

Q1. Download the following datasets: circles0.3, moons1, spiral1, twogaussians33, twogaussians42, and halfkernel.

A1) The following datasets have been downloaded from the resources which are: - circles0.3, moons1, spiral1, twogaussians33, twogaussians42, and halfkernel.

The following can be read in the jupyter notebook using the code: -

Dataset: - Circles0.3

```
dataset1 = pd.read_csv('C:\Python37\datasets\circles0.3.csv')
```

Dataset: -Halfkernel

```
dataset2 = pd.read_csv('C:\Python37\datasets\halfkernel.csv')
```

Dataset: -Moons1

```
dataset3 = pd.read_csv('C:\Python37\datasets\moons1.csv')
```

Dataset: -Spiral1

```
dataset4 = pd.read_csv('C:\Python37\datasets\spiral1.csv')
```

Dataset: -Twogaussians33

```
dataset5 = pd.read_csv('C:\\Python37\\datasets\\twogaussians33.csv')
```

Dataset: -Twogaussians42

```
dataset6 = pd.read_csv('C:\\Python37\\datasets\\twogaussians42.csv')
```

Now all the datasets are connected to the jupyter notebook and hence we can begin with the operation of loading the dataset.

Dataset: - Circles0.3

```
df1=pd.DataFrame(dataset1)  
df1_nolabel1 = df1.drop(['label'],axis=1)
```

Dataset: - HalfKernel

```
df2=pd.DataFrame(dataset2)  
df2_nolabel2 = df2.drop(['label'],axis=1)
```

Dataset: - Moons1

```
df3=pd.DataFrame(dataset3)  
df3_nolabel3 = df3.drop(['label'],axis=1)
```

Dataset: - Spiral1

```
: df4=pd.DataFrame(dataset4)
df4_nolabel4 = df4.drop(['label'],axis=1)
```

Dataset:- Two Gaussian33

```
df5=pd.DataFrame(dataset5)
df5_nolabel5 = df5.drop(['label'],axis=1)
```

Dataset:- Two Gaussian 42

```
: df6=pd.DataFrame(dataset6)
df6_nolabel6 = df6.drop(['label'],axis=1)
```

Q2. For each dataset, remove the class labels, and run the following clustering algorithms,

where $k = 2$:

a. k-means: Choose a distance function and explain why you chose it.

b. EM

c. Spectral clustering: Choose a kernel and number of neighbours and explain why

you chose them. Suggestion: use RBF.

A2)

K Means

K means is a form of unsupervised learning, also the main aim of k means is to find the data groups which are present in the given datasets which is given by the variable "k" in k means, also we use value as $k=2$ in the given datasets for the operation

Also we make use of the euclidean distance in the K means upon all the datasets because squared deviation summation from a reference point which in this case we consider as centroid is the sum of squared euclidean distance of pairwise values under division of number of points which provides better results and hence we consider euclidean distance over any other distances

1) Applying K Means Algorithm to the given datasets:-

● Dataset: - Circles0.3

```
model = KMeans(n_clusters=2)
model.fit(df1_nolabel1)
df1_predarray = model.fit_predict(df1_nolabel1)
df1_pred = pd.DataFrame(df1_predarray, columns = ['predicted'])#converting to dataframe
frames = [df1, df1_pred]
result = pd.concat(frames, axis=1)
```

● Dataset: - HalfKernel

```
model = KMeans(n_clusters=2)
model.fit(df2_nolabel2)
df2_pred = model.fit_predict(df2_nolabel2)
df2_pred = pd.DataFrame(df2_pred, columns = ['predicted'])#converting to dataframe
frames = [df2, df2_pred]
result = pd.concat(frames, axis=1)
```

● Dataset: - Moons1

```
model = KMeans(n_clusters=2)
model.fit(df3_nolabel3)
df3_pred = model.fit_predict(df3_nolabel3)
df3_pred = pd.DataFrame(df3_pred, columns = ['predicted'])#converting to dataframe
frames = [df3, df3_pred]
result = pd.concat(frames, axis=1)
```

● Dataset: - Spiral1

```
: model = KMeans(n_clusters=2)
model.fit(df4_nolabel4)
df4_pred = model.fit_predict(df4_nolabel4)
df4_pred = pd.DataFrame(df4_pred, columns=['predicted'])#converting to dataframe
frames = [df4, df4_pred]
result = pd.concat(frames, axis=1)
```

● Dataset:- Twogaussian33

```
: model = KMeans(n_clusters=2)
model.fit(df5_nolabel5)
df5_pred = model.fit_predict(df5_nolabel5)
df5_pred = pd.DataFrame(df5_pred, columns=['predicted'])#converting to dataframe
frames = [df5, df5_pred]
result = pd.concat(frames, axis=1)
```

● Dataset:- Twogaussian42

```
: model = KMeans(n_clusters=2)
model.fit(df6_nolabel6)
df6_pred = model.fit_predict(df6_nolabel6)
df6_pred = pd.DataFrame(df6_pred, columns=['predicted'])#converting to dataframe
frames = [df6, df6_pred]
result = pd.concat(frames, axis=1)
```

2)

EM

EM is a powerful technique which is used for producing predictions of correct data if there exist any missing data values in the given dataset, hence the E-step if we consider the value of set $n \geq 1$ EM auxiliary function is defined by conditional expectation.

And for the M step it adjusts certain values to for the data based on their probability.

Applying EM Algorithm to the given dataset: -

● Dataset: -Circles0.3

```
model = GaussianMixture(n_components=2)
model.fit(df1_nolabel1)
df1_predarray = model.fit_predict(df1_nolabel1)
df1_pred = pd.DataFrame(df1_predarray, columns=['predicted'])#converting to dataframe
frames = [df1, df1_pred]
result = pd.concat(frames, axis=1)
```

● Dataset: -Half Kernel

```
model = GaussianMixture(n_components=2)
model.fit(df2_nolabel2)
df2_predarray = model.fit_predict(df2_nolabel2)
df2_pred = pd.DataFrame(df2_predarray, columns=['predicted'])#converting to dataframe
frames = [df2, df2_pred]
result = pd.concat(frames, axis=1)
```

● Dataset: -Moons1

```
: model = GaussianMixture(n_components=2)
model.fit(df3_nolabel3)
df3_pred = model.fit_predict(df3_nolabel3)
df3_pred = pd.DataFrame(df3_pred, columns=['predicted'])#converting to dataframe
frames = [df3, df3_pred]
result = pd.concat(frames, axis=1)
```

● Dataset: -Spiral1

```
: model = GaussianMixture(n_components=2)
model.fit(df4_nolabel4)
df4_pred = model.fit_predict(df4_nolabel4)
df4_pred = pd.DataFrame(df4_pred, columns=['predicted'])#converting to dataframe
frames = [df4, df4_pred]
result = pd.concat(frames, axis=1)
```

● Dataset: -Twogaussian33

```
model = GaussianMixture(n_components=2)
model.fit(df5_nolabel5)
df5_pred = model.fit_predict(df5_nolabel5)
df5_pred = pd.DataFrame(df5_pred, columns=['predicted'])#converting to dataframe
frames = [df5, df5_pred]
result = pd.concat(frames, axis=1)
```

● Dataset: -Twogaussian42

```
: model = GaussianMixture(n_components=2)
model.fit(df6_nolabel6)
df6_pred = model.fit_predict(df6_nolabel6)
df6_pred = pd.DataFrame(df6_pred, columns=['predicted'])#converting to dataframe
frames = [df6, df6_pred]
result = pd.concat(frames, axis=1)
```

3) Spectral Clustering

Spectral Clustering is a technique which is used in the graphs to determine the nodes and find the communities as per the connection between the edges also it can be used for a large variety of data. It also makes use of the eigenvalues of spectral matrices for to show their spectrum and and further computes with eigen vectors. In functionality the euclidean distances which are present in this are thereby used to obtain certain values along with “gamma” which shows how much a particular point has influence over the other.

Also the kernel used is RBF as sole reason for it to use is that two points are not directly connected to each other which can be suitable in higher dimensional operation where as in certain cases it can be used by adjusting the value of gamma that is parameter associated with the shape of kernel and hence RBF can be reduced dimensionally and used to obtain clusters.

Applying Spectral Clustering on the following datasets: -

● Dataset:- Circles0.3

```
: model = SpectralClustering(n_clusters=2,affinity="rbf",gamma=10)
model.fit(df1_nolabel1)
df1_predarray = model.fit_predict(df1_nolabel1)
df1_pred = pd.DataFrame(df1_predarray,columns=['predicted'])#converting to dataframe
frames = [df1,df1_pred]
result = pd.concat(frames,axis=1)
```

● Dataset: - half kernel

```
model = SpectralClustering(n_clusters=2,affinity="rbf",gamma=1)
model.fit(df2_nolabel2)
df2_predarray = model.fit_predict(df2_nolabel2)
df2_pred = pd.DataFrame(df2_predarray,columns=['predicted'])#converting to dataframe
frames = [df2,df2_pred]
result = pd.concat(frames,axis=1)
```

● Dataset: - Moons1

```
model = SpectralClustering(n_clusters=2,affinity="rbf",gamma=50)
model.fit(df3_nolabel3)
df3_predarray = model.fit_predict(df3_nolabel3)
df3_pred = pd.DataFrame(df3_predarray,columns=['predicted'])#converting to dataframe
frames = [df3,df3_pred]
result = pd.concat(frames,axis=1)
```

● Dataset: - Spiral1

```
model = SpectralClustering(n_clusters=2,affinity="rbf",gamma=2)
model.fit(df4_nolabel4)
df4_predarray = model.fit_predict(df4_nolabel4)
df4_pred = pd.DataFrame(df4_predarray,columns=['predicted'])#converting to dataframe
frames = [df4,df4_pred]
result = pd.concat(frames,axis=1)
```

● Dataset:- Twogaussian33

```
model = SpectralClustering(n_clusters=2,affinity="rbf")
model.fit(df5_nolabel5)
df5_predarray = model.fit_predict(df5_nolabel5)
df5_pred = pd.DataFrame(df5_predarray,columns=['predicted'])#converting to dataframe
frames = [df5,df5_pred]
result = pd.concat(frames,axis=1)
```

● Dataset:- Twogaussian42

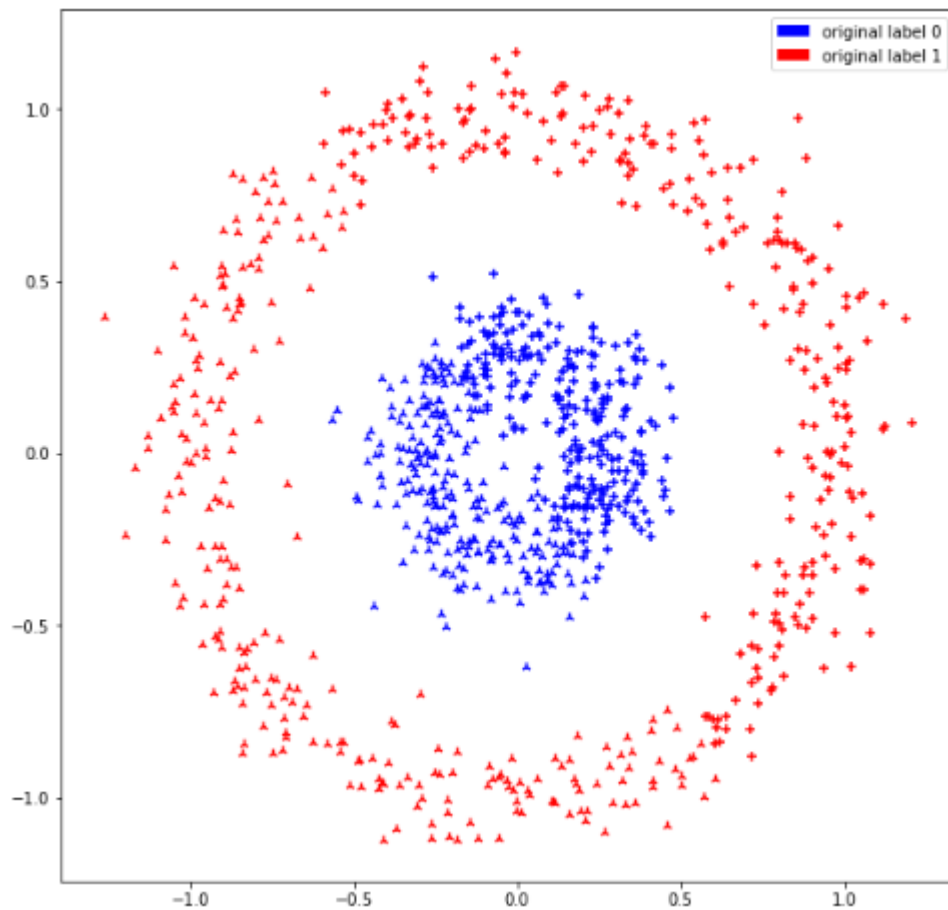
```
model = SpectralClustering(n_clusters=2,affinity="rbf", gamma=10)
model.fit(df6_nolabel6)
df6_predarray = model.fit_predict(df6_nolabel6)
df6_pred = pd.DataFrame(df6_predarray,columns=['predicted'])#converting to dataframe
frames = [df6,df6_pred]
result = pd.concat(frames,axis=1)
```

Q3)Add the original class labels to the existing clusters and plot the points as follows: points in the new cluster should have the same shape; points in the original class should have the same color.

