Machine Learning Assignment (Clustering)

Name: Khuram Shahzad (p21-8742)

Load Data

In [1]:

```
import pandas as pd
import numpy as np
# // Read Data and Evaluation files
Artificial_Eva  = pd.read_csv('Evaluations1.csv')
Spiral_Eva =pd.read_csv('Evaluations2.csv')
Path_Eva=  pd.read_csv('Evaluations3.csv')
Path_Eva.head()


Artificial_data = pd.read_csv('Artificial.csv')
Artificial_data = np.array(Artificial_data.iloc[:, [0, 1]]);

Spiral_data = pd.read_csv('Spiral.csv')
Spiral_data = np.array(Spiral_data.iloc[:, [0, 1]]);

Path_data = pd.read_csv('Path.csv')
Path_data = np.array(Path_data.iloc[:, [0, 1]]);
data= Artificial_data;


def Load_Data(Is_slicing):
    Artificial_data = pd.read_csv('Artificial.csv')
    Spiral_data = pd.read_csv('Spiral.csv')
    Path_data = pd.read_csv('Path.csv')
    if Is_slicing==1:
        Artificial_data = np.array(Artificial_data.iloc[:, [0, 1]]);
        Spiral_data = np.array(Spiral_data.iloc[:, [0, 1]]);
        Path_data = np.array(Path_data.iloc[:, [0, 1]]);
    return (Artificial_data,Spiral_data, Path_data )

Load_Data(1)

Artificial_Eva.head()
```

Out[1]:

	Clustring Algorithm	Root MSE	DBI	Silhouette Score	Purity	Square Error
0	K-Means	0.0	0.0	0.0	0.0	0.0
1	Mini-Batch K-Means	0.0	0.0	0.0	0.0	0.0
2	DBSCAN	0.0	0.0	0.0	0.0	0.0
3	Spectral Clustering	0.0	0.0	0.0	0.0	0.0
4	K means self implemenation	0.0	0.0	0.0	0.0	0.0

Evaluations Metrics

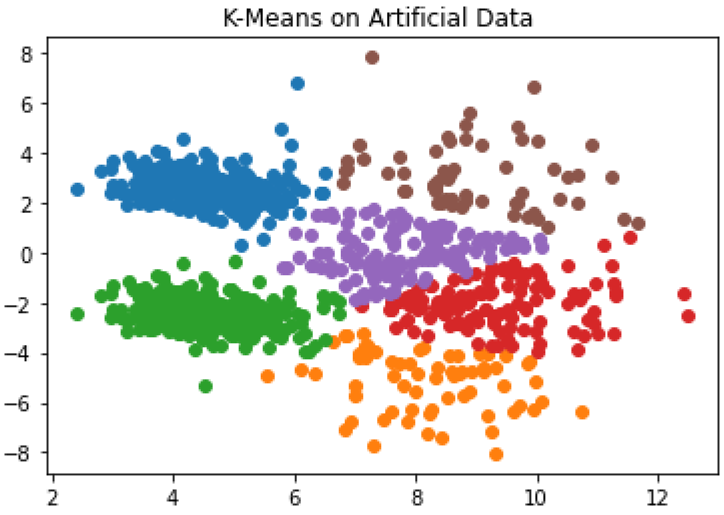
In [2]:

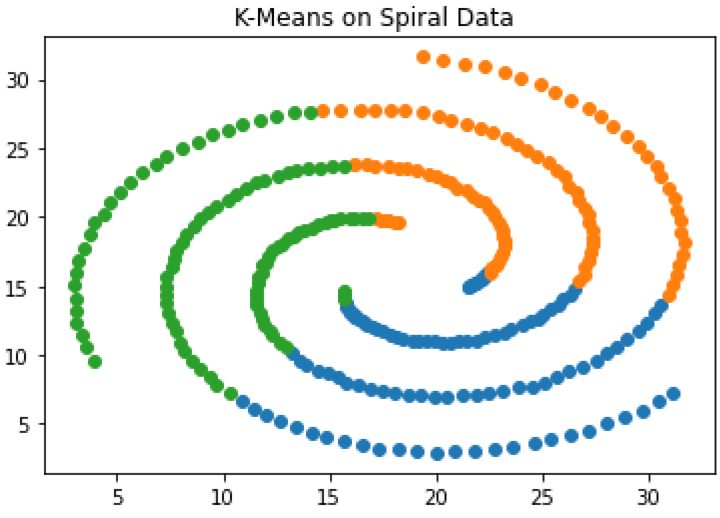
```
import numpy as np
import math
from sklearn import metrics
from sklearn.metrics import davies_bouldin_score
from sklearn.metrics import silhouette_score
from sklearn.metrics import mean_squared_error

def purity_score(y_true,  y_pred):
    # compute contingency matrix (also called confusion matrix)
    contingency_matrix = metrics.cluster.contingency_matrix(y_true, y_pred)
    # return purity
    return np.sum(np.amax(contingency_matrix, axis=0)) / np.sum(contingency_matrix)

def Measure_Evaluation(index, k,  data= [], Eva_df= [], inertia=[], y_pred=[]):
    X =  np.array(data.iloc[:, [0, 1]]);
    db_index = davies_bouldin_score(X, y_per[k])
    sh_score = silhouette_score(X, y_per[k], metric='euclidean')

    X = np.array(data.iloc[:, [2]])
    purity= purity_score(X,y_per[k])
    RMSE=math. sqrt(mean_squared_error(X, y_per[k]))


    Eva_df.at[index,'Root MSE']= RMSE
    Eva_df.at[index,'DBI']=db_index
    Eva_df.at[index,'Silhouette Score']=sh_score
    Eva_df.at[index,'Purity']=purity
    Eva_df.at[index,'Square Error']= inertia[k]
```

