Machine Learning
CSCE 5215

Introductory Data Visualization for Machine Learning

Instructor: Zeenat Tariq

Data Visualization

- Transforming data into visual mode
- Exposing patterns to the eye
- Convey the right information without distortions

Benefits of DataVis:

- Save time
- Group attributes together
- Enhance analysis
- Optimize computational resources

CLARITY	ACCURACY	RELEVANCE

Why Data Visualization

“A picture is worth a thousand words”

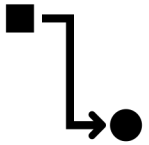
- First step of analysis work.
- It gives intuitive understanding of data.
- Helps you to see data in certain meaningful patterns.
- Visual representations enhances the human cognitive process.

Some more Benefits

- Allow users to see several different perspectives of data
- Makes it possible to interpret vast amounts of data.
- It offers ability to note expectations in data
- Exploring trends within a database through visualization by letting analysts navigate through data and visually orient themselves to the patterns in the data

Data Visualization considerations

Visual effects



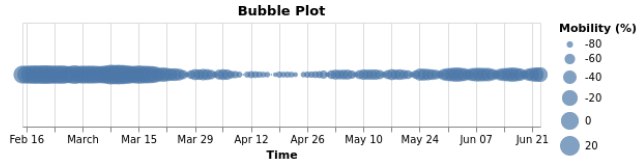
Data types

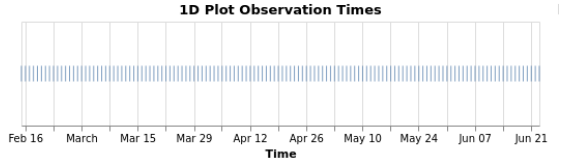
Scale

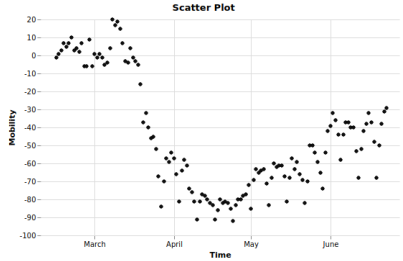
Informative interpretation

Common Visualizations

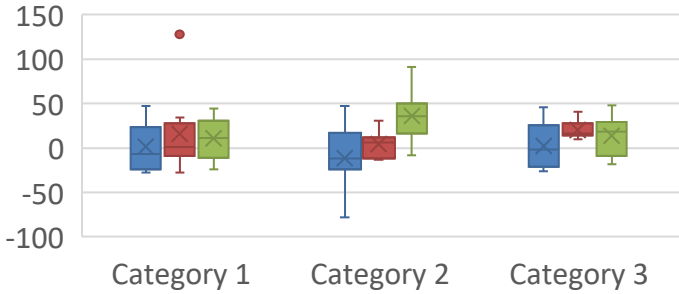
Data Table



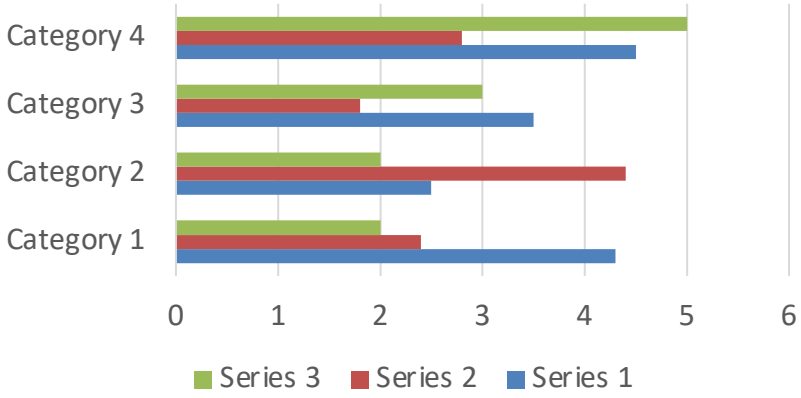


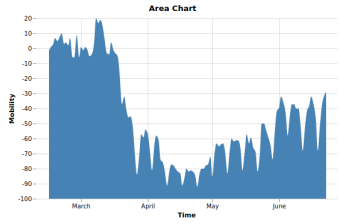


Box plot

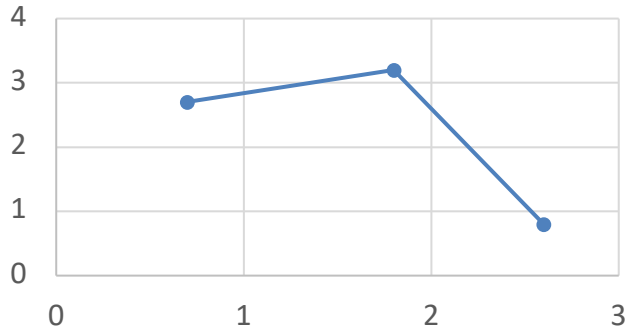


Bar plot / histogram

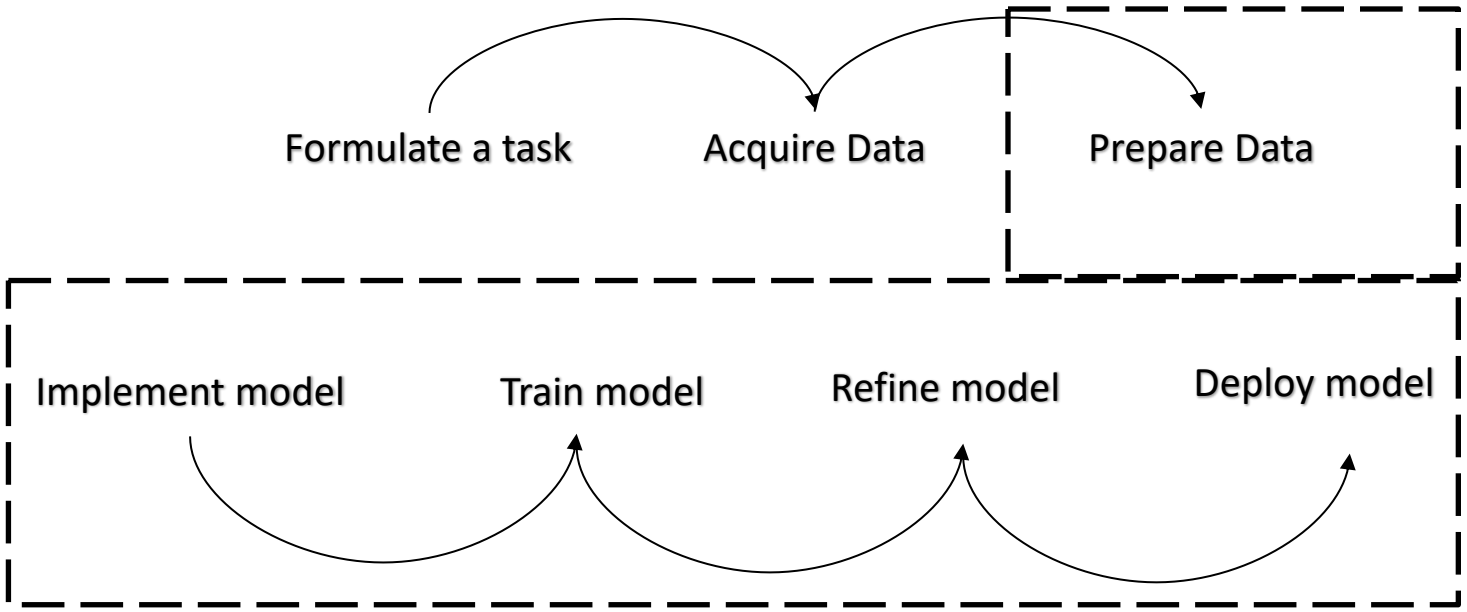




Line graph

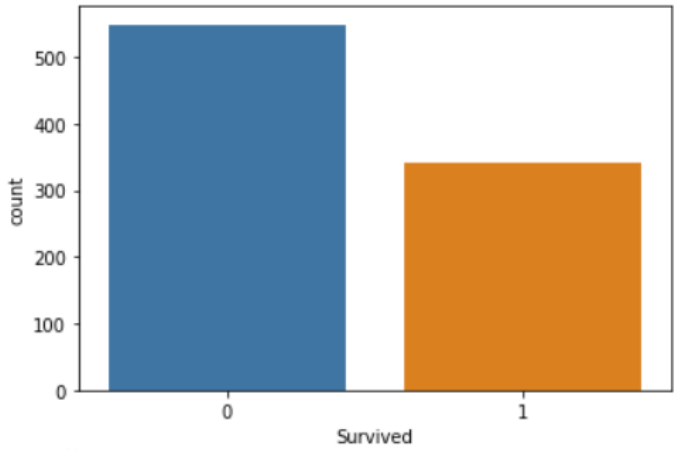


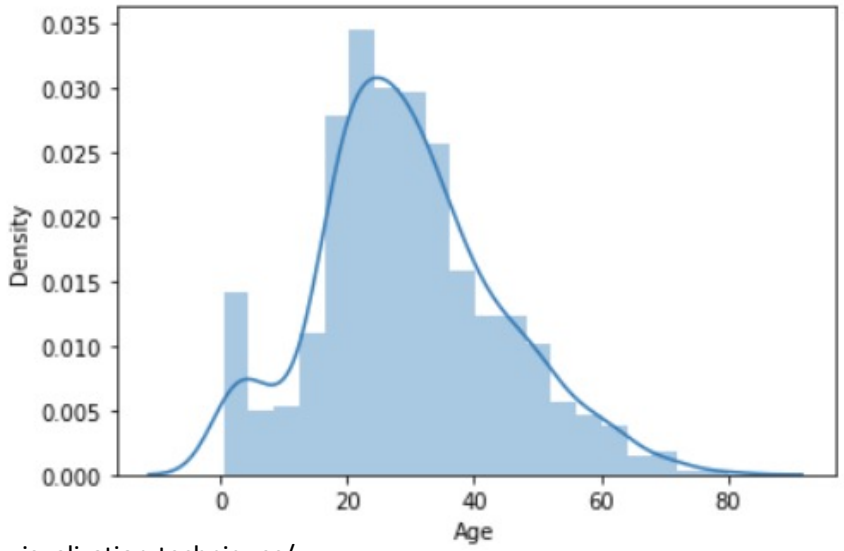
Data Visualization in Machine Learning



Exploratory Data Analysis

```
import numpy as np