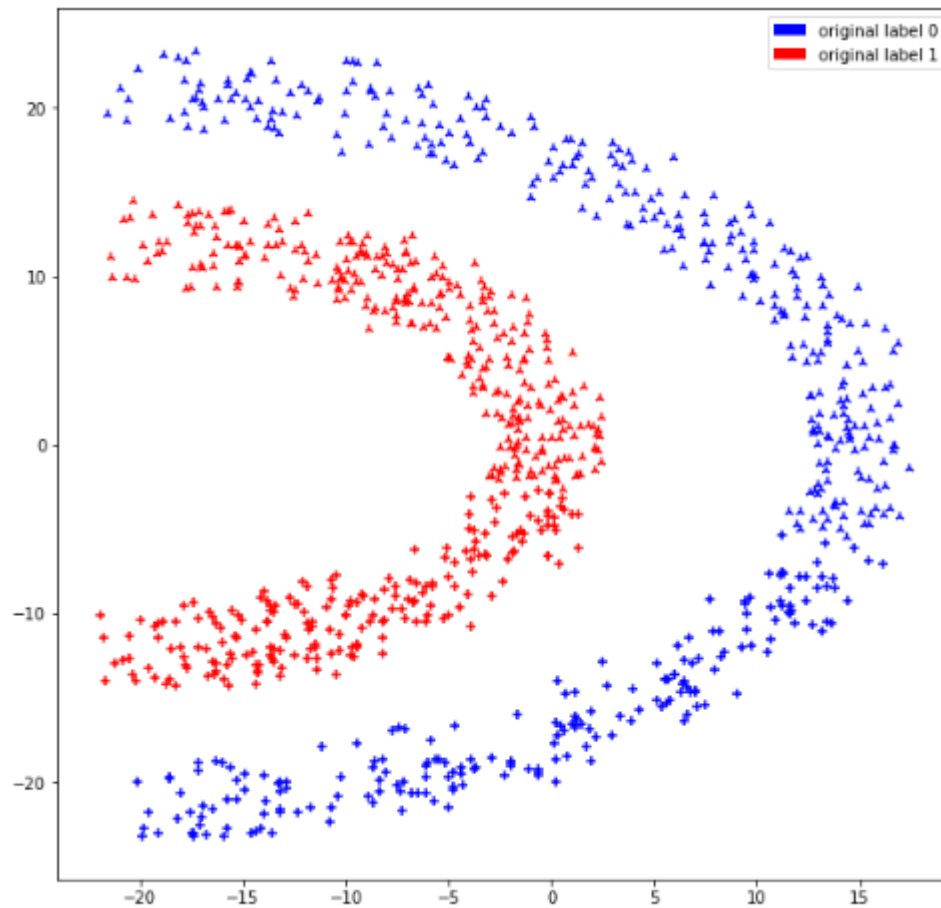
A3)