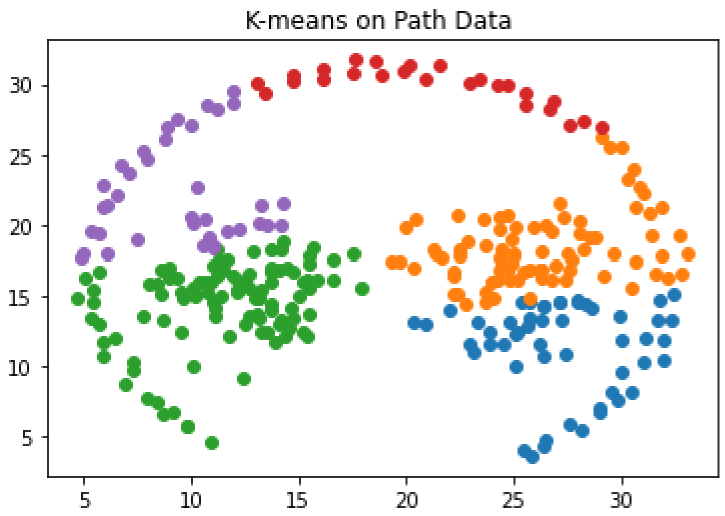
K - Means

In [3]:

```
# k-means clustering
from numpy import unique
from numpy import where
import numpy as np
from sklearn.cluster import KMeans
from matplotlib import pyplot as plt

inertia= []
y_per= []
def Kmeans ( msg, cluster, data= []):
    model = KMeans(n_clusters=cluster, max_iter=1000)
     # fit the model
    kmeans=model.fit(data)
    inertia.append(kmeans.inertia_)

    yhat = model.predict(data)
    y_per.append(yhat)
    # retrieve unique clusters
    clusters = unique(yhat)

    # create scatter plot for samples from each cluster
    for cluster in clusters:
        # get row indexes for samples with this cluster
        row_ix = where(yhat == cluster)
        # create scatter of these samples
        plt.scatter(data[row_ix, 0], data[row_ix, 1])
    # show the plot
    plt.title(msg);
    plt.show()

Artificial_data,Spiral_data,Path_data=  Load_Data(1)
Kmeans( "K-Means on Artificial Data ", 6, Artificial_data)
Kmeans( "K-Means on Spiral Data ", 3,Spiral_data)
Kmeans( "K-means on Path Data ",5, Path_data)

# Evaluations
inertia=[0,0,0]
Artificial_data,Spiral_data,Path_data=  Load_Data(0)
Measure_Evaluation(0, 0,Artificial_data, Artificial_Eva, inertia ,y_per)
Measure_Evaluation(0, 1,Spiral_data, Spiral_Eva, inertia, y_per)
Measure_Evaluation(0, 2,Path_data, Path_Eva, inertia, y_per)

Artificial_Eva.head()
# Spiral_Eva.head()
# Path_Eva.head()
```







Out[3]:

	Clustring Algorithm	Root MSE	DBI	Silhouette Score	Purity	Square Error
0	K-Means	2.719711	0.774641	0.563799	0.957341	0.0
1	Mini-Batch K-Means	0.000000	0.000000	0.000000	0.000000	0.0
2	DBSCAN	0.000000	0.000000	0.000000	0.000000	0.0
3	Spectral Clustering	0.000000	0.000000	0.000000	0.000000	0.0
4	K means self implemenation	0.000000	0.000000	0.000000	0.000000	0.0

In [4]:

```
inertia
```

Out[4]:

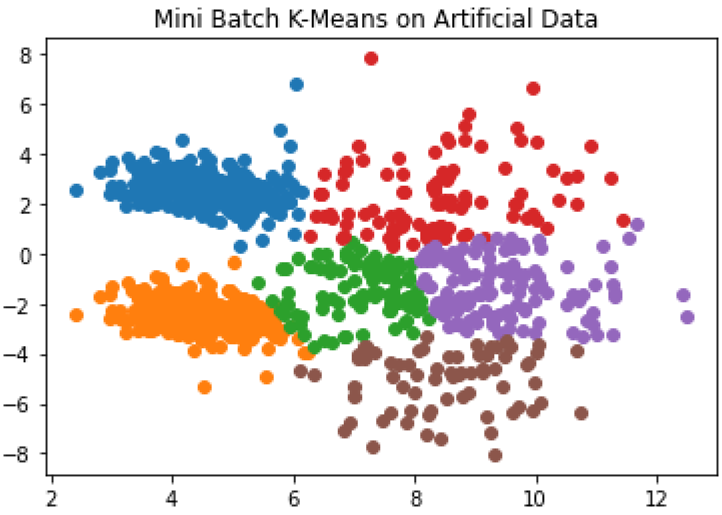
```
[0, 0, 0]
```

Mini-Batch K-Means

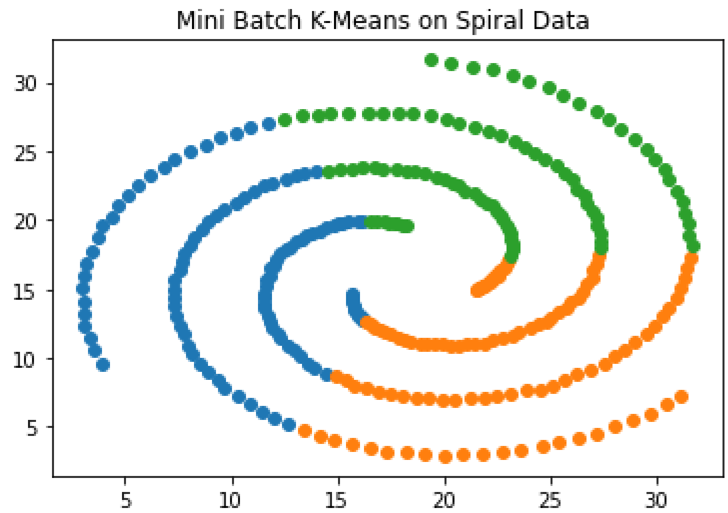
In [5]:

```
# Mini batch k-means clustering
from numpy import unique
from numpy import where
import numpy as np
from sklearn.cluster import MiniBatchKMeans
from matplotlib import pyplot as plt

inertia= []
y_per= []
def Mini_Batch_KMeans ( msg, cluster, data= []):
    model = MiniBatchKMeans(n_clusters=cluster, max_iter=1000)
     # fit the model
    mini_batch_kmeans=model.fit(data)
    inertia.append(mini_batch_kmeans.inertia_)

    yhat = model.predict(data)
    y_per.append(yhat)
    # retrieve unique clusters
    clusters = unique(yhat)

    # create scatter plot for samples from each cluster
    for cluster in clusters:
        # get row indexes for samples with this cluster
        row_ix = where(yhat == cluster)
        # create scatter of these samples
        plt.scatter(data[row_ix, 0], data[row_ix, 1])
    # show the plot
    plt.title(msg);
    plt.show()

Artificial_data,Spiral_data,Path_data=  Load_Data(1)
Mini_Batch_KMeans( "Mini Batch K-Means on Artificial Data ", 6, Artificial_data)
Mini_Batch_KMeans( "Mini Batch K-Means on Spiral Data ", 3,Spiral_data)
Mini_Batch_KMeans( "Mini Batch K-means on Path Data ",5, Path_data)



# Evaluations
inertia=[0,0,0]
Artificial_data,Spiral_data,Path_data=  Load_Data(0)
Measure_Evaluation(1, 0,Artificial_data, Artificial_Eva, inertia ,y_per)
Measure_Evaluation(1, 1,Spiral_data, Spiral_Eva, inertia, y_per)
Measure_Evaluation(1, 2,Path_data, Path_Eva, inertia, y_per)

Artificial_Eva.head()
# Spiral_Eva.head()
# Path_Eva.head()
```

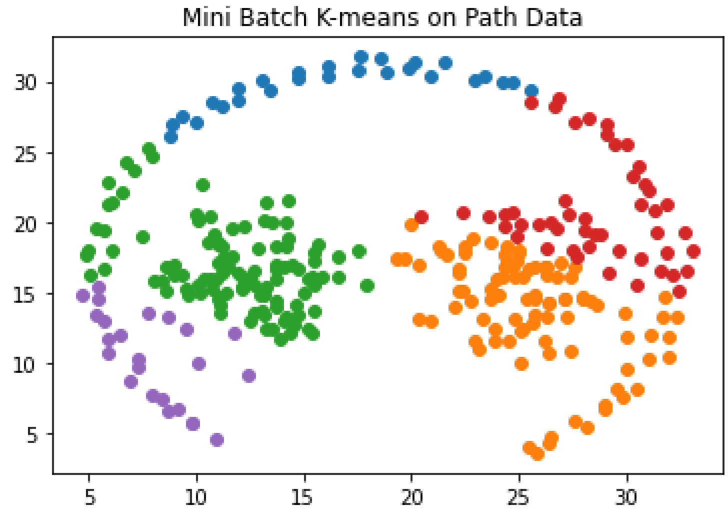
```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:887: UserWarning:
MiniBatchKMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can prevent it by setting batch_size >= 1024 or by se
tting the environment variable OMP_NUM_THREADS=1
  warnings.warn(
```



```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:887: UserWarning:
MiniBatchKMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can prevent it by setting batch_size >= 1024 or by se
tting the environment variable OMP_NUM_THREADS=1
  warnings.warn(
```



```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:887: UserWarning:
MiniBatchKMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can prevent it by setting batch_size >= 1024 or by se
tting the environment variable OMP_NUM_THREADS=1
  warnings.warn(
```



