```
# Q2 - MDS cities
import pandas as pd


df = pd.read_excel('distance.xlsx')


df
```

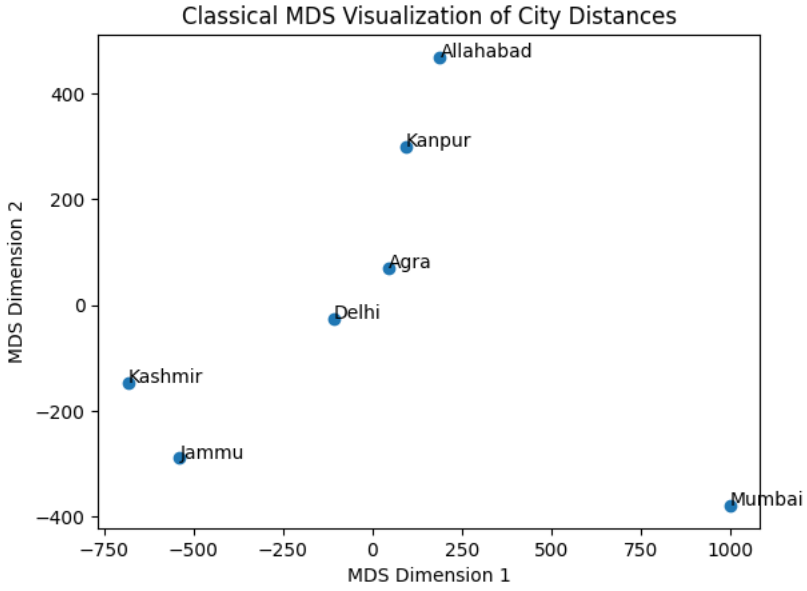
	Unnamed: 0	Jammu	delhi	Kashmir	agra	kanpur	Allahabad	Mumbai
0	Jammu	0	505	202	686	864	1051	1541
1	delhi	505	0	588	181	383	577	1163
2	kashmir	202	588	0	761	896	1068	1699
3	agra	686	181	761	0	233	423	1055
4	kanpur	864	383	896	233	0	194	1133
5	allahabad	1051	577	1068	423	194	0	1173
6	mumbai	1541	1163	1699	1055	1133	1173	0

Next steps: View recommended plots

```
df.set_index('Unnamed: 0', inplace=True)
D_matrix = df.values
print(D_matrix)
```

```
[[   0  505  202  686  864 1051 1541]
 [ 505    0  588  181  383  577 1163]
 [ 202  588    0  761  896 1068 1699]
 [ 686  181  761    0  233  423 1055]
 [ 864  383  896  233    0  194 1133]
 [1051  577 1068  423  194    0 1173]
 [1541 1163 1699 1055 1133 1173    0]]
```