K-Means

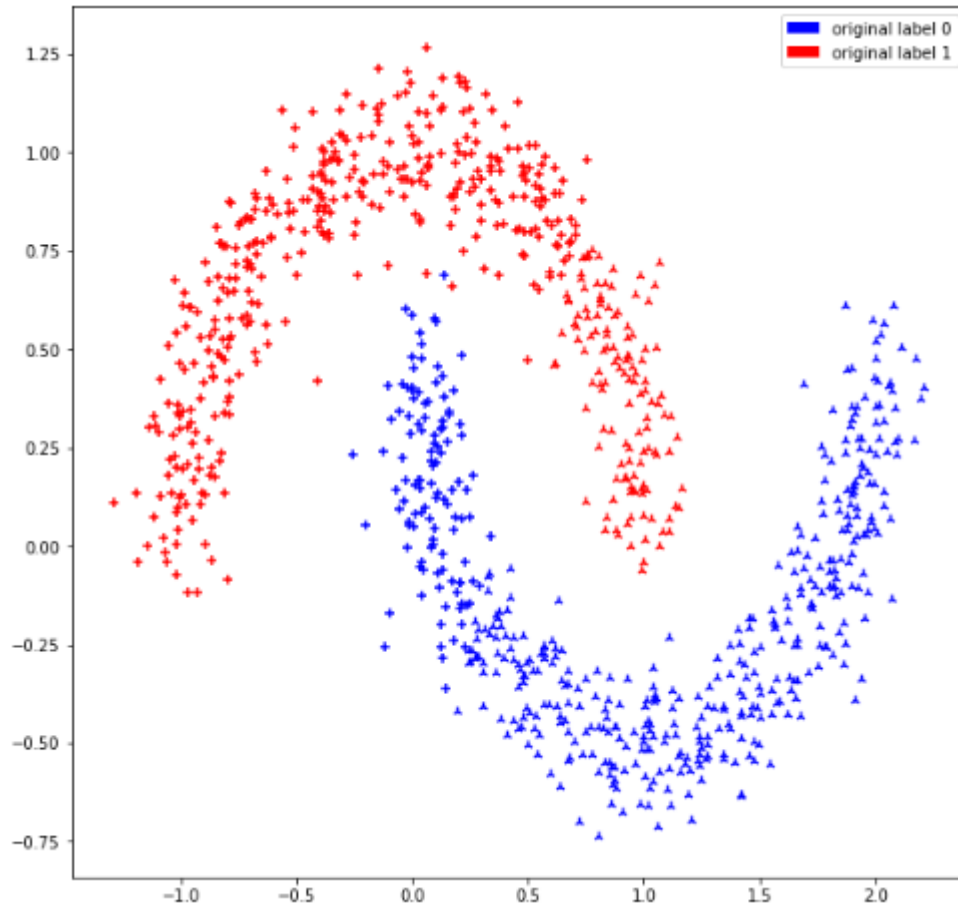
- Dataset: - Circles0.3



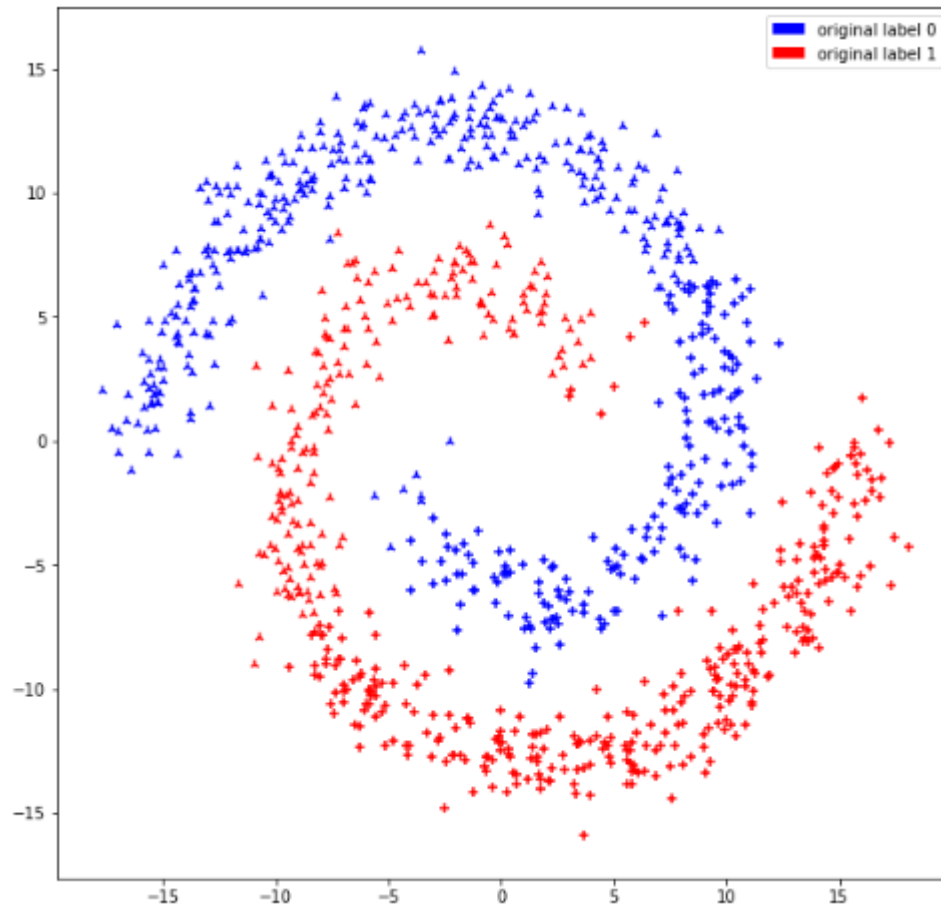
- Dataset: -Half kernel



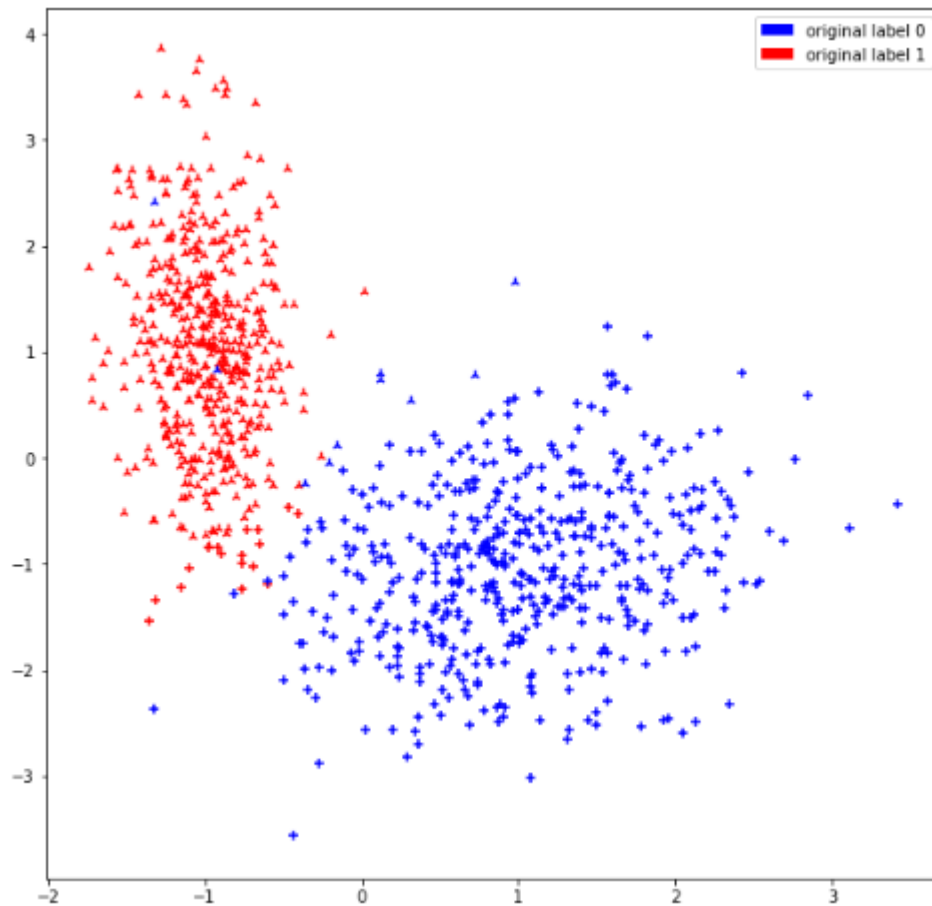
- Dataset: -Moons1



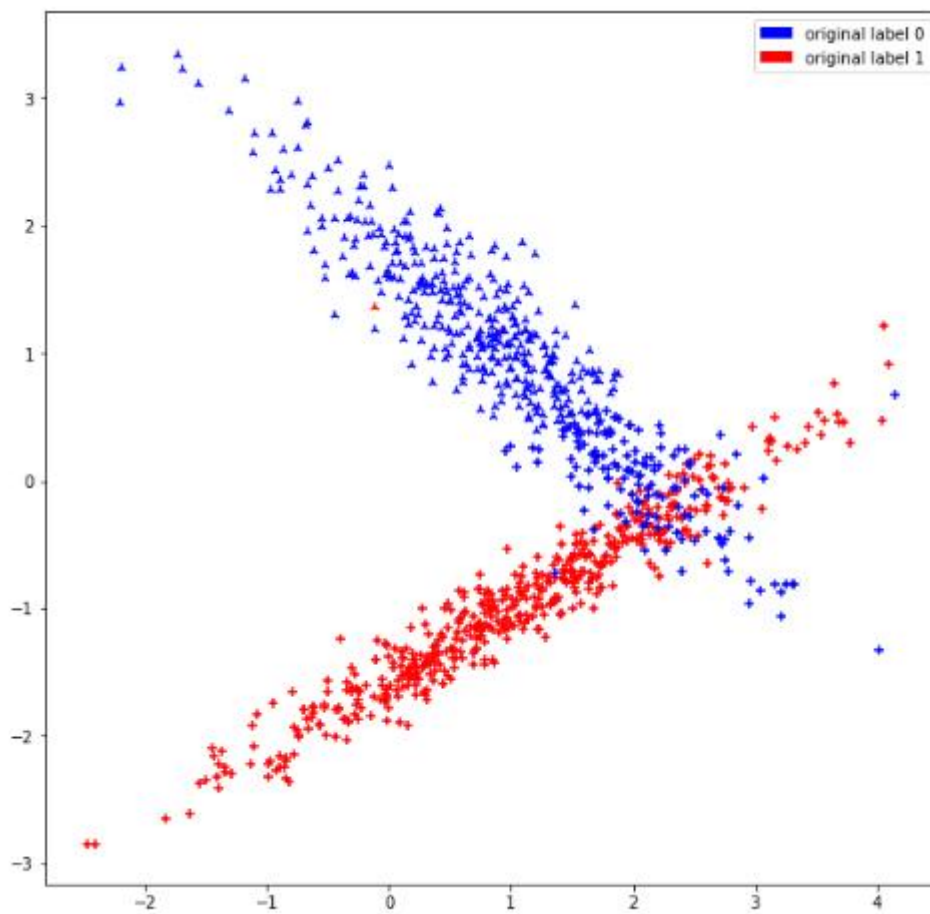
- Dataset: -Spiral1



● Dataset: -Twogaussian33

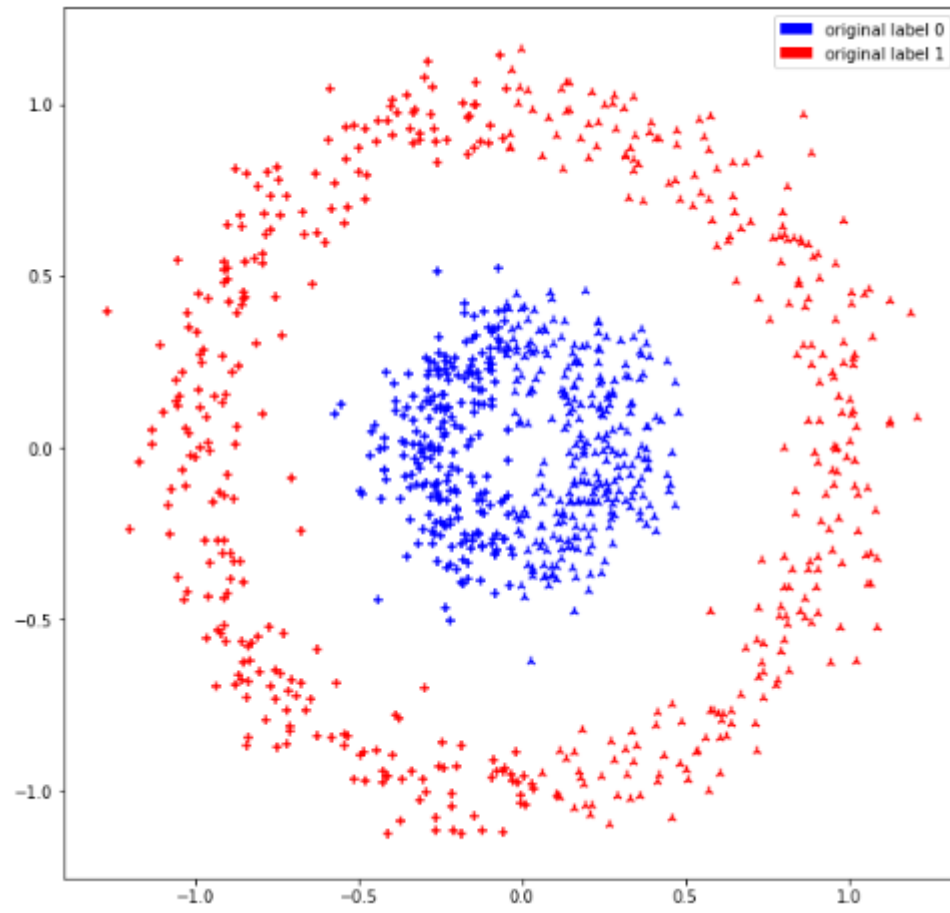


- Dataset: -Twogaussian42

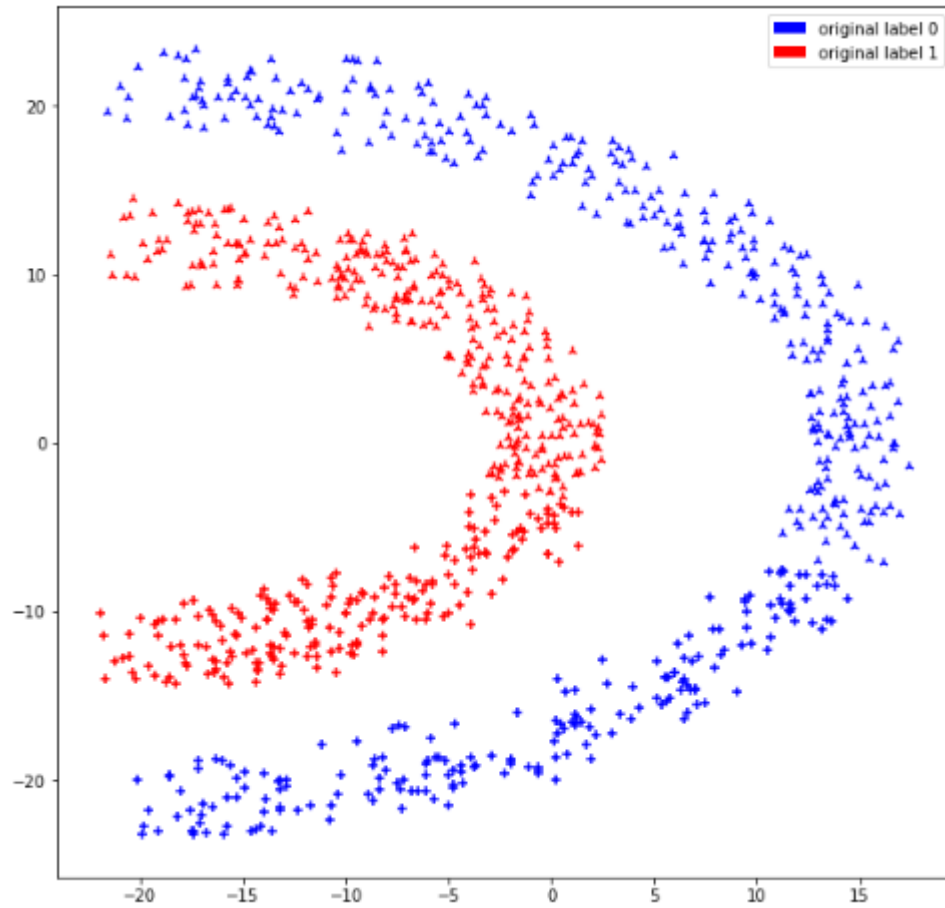


EM

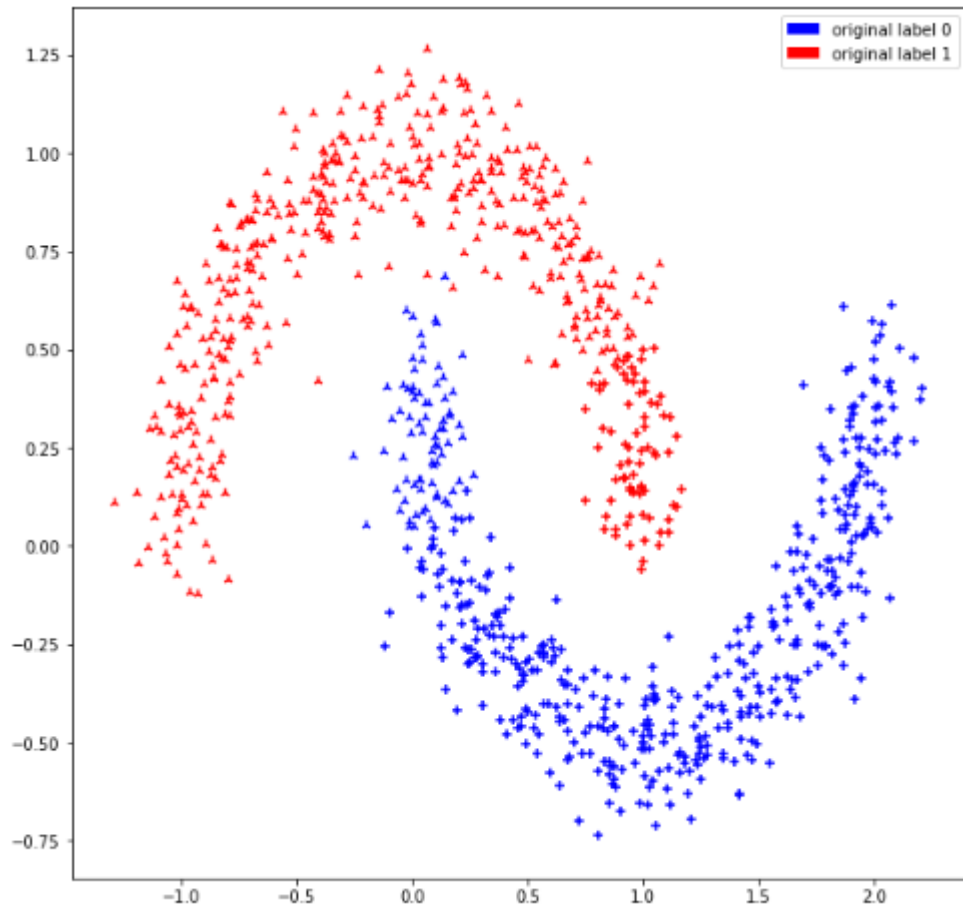
- Dataset: - Circles0.3



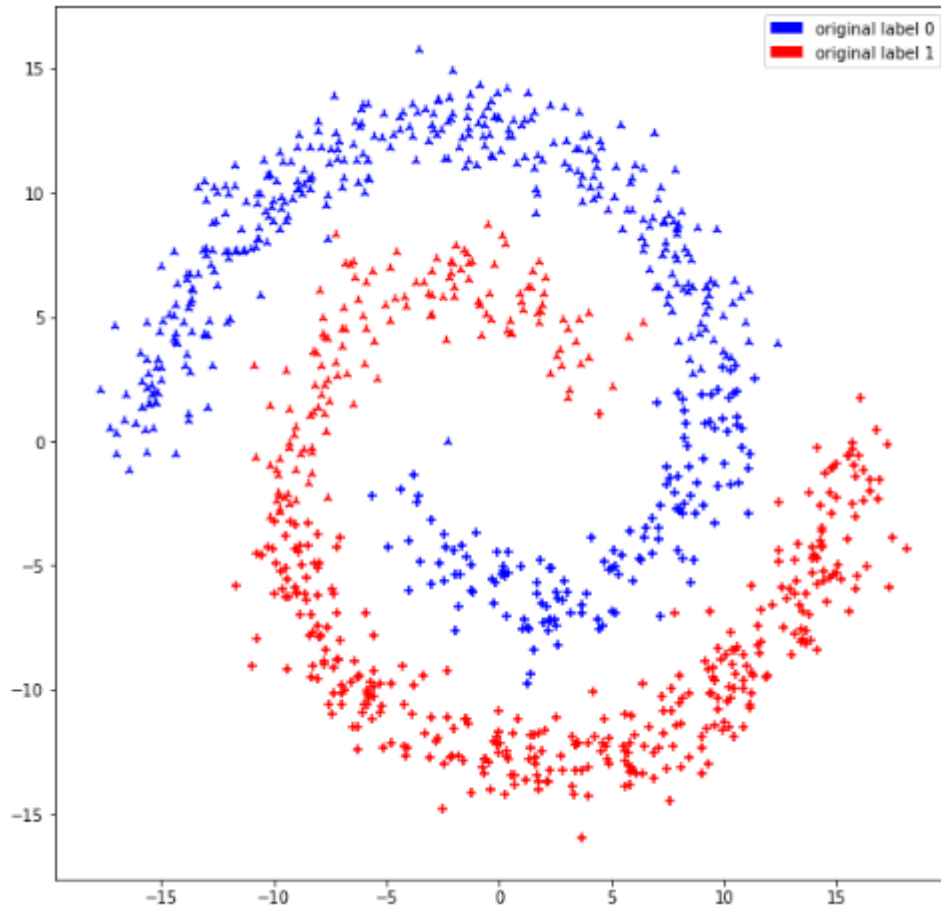
- Dataset: -Half kernel



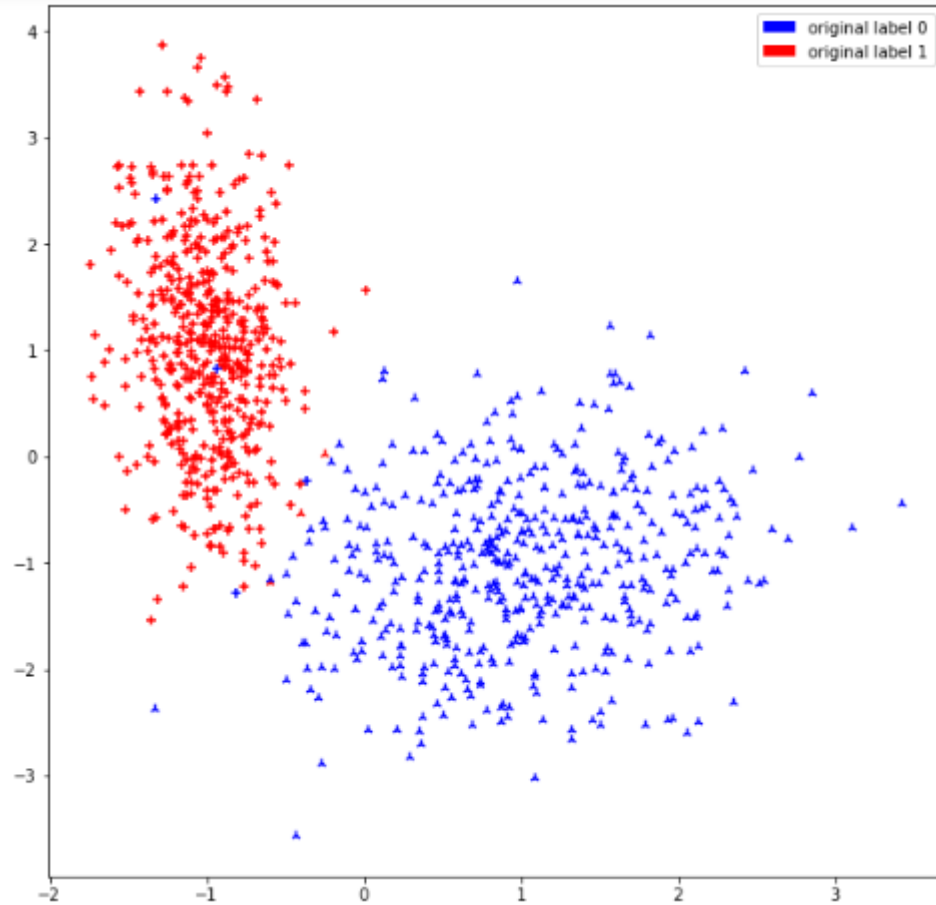
- Dataset: -Moons1



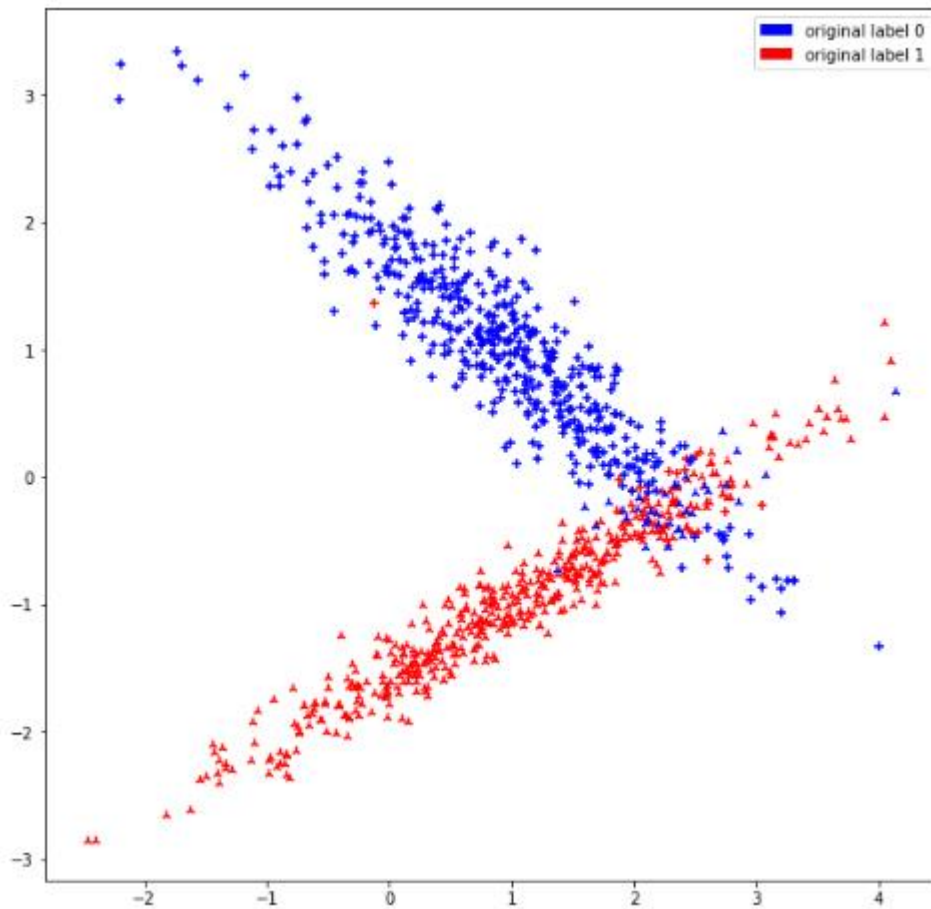
- Dataset: -Spiral1



- Dataset: -Twogaussian33

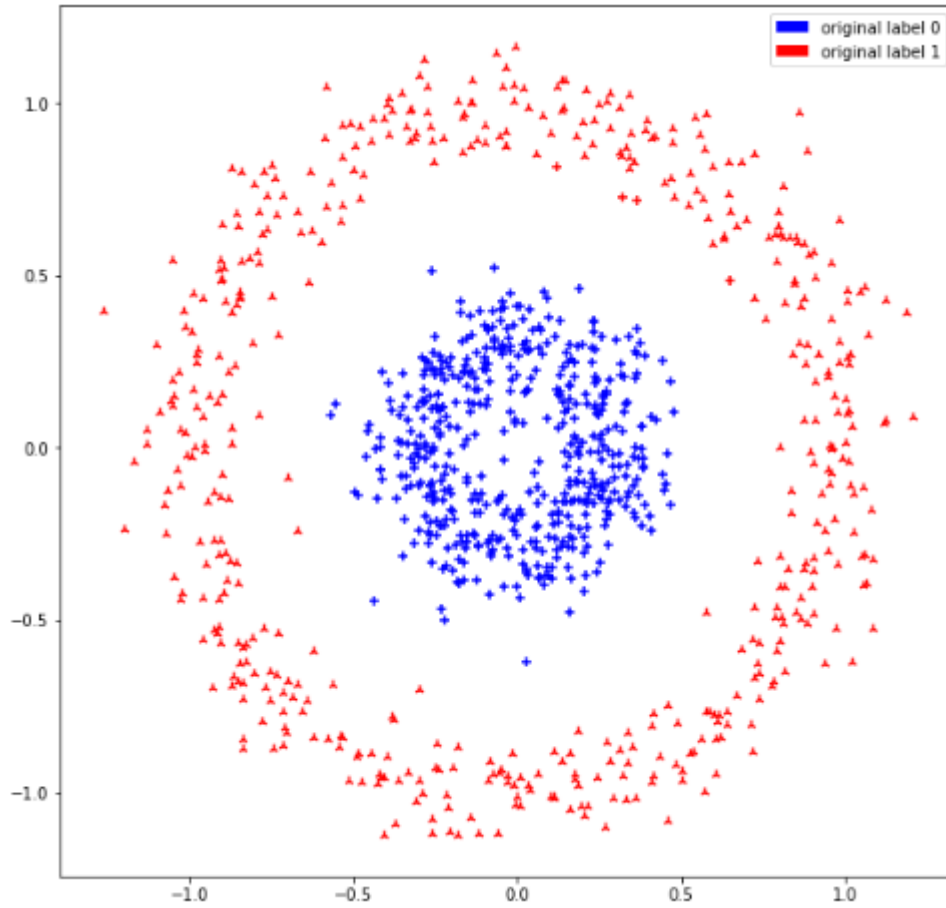


● Dataset: -Twogaussian42

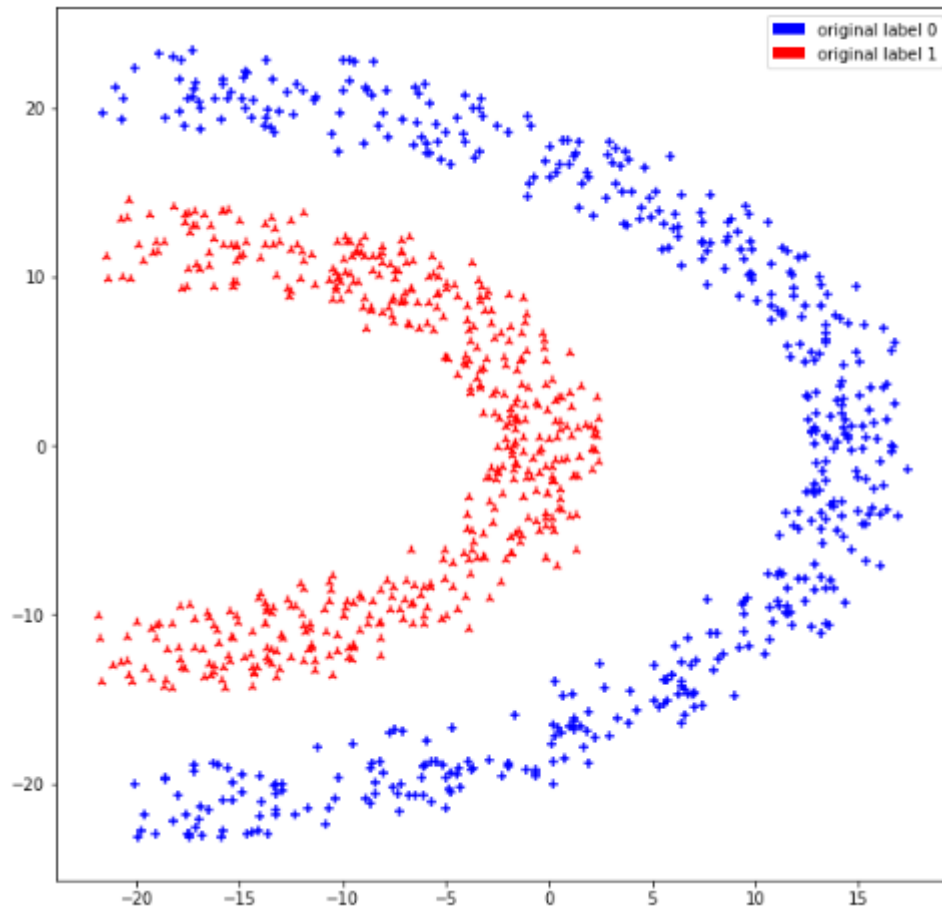


Spectral Clustering

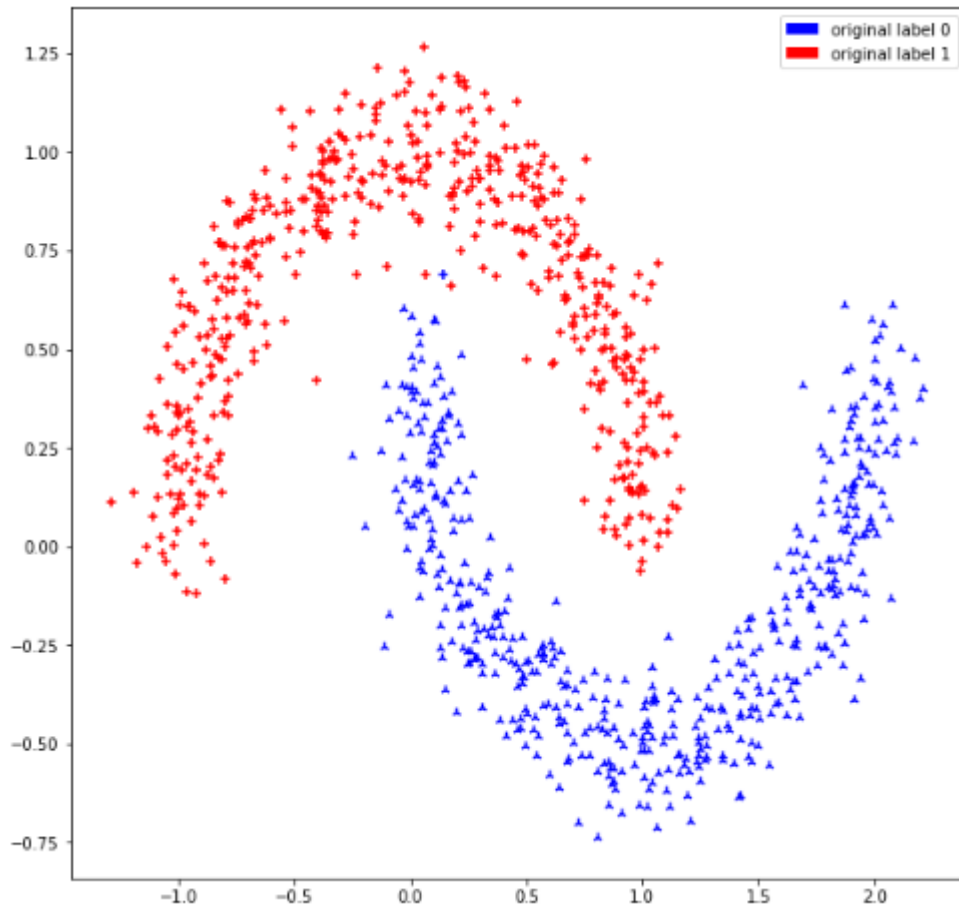
● Dataset: - Circles0.3



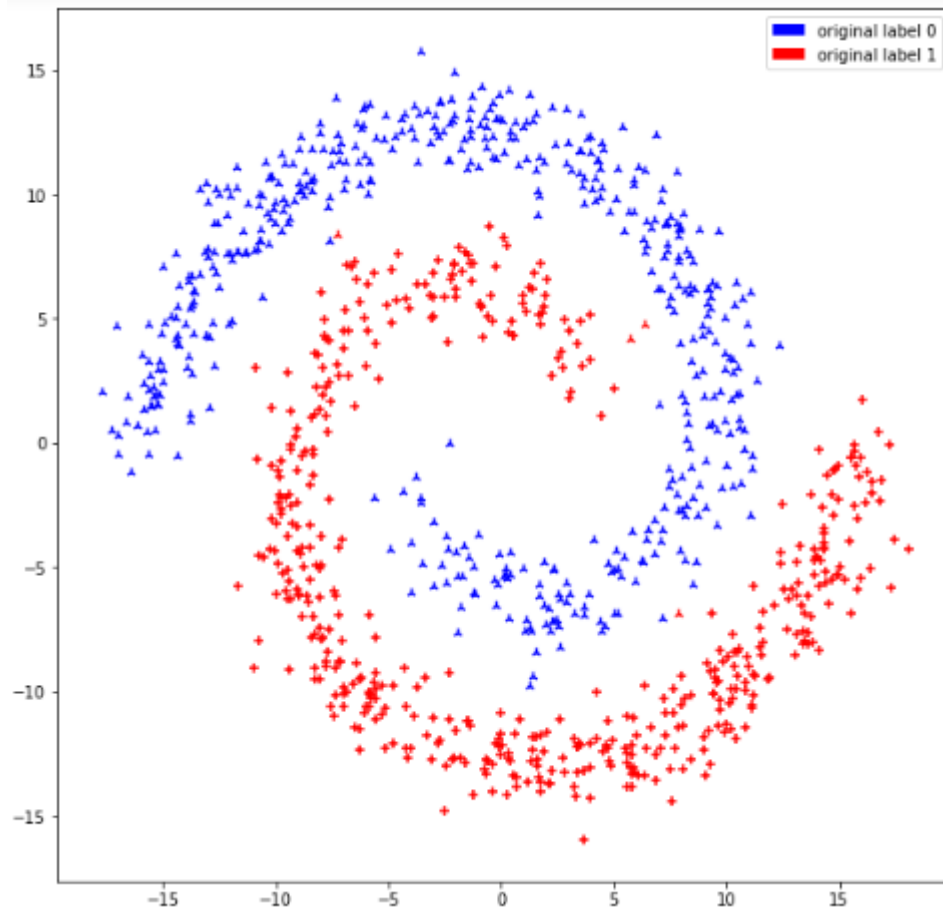
- Dataset: -Half kernel



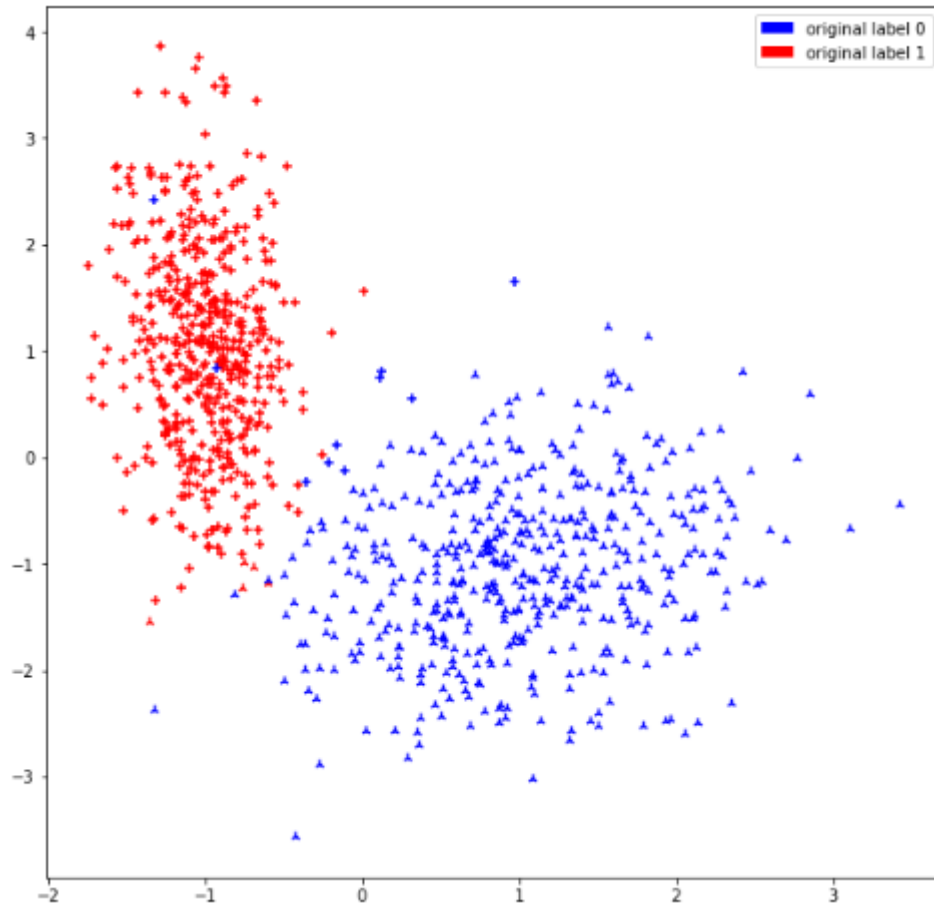
- Dataset: -Moons1



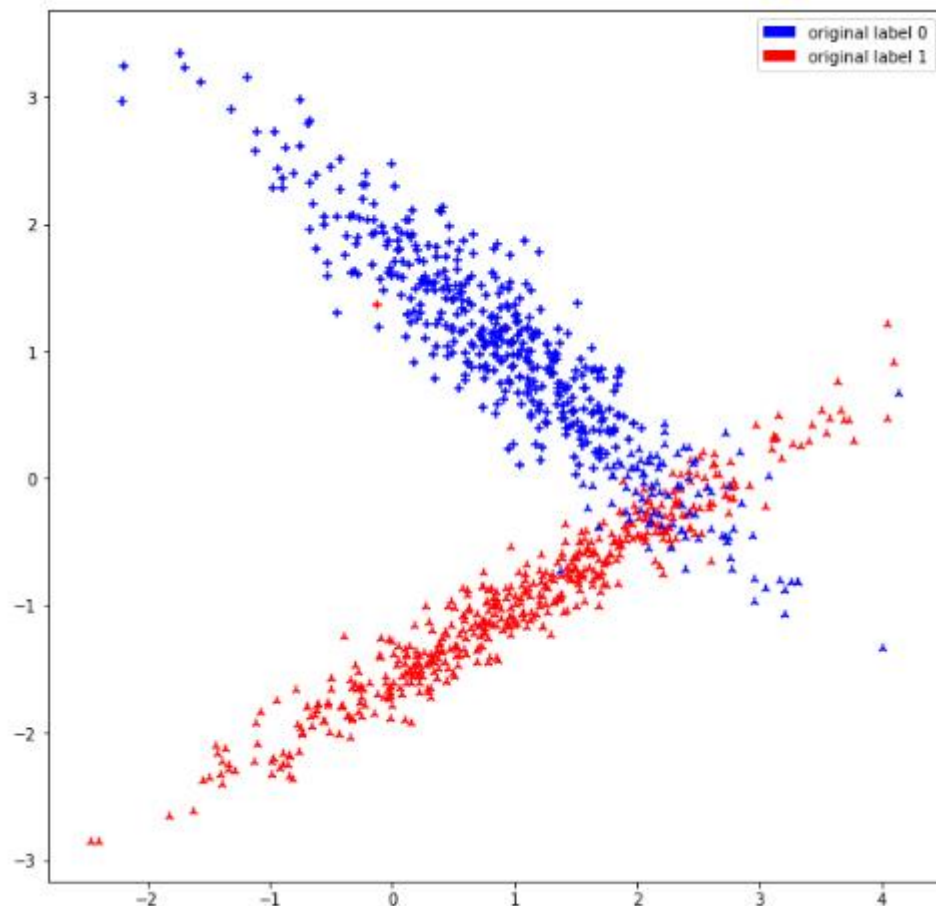
- Dataset: -Spiral1



- Dataset: -TwoGaussian33



- Dataset: -TwoGaussian42



Q4) For all datasets and clustering algorithms, briefly discuss the results you obtained. How good is a particular clustering algorithm on a particular dataset? Why?

A4)

There are 6 datasets

circles0.3, moons1, spiral1, twogaussian33, twogaussian42, half kernel and thus we apply clustering algorithm on the following datasets and hence draw information from the given data.

- **Circles 0.3**

- 1. K means** when applied to **circles 0.3** new clusters are created iteratively along with new centroids which are selected in the beginning. K means will

perform allotment of data points to the given convergence also k means functions well on cloud based data, and doesn't perform well on less round which doesn't appear from the normal data distribution and thus k means is not a good clustering algorithm for circles 0.3.

2. EM when applied to **circles 0.3** it will consider model's parameter and probability distribution is created which is involved in the E step and then when new data is added probability distribution is adjusted as per it which is called as the M step , **thus only difference between K means and EM is that EM takes into importance that when new data points are added it will take into consideration normal distribution.** Thus due to normal distribution of new points are added it will function well and hence give a better result as compared to k means in circles0.3.
3. Spectral Clustering functions upon graph based data which uses dimensions in the space which takes into account every data points as node in the graph. Different types of matrices are created like degree, adjacency matrix and parameters are calculated like eigen values. Gamma value plays a very important role which depicts the shape and spread and hence rbf takes the points into higher dimensions and constructs plot and hence projects them back into lower dimensions. **Thus Spectral clustering gives the best result for circles dataset.**

