# Advanced Artificial Intelligence
# Representational Learning

## Assignment-2


"As students of the University of Windsor, we pledge to pursue all endeavours with honour and integrity, and will not tolerate or engage in academic or personal dishonesty. We confirm that we have not received any unauthorized assistance in preparing for or writing this assignment. We acknowledge that a mark of 0 may be assigned for copied work." **Jaskaran Singh Luthra 110090236**

Submitted By:

1. Harjot Singh
2. Jaskaran Singh Luthra
3. Bikramjeet Singh

***Important Note: In this report we've attached all the screenshots and plots for the "Spiral" dataset. All the plots and outputs are present in a separate file.***

# Solution 1:

```
import pandas as pd


df = pd.read_csv('/content/sample_data/spiral1.csv')


X=df[['x','y']]
y=df['label']
X=X.values
```

## Solution 2:

a. **K-means**: For all the datasets, we 've used "Euclidean" distance as our Distance function as the datasets are symmetrical, and as per the datasets Euclidean distance gives the most accurate finding for k-means.

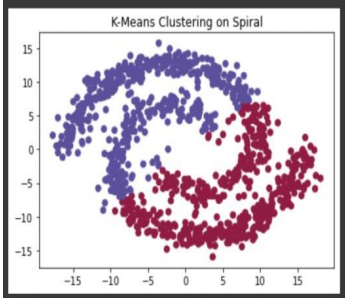
**b. EM**

c. **Spectral Clustering**: In this code, the "Spectral Clustering" class from the sklearn library is used to implement Spectral Clustering. The number of clusters is set to 2, and the affinity parameter is set to 'rbf' to use the RBF kernel. The RBF kernel measures the similarity between two points based on the distance between them. It is well-suited for clustering data that is non-linearly separable, which is the case for all Dataset.

## Spiral dataset

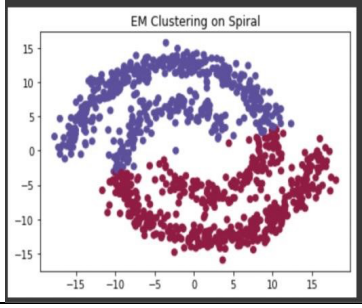
Kmeans

```
from sklearn.cluster import KMeans

#k means
kmeans = KMeans(n_clusters=2, random_state=0)
kmeans.fit(X)
labels_ = kmeans.labels_
plt.scatter(X[:, 0], X[:, 1], c=labels_, cmap='Spectral')
plt.title("K-Means Clustering on Spiral")
plt.show()
```



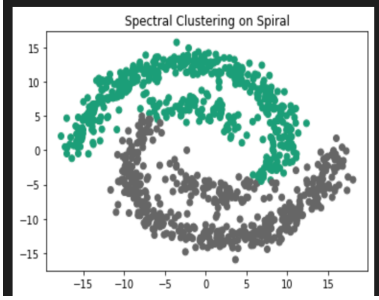

EM

```
# EM Clustering


gmm = GaussianMixture(n_components=2, random_state=0)
gmm.fit(X)
labels_ = gmm.predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels_, cmap='Spectral')
plt.title("EM Clustering on Spiral")
plt.show()
```




Spectral

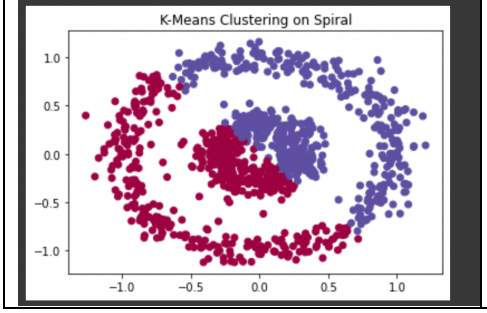
```
# Spectral Clustering
#choosing rbf kernel
from sklearn.cluster import SpectralClustering
sc = SpectralClustering(n_clusters=2, affinity='rbf', rando
labels_ = sc.fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels_,cmap='Dark2')
plt.title("Spectral Clustering on Spiral")
plt.show()
```