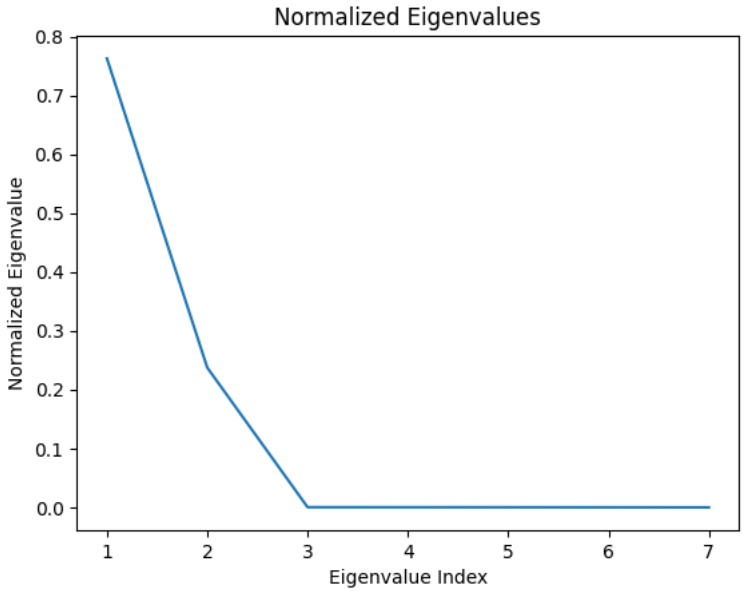
```
# B) local code of MDS
import numpy as np
import matplotlib.pyplot as plt

def classical_mds(distances_matrix, num_dimensions):
    n = len(distances_matrix)

    # create Householder centering matrix
    H = np.eye(n) - np.ones((n, n)) / n
    B = -0.5 * H @ distances_matrix**2 @ H

    # eigenvalue find
    eigenvalues, eigenvectors = np.linalg.eigh(B)

    # Sort eigenvalues
    idx = np.argsort(eigenvalues)[::-1]
    eigenvalues = eigenvalues[idx]
    eigenvectors = eigenvectors[:, idx]

    # top eigenvectors upto the given dimensions
    selected_eigenvectors = eigenvectors[:, :num_dimensions]

    # Map data in space of given dimensions
    coordinates = np.sqrt(np.diag(eigenvalues[:num_dimensions])) @ selected_eigenvectors.T

    return coordinates, eigenvalues



# Perform Classical MDS to reduce to 2D
coordinates, eigenvalues = classical_mds(D_matrix, 2)

# Plot the cities in 2D
plt.scatter(coordinates[0], coordinates[1])
for i, city in enumerate(["Jammu", "Delhi", "Kashmir", "Agra", "Kanpur", "Allahabad", "Mumbai"]):
    plt.annotate(city, (coordinates[0, i], coordinates[1, i]))

plt.title("Classical MDS Visualization of City Distances")
plt.xlabel("MDS Dimension 1")
plt.ylabel("MDS Dimension 2")
plt.show()
```




```
eigenvalues
```

```
array([ 1.81486553e+06,  5.64723323e+05,  2.40193560e+02,  1.05526434e+02,
       -6.64303599e-11, -1.17086338e+02, -4.52631645e+02])
```

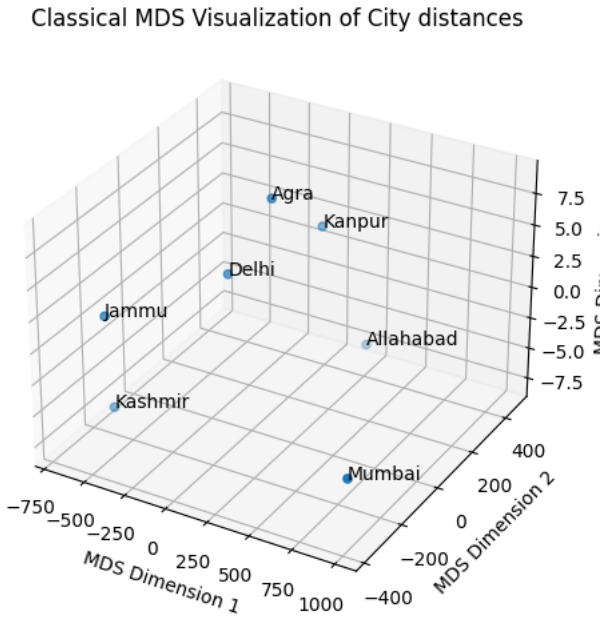
```
# c) plot eigenvalues
normalized_eigenvalues = eigenvalues / np.sum(eigenvalues)

# Plot the normalized eigenvalues
plt.plot(range(1, len(normalized_eigenvalues) + 1), normalized_eigenvalues)
plt.title("Normalized Eigenvalues")
plt.xlabel("Eigenvalue Index")
plt.ylabel("Normalized Eigenvalue")
plt.show()
```




```
There are some negative eigenvalues which shows that D_matrix is not a positive semi-definite matrix
```

```
# d) scatter plot of 3 eigen vectors
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D  # Importing 3D plotting tools

num_dimensions = 3

coordinates, eigenvalues = classical_mds(D_matrix, num_dimensions)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.scatter(coordinates[0], coordinates[1], coordinates[2])
```

```
for i, city in enumerate(["Jammu", "Delhi", "Kashmir", "Agra", "Kanpur", "Allahabad", "Mumbai"]):
   ax.text(coordinates[0, i], coordinates[1, i], coordinates[2, i], city)

ax.set_title("Classical MDS Visualization of City distances")
ax.set_xlabel("MDS Dimension 1")
ax.set_ylabel("MDS Dimension 2")
ax.set_zlabel("MDS Dimension 3")
plt.tight_layout()
plt.show()
```



