Data Visualization

Lecture 3

Matplotlib Visualization with Python

Matplotlib

- Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

- Most of the Matplotlib utilities lies under the pyplot submodule and we usually use plt alias to import them.

- How to install Matplotlib while using Jupyter?
  - C:\Users\Your name> pip install matplotlib

- Note: Anaconda python distribution has already Matplotlib installed.

Matplotlib – Basic Plotting

- Example: Draw a line in a diagram from position (0,0) to position (6,250):

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

xpoints = np.array([0, 6])
ypoints = np.array([0, 250])

plt.plot(xpoints, ypoints)
plt.show()
```

Matplotlib – Basic Plotting

- Exercise: Draw a line in a diagram from position (1, 3) to (2, 8) then to (6, 1) and finally to position (8, 10):

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

xpoints = np.array([1, 2, 6, 8])
ypoints = np.array([3, 8, 1, 10])

plt.plot(xpoints, ypoints)
plt.show()
```

Matplotlib – Basic Plotting

- Exercise: Plotting without x-points:

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

ypoints = np.array([3, 8, 1, 10, 5, 7])

plt.plot(ypoints)
plt.show()
```

- What will happen here?

Matplotlib Line

- Line style:
  - Example: Use a dotted line or a dashed line:

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

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, linestyle = 'dotted’) or linestyle = ‘dashed’
plt.show()
```

Matplotlib Line

- Line style shorter syntax:
  - Linestyle □ ls
  - Dashed □ --
  - Dotted □ :

- Example: plt.plot(ypoints, ls = ':')

Matplotlib Line

- More line styles:

Style	Shorter Syntax
Solid (default)	'-'
Dotted	':'
Dashed	'- -'
Dashdot	'- .'
None	'' or ' '

Matplotlib Line

- Line Color:
  - Example: Set the line color to green:

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

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, color = 'g') or c = 'g'
plt.show()
```

Matplotlib Line

- Line Width:
  - Example: Plot with a 20.5pt wide line:

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

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, linewidth = '20.5’) or lw = ’20.5’
plt.show()
```

Matplotlib Line

- Multiple lines:
  - Exercise: Draw two lines by specifying the x- and y-point values for both lines:

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

x1 = np.array([0, 1, 2, 3])
y1 = np.array([3, 8, 1, 10])
x2 = np.array([0, 1, 2, 3])
y2 = np.array([6, 2, 7, 11])

plt.plot(x1, y1, x2, y2)
plt.show()
```

Matplotlib Bar Plots

- Example: Draw a bar plot consisting of 4 bars:

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

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.bar(x,y)
plt.show()
```

Matplotlib Bar Plots

- Exercise: Change the color of the bars to red, make them horizontal with height=0.2:

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

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.barh(x,y,height=0.1,color='red')
plt.show()
```

Matplotlib Scatter Plots

- Example: A simple scatter plot:

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

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

plt.scatter(x, y)
plt.show()
```

Matplotlib Scatter Plots

- Exercise: Draw two plots on the same figure:

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

#day one, the age and speed of 13 cars:
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x, y)

#day two, the age and speed of 15 cars:
x = np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])
y = np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])
plt.scatter(x, y)

plt.show()
```

Matplotlib Scatter Plots

- Exercise: Create random arrays with 100 values for x-points, y-points, colors and sizes:

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

x = np.random.randint(100, size=(100))
y = np.random.randint(100, size=(100))
colors = np.random.randint(100, size=(100))
sizes = 10 * np.random.randint(100, size=(100))

plt.scatter(x, y, c=colors, s=sizes, alpha=0.5,
cmap='nipy_spectral')

plt.colorbar()

plt.show()
```

Matplotlib Bubble Plots

- A bubble plot is a scatterplot where the circle size is mapped to the value of a third numeric variable.
- Using matplotlib library, a bubble plot can be constructed using the scatter() function.
- In the example, the following parameters are used:
  - x: The data position on the x axis
  - y: The data position on the y axis
  - s: The marker size
  - alpha: Transparency ratio