import pandas pd
import matplotlib.pyplot as plt
import seaborn as sns
from seaborn import load_dataset
#titanic dataset
data = pd.read_csv("titanic_train.csv")
#tips dataset
tips = load_dataset("tips")

sns.countplot(data['Survived'])
plt.show()

sns.distplot(data['Age'])
plt.show()
```



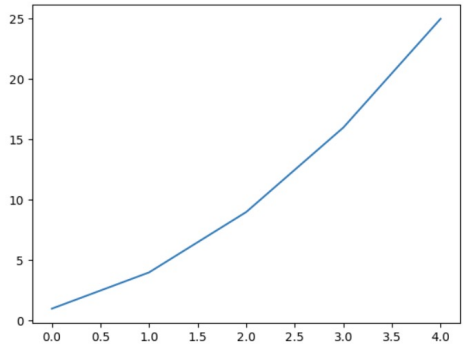


Ref: https://www.analyticsvidhya.com/blog/2021/06/exploratory-data-analysis-using-data-visualization-techniques/

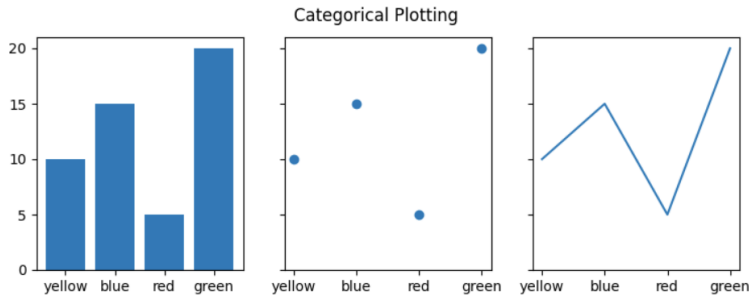
Libraries

- Matplotlib: Popular python library → collection of functions
- Others: plotly, ggplot, seaborn, geoplotlib, pygal, gleam, pandas

```
import matplotlib.pyplot as plt
# Plot some numbers:
plt.plot([1, 4, 9, 16, 25])
# Display the plot:
plt.show()
```




```
import matplotlib.pyplot as plt

data = {'yellow': 10, 'blue': 15, 'red': 5,
'green': 20}
names = list(data.keys())
values = list(data.values())

fig, axs = plt.subplots(1, 3, figsize=(9,
3), sharey=True)
axs[0].bar(names, values)
axs[1].scatter(names, values)
axs[2].plot(names, values)
fig.suptitle('Categorical Plotting')
plt.show()
```



Matplotlib

- Matplotlib is an amazing visualization library in Python for 2D plots of arrays
- Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.
- It was introduced by John Hunter in the year 2002.
- One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals.
- Matplotlib consists of several plots like line, bar, scatter, histogram etc

Basic plots in Matplotlib

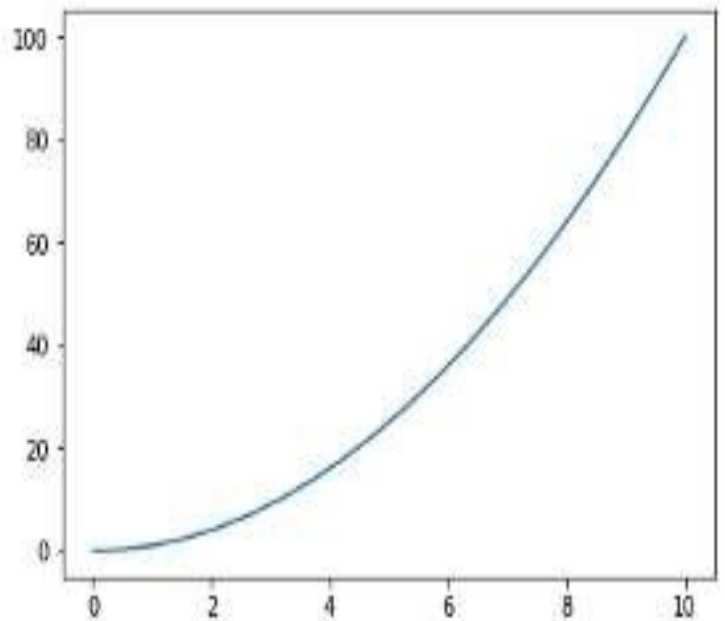
- Matplotlib comes with a wide variety of plots.
- Plots helps to understand trends, patterns, and to make correlations
- They’re typically instruments for reasoning about quantitative information.

