1.(a)

$$h_1^{(1)} = ReLU\left(\sum_{i=1}^{4} x_i w_{1i}^{(1)} + w_{10}^{(1)}\right)$$

$$h_2^{(2)} = ReLU\left(\sum_{i=1}^{4} x_i w_{2i}^{(1)} + w_{20}^{(1)}\right)$$

$$h_2^{(2)} = ReLU\left(\sum_{i=1}^{4} x_i w_{3i}^{(1)} + w_{30}^{(1)}\right)$$

$$z^{(3)} = \sum_{j=1}^{3} h_j^{(2)} w_{1j}^{(2)} + w_{10}^{(2)}$$

1.(b)

$$w_{ji}^{(1)}{}_{new} = w_{ji}^{(1)}{}_{old} - \eta \frac{\partial L}{\partial w_{ji}^{(1)}}$$

$$\frac{\partial L}{\partial w_{ji}^{(1)}} = \frac{\partial L}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial h_j^{(2)}} \frac{\partial h_j^{(2)}}{\partial z_j^{(2)}} \frac{\partial z_j^{(2)}}{\partial w_{ji}^{(1)}}$$

so $$w_{ji}^{(1)}{}_{new} = w_{ji}^{(1)}{}_{old} - \eta \frac{\partial L}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial h_j^{(2)}} \frac{\partial h_j^{(2)}}{\partial z_j^{(2)}} \frac{\partial z_j^{(2)}}{\partial w_{ji}^{(1)}}$$

1.(c)

$$h_1^{(1)} = ReLU\left(\sum_{i=1}^{4} x_i w_{1i}^{(1)} + w_{10}^{(1)}\right)$$

$$= ReLU\left(1\cdot(1) + (-1)(-1) + 1\cdot 0 + 0\cdot(-1) + 1\right)$$

$$= 3$$

$$h_2^{(2)} = ReLU\left(\sum_{i=1}^{4} x_i w_{2i}^{(1)} + w_{20}^{(1)}\right)$$

$$= ReLU\left(1\cdot(-1) + (-1)\cdot 0 + 1\cdot 1 + 0\cdot 0 - 1\right)$$

$$= 0$$

$$h_3^{(3)} = ReLU\left(\sum_{i=1}^{4} x_i w_{3i}^{(1)} + w_{30}^{(1)}\right)$$

$$= ReLU\left(1\cdot 0 + (-1)\cdot 1 + 1\cdot 1 + 0\cdot(-1) + 1\right)$$

$$= 1$$

$$z^{(3)} = \sum_{j=1}^{3} h_j^{(2)} w_{1j}^{(2)} + w_{10}^{(2)}$$

$$= 3\cdot 1 + 0\cdot(-1) + 1\cdot(1) + 1$$

$$= 5$$

$$W_{21}^{(1)}{}_{new} = W_{21}^{(1)}{}_{old} - 1 \cdot \frac{\partial L}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial h_2^{(2)}} \frac{\partial h_2^{(2)}}{\partial z_2^{(2)}} \frac{\partial z_2^{(2)}}{\partial W_{21}^{(1)}}$$

$$W_{21}^{(1)}{}_{old} = -1$$

$$\frac{\partial L}{\partial z^{(3)}} = \frac{\partial \frac{1}{2}\left(y - z^{(3)}\right)^2}{\partial z^{(3)}} = -\left(y - z^{(3)}\right) = -(3-5) = 2$$

$$\frac{\partial z^{(3)}}{\partial h_2^{(2)}} = W_{12}^{(2)} = -1$$

$$\frac{\partial h_2^{(2)}}{\partial z_2^{(2)}} = \frac{\partial \, ReLU\left(z_2^{(2)}\right)}{\partial z_2^{(2)}} = \left[\left[z_2^{(2)} > 0\right]\right]$$

$$= 0$$

$$\frac{\partial z_2^{(2)}}{\partial W_{21}^{(1)}} = x_1 = 1$$

So $W_{21}^{(1)}{}_{new} = -1 - 1 \cdot \left(2 \cdot (-1) \cdot 0 \cdot (1)\right)$

$$= -1$$

2 Clustering

2.1 Spectral Clustering

(a)

