# ML 4 - Data Interpolation By Virat Tiwari

November 7, 2023

## 1 Data Interpolation

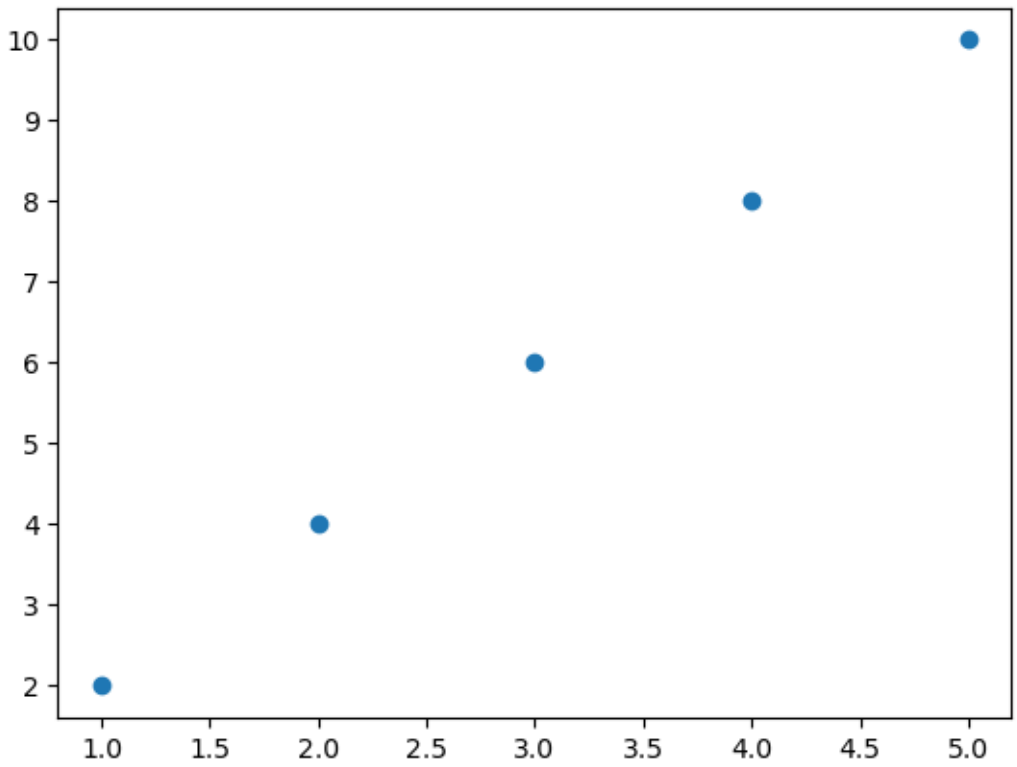
Data Interpolation is the process of estimating unknown values within a dataset based on the known values . In python there are various libraries available that can be used for a data interpolation such as numpy , scipy and pandas . Here is an example of how to perform data interpolation using the numpy library.

TYPE 1 - LINEAR INTERPOLATION USING NUMPY

[5]:
```
# x and y are datapoints that creates the dataset

# We basically create a data in the form of x and y

import numpy as np
x=np.array([1,2,3,4,5])
y=np.array([2,4,6,8,10])
```

[6]:
```
# Import matplotlib . pyplot as plt - for visualizing datapoints x and y

# plt . sactter ( )  function is used for presenting the datapoints in scatter␣
↪graph

# wit

import matplotlib.pyplot as plt
plt.scatter(x,y)
```

[6]:
```
<matplotlib.collections.PathCollection at 0x7f0ef5dabf10>
```




[7]:
```
# NOTE - INE THE PREVIOUS X AND Y ARE LINEAR TO EACH OTHER THATS WHY WE GRAPGH␣
↪GIVES STARINGHT DOT LINE

# NOTE - FOR CREATING DATAPOINTS IN BETWEEN DATAPOINTS WE SPECIFICALLY USE "␣
↪LINEAR INTERPOLATION TECHNIQUE "
```

