Data visualization with Python [Module-3]

→ how to create meaningful, effective and aesthetically pleasing data visuals and plots in python using matplotlib and a couple of other libraries namely seaborn and Folium.

Module 1: matplotlib

stored in pandas dataframe.

Pandas is a python library for data manipulation and analysis.

Module 2: Area plots, histograms, and bar charts

→ create different versions of these plots.

Pie charts, box plots, scatter plots, bubble plots

Module 3:- ~~cat~~ clouds, regression plots

folium which was built primarily to visualize geospatial data.

* Introduction to data visualization

eg:- A transforming a given visual into one which one effective, attractive, and impactive.

Why Build visuals:-

converting complex data in a form that is graphical and easy to understand

1. For exploratory data analysis.
2. Communicate data clearly
3. Share unbiased representation of data
4. Use them to support recommendations to different stakeholders

Best Practices :-

Darkhorse Analytics is a company that spun out of a research lab at the university of Alberta in 2008 and done fasinating work on data visualization. It specializes in quantitative consulting in several areas including data visualization and geo spatial analysis.

When creating a visual, always remember :

1. Less is more effective
2. Less is more attractive
3. Less is more impactive

* comparison bt^n Darkhorse Analysis and pie chart

It is simple, clear,
less distracting, and
much easier to read

* Introduction to Matplotlib

It is most widely used, if not the most popular data visualization library in python.

→ It was created by John Hunter (1968-2012), who was a neurobiologist and reserch in ECOG.

→ It using a proprietary software for the

→ Replaced by MATLAB based version

Date

matplotlib was originally developed as an ECoG visualization tool,

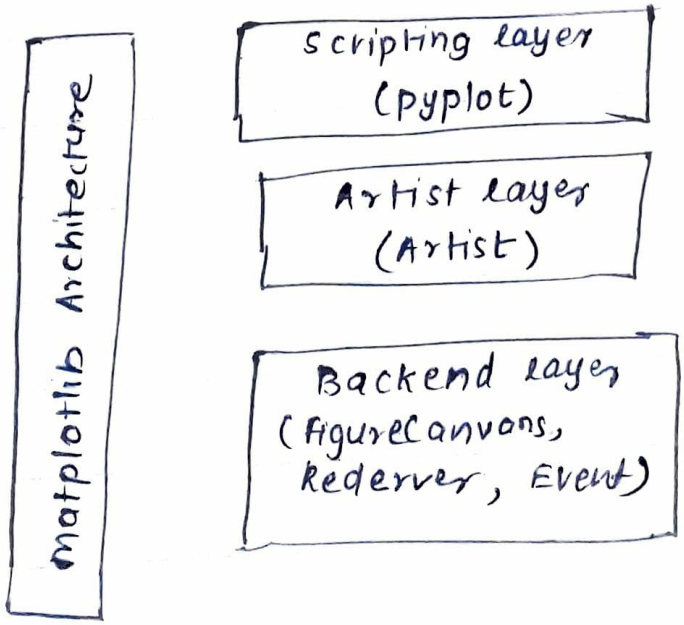
matplotlib was equipped with a scripting interface for quick and easy generation of graphics, represented by pyplot.

* Matplotlib Architecture

3 main layer :- the back-end layer, (where much of the heavy lifting happens)

It is usually the ← the artist layer appropriate programming paradigm writing a web application server, or a UI application, or perhaps a script to be shared with other developers,

→ scripting layer .... which is the appropriate layer for everyday purposes and is considered a lighter scripting interface to simplify common tasks and for a quick and easy generation of graphics and plots.



→ Backend Layer :-

Has 3 built-in abstract interface classes :

1. Figure Canvas : matplotlib.backend_bases.

Figure Canvas

→ Encompasses the area onto which the figure is drawn

2. Renderer : matplotlib.backend_bases.Renderer

→ Knows how to draw on the Figure Canvas

3. Event : matplotlib.backend_bases.Event

→ Handles user i/ps such as keyboard strokes and mouse clicks.

