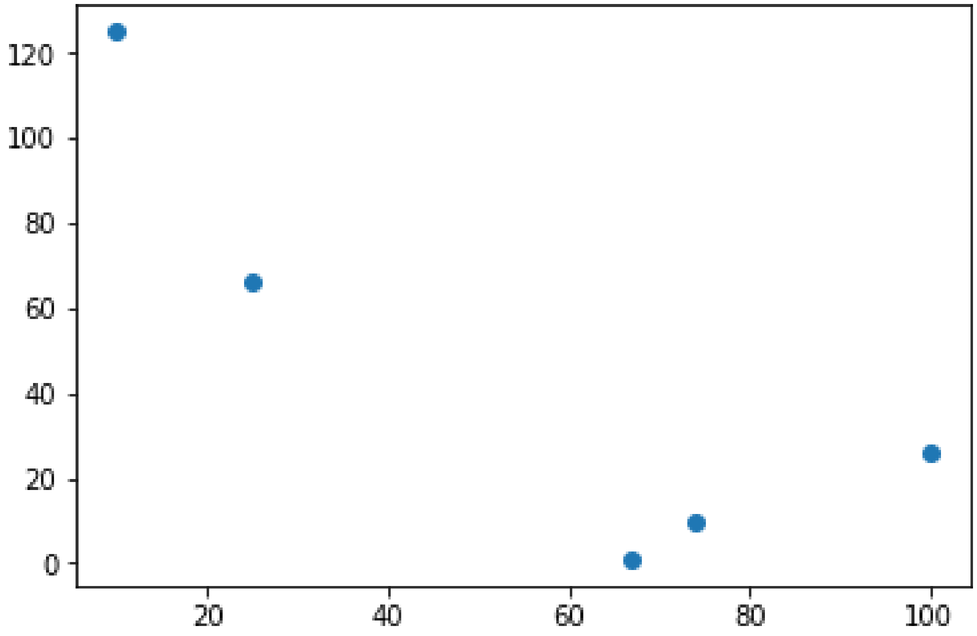
Matplotlib & Numpy

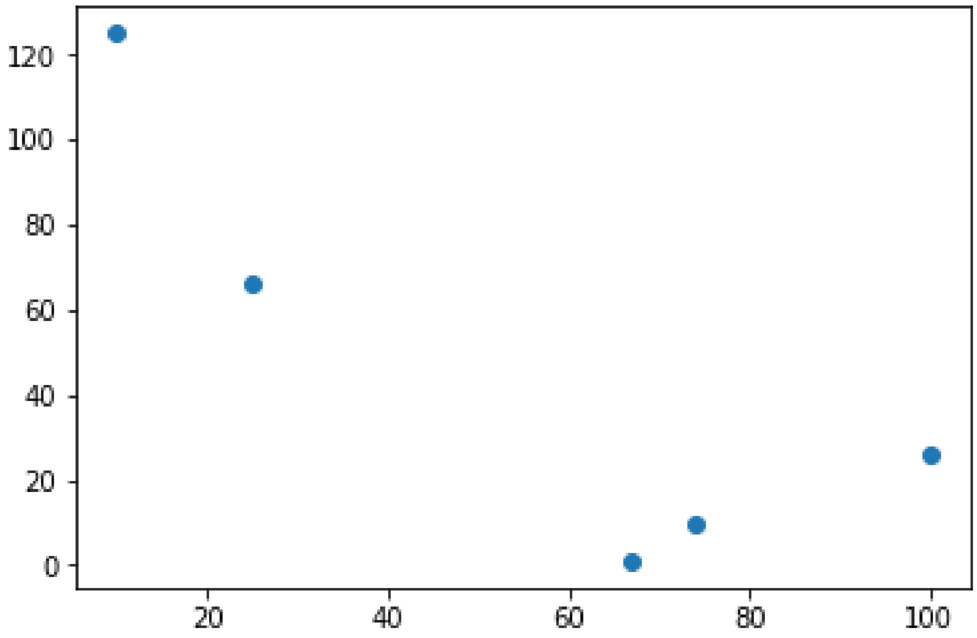
August 11, 2020

```
In [12]: import matplotlib.pyplot as mp
         import numpy as np

         xs = [10, 100, 25, 67, 74]
         ys = [125, 26, 66, 1, 10]
         xs = np.array(xs)
         ys = np.array(ys)
         mp.scatter(xs, ys)  #single variable method
         mp.show()
```



```
In [11]: import matplotlib.pyplot as mp
         import numpy as np

         xys = [[10, 125], [100, 26], [25, 66], [67, 1], [74, 10]]
         xys = np.array(xys)
```

```
#print(xys[:,0])  #prints out only the first column values
#print(xys[:,1]) #prints our the second column

mp.scatter(xys[:,0], xys[:,1])  #both together - more lean
mp.show()
```



```
In [15]: import matplotlib.pyplot as mp
         import numpy as np

         xys = [[10, 125], [100, 26], [25, 66], [67, 1], [74, 10]]
         xys = np.array(xys)

         x_mean = np.mean(xys[:,0])
         y_mean = np.mean(xys[:,1])
         print(x_mean, y_mean)
```

```
55.2 45.6
```

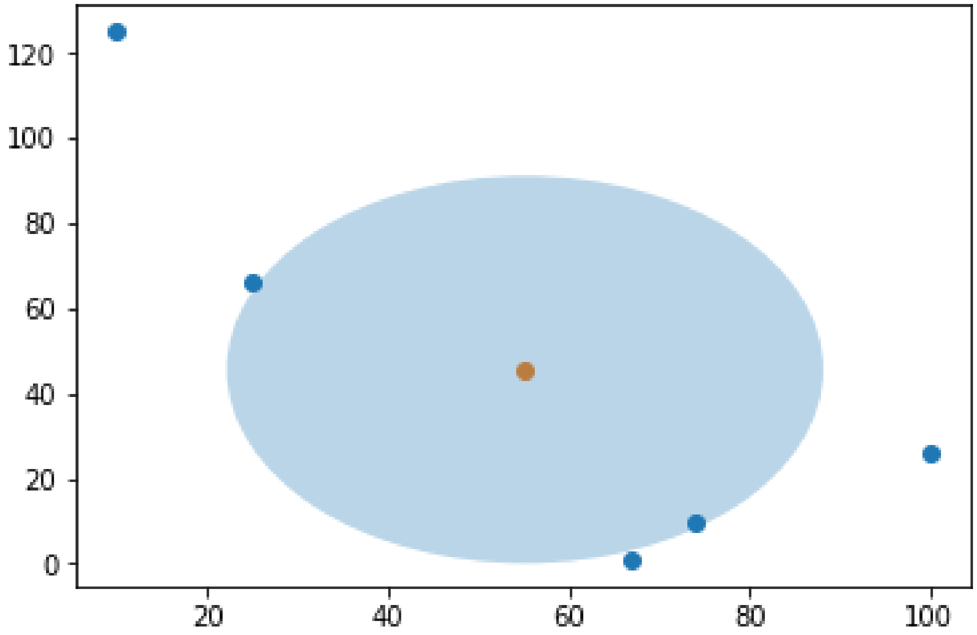
```
In [18]: import matplotlib.pyplot as mp
         import numpy as np

         xys = [[10, 125], [100, 26], [25, 66], [67, 1], [74, 10]]
         xys = np.array(xys)

         #print(xys.mean(0)) #column-wise, cuz 0 NOT 1, mean, [x_mean, y_mean] produced
         x_mean = np.mean(xys[:,0])
         y_mean = np.mean(xys[:,1])
         print(x_mean, y_mean)   #x and y mean
```

```
[55.2 45.6]
55.2 45.6
```

```
In [19]: import matplotlib.pyplot as mp
         import numpy as np

         xys = [[10, 125], [100, 26], [25, 66], [67, 1], [74, 10]]
         xys = np.array(xys)

         print(xys.mean(0))  #lean method (0 - column wise mean, 1 - row wise mean)
```

```
[55.2 45.6]
```

```
In [2]: import matplotlib.pyplot as mp
        import matplotlib.patches as patches