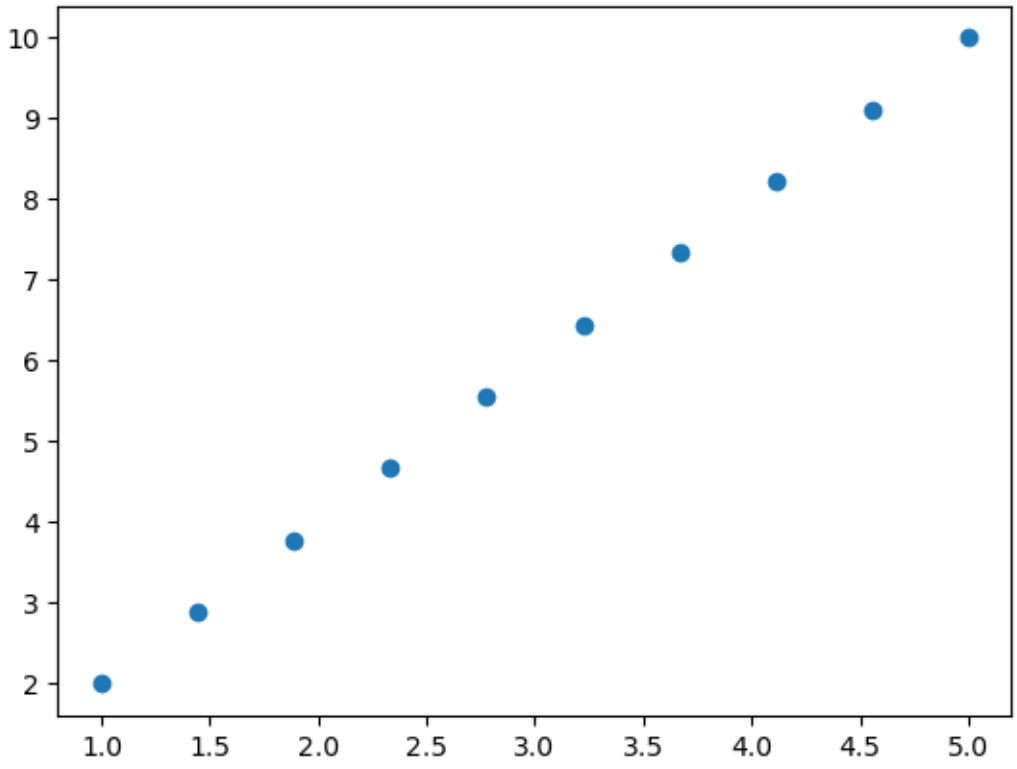
[9]:
```
# INTERPOLATE

# we create new values

# x_new and y_interp are our new values

x_new=np.linspace(1,5,10)
y_interp=np.interp(x_new,x,y)
print(y_interp)
```

```
[ 2.          2.88888889  3.77777778  4.66666667  5.55555556  6.44444444
  7.33333333  8.22222222  9.11111111 10.        ]
```

[14]:
```
# Here we check the new created datapoints in between datapoints using␣
↪interpolate
```

```
# with interpolation

# This is final result of interpolation

plt.scatter(x_new,y_interp)
```

[14]: <matplotlib.collections.PathCollection at 0x7f0ef581aad0>




[11]:
```
# NOTE - IN THIS SCATTER PLOT WE HAVE SEEN THAT HOW DATAPOINTS INCREASES AS␣
↪COMPARING PREVIOUS PLOT BY USING LINEAR INTERPOLATION TECHNIQUE
```

TYPE 2 - CUBIC INTERPOLATION WITH SCIPY

[21]:
```
# x and y are datapoints that creates the dataset

# Here we create data in the form of x and y

import numpy as np
x=np.array([1,2,3,4,5])
y=np.array([1,8,27,64,125])
```