## Circles Dataset

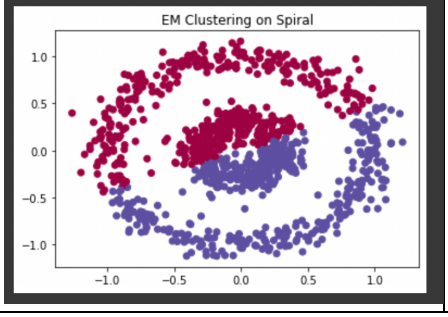
Kmeans

```
#k means
kmeans = KMeans(n_clusters=2, random_state=0)
kmeans.fit(X)
labels_ = kmeans.labels_
plt.scatter(X[:, 0], X[:, 1], c=labels_, cmap='S
plt.title("K-Means Clustering on Spiral")
plt.show()
```



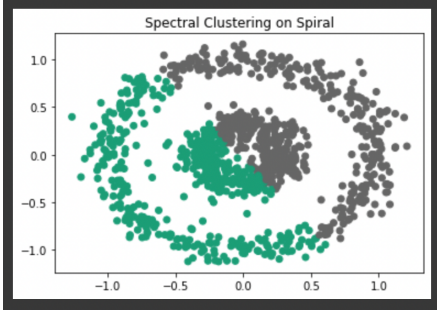

EM

```
# EM Clustering


gmm = GaussianMixture(n_components=2, random_st
gmm.fit(X)
labels_ = gmm.predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels_, cmap='
plt.title("EM Clustering on Spiral")
plt.show()
```



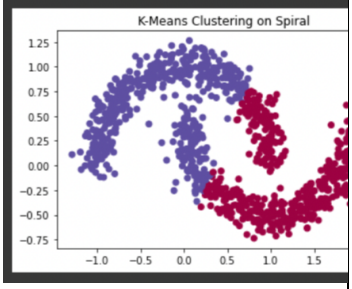

Spectral

```
# Spectral Clustering
#choosing rbf kernel
from sklearn.cluster import SpectralClustering
sc = SpectralClustering(n_clusters=2, affinity='rb
labels_ = sc.fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels_,cmap='Dark
plt.title("Spectral Clustering on Spiral")
plt.show()
```

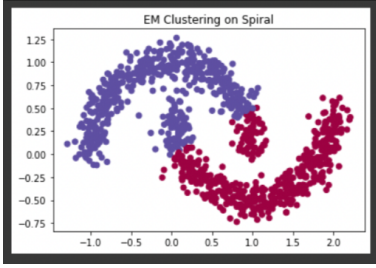


## Moons Dataset

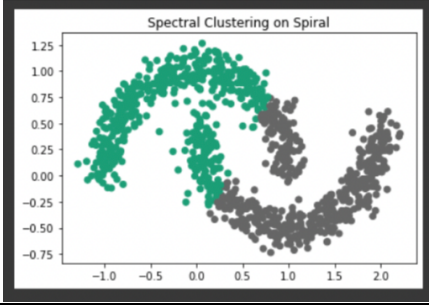
```
#k means
kmeans = KMeans(n_clusters=2, r
kmeans.fit(X)
labels_ = kmeans.labels_
plt.scatter(X[:, 0], X[:, 1], c
plt.title("K-Means Clustering o
plt.show()
```




```
# EM Clustering

gmm = GaussianMixture(n_components=2, random_s
gmm.fit(X)
labels_ = gmm.predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels_, cmap=
plt.title("EM Clustering on Spiral")
plt.show()
```

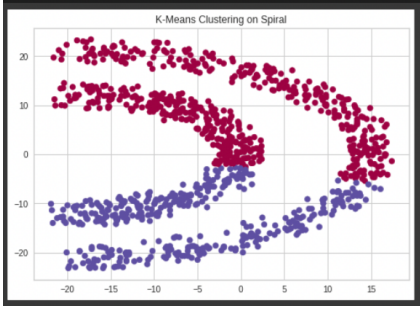



```
# Spectral Clustering
#choosing rbf kernel
from sklearn.cluster import SpectralClustering
sc = SpectralClustering(n_clusters=2, affinity='
labels_ = sc.fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels_,cmap='Da
plt.title("Spectral Clustering on Spiral")
plt.show()
```