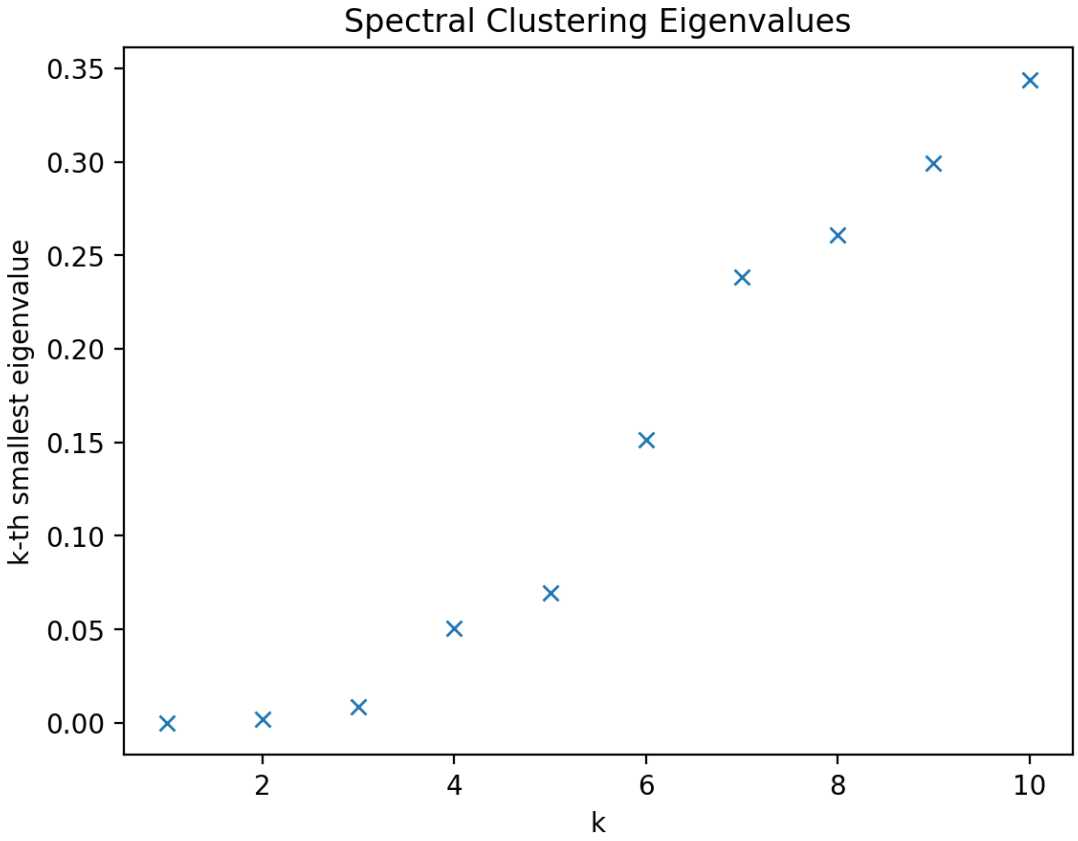
```
def plot_spectral_num_cluster(data, gamma=1):
    '''
    Plot the 10 smallest eigenvalues of the Laplacian matrix
    to determine the number of clusters
    '''
    # TODO: Define a scikit-learn SpectralClustering object
    # - Set argument gamma (rbf parameter) as gamma
    spectral = SpectralClustering(gamma=gamma)

    # Fit data to obtain affinity matrix
    spectral.fit(data)

    # TODO: Compute the Laplacian matrix of the affinity matrix (aff
    # using a scipy function laplacian (scipy.sparse.csgraph.laplaci
    # - Set normed=True
    lap_matrix = laplacian(spectral.affinity_matrix_, normed=True)

    # TODO: Compute the eigenvalues of the Laplacian matrix (lap_mat
    eigenvalues, _ = eigh(lap_matrix)
```

(b)

```
def visualize_spectral(data, gammas, n_clusters=3):
    '''
    Visualize the result of spectral clustering
    '''
    # Build figure and set size
    f, axarr = plt.subplots(1, len(gammas))
    f.set_figheight(5)
    f.set_figwidth(15)

    # for each gamma(rbf parameter)
    for i, gamma in enumerate(gammas):
        # TODO: Define a scikit-learn SpectralClustering object
        # - Set argument n_clusters (number of clusters) as n_clusters and g
        spectral = SpectralClustering(n_clusters=n_clusters, gamma=gamma)

        # Fit data to obtain clusters
        spectral.fit(data)
```