Matplotlib Bubble Plots

- Example:

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

price = [600, 250, 45, 120, 80, 450]
rating = [4.7, 4.5, 4.8, 4.3, 3.8, 4.6]
units_sold = [800, 4500, 15000, 2500, 3000, 1200]

# Bubble sizes scaled (so they're visible)
bubble_sizes = [p * 0.1 for p in units_sold]

plt.scatter(price, rating, s=bubble_sizes, alpha=0.4, c='r')

plt.show()
```

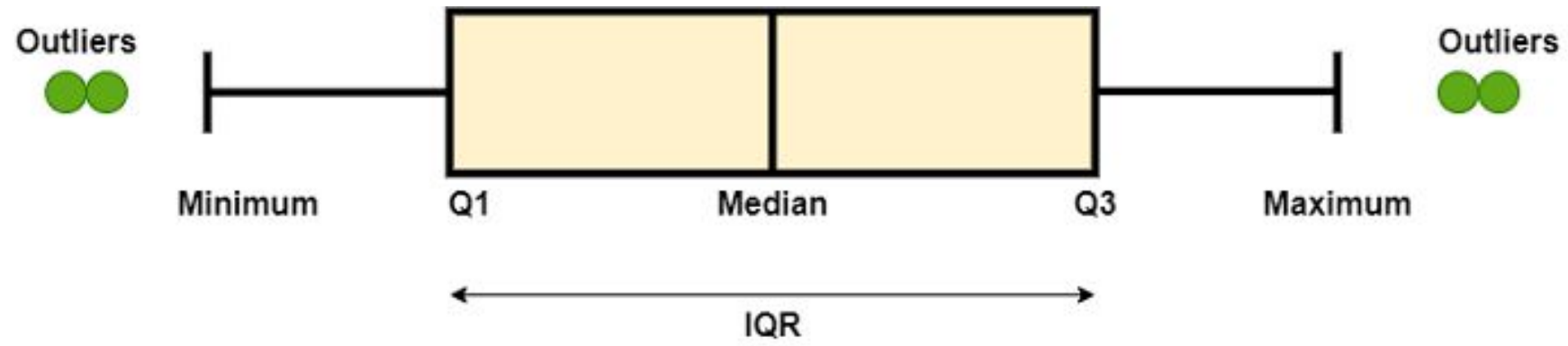
Matplotlib Box Plots

- A Box 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 vertical 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.

Matplotlib Box Plots



- In the box plot, those points which are out of range are called outliers. We can create the box plot of the data to determine the following:

  - The number of outliers in a dataset
  - Is the data skewed or not
  - The range of the data

Matplotlib Box Plots

- Syntax:

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

  - Attributes which can be used to create a more attractive and amazing box plot of the data set:
    - **data:** The data should be an array or sequence of arrays which will be plotted.
    - **notch:** This parameter accepts only Boolean values, either true or false.
    - **vert:** This attribute accepts a Boolean value. If it is set to true (1), then the graph will be vertical. Otherwise (False/0), it will be horizontal.
    - **patch_artist:** this parameter accepts Boolean values, either true or false, and this is an optional parameter to fill the box with color or not.
    - **widths:** It accepts the array of integers which defines the width of the box.

Matplotlib Box Plots

- Example: We will create the random data set of the numpy array and create the box plot.

```
import matplotlib.pyplot as plt
import numpy as np
np.random.seed(15)
dataSet = np.random.normal(100, 25, 200)
print(dataSet)
figure = plt.figure(figsize =(10, 8))
plt.boxplot(dataSet)  or
plt.boxplot(dataSet,vert=0,patch_artist=True,notch='False')
plt.show()
```

Matplotlib Box Plots

- Exercise: Create multiple box plots simultaneously in the same file.