        import numpy as np

        xys = [[10, 125], [100, 26], [25, 66], [67, 1], [74, 10]]
        xys = np.array(xys)

        mean = np.mean(xys, 0)  #holds mean of the x and y values  [55.2 45.6]
        sd = np.std(xys, 0)     #holds sd of the x and y values, 0 is passed to that column-wi

        ellipse = patches.Ellipse([mean[0], mean[1]], sd[0]*2, sd[1]*2, alpha=0.3)

        fig, graph = mp.subplots()  #needed to add overlays

        mp.scatter(xys[:,0], xys[:,1])
        mp.scatter(mean[0], mean[1])  #mean point
        graph.add_patch(ellipse)
        mp.show()
```



```
In [18]: #distance between a point and the mean, to determine to which cluster does the point 
         import matplotlib.pyplot as mp
         import numpy as np

         xys = [[10, 125], [100, 26], [25, 66], [67, 1], [74, 10]]

         dist1 = np.linalg.norm(xys[0] - mean)    #euclidian distance between first point and
         print(dist1)
```

```
125.00167117172208
```

```
In [19]: #if we want the distance between all points and the mean, we'll have to continuosly d
         #list comprehension
         import matplotlib.pyplot as mp
         import matplotlib.patches as patches
         import numpy as np

         a = [2, 4, 6, 8]
         b = [x+5 for x in a]     #add 5 for all elements in a and appends to a list


         xys = [[10, 125], [100, 26], [25, 66], [67, 1], [74, 10]]
         distances = [np.linalg.norm(xy - mean) for xy in xys]
         print(distances[0])
```

```
125.00167117172208
```

```
In [20]: #normalization, scale every data to a specific standard range. This is important to p
         #actual value. For example Engine size of car compared to CO2 emission, engine size m
         #500g, so changing say 100 for each is different, to prevent this scaling we normaliz
         import matplotlib.pyplot as mp