● Half Kernel

1. K means when applied to **half kernel** new clusters are created iteratively along with new centroids which are selected in the beginning. K means will perform allotment of data points to the given convergence also k means functions well on cloud based data, and doesn't perform well on less round which doesn't appear from the normal data distribution and thus k means is not a good clustering algorithm for the given dataset.
2. EM when applied to **Half kernel** it will consider model's parameter and probability distribution is created which is involved in the E step and then when new data is added probability distribution is adjusted as per it which is called as the M step , **thus only difference between K means and EM is that EM takes into importance that when new data points are added it will take into consideration normal distribution.** Thus due to normal

distribution of new points are added it will function well and hence give a better result as compared to k means in the given dataset.

3. Spectral Clustering functions upon graph based data which uses dimensions in the space which takes into account every data points as node in the graph. Different types of matrices are created like degree, adjacency matrix and parameters are calculated like eigen values. Gamma value plays a very important role which depicts the shape and spread and hence rbf takes the points into higher dimensions and constructs plot and hence projects them back into lower dimensions. **Thus Spectral clustering gives the best result for half kernel.**

● **Moons1: -**

1. K means when applied to **Moons1** new clusters are created iteratively along with new centroids which are selected in the beginning. K means will perform allotment of data points to the given convergence also k means functions well on cloud based data, and doesn't perform well on less round which doesn't appear from the normal data distribution and thus k means is not a good clustering algorithm for the given dataset.
2. EM when applied to **Moons1** it will consider model's parameter and probability distribution is created which is involved in the E step and then when new data is added probability distribution is adjusted as per it which is called as the M step , **thus only difference between K means and EM is that EM takes into importance that when new data points are added it will take into consideration normal distribution.** Thus due to normal distribution of new points are added it will function well and hence give a better result as compared to k means in the given dataset.