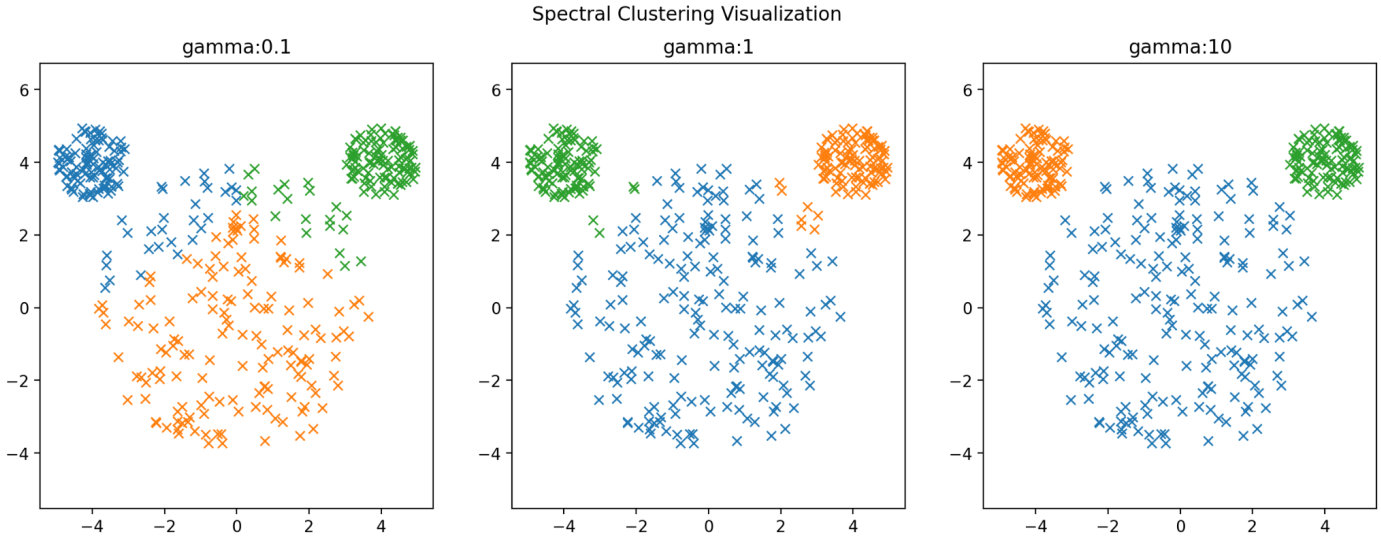
2.1(c)

i.



I will choose k = 5, because the gaps between 12345 are relavently small and the gap between 5 and 6 becomes larger.

2.1(c)ii



I will choose 10.

The larger value of gamma works well because larger gamma weighs closer connections more heavily. For larger gamma, cluster assignments are based on regions of high point density rather than distance to centroids.

2.2 k-Means Clustering

(a)

Poor centroid initialization could cause the model to converge to an arbitrarily bad local minimum. For k-means clustering, each update in each iteration is greedy for the local and not for global optimization.

(b)

```
def visualize_kmeans(data, n_clusters, init):
    """
    Visualize the result of k-means clustering
    """

    # TODO: Define a scikit-learn KMeans object
    # - Set argument n_clusters (number of clusters) to n_clusters
    # - Set argument init ('random' or 'k-means++') to init
    # - Set random_state to 20
    kmeans = KMeans(n_clusters=n_clusters, init=init, random_state=20)

    # Fit data to obtain clusters
    kmeans.fit(data)

    # TODO: print final value of objective function ("inertia_")
    print(kmeans.inertia_)
```

2.2(c)

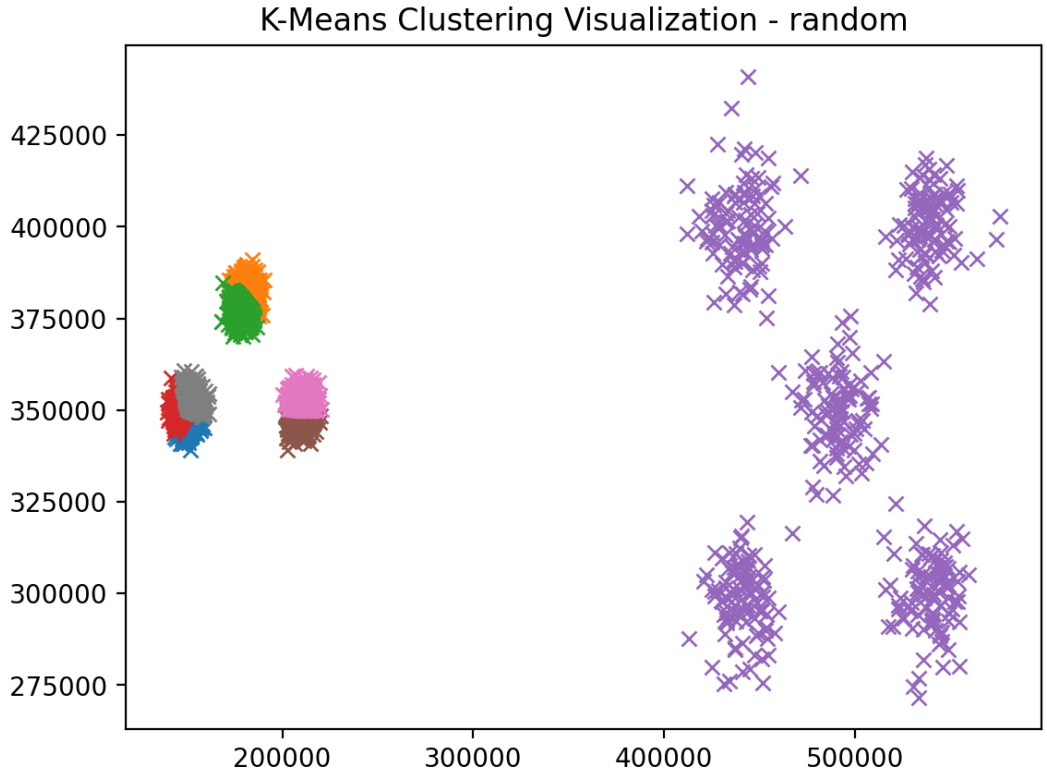
i.