Functional Approach:

```
In [3]: import numpy as np
        x = np.linspace(0, 10, 20) #Generate 20 datapoints between 0 and 10
        y = x**2                   #Generate array 'y' from square of 'x'
```

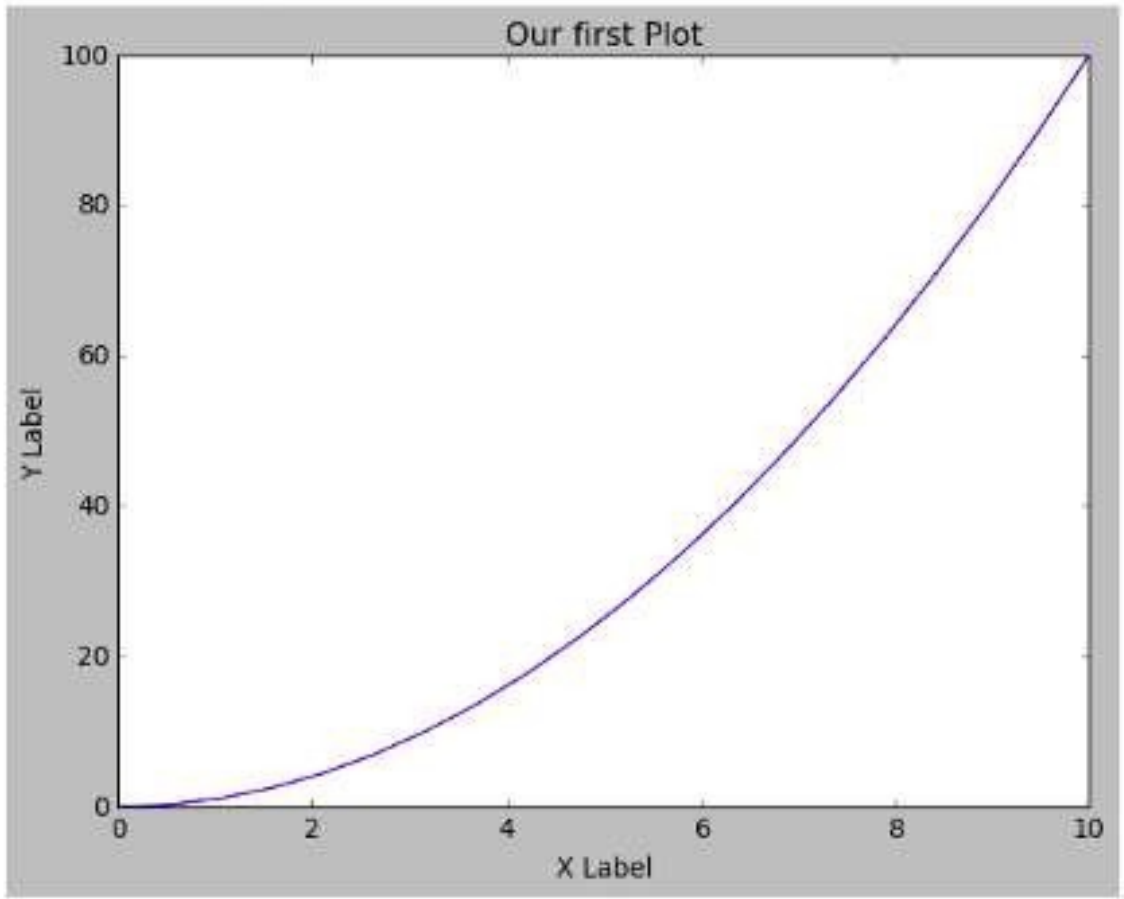
```
In [4]: plt.plot(x,y)
```

```
Out[4]: [<matplotlib.lines.Line2D at 0x7f5a339fcd30>]
```



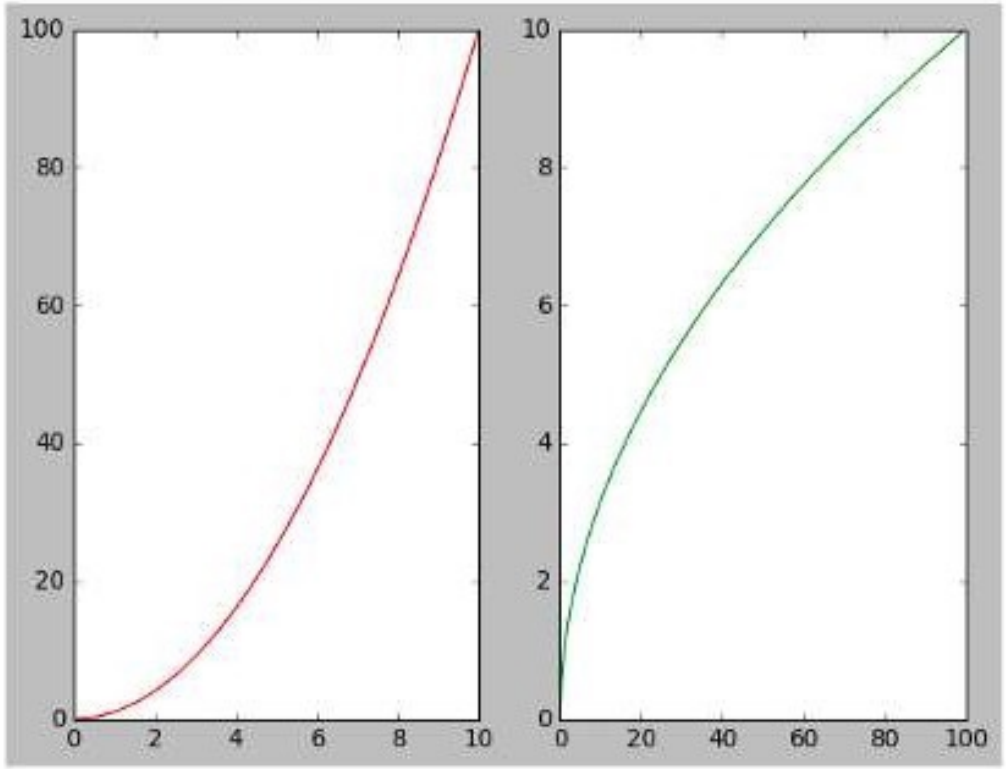
In [12]:
```
plt.plot(x,y)
plt.title('Our first Plot')
plt.xlabel('X Label')
plt.ylabel('Y Label')
```

