Kmeans - Note 01

1. Euclidean Distance

Euclidean Distance represents the **shortest distance between two points.**

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Most machine learning algorithms including **K-Means use this distance metric to measure the similarity between observations.**

```
from scipy.spatial import distance # To calculate distances
```

```
point_1 = (1, 2, 3)
point_2 = (4, 5, 6)
```

```
#computing the euclidean distance
euclidean_distance = distance.euclidean(point_1, point_2)
euclidean_distance
```

```
5.196152422706632
```

2. Manhattan Distance

Manhattan Distance is the sum of absolute differences between points across all the dimensions.

```
# Manhattan Distance vs Euclidean Distance
point_1 = (1, 1)
point_2 = (5, 4)

print('First Data Point:',point_1)
print('Second Data Point:', point_2)
```

```
First Data Point: (1, 1)
Second Data Point: (5, 4)
```

```
#computing the euclidean distance
euclidean_distance = distance.euclidean(point_1, point_2)
print('\nEuclidean Distance b/w', point_1, 'and', point_2, 'is: ',
```

```
euclidean_distance)

# computing the manhattan distance
manhattan_distance = distance.cityblock(point_1, point_2)
print('Manhattan Distance b/w', point_1, 'and', point_2, 'is: ',
manhattan_distance)
```

```
Euclidean Distance b/w (1, 1) and (5, 4) is:  5.0
Manhattan Distance b/w (1, 1) and (5, 4) is:  7
```

3.Minkowski Distance*

```
point_1 = (1, 1)
point_2 = (5, 4)

print('First Data Points :',point_1)
print('Second Data Points :', point_2)
```

```
First Data Points : (1, 1)
Second Data Points : (5, 4)
```

```
#computing the euclidean distance
euclidean_distance = distance.euclidean(point_1, point_2)
print('\nEuclidean Distance b/w', point_1, 'and', point_2, 'is: ',
euclidean_distance)

# computing the manhattan distance
manhattan_distance = distance.cityblock(point_1, point_2)
print('\nManhattan Distance b/w', point_1, 'and', point_2, 'is: ',
manhattan_distance)

# computing the minkowski distance
minkowski_distance = distance.minkowski(point_1, point_2, p=1)
print('\nMinkowski Distance (p=1, Manhattan) b/w', point_1, 'and', point_2,
'is: ', minkowski_distance)
```

```
Euclidean Distance b/w (1, 1) and (5, 4) is:  5.0

Manhattan Distance b/w (1, 1) and (5, 4) is:  7

Minkowski Distance (p=1, Manhattan) b/w (1, 1) and (5, 4) is:  7.0
```

```
minkowski_distance = distance.minkowski(point_1, point_2, p=2)
print('\nMinkowski Distance (p=2, Euclidean) b/w', point_1, 'and', point_2,
'is: ', minkowski_distance)

minkowski_distance = distance.minkowski(point_1, point_2, p=3)
print('\nMinkowski Distance (p=3) b/w', point_1, 'and', point_2, 'is: ',
minkowski_distance)
```

```
Minkowski Distance (p=2, Euclidean) b/w (1, 1) and (5, 4) is:  5.0

Minkowski Distance (p=3) b/w (1, 1) and (5, 4) is:  4.497941445275415
```

3. Hamming Distance

- Hamming Distance measures the **similarity between two strings of the same length**

```
# Hamming Distance
# defining two strings
string_1 = 'euclidean'
string_2 = 'manhattan'

# computing the hamming distance
hamming_distance = distance.hamming(list(string_1),
list(string_2))*len(string_1)
print('Hamming Distance b/w', string_1, 'and', string_2, 'is: ',
hamming_distance)
```

```
Hamming Distance b/w euclidean and manhattan is:  7.0
```