[22]:
```
from scipy.interpolate import interp1d
```

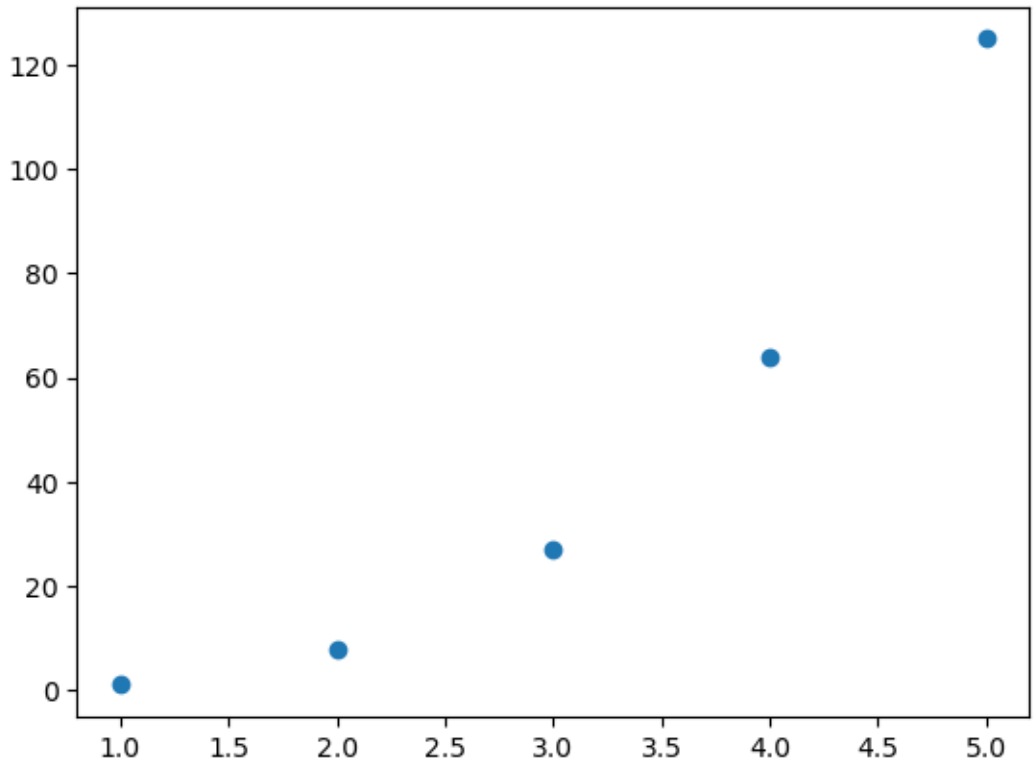
[24]:
```
# Create a cubic interpolation function

f=interp1d(x,y,kind="cubic")
```

[25]:
```
# Interpolate the data

# Given output is a new datapoints

# we create new values
# x_new and y_interp are our new values

x_new=np.linspace(1,5,10)
y_interp=f(x_new)
print(y_interp)
```

```
[  1.           3.01371742   6.739369    12.7037037   21.43347051
  33.45541838  49.2962963   69.48285322  94.54183813 125.        ]
```