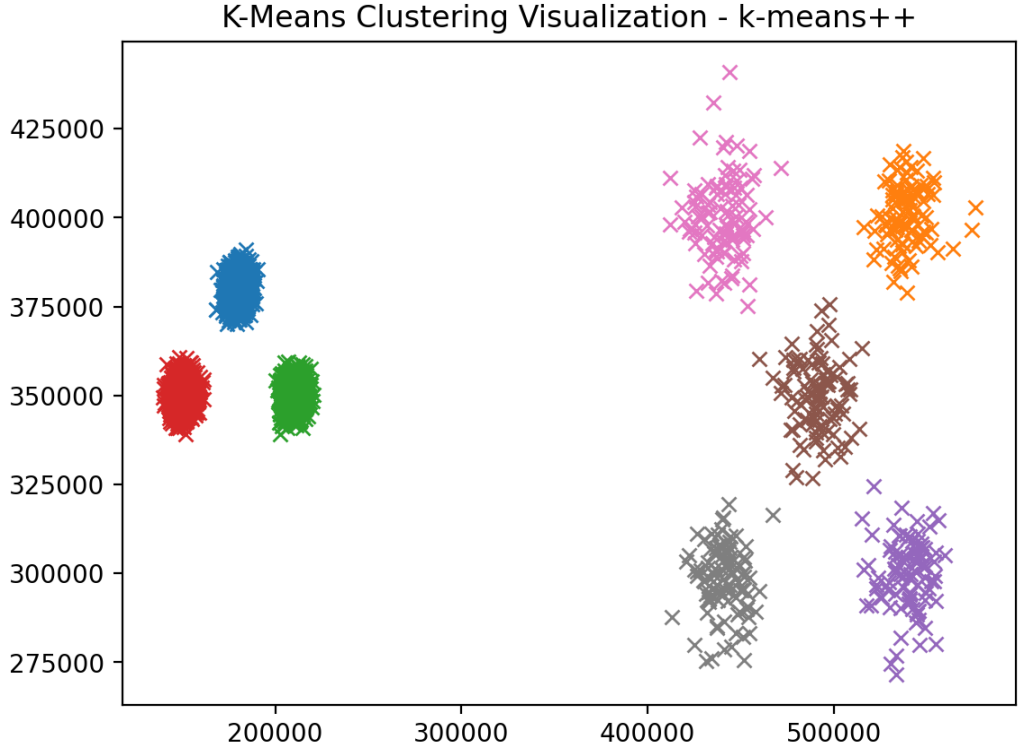
Initialization method: random
2173391383221.5393

Initialization method: k-means++
214492062847.68298

The final value of the objective function using k-means++ is lower than when using random initialization.
The reason is that k-means++ is less likely to ending up in local minima, because the initial centroids are chosen to be as far from each other as possible

ii.


K-Means Clustering Visualization - random
425000
400000
375000
350000
325000
300000
275000
200000
300000
400000
500000


K-Means Clustering Visualization - k-means++
425000
400000
375000
350000
325000
300000
275000
200000
300000
400000
500000

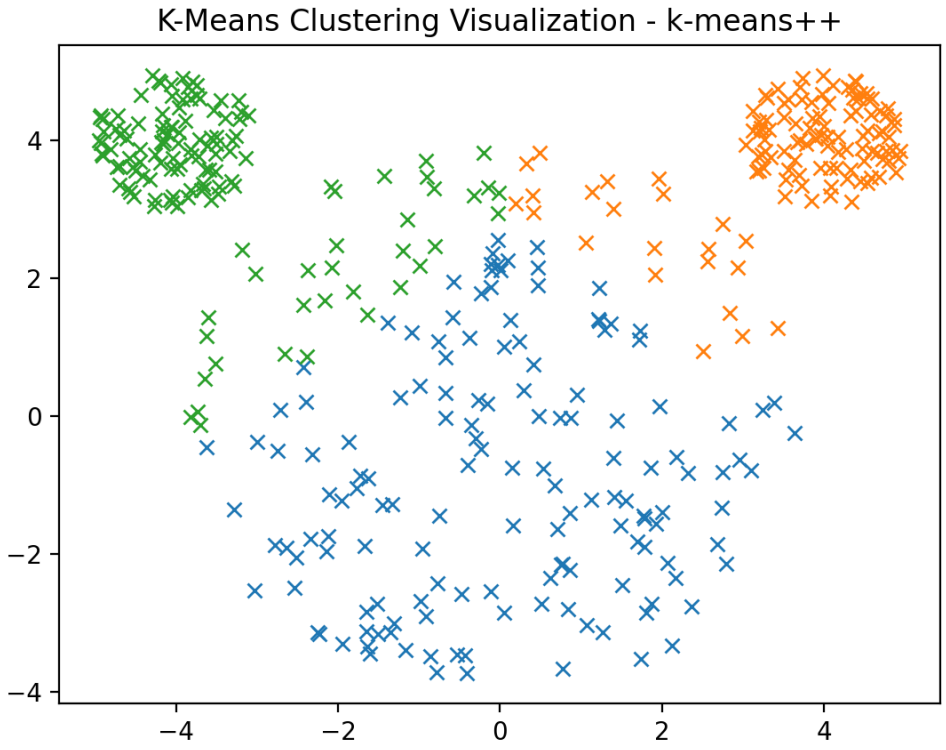
iii.