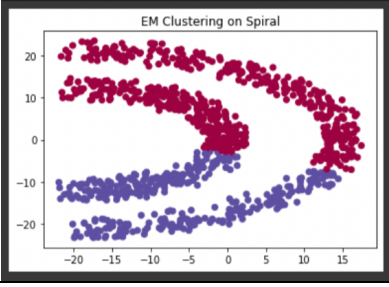



## Half Kernel Dataset:

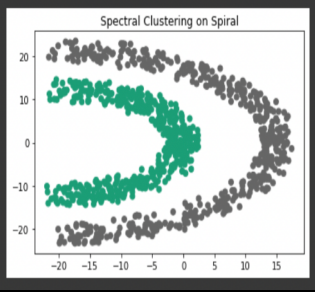
```
#k means
kmeans = KMeans(n_clusters=2, random_state=0)
kmeans.fit(X)
labels_ = kmeans.labels_
plt.scatter(X[:, 0], X[:, 1], c=labels_, cmap='Spe
plt.title("K-Means Clustering on Spiral")
plt.show()
```




```
# EM Clustering

gmm = GaussianMixture(n_components=2, randor
gmm.fit(X)
labels_ = gmm.predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels_, cma
plt.title("EM Clustering on Spiral")
plt.show()
```

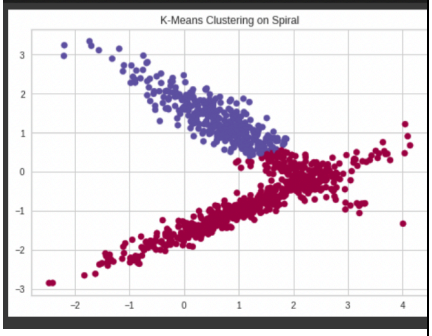



```
# Spectral Clustering
#choosing rbf kernel
from sklearn.cluster import SpectralClustering
sc = SpectralClustering(n_clusters=2, affinity='rb
labels_ = sc.fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels_,cmap='Dark
plt.title("Spectral Clustering on Spiral")
plt.show()
```

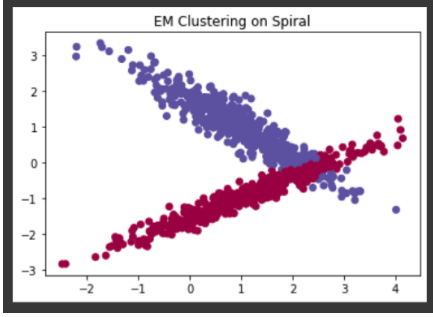



## Two Gaussian:

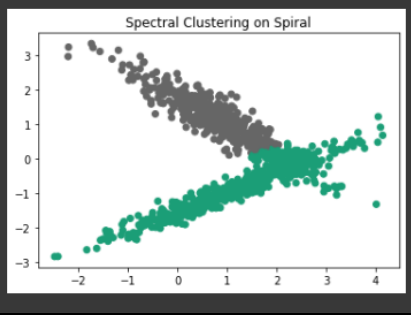
```
#k means
kmeans = KMeans(n_clusters=2, random_state=(
kmeans.fit(X)
labels_ = kmeans.labels_
plt.scatter(X[:, 0], X[:, 1], c=labels_, cma
plt.title("K-Means Clustering on Spiral")
plt.show()
```




```
gmm = GaussianMixture(n_components=2, ran
gmm.fit(X)
labels_ = gmm.predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels_,
plt.title("EM Clustering on Spiral")
plt.show()
```




```
# Spectral Clustering
#choosing rbf kernel
from sklearn.cluster import SpectralClustering
sc = SpectralClustering(n_clusters=2, affinity='rbf
labels_ = sc.fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels_,cmap='Dark2
plt.title("Spectral Clustering on Spiral")
plt.show()
```