Out[12]:
```
Text(0,0.5,'Y Label')
```



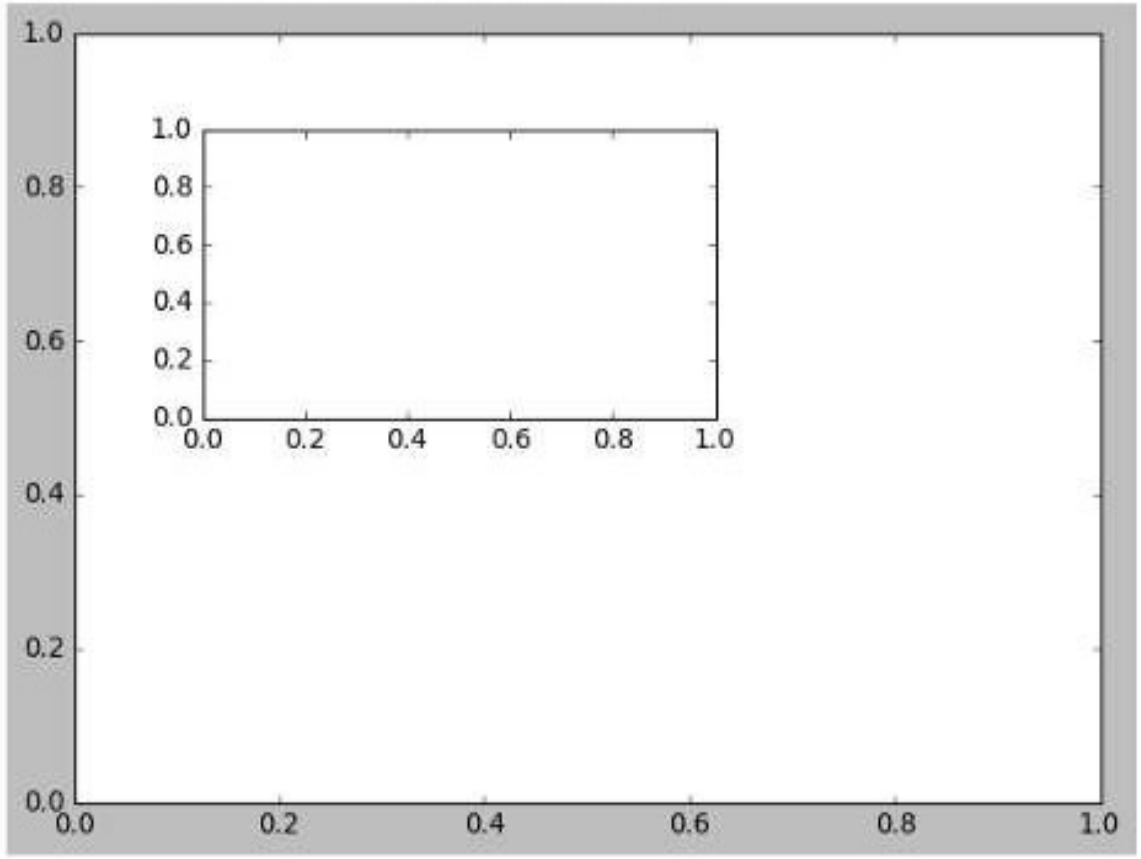
Using .subplot() we will create a two plots on the same canvas:

In [13]:
```
# plt.subplot(nrows, ncols, plot_number)
plt.subplot(1, 2, 1)
plt.plot(x, y, 'red') # More on color options later

plt.subplot(1,2,2)
plt.plot(y, x, 'green');
```



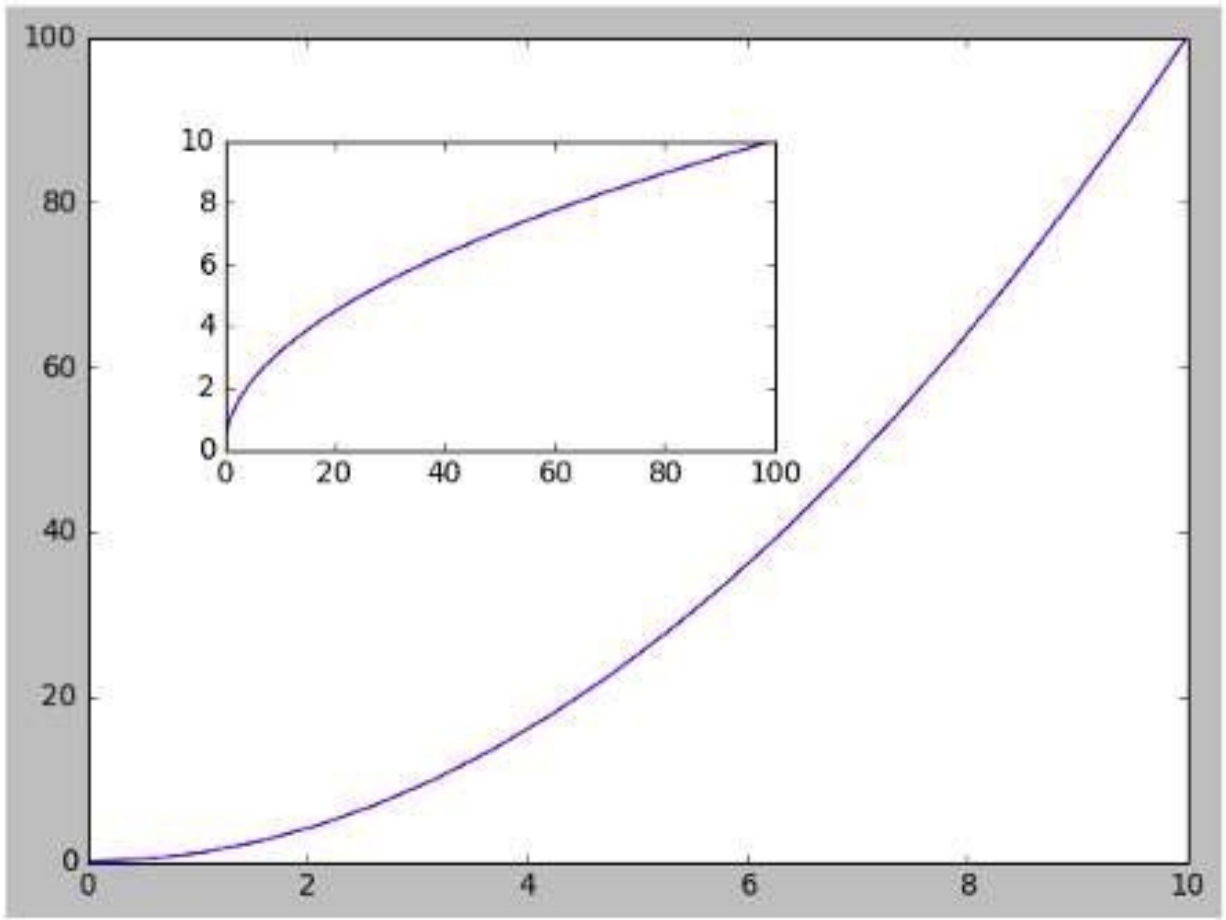
Something interesting ,figure in figure

```
In [24]: fig = plt.figure()

         axes1 = fig.add_axes([0.1, 0.1, 0.8, 0.8])
         axes2 = fig.add_axes([0.2, 0.5, 0.4, 0.3])
```