3. Spectral Clustering functions upon graph based data which uses dimensions in the space which takes into account every data points as node in the graph. Different types of matrices are created like degree, adjacency matrix and parameters are calculated like eigen values. Gamma value plays a very important role which depicts the shape and spread and hence rbf takes the points into higher dimensions and constructs plot and hence projects them back into lower dimensions. **Thus Spectral clustering gives the best result for moons1 dataset.**

● **Spiral 1: -**

1. K means when applied to **Spiral1** new clusters are created iteratively along with new centroids which are selected in the beginning. K means will perform allotment of data points to the given convergence also k means functions well on cloud based data, and doesn't perform well on less round which doesn't appear from the normal data distribution and thus k means is not a good clustering algorithm for the given dataset.
2. EM when applied to **Spiral1** it will consider model's parameter and probability distribution is created which is involved in the E step and then when new data is added probability distribution is adjusted as per it which is called as the M step , **thus only difference between K means and EM is that EM takes into importance that when new data points are added it will take into consideration normal distribution.** Thus due to normal distribution of new points are added it will function well and hence give a better result as compared to k means in the given dataset.
3. Spectral Clustering functions upon graph based data which uses dimensions in the space which takes into account every data points as node in the graph. Different types of matrices are created like degree, adjacency matrix and parameters are calculated like eigen values. Gamma value plays a very important role which depicts the shape and spread and hence rbf takes the points into higher dimensions and constructs plot and hence projects them back into lower dimensions. **Thus Spectral clustering gives the best result for Spiral1 dataset.**

● **Twogaussian33: -**

1. K means when applied to **Twogaussian33** new clusters are created iteratively along with new centroids which are selected in the beginning. K means will perform allotment of data points to the given convergence also k means functions well on cloud based data, and doesn't perform well on less round which doesn't appear from the normal data distribution and thus k means is not a good clustering algorithm for the given dataset.

2. EM when applied to **Twogaussian33** it will consider model's parameter and probability distribution is created which is involved in the E step and then when new data is added probability distribution is adjusted as per it which is called as the M step , **thus only difference between K means and EM is that EM takes into importance that when new data points are added it will take into consideration normal distribution.** Thus due to normal distribution of new points are added it will function well and hence give a better result as compared to k means in the given dataset.
