đài hoa')
```

Top



hàm `plt.subplots()` sẽ định hình hiển thị các biểu đồ con theo vị trí dòng, cột dựa trên khai báo số lượng (dòng, cột). Chẳng hạn nếu chúng ta muốn có biểu đồ gồm 2 đồ thị được hiển thị trên 1 dòng, 2 cột thì sẽ cần truyền vào là `plt.subplots(1, 2)` . Các tham số trả về: `fg, ax` lần lượt là hình dạng đồ thị (figures) và trục tọa độ (axis). Trong trường hợp có nhiều đồ thị thì ax sẽ là 1 list tương ứng các đồ thị con. Tại mỗi đồ thị con ta có thể visualize các biểu đồ theo ý muốn thông qua các hàm `set_` như `set_title, set_label, set_xlim, set_ylim, set_xticks, set_yticks, ....`

Chúng ta có thể sử dụng biểu đồ `plt.subplots()` để biểu diễn hình ảnh của các nhãn trong phân loại hình ảnh.

```
from google.colab import drive
import os

drive.mount('/content/gdrive')
path = '/content/gdrive/My Drive/Colab Notebooks/visualization'
os.chdir(path)
os.listdir()
```

```
['common_pdf_shape.png',
 'kde_shape.png',
 'n01443537_9.JPEG',
 'n01443537_1.JPEG',
 'n01443537_3.JPEG',
 'n01443537_6.JPEG',
 'n01443537_2.JPEG',
 'n01443537_4.JPEG',
 'n01443537_0.JPEG',
 'n01443537_8.JPEG',
 'n01443537_5.JPEG',
 'n01443537_7.JPEG']
```

Top

```
import matplotlib.pyplot as plt
import numpy as np
import cv2
import glob

# Sử dụng glob để lấy toàn bộ các file có extension là '.JPEG'.
images = []
for path in glob.glob('*.JPEG'):
  image = plt.imread(path)
  images.append(image)

# Khởi tạo subplot với 2 dòng 5 cột.
fg, ax = plt.subplots(2, 5, figsize=(20, 8))
fg.suptitle('Images of fish')

for i in np.arange(2):
  for j in np.arange(5):
    ax[i, j].imshow(images[i + j + j*i])
    ax[i, j].set_xlabel('Fish '+str(i+j+j*i))
```

4. Tổng kết.

Như vậy thông qua bài này chúng ta đã làm quen được với các dạng biểu đồ: Biểu đồ barchart, line, tròn, diện tích, heatmap và các dạng biểu đồ về phân phối như: Histogram, density, boxplot, swarn. Ngoài ra chúng ta cũng làm quen được cách sử dụng các packages như matplotlib, seaborn trong visualization.

Trên đây mới chỉ là những dạng biểu đồ phổ biến. Ngoài ra còn rất nhiều các biểu đồ visualize khác mà chúng ta sẽ bắt gặp khi làm việc với khoa học dữ liệu. Đồng thời các packages về visualize trong python cũng không chỉ giới hạn ở matplotlib. Một số packages khác cũng được sử dụng nhiều như: `plotly, waterfall,....`

Top

5. Tài liệu tham khảo.

Bài viết có sử dụng một số tài liệu tham khảo sau đây:

1. matplotlib - tutorials (https://matplotlib.org/3.1.1/tutorials/index.html)
2. giới thiệu về pandas (https://www.kaggle.com/phamdinhkhanh/gi-i-thi-u-pandas)
3. các dạng biểu đồ chính trong ggplot2 - R (https://rpubs.com/phamdinhkhanh/385843)

Top