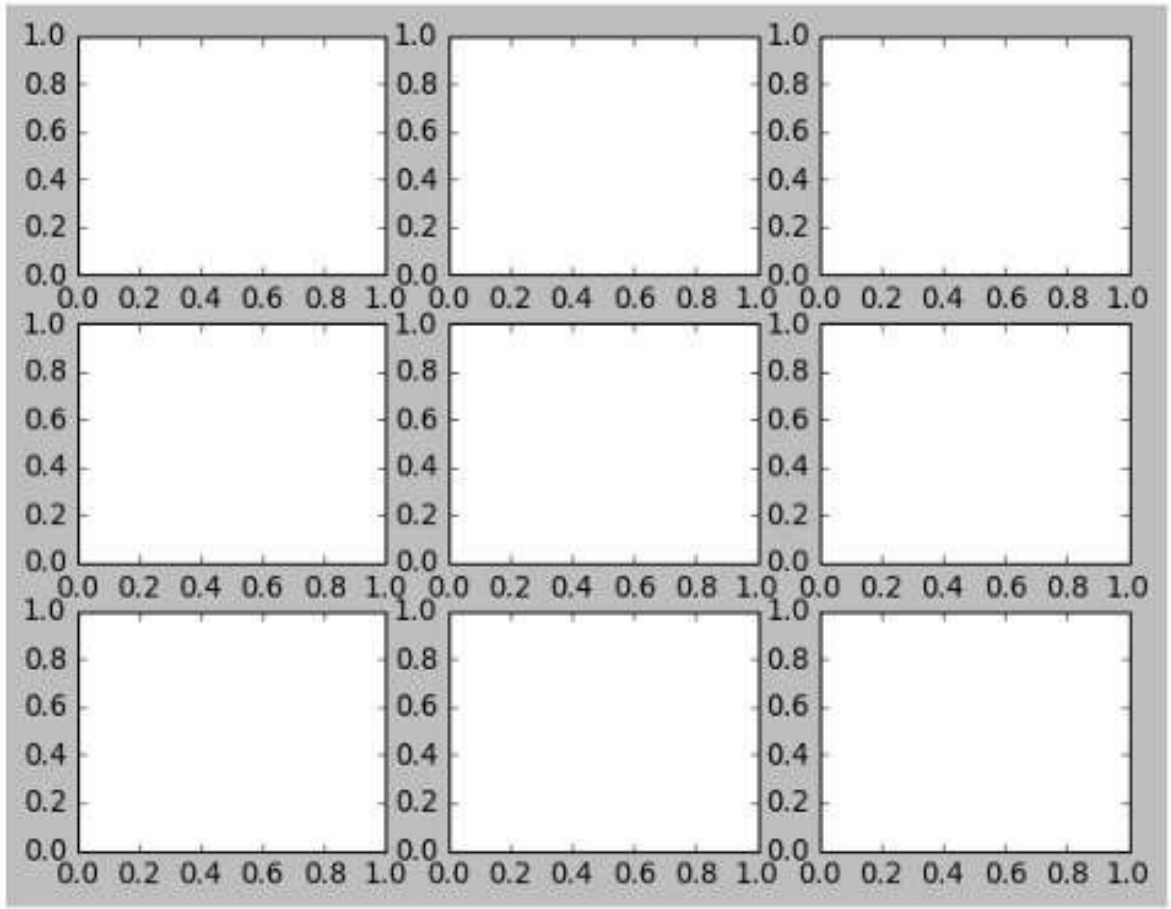
In [26]:
```
fig = plt.figure()

axes1 = fig.add_axes([0.1, 0.1, 0.8, 0.8])
axes2 = fig.add_axes([0.2, 0.5, 0.4, 0.3])

axes1.plot(x,y)
axes2.plot(y,x)
```

Out[26]:
```
[<matplotlib.lines.Line2D at 0x7f00630991d0>]
```



We can create a matrix of subplot for example 3*3

```
In [30]: # Empty canvas of 3 by 3 subplots
         fig, axes = plt.subplots(nrows=3, ncols=3)
```