3. Spectral Clustering functions upon graph based data which uses dimensions in the space which takes into account every data points as node in the graph. Different types of matrices are created like degree, adjacency matrix and parameters are calculated like eigen values. Gamma value plays a very important role which depicts the shape and spread and hence rbf takes the points into higher dimensions and constructs plot and hence projects them back into lower dimensions. **Thus Spectral clustering gives the similar result as EM for twogaussian 33.**

Performance of EM= Performance of Spectral Clustering

● **Twogaussian42**

1. K means when applied to **Twogaussian42** new clusters are created iteratively along with new centroids which are selected in the beginning. K means will perform allotment of data points to the given convergence also k means functions well on cloud based data, and doesn't perform well on less round which doesn't appear from the normal data distribution and thus k means is not a good clustering algorithm for the given dataset.
2. EM when applied to **Twogaussian42** it will consider model's parameter and probability distribution is created which is involved in the E step and then when new data is added probability distribution is adjusted as per it which is called as the M step , **thus only difference between K means and EM is that EM takes into importance that when new data points are added it will take into consideration normal distribution.** Thus due to normal distribution of new points are added it will function well and hence give a better result as compared to k means in the given dataset.

Since the data is quite linear the structural formation of dataset is such that EM gives the best output for clustering in twogaussian42

3. Spectral Clustering functions upon graph based data which uses dimensions in the space which takes into account every data points as node in the graph. Different types of matrices are created like degree, adjacency matrix and parameters are calculated like eigen values. Gamma value plays a very important role which depicts the shape and spread and hence rbf takes the points into higher dimensions and constructs plot and hence projects them back into lower dimensions.

Clustering Performance Summary

Parameter	Circles0.3	Halfkernel	Moons1	Spiral 1	Twogaussian33	Twogaussian 42
K means	Bad	Bad	Bad	Bad	Bad	Bad
EM	Average	Average	Average	Average	Ideal	Best
Spectral Clustering	Best	Best	Best	Best	Similar to EM	Average

Q5) Pick an index of clustering validity discussed in class, and find the best value of k for each dataset, for both k-means and EM. Explain: (a) how you choose the best number of clusters (textually and graphically), and (b) how the index of validity works.

A5) I have picked Calinski and Harabasz validation index (CH value) for the given clusters as they consist of parameters like ratio between inter cluster dispersion and within cluster dispersion. The values of k used for the given datasets is between 2 to 10 for as the specified range.