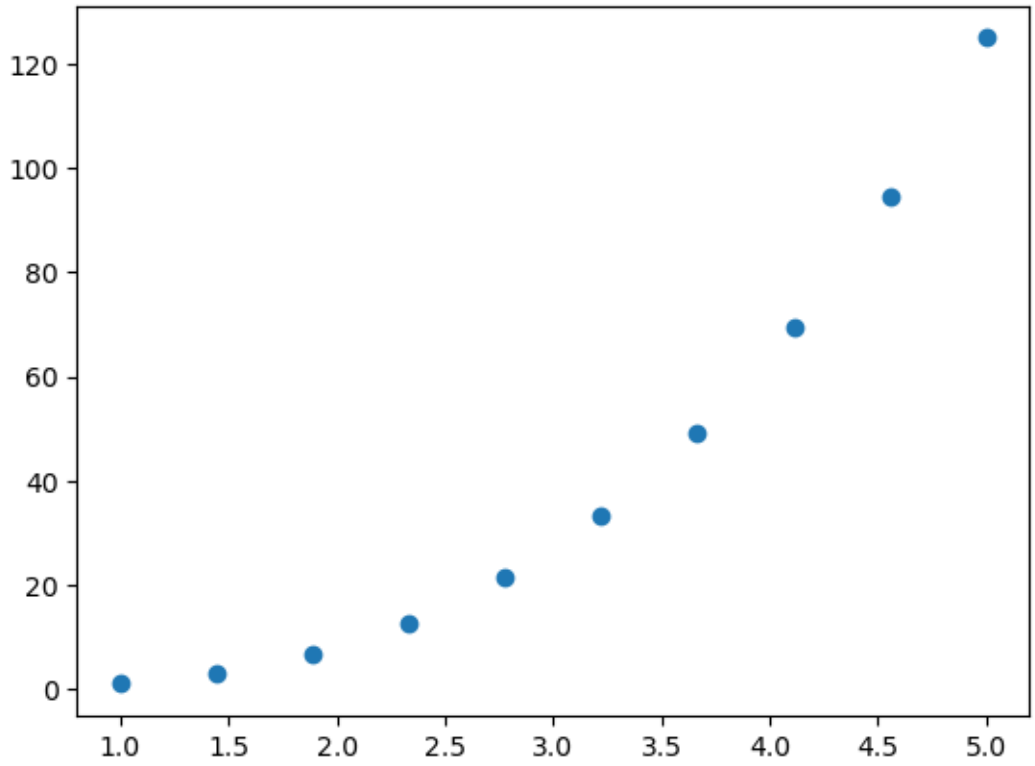
[26]:
```
# Without interpolate

plt.scatter(x,y)
```

[26]: <matplotlib.collections.PathCollection at 0x7f0ef5af7580>



[27]:
```
# with interpolate

plt.scatter(x_new,y_interp)
```

[27]:
```
<matplotlib.collections.PathCollection at 0x7f0ef42c62f0>
```




TYPE 3 - POLYNOMIAL INTERPOLATION

[29]:
```
# x and y are datapoints that creates the dataset

# Here we create data in the form of x and y

import numpy as np
x=np.array([1,2,3,4,5])
y=np.array([1,8,27,64,125])
```

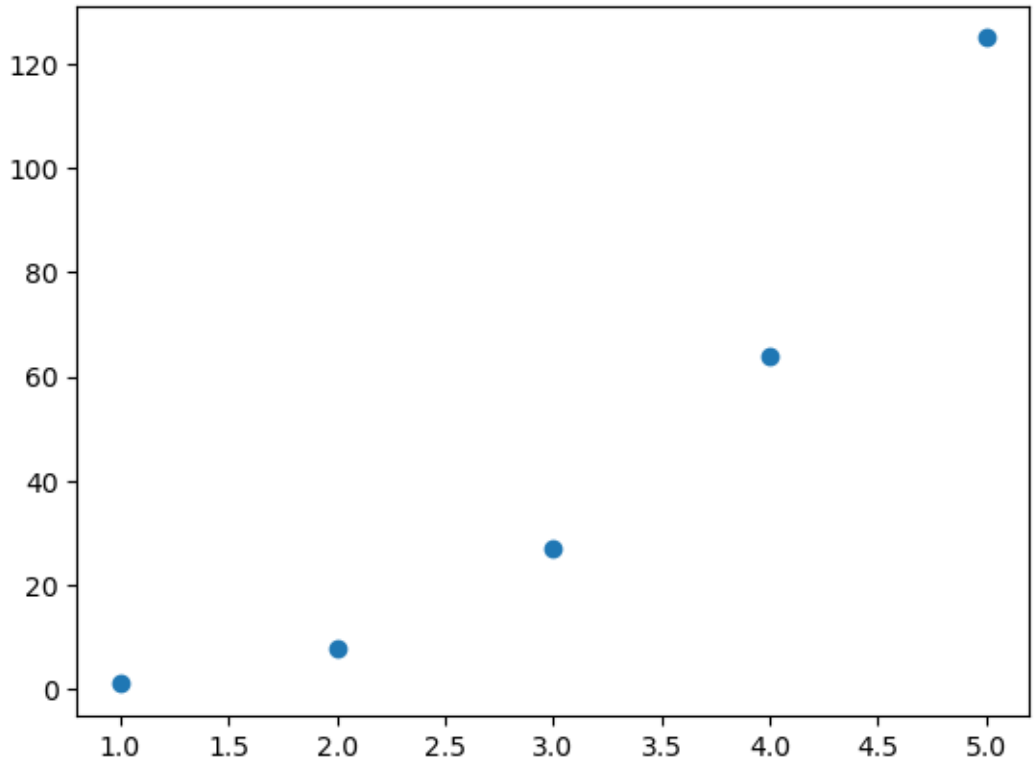
[30]:
```
# INTERPOLATE THE DATA USING POLYNOMIAL INTERPOLATION

# IN POLYNOMIAL WE GIVE THE DEGREE VALUE THAT HOW MUCH WE WANT CURVE IN DATASET␣
 ↪OR GRAPH
```

```
# HERE WE GIVE "2" AS A DEGREE VALUE

p=np.polyfit(x,y,2)
```