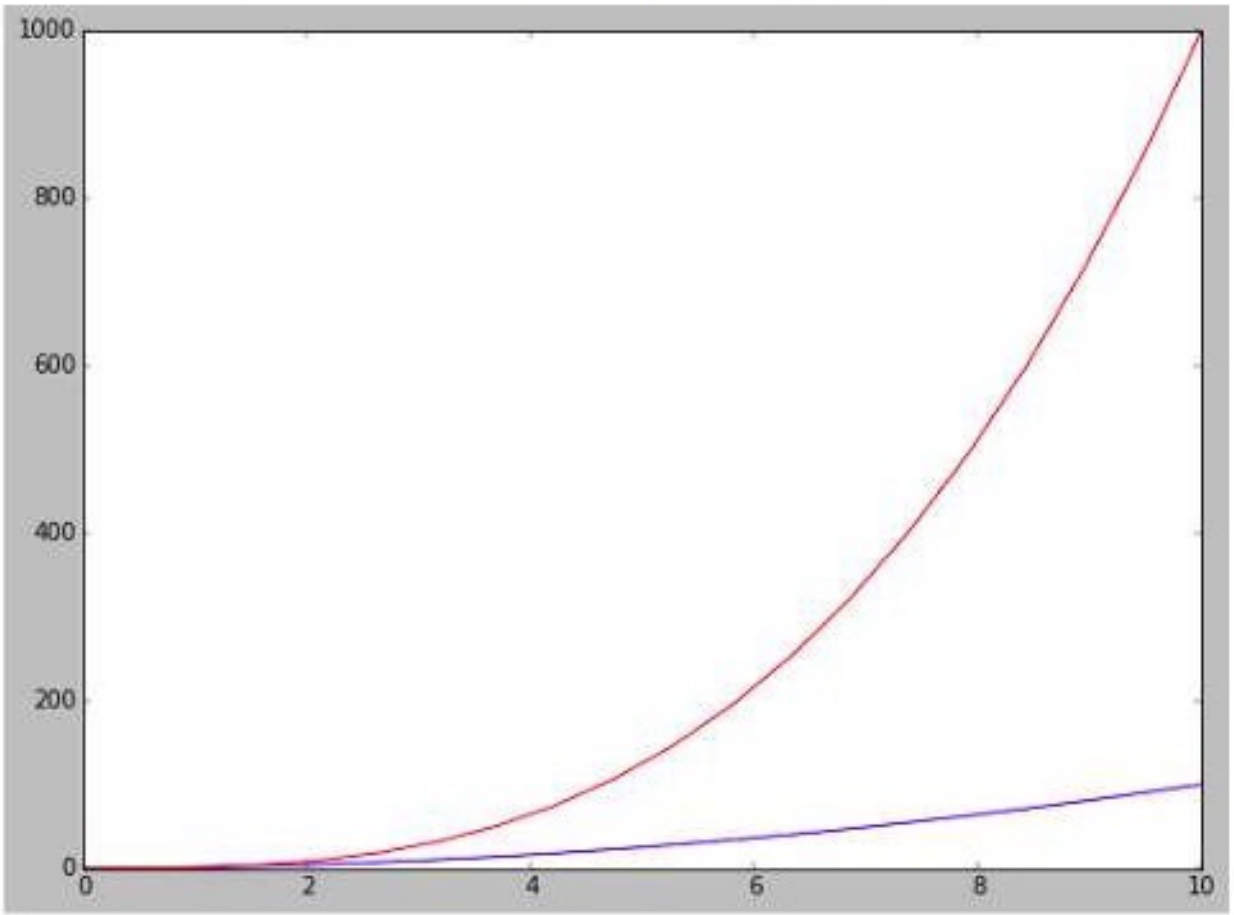


Legends

Legends allows us to distinguish between plots. With Legends, you can use label texts to identify or differentiate one plot from another. For example, say we have a figure having two plots

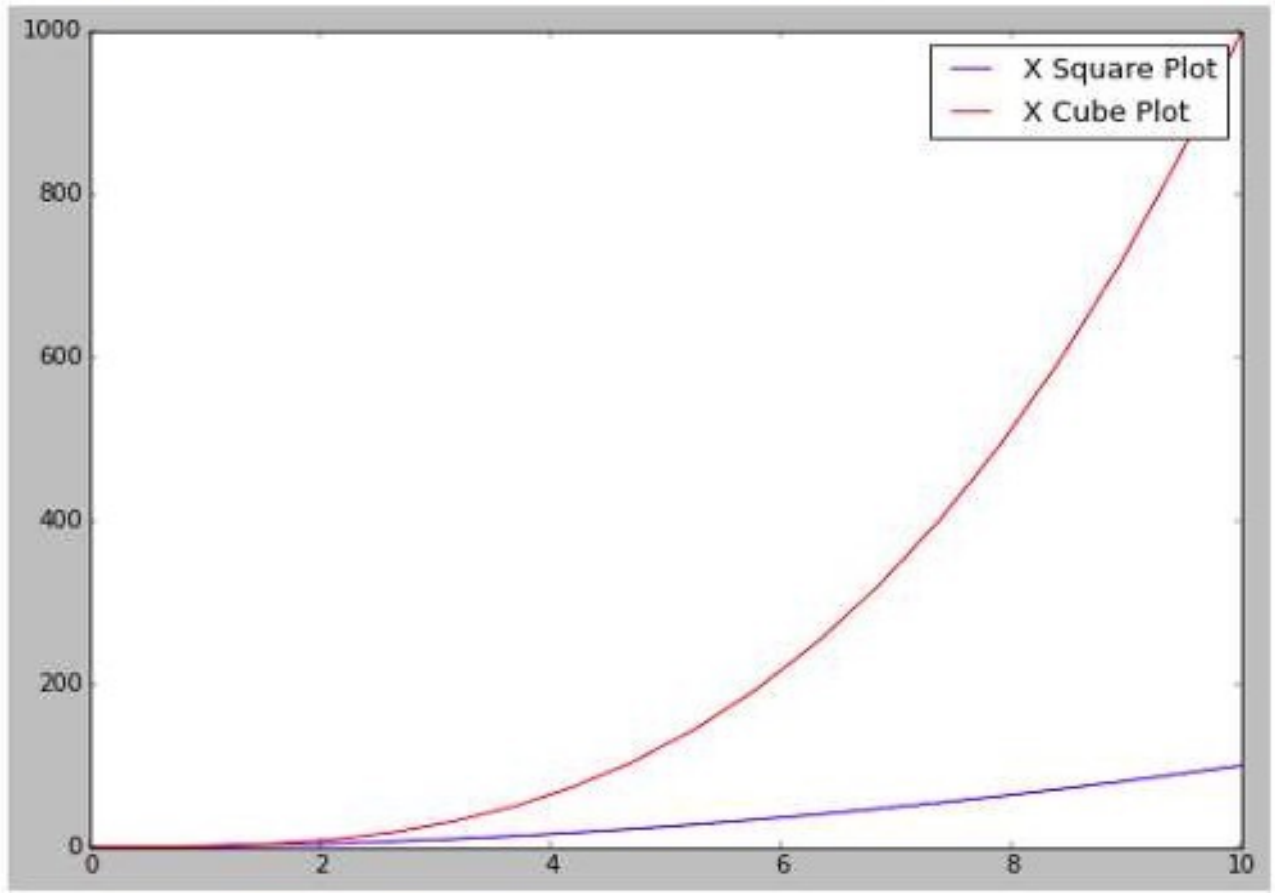
In [74]:
```
fig = plt.figure(figsize=(8,6), dpi = 60)
ax = fig.add_axes([0,0,1,1])
ax.plot(x,x**2)
ax.plot(x,x**3, 'red')
```

Out[74]:
```
[<matplotlib.lines.Line2D at 0x7f0062338f60>]
```



In [75]:
```
fig = plt.figure(figsize=(8,6), dpi = 60)
ax = fig.add_axes([0,0,1,1])
ax.plot(x,x**2, label="X Square Plot")
ax.plot(x,x**3, 'red', label='X Cube Plot')

ax.legend()
```