# Solution 3:

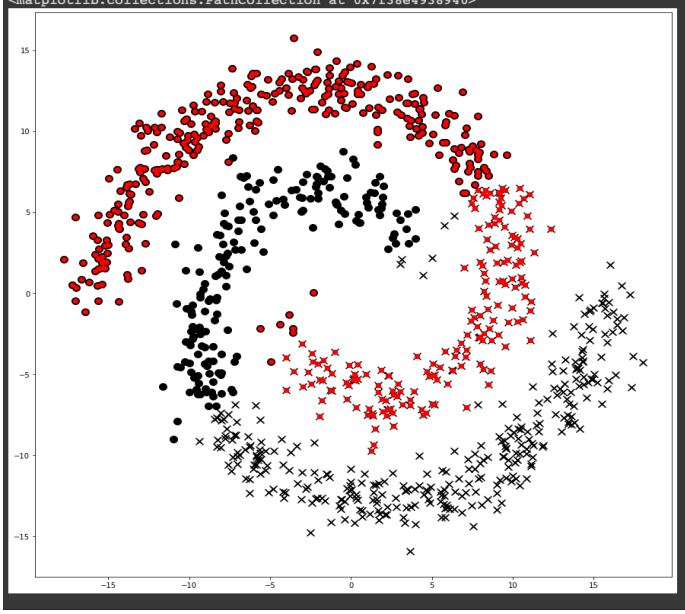
If we see here, we've followed the convention that same class have the same class whereas same clusters have the same colour.

***Note:** These Plots are for SPIRAL Dataset other datasets plots are in Appendix file.*

## I. K-MEANS:

```
markers = ["x","o"]
colors = ['black','r']
plt.figure(figsize =(15, 15))
#plt.scatter(X[:, 0], X[:, 1], c=labels_, cmap='Spectral')

plt.scatter(df_2[df_2[0]==0.0]['x'], df_2[df_2[0]==0.0]['y'],c=colors[0], cmap='Spectral',s=100,marker=markers[0])
plt.scatter(df_2[df_2[0]==1]['x'], df_2[df_2[0]==1]['y'],c=colors[0],  cmap='Spectral',s=100,marker=markers[1])

plt.scatter(df[df['label']==0.0]['x'], df[df['label']==0.0]['y'], marker=markers[0], cmap='Spectral',s=40,c=colors[0])
plt.scatter(df[df['label']==1.0]['x'], df[df['label']==1.0]['y'], marker=markers[1], cmap='Spectral',s=40,c=colors[1])
```

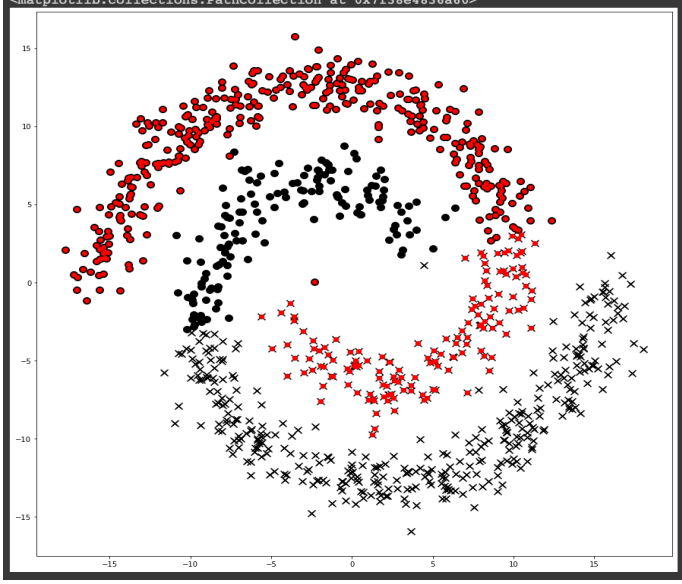
```
<matplotlib.collections.PathCollection at 0x7f38e4938940>
```