         import matplotlib.patches as patches
         import numpy as np
         xys = [[10, 125], [100, 26], [25, 66], [67, 1], [74, 10]]
         xys = np.array(xys)

         #domain standardization
         x_min = np.min(xys[:,0]) #min of first column
         x_max = np.max(xys[:,0])
         normalized_x = (xys[:,0] - x_min) / (x_max-x_min)
         print(normalized_x)
```

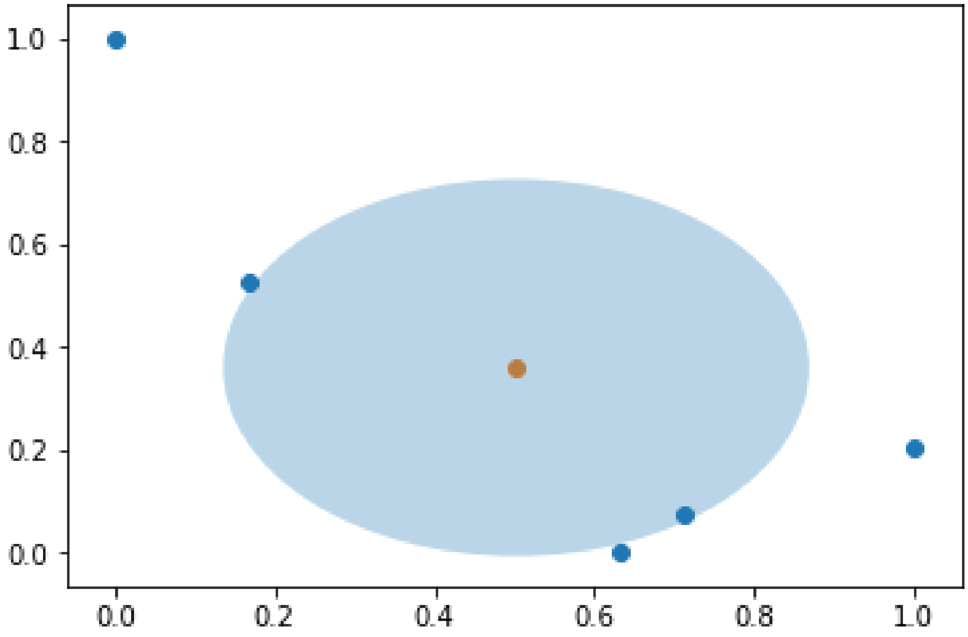
```
[0.         1.         0.16666667 0.63333333 0.71111111]
```

```
In [16]: #domain standardization of both x and y values

         import matplotlib.pyplot as mp
         import matplotlib.patches as patches
         import numpy as np
         xys = [[10, 125], [100, 26], [25, 66], [67, 1], [74, 10]]
         xys = np.array(xys)

         xy_min = np.min(xys, 0)
         xy_max = np.max(xys, 0)
         normalized = (xys - xy_min) / (xy_max-xy_min)
         print(normalized)
```

```
[[0.         1.        ]
 [1.         0.2016129 ]
 [0.16666667 0.52419355]
 [0.63333333 0.        ]
 [0.71111111 0.07258065]]
```

```
In [15]: #normalized data plotting, data that is two standard deviations from the mean mustn't
         #as to how they were recorded
         import matplotlib.pyplot as mp
         import matplotlib.patches as patches
         import numpy as np

         xys = [[10, 125], [100, 26], [25, 66], [67, 1], [74, 10]]
         xys = np.array(xys)
```

```
xy_min = np.min(xys, 0)
xy_max = np.max(xys, 0)
normalized = (xys - xy_min) / (xy_max-xy_min)

mean = np.mean(normalized_x, 0)
sd = np.std(normalized_x, 0)


ellipse = patches.Ellipse([mean[0], mean[1]], sd[0]*2, sd[1]*2, alpha=0.3)

fig, graph = mp.subplots()

mp.scatter(normalized[:,0], normalized[:,1])
mp.scatter(mean[0], mean[1])
graph.add_patch(ellipse)
mp.show()
```


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0