K-means++ procides a better solution

K-means++ gives a better initialization because it avoids picking centroids that are close to each other, so we can avoid getting stuck in a local minima of the objective function such as the situation shown in the first graph with random initialization where the clusters vary greatly in size.

2.3 Comparison

Data file: mickey.csv
Number of clusters: 3
Initialization method: k-means++
1366.4033341433646



The spectral clustering works better. It better captures the three clusters and recognizes the ear of Mickey.

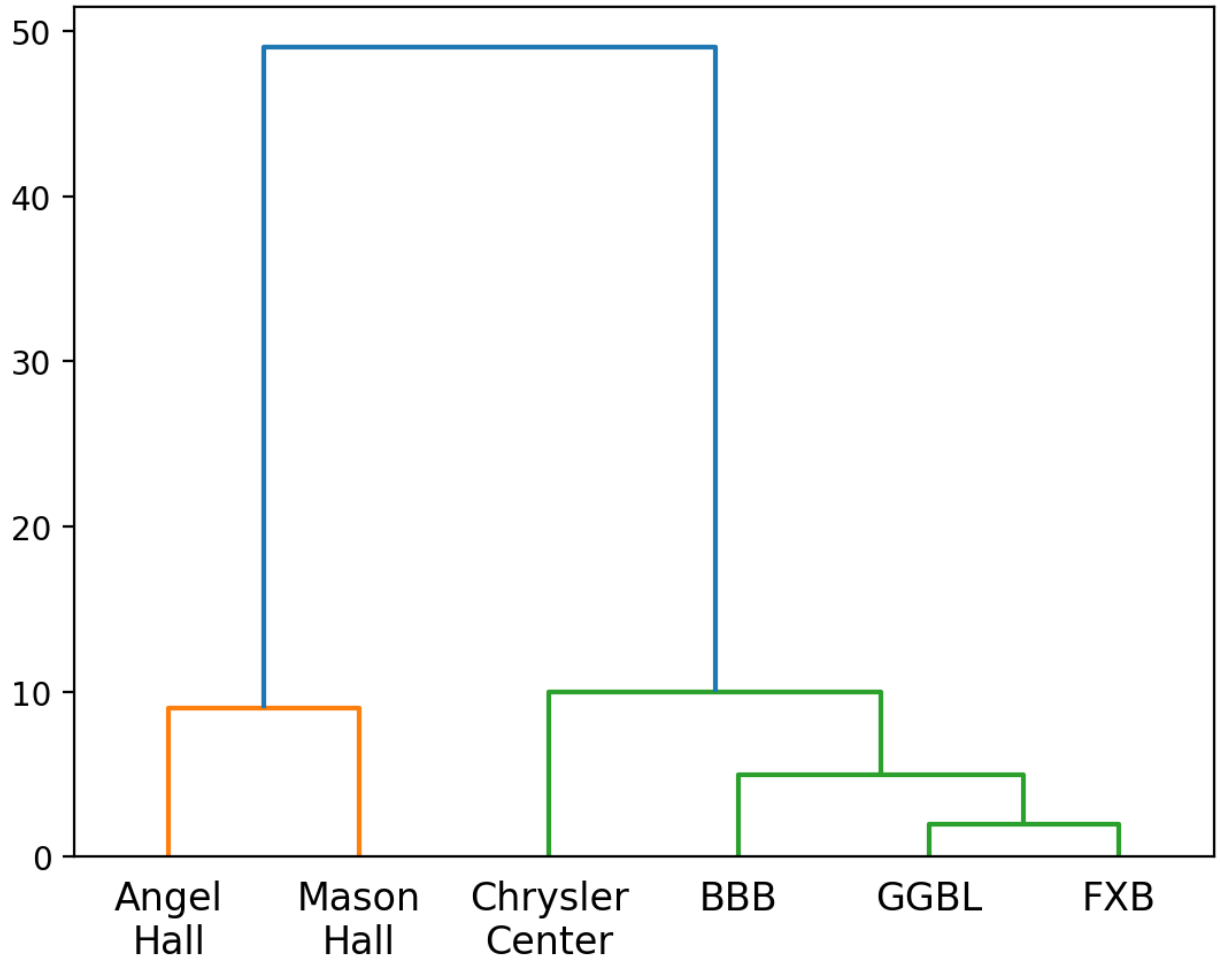
This is because k-means clustering works best on globular clusters, and spectral clustering is better at handling the clusters of varying density which is the case in the Mickey.