coordinates

```
array([[-538.2975014 , -107.7592322 , -682.98702346,   45.93533678,
          93.39190769,  189.59709861, 1000.11941399],
       [-289.17025498,  -25.3603316 , -146.99747161,   71.14570763,
         300.40744179,  468.95584873, -378.98093997],
       [   2.05225821,    3.59062404,   -7.82242259,    8.86346371,
           4.0506644 ,   -7.52534419,   -3.20924358]])
```

We can observe the distance in 3 coordinate system of eigen values. The relative distance is preserved in a local coordinate system.

No.

Date

3a) K is P.S.d ($K \geq 0$) iff $\lambda_i \geq 0$

$K \geq 0$ eigen value $Kv_i = \lambda_i v_i$

Multiply both sides by v_i^T

$\Rightarrow v_i^T K v_i = \lambda_i \|v_i\|^2$

$\|v_i\|^2 \geq 0 \Rightarrow v_i^T K v_i \geq 0$ iff $\boxed{\lambda_i \geq 0}$ $\forall i$

All eigen values be non-negative

b) $d_{ij} = k_{ii} + k_{jj} - 2k_{ij}$ K - symmetric, P.S.d

$= v_i^T K v_i + v_j^T K v_j - 2 v_i^T K v_j$

$= (v_i - v_j)^T K (v_i - v_j)$

K is symmetric matrix $\Rightarrow d_{ij}$ is squared distance function

c) $D = [d_{ij}]$ is conditionally negative definite (c.n.d)

$B_\alpha = -\frac{1}{2} H_\alpha D H_\alpha^T \geq 0$ (P.S.d) To prove

$= \frac{1}{2} (1 - e\alpha^T) D (1 - e\alpha^T)^T$

$1 - e\alpha^T = y^T \Rightarrow -\frac{1}{2} y^T D y$

D is c.n.d $\Rightarrow y^T D y \leq 0$

$\Rightarrow -\frac{1}{2} y^T D y \geq 0$

$\leftarrow$ B_α is P.S.d

No.

Date

d) $A+B = (A_{ij}+B_{ij})_{ij} \geq 0$

$x^T(A+B)x = x^TAx + x^TBx$

$\geq 0 \quad \geq 0$

$\Rightarrow x^T(A+B)x \geq 0$

$A+B$ is p.s.d

e) $A \circ B = [A_{ij}B_{ij}]_{ij} \geq 0$

$A = \sum_i \lambda_i u_i u_i^T \quad , B = \sum_j \beta_j v_j v_j^T$

$A \circ B = \sum_{ij} \lambda_i \beta_j (u_i u_i^T)(v_j v_j^T)$

$= \sum_{ij} \lambda_i \lambda_j (u_i v_j)(u_i v_j)^T$

$(u_i \circ v_j)(u_i v_j)^T$ is p.s.d $\forall \lambda_i \lambda_j \geq 0$

$A \circ B$ is p.s.d

No.
Date

Q.4) Distance function has properties :-

1. Non negative : $d(x,y) \geq 0$
2. Symmetry : $d(x,y) = d(y,x)$
3. Triangle inequality $d(x,y) + d(y,z) \geq d(x,z)$ $\forall x,y,z$

a) d^2 violates triangle inequality

If $d(x,y) = 1$, $d(y,z) = 2$, $d(x,z) = 3$

$d^2(x,y) = 1$, $d^2(y,z) = 4$, $d^2(x,z) = 9$

$1 + 4 < 9$

$\therefore$ d^2 is not a distance function

b) $\sqrt{d}$ is distance function

Triangle inequality $\sqrt{(x-y)^2} + \sqrt{(y-z)^2} \geq \sqrt{(x-z)^2}$

Using Cauchy-Schwarz inequality and Euclidean distance property

d^p is a valid Euclidean distance function $\forall p \in (0,1)$