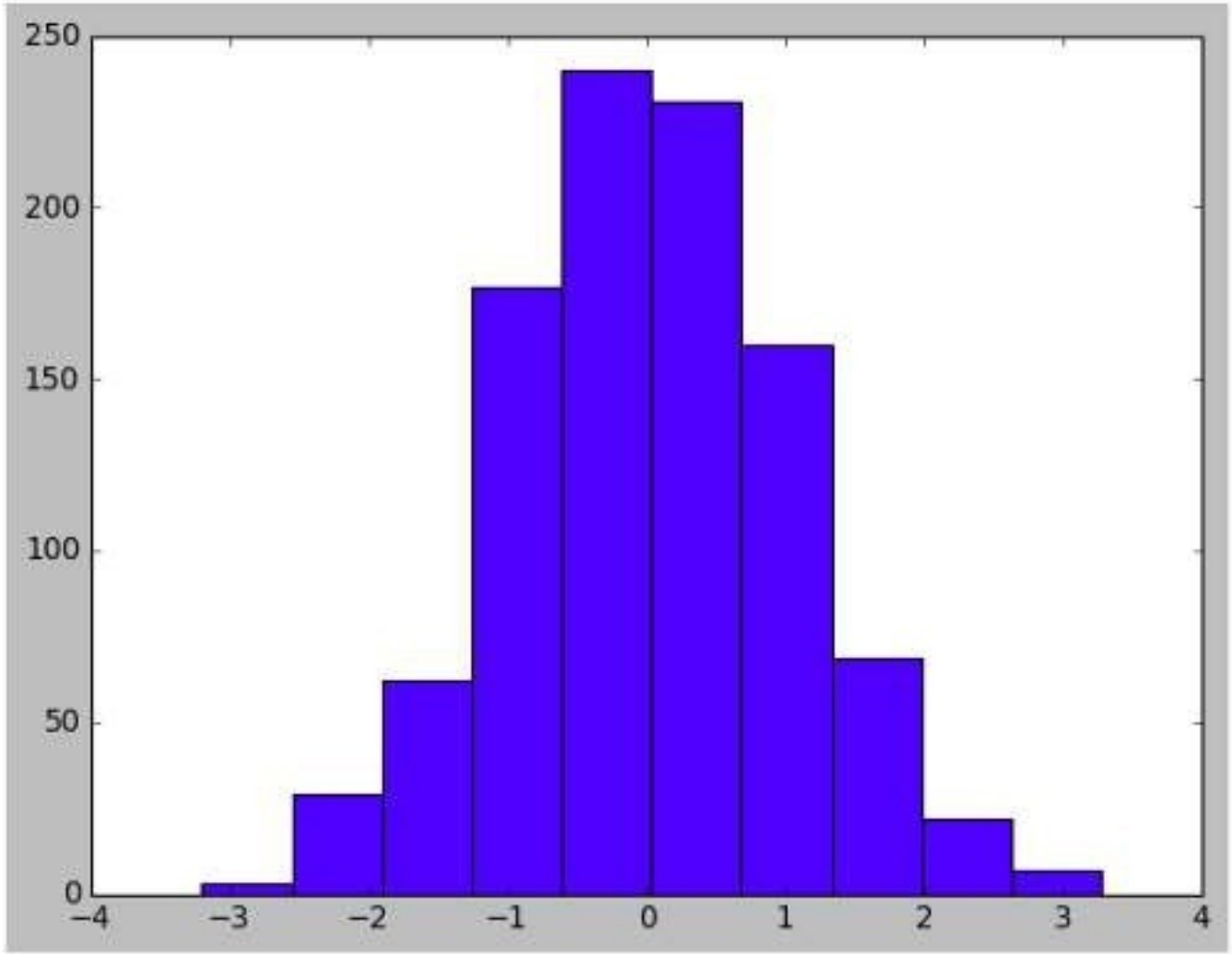
Out[75]: <matplotlib.legend.Legend at 0x7f0061e0ee80>



Plot Types

- Histogram
- Helps us understand the distribution of numeric value in a way that you can not do with mean, median and mode.

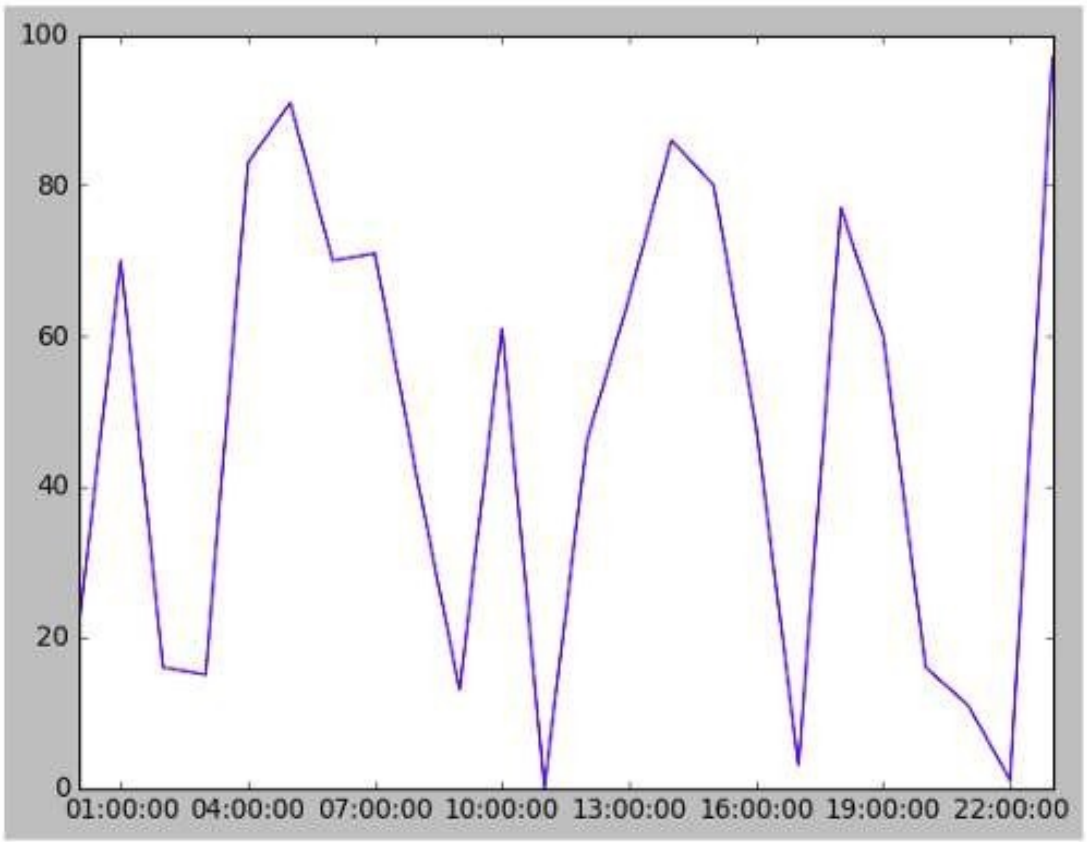
In [90]:
```
x = np.random.randn(1000)
plt.hist(x);
```



Time series (LinePlot)

- A chart that shows a trend over a period of time
- It allows you to test various hypotheses under certain conditions, like what happens different days of the week or between different times of the day