3 Hierarchical Clustering

(a)

```python
def plot_hierarchical_dendrogram(buildings, ETAs):
    '''
    plot the dendrogram based on hierarchical clustering
    '''
    # TODO: Define a scikit-learn AgglomerativeClustering object
    # Set argument n_clusters(number of clusters) as None, affinity as precomputed, linkage as comple
    agglomerative = AgglomerativeClustering(
        n_clusters=None, affinity='precomputed', linkage='complete', distance_threshold=0)

    # Fit data to obtain clusters
    agglomerative.fit(ETAs)

    # Build the linkage matrix that represents the hierarchical cluster structure
    linkage_matrix = np.column_stack(
        [agglomerative.children_, agglomerative.distances_, np.ones_like(agglomerative.distances_)])

    # TODO: Plot dendrogram using a scipy function dendrogram (scipy.cluster.hierarchy.dendrogram imp
    # Set argument labels as buildings
    dendrogram(linkage_matrix, labels=buildings)

    plt.savefig('hierarchical_dendrogram.png', dpi=200, bbox_inches='tight')
```

3(b)



2 clusters:
Cluster1: Angel Hall, Mason Hall
Cluster2: Chrysler Center, BBB, GGBL, FXB

4 clusters:
Cluster1: Angel Hall
Cluster2: Mason Hall
Cluster3: Chrysler Center
Cluster4: GGBL, FXB, BBB

4 Investigating Spectral Clustering

4.(a)

$$W = \begin{bmatrix} 1 & \frac{3}{4} & \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{3}{4} & 1 & \frac{1}{4} & \frac{1}{2} & 0 \\ \frac{1}{3} & \frac{1}{4} & 1 & \frac{4}{5} & \frac{1}{5} \\ 0 & \frac{1}{2} & \frac{4}{5} & 1 & 0 \\ \frac{1}{3} & 0 & \frac{1}{5} & 0 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} \frac{29}{12} & 0 & 0 & 0 & 0 \\ 0 & \frac{5}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{31}{12} & 0 & 0 \\ 0 & 0 & 0 & \frac{23}{10} & 0 \\ 0 & 0 & 0 & 0 & \frac{23}{15} \end{bmatrix}$$

$$L = \begin{bmatrix} \frac{17}{12} & -\frac{3}{4} & -\frac{1}{3} & 0 & -\frac{1}{3} \\ -\frac{3}{4} & \frac{3}{2} & -\frac{1}{4} & -\frac{1}{2} & 0 \\ -\frac{1}{3} & -\frac{1}{4} & \frac{19}{12} & -\frac{4}{5} & -\frac{1}{5} \\ 0 & -\frac{1}{2} & -\frac{4}{5} & \frac{13}{10} & 0 \\ -\frac{1}{3} & 0 & -\frac{1}{5} & 0 & \frac{8}{15} \end{bmatrix}$$

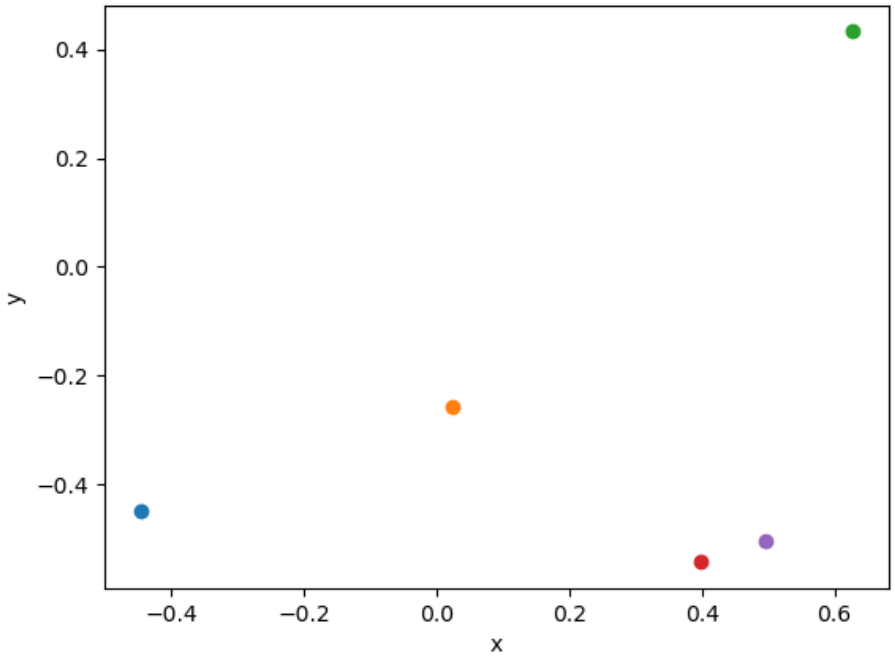
4(b)

```
def spectral_clustering():
    '''
    As specified in Question 4, you will initialize the laplacia
    to the k lowest eigenvalues, and plot the rows of the eigenv
    '''
    lap_matrix = np.matrix([[17/12, -3/4, -1/3, 0, -1/3],
                            [-3/4, 3/2, -1/4, -1/2, 0],
                            [-1/3, -1/4, 19/12, -4/5, -1/5],
                            [0, -1/2, -4/5, 13/10, 0],
                            [-1/3, 0, -1/5, 0, 8/15]])
    print(lap_matrix)
    eigenvalues, eigenvectors = eigh(lap_matrix)
    print(eigenvalues[:2])
    print(eigenvectors[:2])
```