● Circles0.3

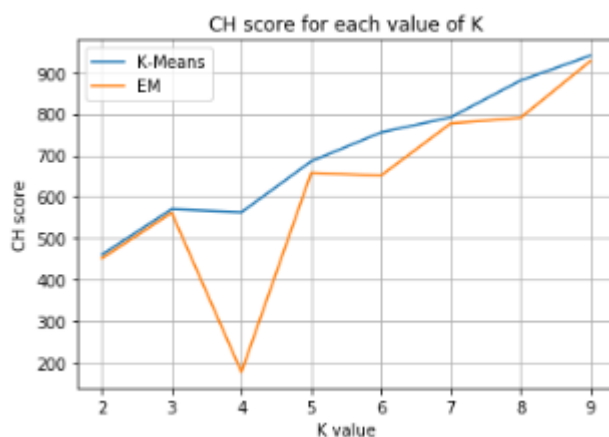
```
# metrics.calinski_harabaz_score(df1.iloc[:,0:2].values,df1_predarray)
print("CH Score for K means")
km_scores,em_scores = [],[]
for k in range(2,10):
    km_model = KMeans(n_clusters=k)
    km_model.fit(df1_nolabel1)
    km_df1_predarray = km_model.fit_predict(df1_nolabel1)
    km_df1_pred = pd.DataFrame(km_df1_predarray,columns =['predicted'])#converting to dataframe
    # print(k,metrics.calinski_harabaz_score(df1.iloc[:,0:2].values,df1_predarray))
    km_scores.append(metrics.calinski_harabaz_score(df1.iloc[:,0:2].values,km_df1_predarray))
    em_model = GaussianMixture(n_components=k)
    em_model.fit(df1_nolabel1)
    em_df1_predarray = em_model.fit_predict(df1_nolabel1)
    em_df1_pred = pd.DataFrame(em_df1_predarray,columns =['predicted'])#converting to dataframe
    # print(k,metrics.calinski_harabaz_score(df1.iloc[:,0:2].values,em_df1_predarray))
    em_scores.append(metrics.calinski_harabaz_score(df1.iloc[:,0:2].values,em_df1_predarray))
print("K-Means: ",km_scores)
print("EM: ", em_scores)

plt.xlabel('K value')
plt.ylabel('CH score')
plt.title('CH score for each value of K')
plt.grid(True)
plt.plot(list(range(2,10)),km_scores,label="K-Means")
plt.plot(list(range(2,10)),em_scores,label="EM")
plt.legend()
```

CH Score for K means

K-Means: [460.2872809004318, 570.1508618125067, 562.541462605339, 685.5731073276743, 755.3023616582724, 791.7465205502288, 880.5186812137996, 940.5853601561204]

EM: [451.27711660283416, 561.2450799300696, 176.98957000599583, 657.7838266956265, 650.9723545846996, 777.0816880815079, 790.4201045913392, 927.2754510546118]



We select K=10, as it has maximum ch score textually and graphically for K means

We select K=10, as it has maximum ch score and graphically select K=7 using elbow method for EM

● Half kernel

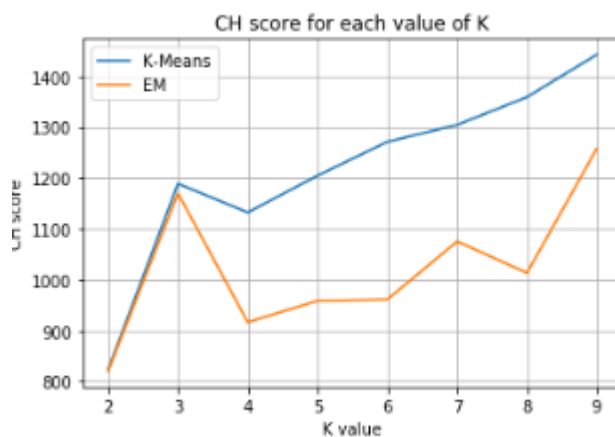
```
: # metrics.calinski_harabaz_score(df1.iloc[:,0:2].values,df1_predarray)
print("CH Score for K means")
km_scores,em_scores = [],[]
for k in range(2,10):
    km_model = KMeans(n_clusters=k)
    km_model.fit(df2_nolabel2)
    km_df2_predarray = km_model.fit_predict(df2_nolabel2)
    km_df2_pred = pd.DataFrame(km_df2_predarray,columns=['predicted'])#converting to dataframe
    # print(k,metrics.calinski_harabaz_score(df1.iloc[:,0:2].values,df1_predarray))
    km_scores.append(metrics.calinski_harabaz_score(df2.iloc[:,0:2].values,km_df2_predarray))
    em_model = GaussianMixture(n_components=k)
    em_model.fit(df2_nolabel2)
    em_df2_predarray = em_model.fit_predict(df2_nolabel2)
    em_df2_pred = pd.DataFrame(em_df2_predarray,columns=['predicted'])#converting to dataframe
    # print(k,metrics.calinski_harabaz_score(df1.iloc[:,0:2].values,em_df1_predarray))
    em_scores.append(metrics.calinski_harabaz_score(df2.iloc[:,0:2].values,em_df2_predarray))
print("K-Means: ", km_scores)
print("EM: ", em_scores)

plt.xlabel('K value')
plt.ylabel('CH score')
plt.title('CH score for each value of K')
plt.grid(True)
plt.plot(list(range(2,10)),km_scores,label="K-Means")
plt.plot(list(range(2,10)),em_scores,label="EM")
plt.legend()
```

CH Score for K means

K-Means: [823.5030813094922, 1188.776572834821, 1132.2479775414454, 1204.6445772506488, 1271.172402343824, 1304.4435011131552, 1359.0493747410158, 1442.4303176722951]

EM: [820.6276169080147, 1169.8787841165247, 916.415387592172, 958.551169156231, 961.2964568646197, 1075.33181163376, 1013.3019208211987, 1257.5966764310092]



We select k=10 for which ch score is maximum for K means

We select k=10 for which ch score is maximum and K=9 graphically using elbow method for EM.

Moons1

```
: # metrics.calinski_harabaz_score(df1.iloc[:,0:2].values,df1_predarray)
print("CH score for K means")
km_scores,em_scores = [],[]
for k in range(2,10):
    km_model = KMeans(n_clusters=k)
    km_model.fit(df3_nolabel3)
    km_df3_predarray = km_model.fit_predict(df3_nolabel3)
    km_df3_pred = pd.DataFrame(km_df3_predarray,columns=['predicted'])#converting to dataframe
    # print(k,metrics.calinski_harabaz_score(df1.iloc[:,0:2].values,df1_predarray))
    km_scores.append(metrics.calinski_harabaz_score(df3.iloc[:,0:2].values,km_df3_predarray))
    em_model = GaussianMixture(n_components=k)
    em_model.fit(df3_nolabel3)
    em_df3_predarray = em_model.fit_predict(df3_nolabel3)
    em_df3_pred = pd.DataFrame(em_df3_predarray,columns=['predicted'])#converting to dataframe
    # print(k,metrics.calinski_harabaz_score(df1.iloc[:,0:2].values,em_df1_predarray))
    em_scores.append(metrics.calinski_harabaz_score(df3.iloc[:,0:2].values,em_df3_predarray))
print("K-Means: ", km_scores)
print("EM: ", em_scores)

plt.xlabel('K value')
plt.ylabel('CH score')
plt.title('CH score for each value of K')
plt.grid(True)
plt.plot(list(range(2,10)),km_scores,label="K-Means")
plt.plot(list(range(2,10)),em_scores,label="EM")
plt.legend()
```

CH Score for K means

K-Means: [1432.961728513351, 1299.6285955537235, 1492.4062838039372, 1541.6765477611286, 1844.4943693592045, 1915.726648702847, 2115.563870793936, 2209.905366945294]

EM: [1234.4494044143364, 1138.0712626231962, 1276.2582275757538, 1329.3400448970579, 1770.1183982449675, 1759.9977255940512, 1777.6937145464558, 2042.9787833969372]



We select K=10 for which the ch score is maximum for K means

We select k=10 textually for which CH score is maximum and K=8 graphically for EM

● Spiral1

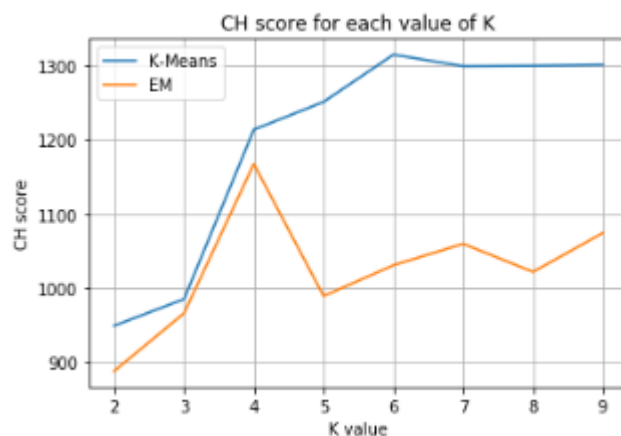
```
: # metrics.calinski_harabaz_score(df1.iloc[:,0:2].values,df1_predarray)
print("CH Score for K means")
km_scores,em_scores = [],[]
for k in range(2,10):
    km_model = KMeans(n_clusters=k)
    km_model.fit(df4_nolabel4)
    km_df4_predarray = km_model.fit_predict(df4_nolabel4)
    km_df4_pred = pd.DataFrame(km_df4_predarray,columns=['predicted'])#converting to dataframe
    # print(k,metrics.calinski_harabaz_score(df1.iloc[:,0:2].values,df1_predarray))
    km_scores.append(metrics.calinski_harabaz_score(df4.iloc[:,0:2].values,km_df4_predarray))
    em_model = GaussianMixture(n_components=k)
    em_model.fit(df4_nolabel4)
    em_df4_predarray = em_model.fit_predict(df4_nolabel4)
    em_df4_pred = pd.DataFrame(em_df4_predarray,columns=['predicted'])#converting to dataframe
    # print(k,metrics.calinski_harabaz_score(df1.iloc[:,0:2].values,em_df1_predarray))
    em_scores.append(metrics.calinski_harabaz_score(df4.iloc[:,0:2].values,em_df4_predarray))
print("K-Means: ", km_scores)
print("EM: ", em_scores)

plt.xlabel('K value')
plt.ylabel('CH score')
plt.title('CH score for each value of K')
plt.grid(True)
plt.plot(list(range(2,10)),km_scores,label="K-Means")
plt.plot(list(range(2,10)),em_scores,label="EM")
plt.legend()
```

CH Score for K means

K-Means: [949.1564080865779, 985.0121094218789, 1213.377735170917, 1250.7931076907043, 1314.4710960343734, 1298.8722209833063, 1299.4296196739087, 1300.7563713113996]

EM: [888.1435032677558, 965.9836205064893, 1167.2393651686684, 989.0890103581493, 1030.6649625405446, 1059.58400461722, 1021.659718396471, 1074.1806494060638]



As per the graph we select k=10 textually will select k=6 graphically using the elbow method as it gives the maximum value for K means graphically

We select K=4 as ch score is maximum textually and graphically K=4 for EM

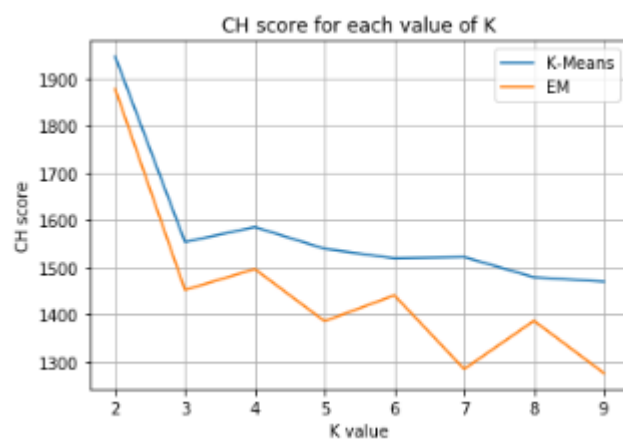
● Twogaussian33

```
# metrics.calinski_harabaz_score(df1.iloc[:,0:2].values,df1_predarray)
print("CH score for K means")
km_scores,em_scores = [],[]
for k in range(2,10):
    km_model = KMeans(n_clusters=k)
    km_model.fit(df5_nolabels)
    km_df5_predarray = km_model.fit_predict(df5_nolabels)
    km_df5_pred = pd.DataFrame(km_df5_predarray,columns=['predicted'])#converting to dataframe
    # print(k,metrics.calinski_harabaz_score(df1.iloc[:,0:2].values,df1_predarray))
    km_scores.append(metrics.calinski_harabaz_score(df5.iloc[:,0:2].values,km_df5_predarray))
    em_model = GaussianMixture(n_components=k)
    em_model.fit(df5_nolabels)
    em_df5_predarray = em_model.fit_predict(df5_nolabels)
    em_df5_pred = pd.DataFrame(em_df5_predarray,columns=['predicted'])#converting to dataframe
    # print(k,metrics.calinski_harabaz_score(df1.iloc[:,0:2].values,em_df1_predarray))
    em_scores.append(metrics.calinski_harabaz_score(df5.iloc[:,0:2].values,em_df5_predarray))
print("K-Means: ", km_scores)
print("EM: ", em_scores)

plt.xlabel('K value')
plt.ylabel('CH score')
plt.title('CH score for each value of K')
plt.grid(True)
plt.plot(list(range(2,10)),km_scores,label="K-Means")
plt.plot(list(range(2,10)),em_scores,label="EM")
plt.legend()
```

CH Score for K means

K-Means: [1946.2649408634147, 1553.357504234373, 1585.1007988289398, 1539.1194730909926, 1518.6376339615722, 1521.548734206348
6, 1478.685946525435, 1469.413714274159]
EM: [1877.6783523820309, 1451.7287498094558, 1496.1028664413602, 1385.5711377498742, 1440.4140168781505, 1284.1119551053384, 1
386.4613886558418, 1275.7833776546743]



We will take K=2 for both graph as well as text because no elbow is created and hence gives the maximum value.