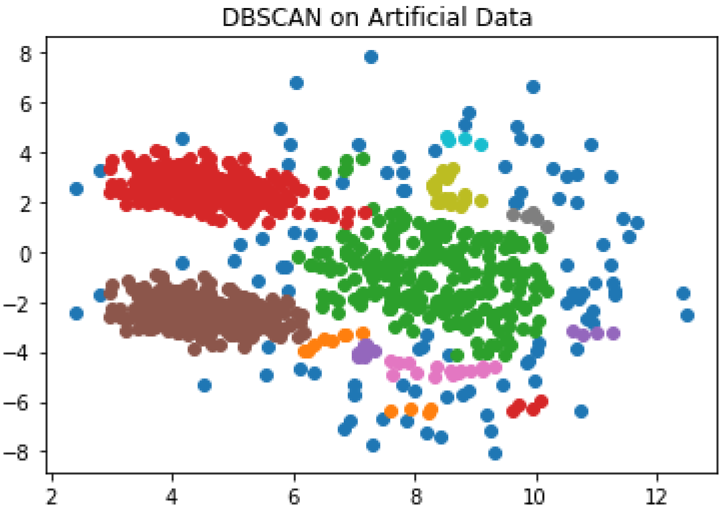
Out[5]:

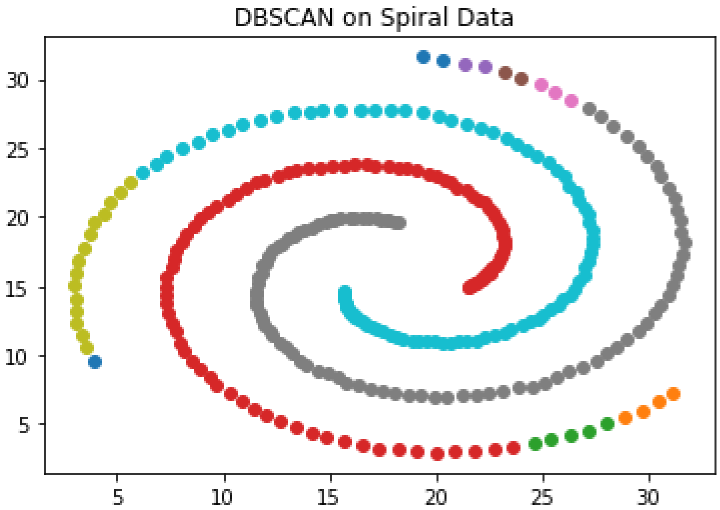
	Clustring Algorithm	Root MSE	DBI	Silhouette Score	Purity	Square Error
0	K-Means	2.719711	0.774641	0.563799	0.957341	0.0
1	Mini-Batch K-Means	2.838231	0.820054	0.519345	0.963294	0.0
2	DBSCAN	0.000000	0.000000	0.000000	0.000000	0.0
3	Spectral Clustering	0.000000	0.000000	0.000000	0.000000	0.0
4	K means self implemenation	0.000000	0.000000	0.000000	0.000000	0.0

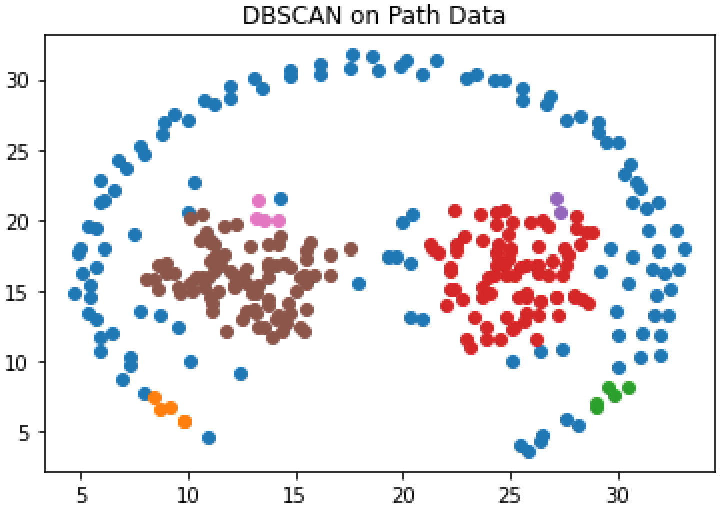
DBSCAN

In [6]:

```
# Mini batch k-means clustering
from numpy import unique
from numpy import where
import numpy as np
from sklearn.cluster import DBSCAN
from matplotlib import pyplot as plt

test=0
y_per= []
def DbScan ( msg,window_size ,sample, data= []):

    model = DBSCAN(eps=window_size, min_samples=(sample))
     # fit the model
    test=model.fit(data)

    yhat = model.fit_predict(data)
    y_per.append(yhat)
    # retrieve unique clusters
    clusters = unique(yhat)

    # create scatter plot for samples from each cluster
    for cluster in clusters:
        # get row indexes for samples with this cluster
        row_ix = where(yhat == cluster)
        # create scatter of these samples
        plt.scatter(data[row_ix, 0], data[row_ix, 1])
    # show the plot
    plt.title(msg);
    plt.show()

Artificial_data,Spiral_data,Path_data=  Load_Data(1)
DbScan( "DBSCAN on Artificial Data ", 0.4,4, Artificial_data)
DbScan( "DBSCAN on Spiral Data ", 0.99999999,1.5,Spiral_data)
DbScan( "DBSCAN on Path Data ",1.333,4.1, Path_data)


# Evaluations
inertia= [0,0,0]

Artificial_data,Spiral_data,Path_data=  Load_Data(0)
Measure_Evaluation(2, 0,Artificial_data, Artificial_Eva, inertia ,y_per)
Measure_Evaluation(2, 1,Spiral_data, Spiral_Eva, inertia, y_per)
Measure_Evaluation(2, 2,Path_data, Path_Eva, inertia, y_per)

Artificial_Eva.head()
# Spiral_Eva.head()
# Path_Eva.head()
```