In [227]:
```
import matplotlib.pyplot as plt
import datetime
import numpy as np

x = np.array([datetime.datetime(2018, 9, 28, i, 0) for i in range(24)])
y = np.random.randint(100, size=x.shape)

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

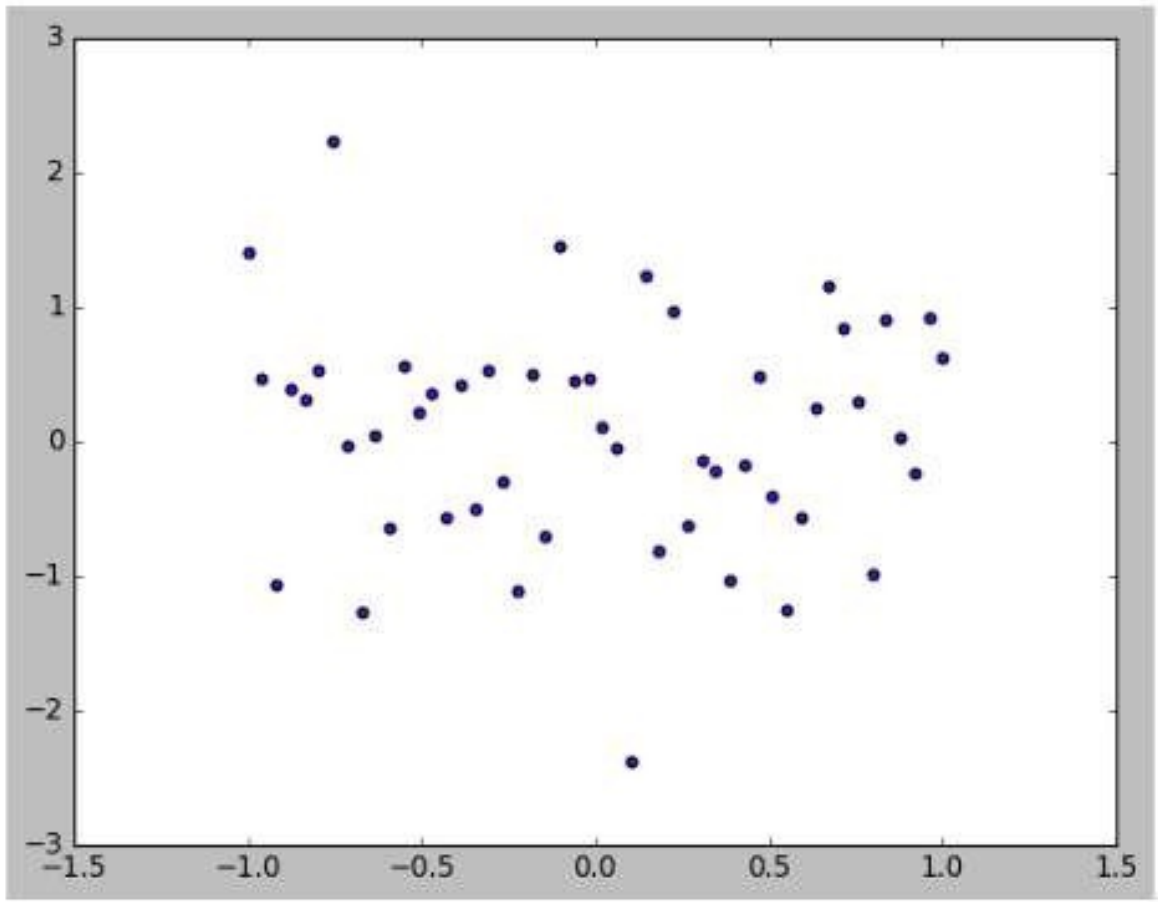


Scatter plots

- Offer a convenient way to visualize how two numeric values are related in your data
- It helps in understanding relationships between multiple variables
- Using .scatter() method, we can create a scatter plot:

In [214]:
```
fig, ax = plt.subplots()
x = np.linspace(-1, 1, 50)
y = np.random.randn(50)
ax.scatter(x, y)
```

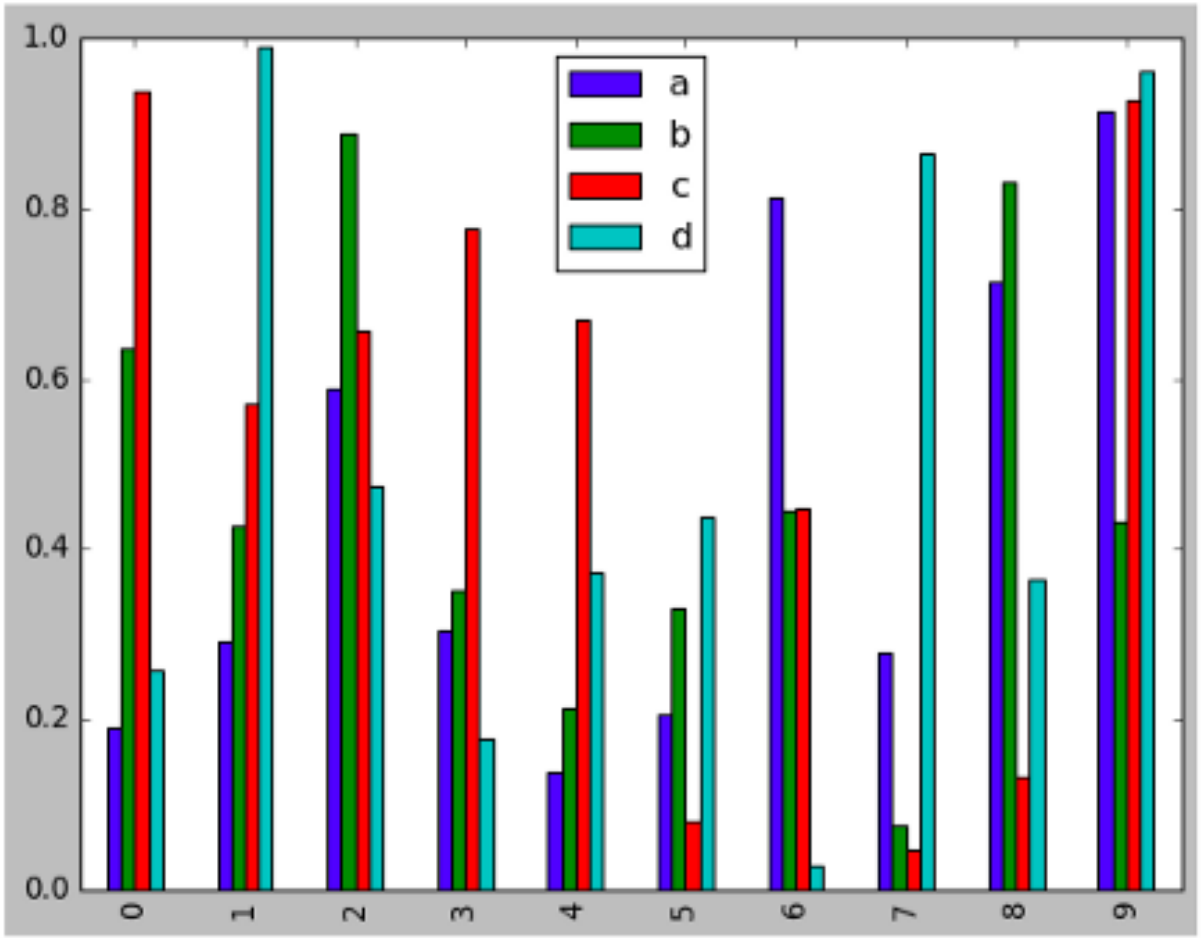
Out[214]:
```
<matplotlib.collections.PathCollection at 0x7f004eaa30f0>
```



Bar graphs

Convenient for comparing numeric values of several groups. Using .bar() method, we can create a bar graph:

In [216]:
```
my_df = pd.DataFrame(np.random.rand(10, 4), columns=['a', 'b', 'c', 'd'])

my_df.plot.bar();
```



References

- https://www.simplifiedpython.net/data-visualization-python-tutorial/
- https://matplotlib.org/
- https://mode.com/blog/python-data-visualization-libraries/
- https://towardsdatascience.com/top-6-python-libraries-for-visualization-which-one-to-use-fe43381cd658