## II. EM – Clustering:

```
markers = ["x","o"]
colors = ['black','r']
plt.figure(figsize =(15, 15))
#plt.scatter(X[:, 0], X[:, 1], c=labels_, cmap='Spectral')

plt.scatter(df_2[df_2[0]==0.0]['x'], df_2[df_2[0]==0.0]['y'],c=colors[0], cmap='Spectral',s=100,marker=markers[0])
plt.scatter(df_2[df_2[0]==1]['x'], df_2[df_2[0]==1]['y'],c=colors[0],  cmap='Spectral',s=100,marker=markers[1])

plt.scatter(df[df['label']==0.0]['x'], df[df['label']==0.0]['y'], marker=markers[0], cmap='Spectral',s=40,c=colors[0])
plt.scatter(df[df['label']==1.0]['x'], df[df['label']==1.0]['y'], marker=markers[1], cmap='Spectral',s=40,c=colors[1])
```

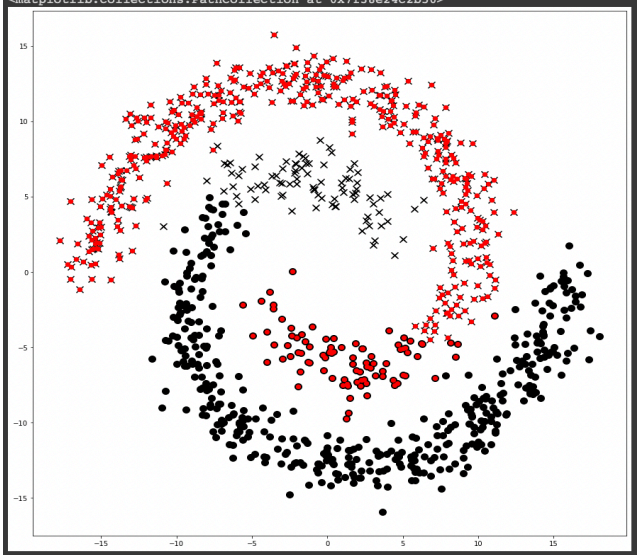
```
<matplotlib.collections.PathCollection at 0x7f38e4836a60>
```



## III. SPECTRAL CLUSTERING:

```
markers = ["x","o"]
colors = ['black','r']
plt.figure(figsize =(15, 15))
#plt.scatter(X[:, 0], X[:, 1], c=labels_, cmap='Spectral')

plt.scatter(df_2[df_2[0]==0.0]['x'], df_2[df_2[0]==0.0]['y'],c=colors[0], cmap='Spectral',s=100,marker=markers[0])
plt.scatter(df_2[df_2[0]==1]['x'], df_2[df_2[0]==1]['y'],c=colors[0],  cmap='Spectral',s=100,marker=markers[1])

plt.scatter(df[df['label']==0.0]['x'], df[df['label']==0.0]['y'], marker=markers[0], cmap='Spectral',s=40,c=colors[0]
plt.scatter(df[df['label']==1.0]['x'], df[df['label']==1.0]['y'], marker=markers[1], cmap='Spectral',s=40,c=colors[1]
```

```
<matplotlib.collections.PathCollection at 0x7f38e24c2b50>
```




# Solution 4:

Here's a summary of the results obtained for each dataset and clustering algorithm:

1. **circles0.3:**

- k-means: K-means algorithm is not well-suited for circular data like the circles0.3 dataset, and it produces poor results with high within-cluster variance.
- EM: EM algorithm performs better than k-means, and it can identify the circular structure in the dataset. However, the algorithm struggles to separate the overlapping circles due to the data's high noise level.
- Spectral clustering: Spectral clustering with an RBF kernel performs well on circles0.3 dataset as it can capture the circular structure in the data and separate the overlapping circles.

2. **moons1:**

- k-means: K-means algorithm fails to capture the moon-shaped structure of the dataset and produces poor results with high within-cluster variance.
- EM: EM algorithm performs better than k-means, and it can identify the two moon-shaped clusters. However, the algorithm struggles to separate the overlapping parts of the moons.
- Spectral clustering: Spectral clustering with a 3-NN or RBF kernel performs well on moons1 dataset as it can capture the moon-shaped structure in the data and separate the two moons.