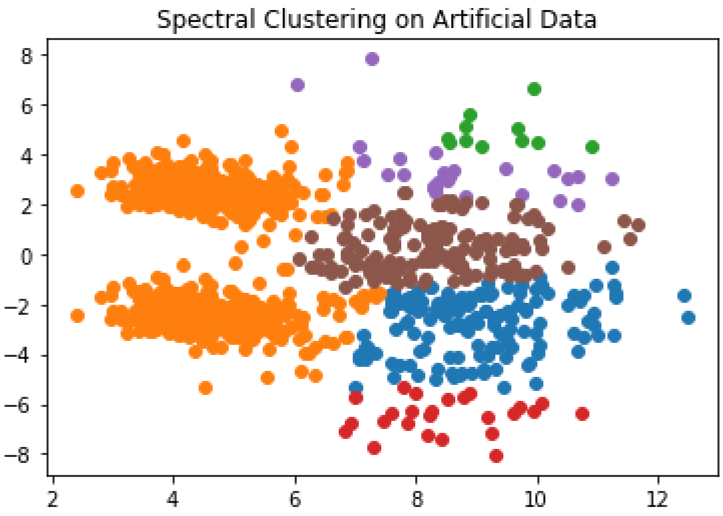
Out[6]:

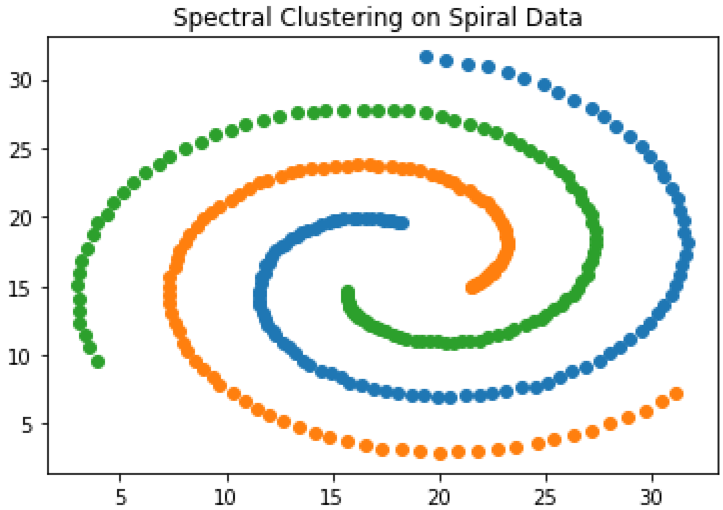
	Clustring Algorithm	Root MSE	DBI	Silhouette Score	Purity	Square Error
0	K-Means	2.719711	0.774641	0.563799	0.957341	0.0
1	Mini-Batch K-Means	2.838231	0.820054	0.519345	0.963294	0.0
2	DBSCAN	2.559025	5.331623	0.255895	0.962302	0.0
3	Spectral Clustering	0.000000	0.000000	0.000000	0.000000	0.0
4	K means self implemenation	0.000000	0.000000	0.000000	0.000000	0.0