[32]:
```
# we create new values
# x_new and y_interp are our new values

x_new=np.linspace(1,5,10)
y_interp=np.polyval(p,x_new)
```

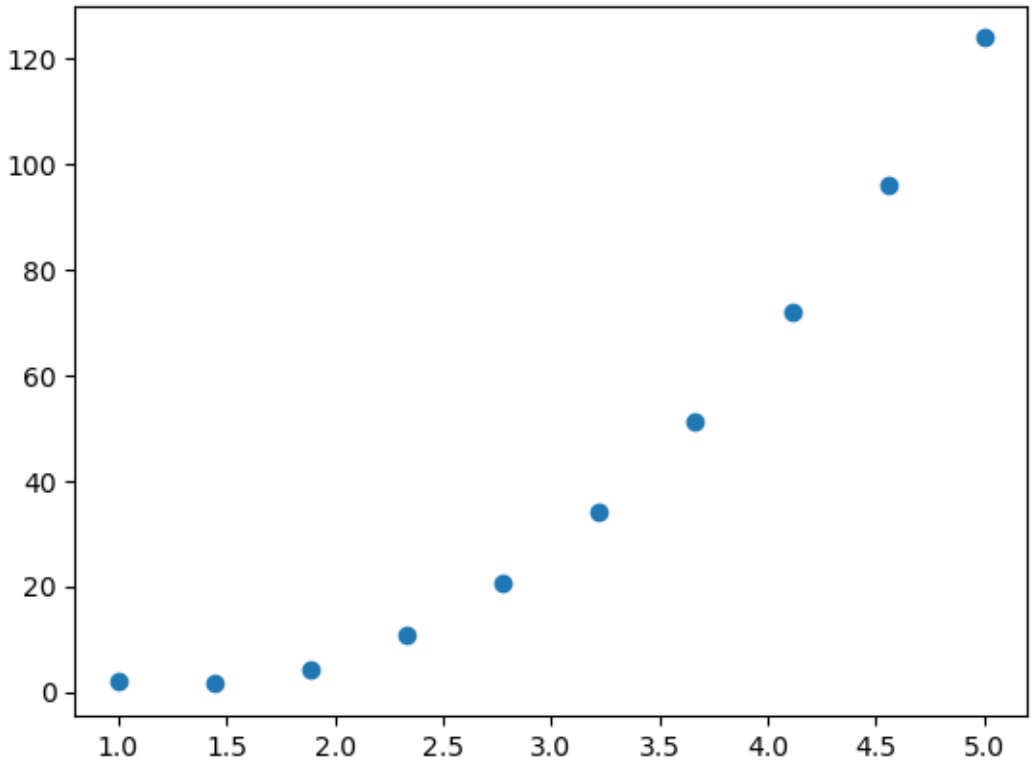
[33]:
```
# Without interpolation

plt.scatter(x,y)
```

[33]: <matplotlib.collections.PathCollection at 0x7f0ef4336470>




[34]:
```
# With interpolation

plt.scatter(x_new,y_interp)
```

[34]: <matplotlib.collections.PathCollection at 0x7f0ef419e200>




NOTE - AT THE END OF THE DAY WE CREATE THE DATA WITHIN SPECIFIC RANGE OF DATASET

THANK YOU SO MUCH !!

YOURS VIRAT TIWARI :)