We select K = 2 textually and K=2 graphically using the elbow method for EM.

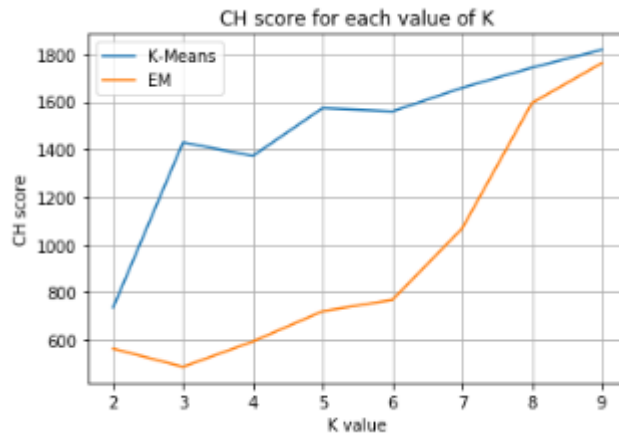
\

● Twogaussian42

```
: # metrics.calinski_harabaz_score(df1.iloc[:,0:2].values,df1_predarray)
print("CH Score for K means")
km_scores,em_scores = [],[]
for k in range(2,10):
    km_model = KMeans(n_clusters=k)
    km_model.fit(df6_nolabel6)
    km_df6_predarray = km_model.fit_predict(df6_nolabel6)
    km_df6_pred = pd.DataFrame(km_df6_predarray,columns=['predicted'])#converting to dataframe
    # print(k,metrics.calinski_harabaz_score(df1.iloc[:,0:2].values,df1_predarray))
    km_scores.append(metrics.calinski_harabaz_score(df6.iloc[:,0:2].values,km_df6_predarray))
    em_model = GaussianMixture(n_components=k)
    em_model.fit(df6_nolabel6)
    em_df6_predarray = em_model.fit_predict(df6_nolabel6)
    em_df6_pred = pd.DataFrame(em_df6_predarray,columns=['predicted'])#converting to dataframe
    # print(k,metrics.calinski_harabaz_score(df1.iloc[:,0:2].values,em_df1_predarray))
    em_scores.append(metrics.calinski_harabaz_score(df6.iloc[:,0:2].values,em_df6_predarray))
print("K-Means: ", km_scores)
print("EM: ", em_scores)

plt.xlabel('K value')
plt.ylabel('CH score')
plt.title('CH score for each value of K')
plt.grid(True)
plt.plot(list(range(2,10)),km_scores,label="K-Means")
plt.plot(list(range(2,10)),em_scores,label="EM")
plt.legend()
```

CH Score for K means
K-Means: [736.8651433450522, 1429.0889412496765, 1372.0996760853177, 1573.103925162103, 1559.055720976297, 1658.2528373774903, 1742.648742528749, 1817.652054103163]
EM: [563.4093768736412, 487.1869904079199, 593.4117080272515, 720.652442563565, 768.2408879596338, 1067.964970929543, 1594.357324957335, 1761.8220218484992]



Graphically we take $k=5$ because of elbow method and $k=10$ for textual method as it is maximum.

We select $k = 9$ for both textually and graphically method in EM.

SUMMARY

Kmeans

Parameter	Circles0.3	Half Kernel	Moons1	Spiral 1	Twogaussian 33	Twogaussian 42
Textual	K=10	K=10	K=10	K=10	K=2	K=10
Graphical	K=10	K=3	K=10	K=6	K=2	K=5

EM

Parameter	Circles0.3	Half Kernel	Moons1	Spiral 1	Twogaussian 33	Twogaussian 42
Textual	K=10	K=10	K=10	K=4	K=2	K=9
Graphical	K=7	K=9	K=8	K=4	K=2	K=9

(b) The Calinski and Harabasz score is defined as ratio between and the between-cluster dispersion and within cluster dispersion. The trace of between class scatter matrix S_a is given by:

$$S_a = \sum_{i=1}^k |P_i| \|\mu - \mu_i\|^2$$

The trace of within class scatter matrix S_b is given by,

$$S_b = \sum_{i=1}^k \sum_{j=1}^{|P_i|} \|x_j - \mu_i\|^2$$

CH index maximizes,

$$CH = (S_a / (k-1)) / (S_b / n-k)$$

Here μ is the mean of the means and μ_i is the mean of cluster P_i .

The above mathematical formula gives the relation between the distances from the mean and hence determines the functioning of the given CH index score which is used for the validation.

Q6) Explain mathematically (sketch of proof) how k-means is a particular case of EM.

A6)

Functioning of the EM clustering is in two steps 1) E Step 2) M Step : -

1) In the E-step an initial guess is made for the model's parameters and a probability distribution is created.

2) In the M-step probability distribution is adjusted to include the new data. Shown as follows.

$$P(\omega_i | x_k, \Theta) = \left| \sum_i \right|^{-1/2} e^{-1/2(x_k - \mu_i)^T \Sigma_i^{-1} (x_k - \mu_i)} * P(\omega_i) / \sum_{j=1}^c \left| \sum_j \right|^{-1/2} e^{-1/2(x_k - \mu_j)^T \Sigma_j^{-1} (x_k - \mu_j)} * P(\omega_j)$$

When squared value of mahalanobis distance is decreased the term $P(\omega_i | x_k, \Theta)$ becomes greater. Also the points with the same mahalanobis distances are

equally likely when the data has a normal distribution. Thus smaller is the distance the higher is the probability of a point belonging to a particular cluster. Lets consider we make an assumption of $\sum_i = 1$ then we just calculate the Euclidean distance given by the formula.

$$(x_k - \mu_i)^t (x_k - \mu_i) = ||x_k - \mu_i||^2$$

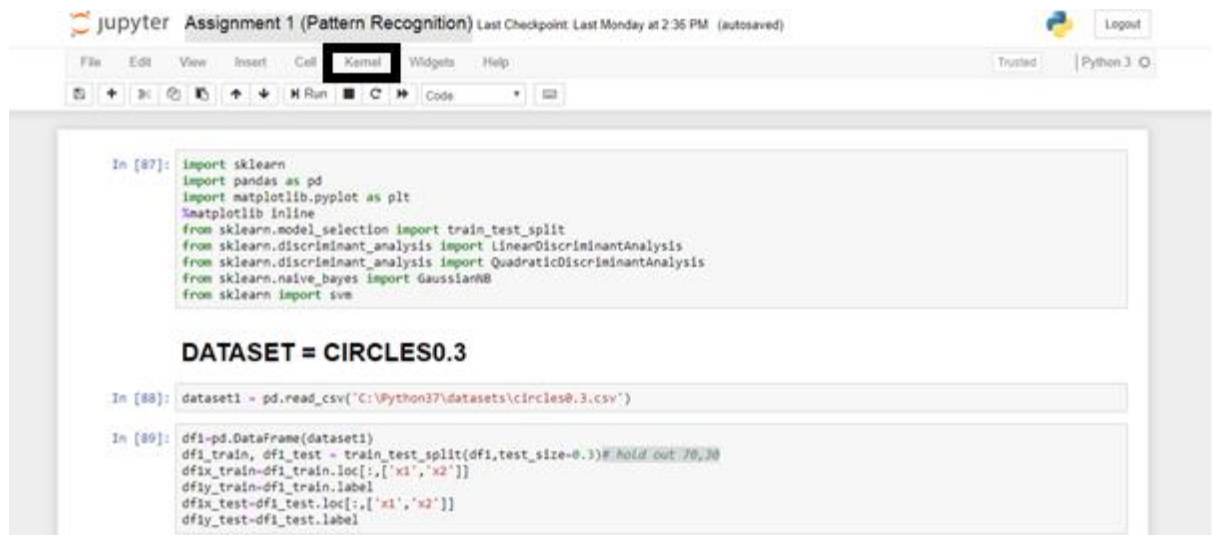
Finding the euclidean distance between a point x_k and the mean. So, we say that,

$$P(\omega_i | x_k, \Theta) = 1 \text{ when } i=j \\ = 0 \text{ else}$$

The above formulation leads to similarity with K means.

Steps to run the code: -

- Open python jupyter book in terminal of your folder in which there is python program
- Create a new file of assignment 2 and follow the code
- Each cell has different functions either run the cells by using shift + enter or run entire code by selecting the kernel option



The image shows a Jupyter Notebook interface titled "Assignment 1 (Pattern Recognition)". The "Kernel" menu is highlighted with a black box. The code in the notebook is as follows:

```
In [87]: import sklearn
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn import svm

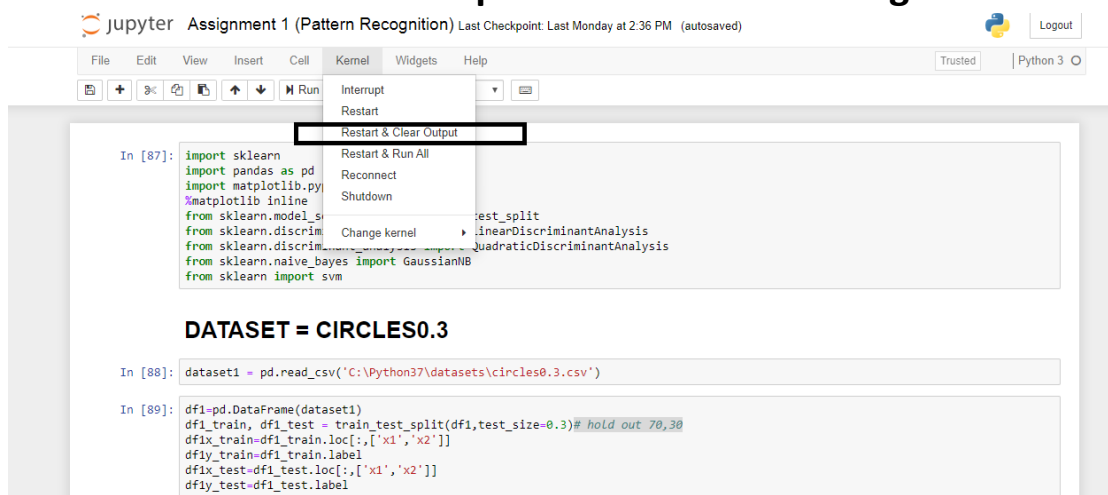
DATASET = CIRCLES0.3

In [88]: dataset1 = pd.read_csv('C:\Python37\datasets\circles0.3.csv')

In [89]: df1=pd.DataFrame(dataset1)
df1_train, df1_test = train_test_split(df1,test_size=0.3)# hold out 70,30
df1x_train=df1_train.loc[:,['x1','x2']]
df1y_train=df1_train.label
df1x_test=df1_test.loc[:,['x1','x2']]
df1y_test=df1_test.label
```

The black box shows the following option

- Select restart and run all option from the following



The image shows the same Jupyter Notebook interface, but with the "Kernel" menu open. The "Restart & Clear Output" option is highlighted with a black box. The code in the notebook is the same as in the previous image.

The black box shows the following option

- Hence each cell runs showing the clusters along with the following results and graph as needed

Name: - Sami Ali Choudhry
Assignment 2

Student No: -105187035 Course: - Pattern Recognition (8740)