K = 2:
Eigenvalue:
3.9968e-15
5.7311e-01

Eigenvector:
[[-0.4472 0.0246 0.6281 0.3977 0.4967]
 [-0.4472 -0.2568 0.4326 -0.5426 -0.5025]]

(c)



I plot the rows of eigenvectors and based on the plot, we can see that node 1 2 4 5 will be grouped together and the green node 3 itself as a group (the green dot on the upper right corner is node 3.

```
def spectral_clustering():
    '''
    As specified in Question 4, you will initialize the laplacian
    to the k lowest eigenvalues, and plot the rows of the eigenve
    '''
    lap_matrix = np.matrix([[17/12, -3/4, -1/3, 0, -1/3],
                            [-3/4, 3/2, -1/4, -1/2, 0],
                            [-1/3, -1/4, 19/12, -4/5, -1/5],
                            [0, -1/2, -4/5, 13/10, 0],
                            [-1/3, 0, -1/5, 0, 8/15]])
    eigenvalues, eigenvectors = eigh(lap_matrix)
    print(eigenvalues[:2])
    print(eigenvectors[:2])
    for i in range(len(eigenvectors)):
        plt.scatter(eigenvectors[0][i], eigenvectors[1][i])
    plt.xlabel("x")
    plt.ylabel("y")
    plt.savefig('q4 k=2')
```

4(d)

Eigenvalue:

3.9968e-15

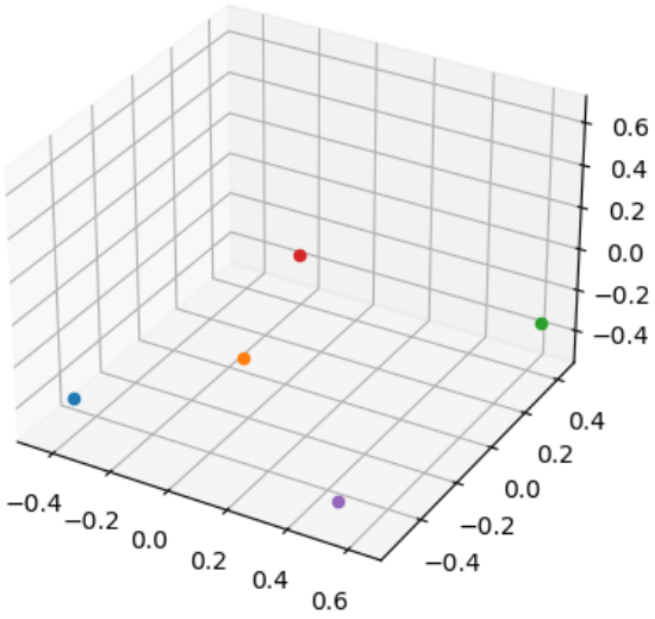
5.7311e-01

1.1935e+00

Eigenvector:

[[-0.4472 0.0246 0.6281 0.3977 0.4967]
 [-0.4472 -0.2568 0.4326 -0.5426 -0.5025]
 [-0.4472 -0.2101 -0.3380 0.6420 -0.4789]]

(e)



The green node 3 itself

The purple node 5 itself

The red node 4, blue node 1 and orange ndoe 2

```python
def spectral_clustering():
    '''
    As specified in Question 4, you will initialize the laplac
    to the k lowest eigenvalues, and plot the rows of the eige
    '''
    lap_matrix = np.matrix([[17/12, -3/4, -1/3, 0, -1/3],
                            [-3/4, 3/2, -1/4, -1/2, 0],
                            [-1/3, -1/4, 19/12, -4/5, -1/5],
                            [0, -1/2, -4/5, 13/10, 0],
                            [-1/3, 0, -1/5, 0, 8/15]])
    eigenvalues, eigenvectors = eigh(lap_matrix)
    print(eigenvalues[:3])
    print(eigenvectors[:3])
    ax = plt.axes(projection='3d')

    for i in range(len(eigenvectors)):
        ax.scatter(eigenvectors[0][i], eigenvectors[1]
                   [i], eigenvectors[2][i])
    plt.savefig('q4 k=3')
```

5.(a) If we fix V, we reduce the problem to :

