# lab5-shridhargupta-211220051

April 29, 2024

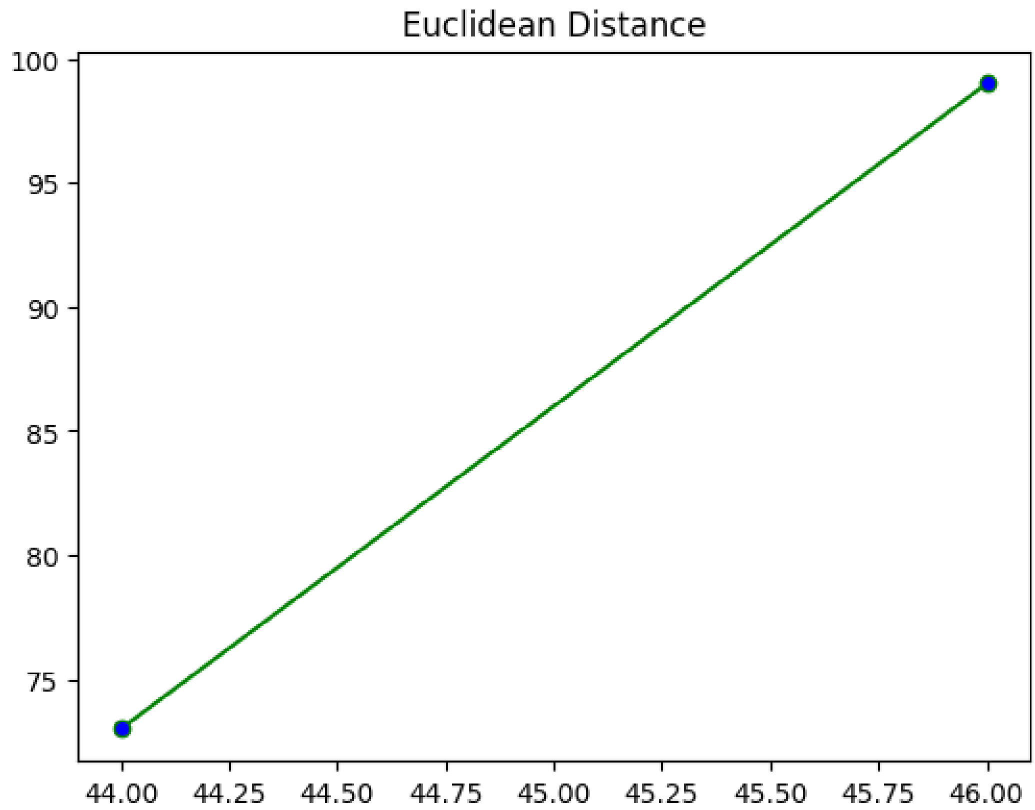
[ ]:
```
from math import*
from decimal import Decimal
import random
import pandas as pd
import numpy as np
import seaborn as sns

x = np.random.randint(-50,50, size = 10)
y = np.random.randint(-50,50, size = 10)
print(x)
print(y)
```

```
[ 49  -2  40  12  45 -46 -36 -42  19 -31]
[ 48 -27  18  31 -19 -38  -1  46  22   7]
```