```
import matplotlib.pyplot as plt
import numpy as np
np.random.seed(10)
dataSet1 = np.random.normal(100, 10, 220)
dataSet2 = np.random.normal(80, 20, 200)
dataSet3 = np.random.normal(60, 35, 220)
dataSet4 = np.random.normal(50, 40, 200)
dataSet = [dataSet1, dataSet2, dataSet3, dataSet4]
figure = plt.figure(figsize =(10, 7))
ax = figure.add_axes([0, 0, 1, 1])
bp = ax.boxplot(dataSet)
plt.show()
```

Matplotlib Waffle Charts

- Waffle charts can be an interesting element in a dashboard.
- It is especially useful to display the progress towards goals and seeing how each item contributes to the whole.
- They are not very useful if you try to put too many items in it.
- Install using jupyter notebook:

```
!pip install pywaffle
```

Matplotlib Waffle Charts

- Example:
  - First, import Waffle from pywaffle:

```
from pywaffle import Waffle
```

  - Second, create a DataFrame. It will fakely represent the number of immigrants from Argentina, Brazil, Cuba, and Peru to the US.

```
df = pd.DataFrame({
    'country': ['Argentina', 'Brazil', 'Cuba', 'Peru'],
    'number': [212, 334, 1500, 130]
})
```

Matplotlib Waffle Charts

- Example:
  - Finally, plot the figure:

```
fig = plt.figure(
    FigureClass=Waffle,
    rows=10,
    values=list(df.number/5),
    labels=list(df.country),
    figsize=(12, 8),
    legend={'bbox_to_anchor': (0.5, 0.5)}
)
```

Matplotlib Wordclouds

- It is basically a visualization technique to represent the frequency of words in a text where the size of the word represents its frequency.

- To work with wordclouds in python we should install some libraries:
  - pip install numpy
  - pip install pillow image handeling
  - pip install matplotlib generating plots
  - pip install wordcloud generating wordclouds

Matplotlib Wordclouds

- Example: We will use the Restaurant reviews data from Kaggle

  - First, we will import needed libraries:

```
import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS
```

  - Second, read the data as a dataframe using pandas:

```
df = pd.read_csv(r'E:/Restaurant_Reviews.tsv',sep='\t')
```

Matplotlib Wordclouds

- Example: We will use the Restaurant reviews data from Kaggle

  - The data contains two columns namely 'Review' and 'Liked'. The 'Review' column is the actual review written by the customer and the 'Liked' column is a binary variable which states whether or not the customer liked the food.
  - Then we will generate a text variable which contains all the reviews combined as a single string, which is done using python join function:

```
text = " ".join(cat for cat in df.Review)
```

Matplotlib Wordclouds

- Example: We will use the Restaurant reviews data from Kaggle

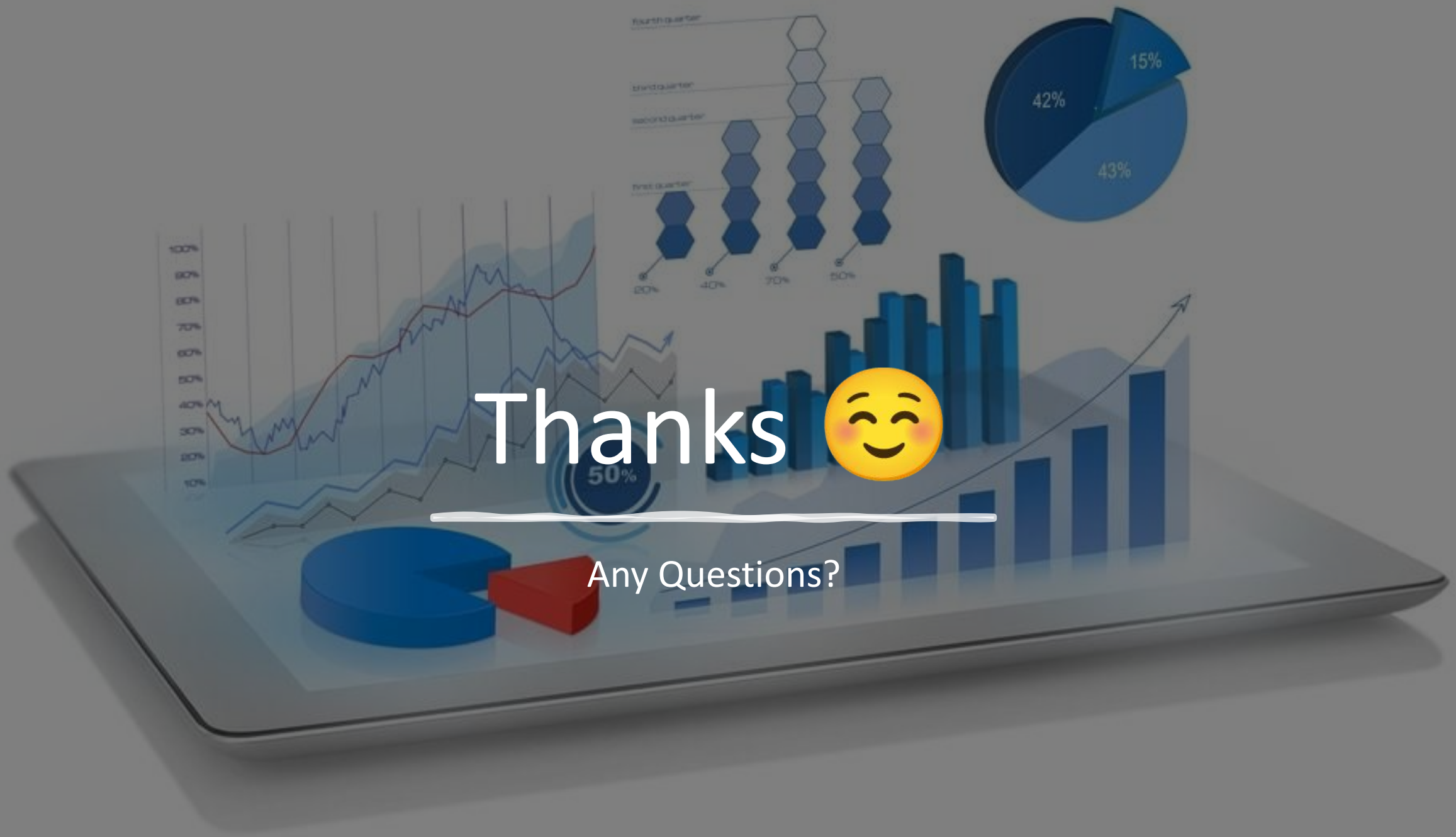
  - Finally, generate the wordcloud and display it:

```
word_cloud = WordCloud(
        width=3000,
        height=2000,
        random_state=1,
        background_color="white",
        colormap="viridis",
        collocations=False,
        stopwords=STOPWORDS,
        ).generate(text)

plt.imshow(word_cloud)
plt.axis("off")
plt.show()
```

Matplotlib Wordclouds

- The WordCloud function provides a lot of parameters that we can choose from:
  - **width/height:** To adjust the height and width of the wordcloud.
  - **random_state:** To recreate the same plot every time we run the function. The random_state parameter has to be an integer value.
  - **background_color:** To set a background_color. The default value for this parameter is 'black'.
  - **colormap:** To set up the color theme for the words.
  - **collocations:** To include bigrams of two words when set to True. The default value is True
  - **stopwords**: To set the list of words that needs to be eliminated. This list can include trivial words like this, that, is, was, the, etc. If this parameter is set to None, then function will consider a built-in list of STOPWORDS.
  - **max_font_size:** To set the maximum font size of the largest word.
  - **normalize_plurals:** To keep or remove the trailing 's' from the words.



Thanks 😊

Any Questions?