# Data Viusalisation with Python Programming

- Presentation By Uplatz
- Contact us: https://training.uplatz.com
- Email: info@uplatz.com
- Phone: +44 7836 212635

Uplatz

# More Data Visualisation tools

# Learning outcomes:

- Scatter Plot
- Area Plot
- STACKED Area Plot
- Box Plot

# Scatter Plot

A **scatter plot** is a type of plot that shows the data as a collection of points. The position of a point depends on its two-dimensional value, where each value is a position on either the horizontal or vertical dimension.

Scatter plots are used to plot data points on horizontal and vertical axis in the attempt to show how much one variable is affected by another. Each row in the data table is represented by a marker the position depends on its values in the columns set on the X and Y axes.

# Scatter Plot

**Creating a scatter plot:**

The **scatter()** method in the **matplotlib** library is used to draw a scatter plot.

The syntax for **scatter()** method is given below:

**matplotlib.pyplot.scatter(x_axis_data, y_axis_data, s=None, c=None, marker=None, cmap=None, vmin=None, vmax=None, alpha=None, linewidths=None, edgecolors=None)**

# Scatter Plot

## Creating a scatter plot:

The scatter() method takes in the following parameters:

**x_axis_data-** An array containing x-axis data

**y_axis_data-** An array containing y-axis data

**s-** marker size (can be scalar or array of size equal to size of x or y)

**c-** color of sequence of colors for markers marker style

**cmap-** cmap name

**linewidths-** width of marker border

**edgecolor-** marker border color

**alpha-** blending value, between 0 (transparent) and 1 (opaque)

# Scatter Plot

**Creating a scatter plot:**

**Example 1:** This is the most basic example of a scatter plot.

```
import matplotlib.pyplot as plt
x =[5, 7, 8, 10, 12, 17]
y =[99, 86, 67, 58, 100, 120]
plt.scatter(x, y, c ="blue")
# To show the plot
plt.show()
```

# Scatter Plot

**Creating a scatter plot:**

**Example 2:** Scatter plot with different shape and colour for two datasets.

```
x1 = [89, 43, 36, 36, 95,10]
y1 = [21, 46, 3, 35, 67, 95]
x2 = [26, 29, 48, 64, 6, 5]
y2 = [26, 34, 90, 33, 38, 58]
plt.scatter(x1, y1, c ="pink", linewidths = 2,
        marker ="s", edgecolor ="green",  s = 50)
plt.scatter(x2, y2, c ="yellow", linewidths = 2,
        marker ="^", edgecolor ="red", s = 200)
plt.show()
```

# Scatter Plot

## Creating a scatter plot:

### Example 3: Random Data Distributions

In Machine Learning the data sets can contain thousands-, or even millions, of values.

You might not have real world data when you are testing an algorithm, you might have to use randomly generated values. The NumPy module can help us with that!

**Let's see the example.**

# Area Plot

An **Area Plot** is an extension of a Line Chart.
An **Area Plot** is obtained by filling the region between the Line Chart and the axes with a color. When more than one **Area Plot** is shown in the same graph, each area plot is filled with a different color. It represents the evolution of a **numerical variable** following **another** numerical variable. If you want to represent this evolution for several groups in the same time, you are probably interested by stacked area chart, where every groups are displayed one of top of each other.

Uplatz

# Area Plot

**Creating Area plot:**

In python, area chart can be done using the fill_between() function of matplotlib. Note that there are 2 main functions allowing to draw them. I advise to use the **fill_between** function that allows easier customisation. The **stackplot** function could work as well, but is more adapted for stacked area charts.

**matplotlib.pyplot. fill_between(x, y)**

Uplatz

# Area Plot

**Creating Area plot:**

Example_1:

```
import matplotlib.pyplot as plt
# Create data
x=range(1,6)
y=[1,4,6,8,4]
plt.plot(x,y) #Line plot
plt.show()
# Area plot
plt.fill_between(x, y)
plt.show()
```

# Area Plot

**Creating Area plot:**

Example_2: Adding line to area chart and controlling the colour of area chart.

```
import matplotlib.pyplot as plt
x=range(1,11)
y=[1,4,6,8,4,12,14,16,9,10]
plt.fill_between(x, y, color="green", alpha=0.4)
plt.plot(x,y, color="red", alpha=0.5) #Line plot
plt.show()
```

# Stacked Area Plot

A stacked area chart/plot is the extension of a basic area chart to display the evolution of the value of several groups on the same graphic. The values of each group are displayed on top of each other. It allows to check on the same figure the evolution of both the total of a numeric variable, and the importance of each group. Note that this chart becomes hard to read if too many groups are displayed and if the patterns are really different between groups.

# Stacked Area Plot

**Creating Stacked Area Plot:**

You can create stacked area chart using the Matplotlib library of python. This is done using the **stackplot()** function.

# Stacked Area Plot

**Creating Stacked Area Plot:**

Example_1:

```
import matplotlib.pyplot as plt
x=range(1,6)
y=[ [1,4,6,8,9], [2,2,7,10,12], [2,8,5,10,6] ]
# Basic stacked area chart.
plt.stackplot(x,y, labels=['A','B','C'])
plt.legend(loc='upper left')
plt.show()
```

# Stacked Area Plot

**Creating Stacked Area Plot:**

Example_2: Adding colour.

```
import matplotlib.pyplot as plt
x1=range(1,6)
y1=[1,4,6,8,9]
y2=[2,2,7,10,12]
y3=[2,8,5,10,6]
col = ["red", "blue","green"]
plt.stackplot(x1,y1, y2, y3, labels=['A','B','C'],
colors=col)
plt.legend(loc='upper right')
plt.show()
```

# Box Plot

A **Box Plot** is also known as **Whisker plot** is created to display the summary of the set of data values having properties like minimum, first quartile, median, third quartile and maximum. In the box plot, a box is created from the first quartile to the third quartile, a verticle line is also there which goes through the box at the median. Here x-axis denotes the data to be plotted while the y-axis shows the frequency distribution.

# Box Plot

**Creating Box Plot:**

The matplotlib.pyplot module of matplotlib library provides boxplot() function with the help of which we can create box plots.

**Syntax:**

matplotlib.pyplot.boxplot(data, notch=None, vert=None, patch_artist=None, widths=None)

**Here you can see the arguments details:**

**data: array or sequence of array to be plotted**

**notch: optional parameter accepts Boolean values**

# Box Plot

**Creating Box Plot:**

**vert:** optional parameter accepts Boolean values false and true for horizontal and vertical plot respectively

**patch_artist :** optional parameter having boolean values

**Widths:** optional parameter accepts array and sets the width of boxes

# Box Plot

**Creating Box Plot:**

Example: We use the **numpy.random.normal()** function to create the fake data. It takes three arguments, mean and standard deviation of the normal distribution, and the number of values desired.

```
import matplotlib.pyplot as plt
import numpy as np
data = np.random.normal(100, 20, 5)
plt.boxplot(data)
plt.show()
```

# Box Plot

**Creating Box Plot:**

Example: Boxplot can be drawn calling DataFrame.plot.box(), or DataFrame.boxplot() to visualize the distribution of values within each column. For instance, here is a boxplot representing five trials of 10 observations of a uniform random variable on [0,1].

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.rand(10, 5),
columns=['A', 'B', 'C', 'D', 'E'])
df.plot.box(grid='True')
plt.show()
```