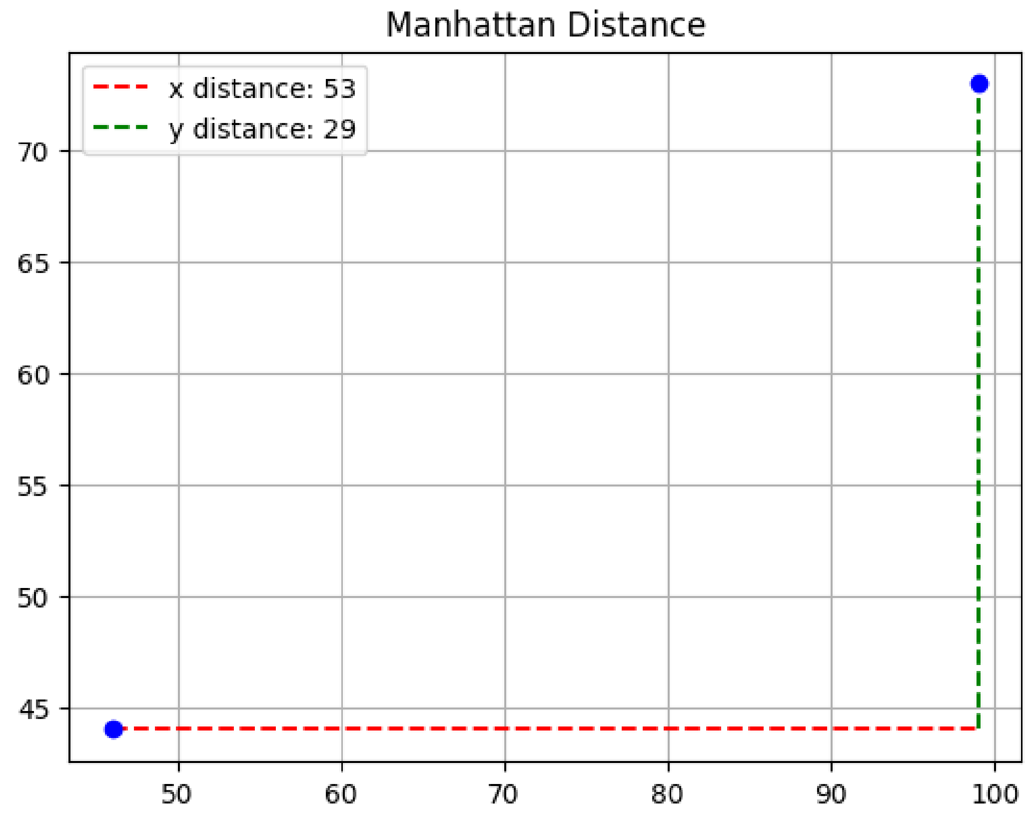
[ ]:
```
import random as ra
import numpy as np
import matplotlib.pyplot as plt

x1 = ra.randrange(100)
y1 = ra.randrange(100)
x2 = ra.randrange(100)
y2 = ra.randrange(100)
print("Point 1 : ",x1,",",y1)
print("Point 2 : ",x2,",",y2)

point1 = [x1,x2]
point2 = [y1,y2]

p1 = np.array((x1,y1))
p2 = np.array((x2,y2))

# Eulcidean distance matrix (PLOT)

res = np.sum(np.square(p1 - p2))
ans = np.sqrt(res)
print("Euclidean Distance : ",ans)
```

```python
plt.plot(point1,point2,marker="o",markerfacecolor="blue",color="green")
plt.title('Euclidean Distance')
plt.show()

# Manhattan Distance matrix (PLOT)

def manhattan_distance(point1, point2):
  return abs(x1-x2)+abs(y1-y2)

dist = manhattan_distance(point1,point2)
print("Manhattan Distance : ",dist)

plt.plot([point1[0], point2[0]], [point1[1], point1[1]], 'r--', label=f"x␣
↪distance: {abs(point2[0] - point1[0])}")
plt.plot([point2[0], point2[0]], [point1[1], point2[1]], 'g--', label=f"y␣
↪distance: {abs(point2[1] - point1[1])}")

plt.scatter([point1[0], point2[0]], [point1[1], point2[1]], color='blue',␣
↪zorder=5)
plt.legend()
plt.title('Manhattan Distance')
plt.grid(True)
plt.show()
```

```
Point 1 :  46 , 99
Point 2 :  44 , 73
Euclidean Distance :  26.076809620810597
```




```
Manhattan Distance :  28
```




[ ]:
```
#Manhattan Distance
def manhattan_distance(x,y):
  return sum(abs(a-b) for a,b in zip(x,y))

dist = []
ans = []
def manhattan_distance(x,y):
  for a,b in zip(x,y):
    dist.append(abs(b-a))
  for i in dist:
    ans.append(i[0] + i[1])

manhattan_distance(x, y)
print(ans)
```

```
[81, 40, 70, 74, 19, 111, 70, 83, 137, 66]
```

[ ]:
```
#Euclidean Distance
def euclidean_distance(x,y):
  return sqrt(sum(pow(a-b,2) for a, b in zip(x, y)))
```

```
# dist = []
# ans = []

# def manhattan_distance(x,y):
#   for a,b in zip(x,y):
#     dist.append(sqrt(pow(a-b,2)))
#   for i in dist:
#     ans.append(i[0] + i[1])

# manhattan_distance(x, y)
# print(ans)

# # def euclidean_distance(x,y):
# #   for a, b in zip(x, y):


# # print(euclidean_distance(x, y))
```

[ ]:
```
#Minkowski Distance
def nth_root(value, n_root):
 root_value = 1/float(n_root)
 return round (Decimal(value) ** Decimal(root_value),3)

def minkowski_distance(x,y,p_value):
 return nth_root(sum(pow(abs(a-b),p_value) for a,b in zip(x, y)),p_value)

print(minkowski_distance(x, y, 1))
```

```
303.000
```

[ ]:
```
#Cosine Similarity
def square_rooted(x):
   return round(sqrt(sum([a*a for a in x])),3)

def cosine_similarity(x,y):
 numerator = sum(a*b for a,b in zip(x,y))
 denominator = square_rooted(x)*square_rooted(y)
 return round(numerator/float(denominator),3)

print(cosine_similarity(x, y))
```

```
0.256
```