3. **spiral1:**

- k-means: K-means algorithm is not well-suited for spiral-shaped data like the spiral1 dataset, and it produces poor results with high within-cluster variance.
- EM: EM algorithm performs better than k-means, and it can identify the two spirals in the dataset. However, the algorithm struggles to separate the overlapping parts of the spirals.
- Spectral clustering: Spectral clustering with an RBF kernel performs well on spiral1 dataset as it can capture the spiral-shaped structure in the data and separate the two spirals.

4. **twogaussians42:**

- k-means: K-means algorithm performs well on twogaussians42 dataset as it can identify the two Gaussian distributions and separate them.
- EM: EM algorithm also performs well on twogaussians42 dataset and produces similar results to k-means.
- Spectral clustering: Spectral clustering with an RBF kernel performs well on twogaussians42 dataset and produces similar results to k-means and EM.

5. **halfkernel:**

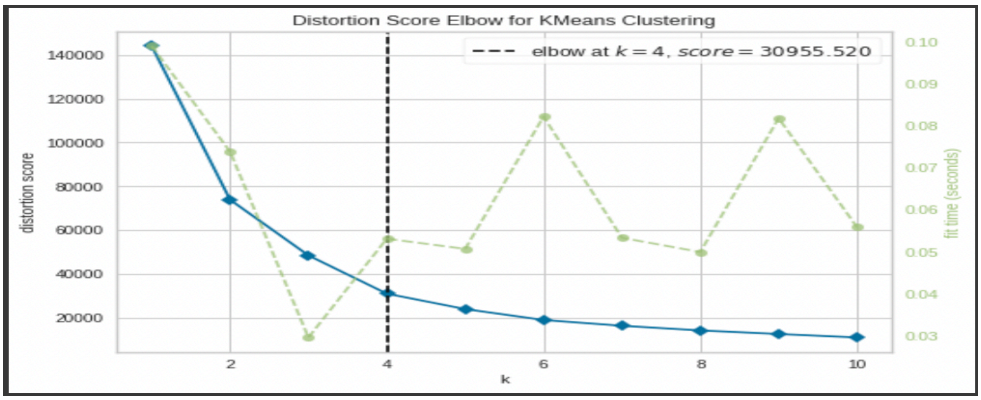
- k-means: K-means algorithm is not well-suited for halfkernel dataset as it produces poor results with high within-cluster variance.
- EM: EM algorithm performs better than k-means, and it can identify the two clusters in the dataset. However, the algorithm struggles to separate the overlapping parts of the half kernel.
- Spectral clustering: Spectral clustering with an RBF kernel performs well on halfkernel dataset as it can capture the half kernel-shaped structure in the data and separate the two clusters.

In summary, the choice of clustering algorithm and parameters depends on the characteristics of the dataset. In general, spectral clustering with an RBF kernel performs well on most datasets as it can capture the underlying structure in the data and separate the clusters. However, EM algorithm may be more suitable for datasets with overlapping clusters. K-means algorithm may be suitable for datasets with well-separated clusters and a clear underlying structure.

# Solution 5:

**5a)**

```
from yellowbrick.cluster import KElbowVisualizer
# Instantiate the clustering model and visualizer
model = KMeans()
visualizer = KElbowVisualizer(model, k=(1, 11))

visualizer.fit(X) # Fit the data to the visualizer
visualizer.show() # Finalize and render the figure
plt.show()
#Therefore, from figure it can be seen that the optimal number of clusters is 4.
```




## K for
## SPIRAL: 4, MOONS: 3, CIRCLES: 5, TWO GAUSIAN: 3, HALF KERNEL: 3

***(Note: Elbow Plots of other datasets is in Appendix file)***

**5b)**

The index of validity is a measure of the quality of the clustering solution. It compares the ratio of the within-cluster sum of squared distances (WCSS) to the between-cluster sum of squared distances (BCSS). The BCSS measures the dissimilarity between the clusters, while the WCSS measures the dissimilarity within the clusters. The index of validity is defined as the ratio BCSS/WCSS, and a higher value indicates a better clustering solution. The index of validity can be used to compare different clustering solutions for a fixed value of k, or to compare different values of k for a fixed clustering algorithm.