For each $u^{(i)}$, $i = 1 \dots n$

$$\min_{\bar{u}^{(i)}, \varepsilon_{ij}} \frac{1}{2} \left\| \bar{u}^{(i)} \right\|^2 + C \sum_{\bar{j} : Y_{i\bar{j}} \neq 0} \varepsilon_{i\bar{j}}$$

Subject to $Y_{i\bar{j}} \left(\bar{u}^{(i)} \cdot \bar{v}^{(\bar{j})} \right) \geq 1 - \varepsilon_{i\bar{j}}$,

$$\varepsilon_{i\bar{j}} \geq 0 \text{ for all } \bar{j} \text{ where } Y_{i\bar{j}} \neq 0$$

We now have n independent soft-margin SVM problems, one for each $\bar{u}^{(i)}$ with labeled examples $\left\{ \left(\bar{v}^{(\bar{j})}, Y_{i\bar{j}} \right) \right\}_{\bar{j} : Y_{i\bar{j}} \neq 0}$

5(b) i.

$$\min_{\bar{u}^{(1)},\, \varepsilon_{11},\, \varepsilon_{12}} \frac{1}{2}\left\|\bar{u}^{(1)}\right\|^2 + C\varepsilon_{11} + C\varepsilon_{12}$$

Subject to $Y_{11}(\bar{u}^{(1)} \cdot \bar{v}^{(1)}) \geq 1 - \varepsilon_{11}$,

$Y_{12}(\bar{u}^{(1)} \cdot \bar{v}^{(2)}) \geq 1 - \varepsilon_{12}$,

Plugging in $\bar{v}^{(1)}, \bar{v}^{(2)}, Y_{11}, Y_{12}$, we have

(1) $\bar{u}_1^{(1)} \geq 1 - \varepsilon_{11}$

(1) $\bar{u}_2^{(1)} \geq 1 - \varepsilon_{12}$

$$\min_{\bar{u}^{(2)},\, \varepsilon_{21}} \frac{1}{2}\left\|\bar{u}^{(2)}\right\|^2 + C\varepsilon_{21}$$

Subject to $Y_{21}(\bar{u}^{(2)} \cdot \bar{v}^{(1)}) \geq 1 - \varepsilon_{21}$,

Plugging in $\bar{v}^{(1)}, Y_{21}$ we have

(-1) $\bar{u}_1^{(2)} \geq 1 - \varepsilon_{21}$

5. (b) ii. When $C = \infty$, this is hard margin SVM

$$\varepsilon_{11}, \varepsilon_{12}, \varepsilon_{21} = 0$$

For $u^{(1)}$, $\min_{\bar{u}^{(1)}} \frac{1}{2}\|\bar{u}^{(1)}\|^2$

subject to $Y_{11}(\bar{u}^{(1)} \cdot \bar{v}^{(1)}) \geq 1$

$Y_{12}(\bar{u}^{(1)} \cdot \bar{v}^{(2)}) \geq 1$

plugging in $\bar{v}^{(1)}, \bar{v}^{(2)}, Y_{11}, Y_{12}$, we get:

$(1)\ \bar{u}_1^{(1)} \geq 1$

$(1)\ \bar{u}_2^{(1)} \geq 1$

The problem is minimized when $\bar{u}_1^{(1)} = 1$, $\bar{u}_2^{(1)} = 1$

Therefore, $\bar{u}^{(1)} = [1, 1]^T$

For $\bar{u}^{(2)}$, $\min_{\bar{u}^{(2)}, \varepsilon_{21}} \frac{1}{2}\|\bar{u}^{(2)}\|^2$

subject to $Y_{21}(\bar{u}^{(2)} \cdot \bar{v}^{(1)}) \geq 1$

plugging in $\bar{v}^{(1)}, Y_{21}$ we have

$(-1)\ \bar{u}_1^{(2)} \geq 1$

The problem is minimized when $\bar{u}_1^{(2)} = -1$ $\bar{u}_2^{(2)} = 0$

Therefore, $\bar{u}^{(2)} = [-1, 0]^T$

5 (b) iii.

$$\hat{Y}_{22} = \bar{u}^{(2)} \cdot \bar{v}^{(2)} = [-1, 0]^T [0, 1]^T$$

$$= 0$$

No, we can't predict $\hat{Y}_{22}$ as -1 or 1

$\hat{Y}_{22}$ is 0. It is unclear whether it should be predict as -1 or 1

5.(b) iv.

New user is a new user registers and has not provided any interaction yet. When a new user enrolls in the system and for a certain period of time the recommender has to provide recommendation without relying on the user's past interactions, we can rely on user's features (e.g. age, gender, country) and try to find similar users and recommend the items they interacted with in a positive way.
The other method is that if no demographic feature is present or their quality is too poor, a common strategy is to offer them non-personalized recommendations. This means that they could be recommended simply the most popular items either globally or for their specific geographical region or language.

When items added to the catalogue have either none or very little interactions. We can use content-based filtering algorithms. Since content based recommenders choose which items to recommend based on the feature the items possess, even if no interaction for a new item exist, still its features will allow for a recommendation to be made.