Artist Layer :-

- Comprised of one main object - Artist :

→ knows how to use the Renderer to draw on the canvas.

- Titles, lines, tick lables, and images, all correspond to individual Artist instances.
- Two types of Artist objects :

1. Primitive : Line2D, Rectangle, Circle and Text
2. composite : Axis, Tick, Axes, and figure

↓

composite artist

(because matplotlib API plotting methods are defined, including methods to crete and manipulate the Figs,

- Each composite artist may contain other composite artists as well as primitive artists.

* Putting the Artist Layer to use :

→ Let's try to generate a histogram of some data using the Artist layer :-

agg stands for anti grain geometry which is from matplotlib a high-performance library that produces attractive images.

* import the Numpy library to generate the random numbers.

(111) → one row, cloumn and uses the 1st cell in that grid for the location of axes.

Hist creates a sequence of rectangle artists for each histogram bar and adds them to the axes container.

Scripting layer :- Who was not programmers

- Comprised mainly of pyplot, a scripting Interface that is lighter that the Artist layer.
- Let's see how we can generate the same histogram of 10000 random values using the pyplot interface.

```
import matplotlib.pyplot as plt
import numpy as np
x = np.random.randn(10000)
plt.hist(x, 100)
plt.title(r'Normal distribution wis $\mu=0,
          \sigma=1$')
plt.savefig('matplotlib-histogram.png')
plt.show()
```

Basic Plotting with Matplotlib

Matplotlib - Jupyter Notebook

It is a well-established data visualization library that is well supported in different environments such as python scripts, in the iPython shell, web application servers, in graphical user interface toolkits as well as the Jupyter Notebook.

→ In Jupyter notebook, all you have to do is import Matplotlib

* In this course → scripting interface

Visualization tools using the scripting interface.

magic function :- A magic function starts with % matplotlib, and to enforce plots to be rendered within the browser, you pass in inline as the backend.

→ Matplotlib has a no. of different backends available.

Dis :- You cannot modify a figure once it's rendered.

A backend that overcomes this limitation is the notebook backend.

With the notebook backend in place, if a plt function is called, it checks if an active figure exists and any function called will applied to this figure.

matplotlib - Pandas

Pandas has a built-in implementation of it. Calling the plot function on a given pandas series or dataframe.

⇒ india-china_df.plot(kind="line")

⇒ india_china_df["India"].plot(kind="hist")

* Dataset on Immigration

Dataset

→ The poplution Division of the United Nations compiled data pertaining to 45 countries.

→ For each country, annual data on the flows of international migrats is reported in addition to other metadata.

→ We will primarily work with a United Nations data on immigration to Canada.

* Read Data into Pandas DataFrame

```
import numpy as np # useful for many scientific computing in python
import pandas as pd # primary data structure library
from __future__ import print_function # adds compatibility to python 2
# install xlrd
print('xlrd installed')
```

→ Required to extract data from Excel spreadsheet files.

```
df_can = pd.read_excel (
    "          url          "
    sheetname = "Canada by Citizenship",
    skiprows = rang(20),
    skip_footer = 2)
```

* Display Dataframe :-

```
df_can.head()
```

→ head function to display the first five rows of the dataframe.

Line Plots :- A line plot is a type of plot which displays information as a series of data points called 'markers' connected by straight line segments.

→ One of the most basic type of chart

When to use line plots :-

The best use case for a line plot is when you have a continuous dataset and you're interested in visualizing the data over a period of time.

Creating Line Plots

```
import matplotlib as mp1
import matplotlib.pyplot as plt
years = list(map(str, range(1980, 2014)))
df_canada.loc['Haiti', years].plot(kind = 'line')
plt.title('          ')
plt.ylabel('          ')
plt.xlabel('    ')
plt.show()
```