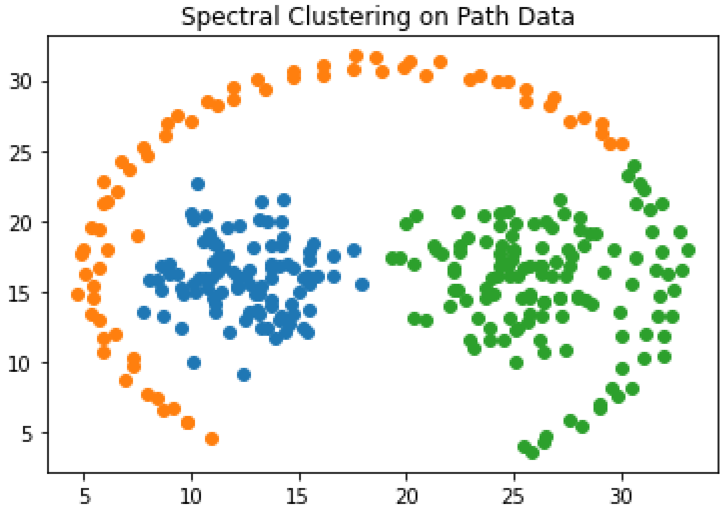
Spectral Clustering

In [7]:

```python
# Mini batch k-means clustering
from numpy import unique
from numpy import where
import numpy as np
from sklearn.cluster import SpectralClustering
from matplotlib import pyplot as plt

y_per= []
def Spectral_Clustering ( msg,cluster, data= []):

    model = SpectralClustering(n_clusters=cluster)
    # fit model and predict clusters
    yhat = model.fit_predict(data)

    y_per.append(yhat)
    # retrieve unique clusters
    clusters = unique(yhat)

    # create scatter plot for samples from each cluster
    for cluster in clusters:
        # get row indexes for samples with this cluster
        row_ix = where(yhat == cluster)
        # create scatter of these samples
        plt.scatter(data[row_ix, 0], data[row_ix, 1])
    # show the plot
    plt.title(msg);
    plt.show()

# Load Data
Artificial_data,Spiral_data,Path_data=  Load_Data(1)

Spectral_Clustering( "Spectral Clustering on Artificial Data ", 6, Artificial_data)
Spectral_Clustering( "Spectral Clustering on Spiral Data ",3,Spiral_data)
Spectral_Clustering( "Spectral Clustering on Path Data ",3, Path_data)


# Evaluations
inertia= [0,0,0]

Artificial_data,Spiral_data,Path_data=  Load_Data(0)
Measure_Evaluation(3, 0,Artificial_data, Artificial_Eva, inertia ,y_per)
Measure_Evaluation(3, 1,Spiral_data, Spiral_Eva, inertia, y_per)
Measure_Evaluation(3, 2,Path_data, Path_Eva, inertia, y_per)

Artificial_Eva.head()
# Spiral_Eva.head()
# Path_Eva.head()
```







Out[7]:

	Clustring Algorithm	Root MSE	DBI	Silhouette Score	Purity	Square Error
0	K-Means	2.719711	0.774641	0.563799	0.957341	0.0
1	Mini-Batch K-Means	2.838231	0.820054	0.519345	0.963294	0.0
2	DBSCAN	2.559025	5.331623	0.255895	0.962302	0.0
3	Spectral Clustering	2.432624	1.089437	0.273916	0.648810	0.0
4	K means self implemenation	0.000000	0.000000	0.000000	0.000000	0.0

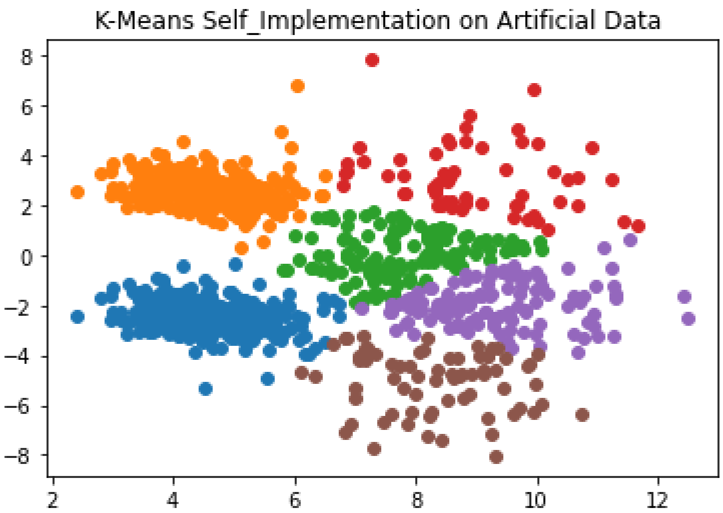
In [ ]:

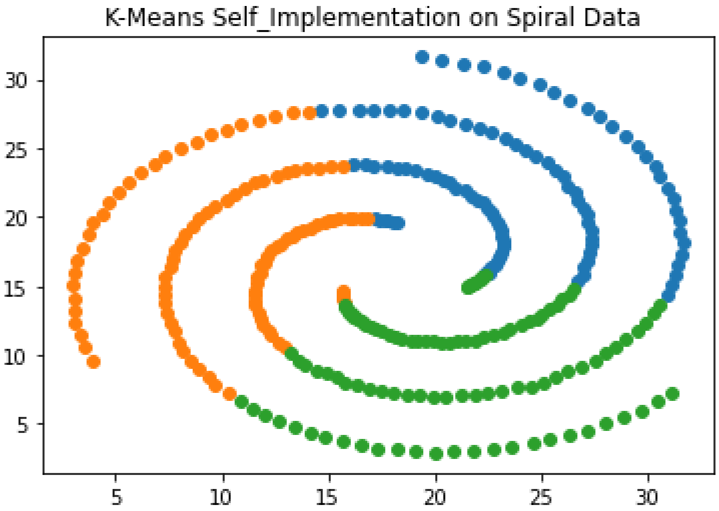
K means self implemenation

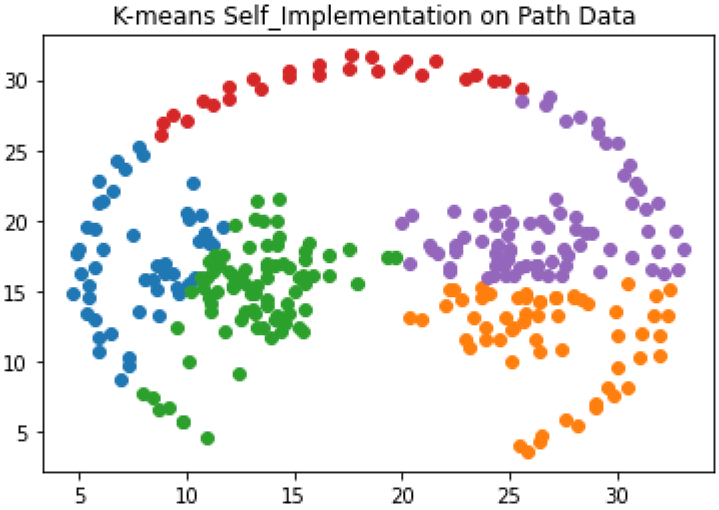
In [8]:

```
#Loading the required modules
import pandas as pd
import numpy as np
from scipy.spatial.distance import cdist
from sklearn.datasets import load_digits
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
```

```
import matplotlib.pyplot as plt
# define dataset
inertia= []
y_per= []
def Kmeans_Self_Implementation ( msg, k, no_of_iterations,  data= []):
    idx = np.random.choice(len(data), k, replace=False)
    #Randomly choosing Centroids
    centroids = data[idx, :] #Step 1
    distance2=0
    #finding the distance between centroids and all the data points
    distances = cdist(data, centroids ,'euclidean') #Step 2

    #Centroid with the minimum Distance
    yhat = np.array([np.argmin(i) for i in distances]) #Step 3

    #Repeating the above steps for a defined number of iterations
    #Step 4
    for _ in range(no_of_iterations):
        centroids = []
        for idx in range(k):
            #Updating Centroids by taking mean of Cluster it belongs to
            temp_cent = data[yhat==idx].mean(axis=0)
            centroids.append(temp_cent)

        centroids = np.vstack(centroids) #Updated Centroids
        distances = cdist(data, centroids ,'euclidean')

        yhat = np.array([np.argmin(i) for i in distances])

    y_per.append(yhat)
    inertia.append(distances)
    clusters = unique(yhat)
    # create scatter plot for samples from each cluster
    for cluster in clusters:
        # get row indexes for samples with this cluster
        row_ix = where(yhat == cluster)
        # create scatter of these samples
        plt.scatter(data[row_ix, 0], data[row_ix, 1])
    # show the plot
    plt.title(msg);
    plt.show()

Artificial_data,Spiral_data,Path_data=  Load_Data(1)
Kmeans_Self_Implementation("K-Means Self_Implementation on Artificial Data ", 6,1000, A
Kmeans_Self_Implementation("K-Means Self_Implementation on Spiral Data ", 3,1000, Spira
Kmeans_Self_Implementation("K-means Self_Implementation on Path Data ",5, 1000, Path_da

# Evaluations
inertia[0]= 0
inertia[1]= 0
inertia[2]= 0
Artificial_data,Spiral_data,Path_data=  Load_Data(0)
Measure_Evaluation(4, 0,Artificial_data, Artificial_Eva, inertia ,y_per)
Measure_Evaluation(4, 1,Spiral_data, Spiral_Eva, inertia, y_per)
Measure_Evaluation(4, 2,Path_data, Path_Eva, inertia, y_per)

Artificial_Eva.head()
# Spiral_Eva.head()
# Path_Eva.head()
```







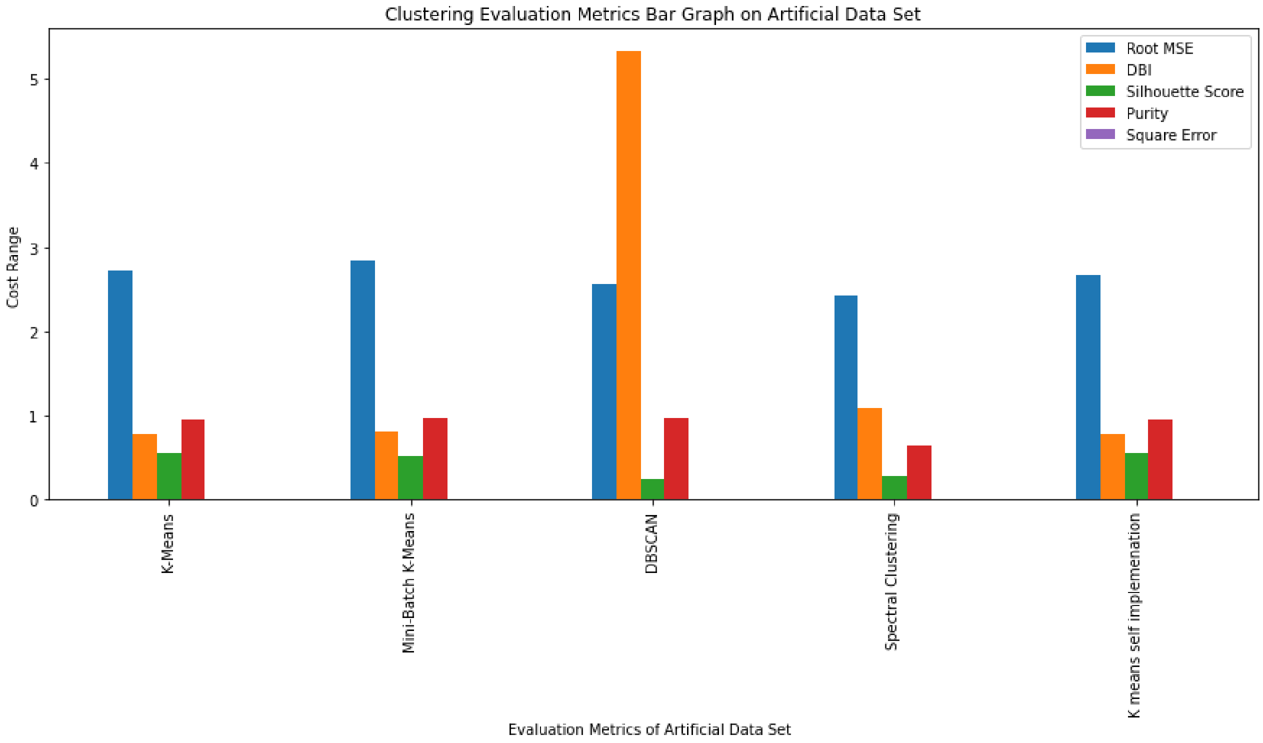
Out[8]:

	Clustring Algorithm	Root MSE	DBI	Silhouette Score	Purity	Square Error
0	K-Means	2.719711	0.774641	0.563799	0.957341	0.0
1	Mini-Batch K-Means	2.838231	0.820054	0.519345	0.963294	0.0
2	DBSCAN	2.559025	5.331623	0.255895	0.962302	0.0
3	Spectral Clustering	2.432624	1.089437	0.273916	0.648810	0.0
4	K means self implemenation	2.659216	0.779103	0.562926	0.956349	0.0

Result for Artificial data set

In [12]:

```
from matplotlib import pyplot as plt
Artificial_Eva.set_index('Clustring Algorithm').plot(kind = 'bar')
plt.title("Clustering Evaluation Metrics Bar Graph on Artificial Data Set")
plt.xlabel("Evaluation Metrics of Artificial Data Set ")
plt.ylabel("Cost Range")
plt.rcParams["figure.figsize"] = (15,6)
plt.show()
Artificial_Eva.head()
```



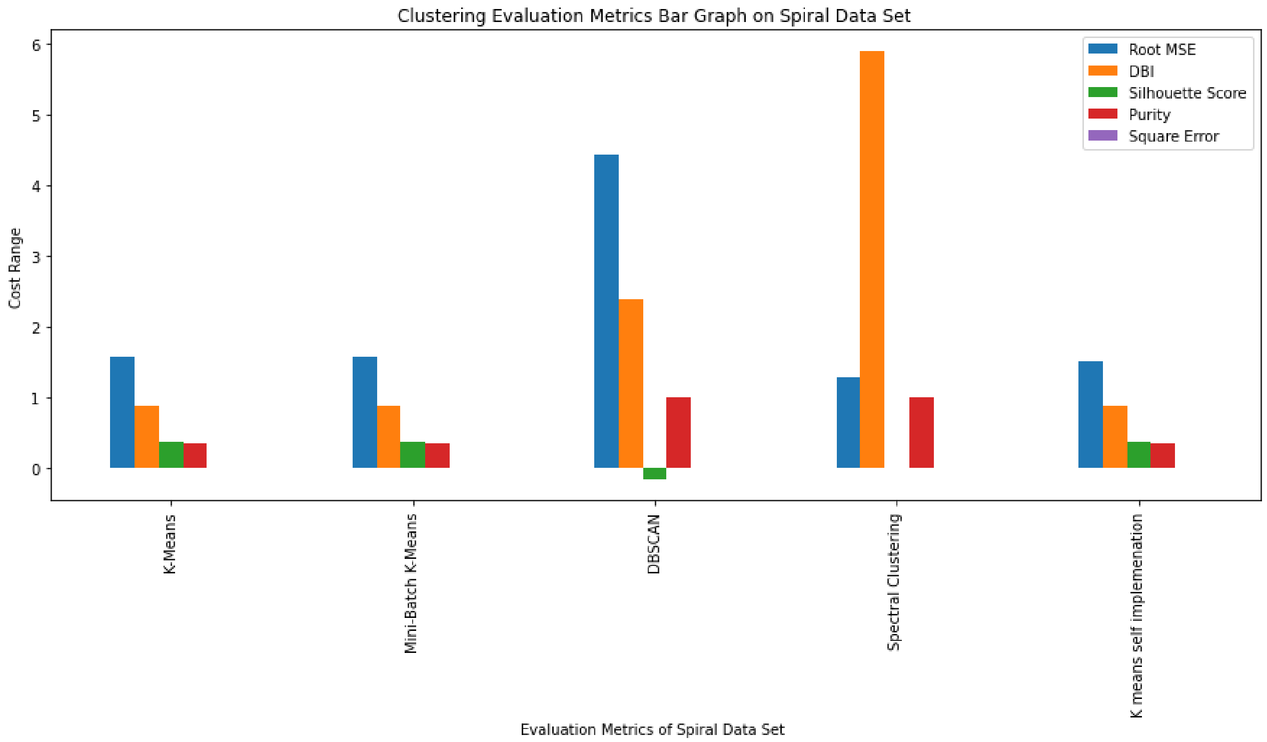
Out[12]:

	Clustring Algorithm	Root MSE	DBI	Silhouette Score	Purity	Square Error
0	K-Means	2.719711	0.774641	0.563799	0.957341	0.0
1	Mini-Batch K-Means	2.838231	0.820054	0.519345	0.963294	0.0
2	DBSCAN	2.559025	5.331623	0.255895	0.962302	0.0
3	Spectral Clustering	2.432624	1.089437	0.273916	0.648810	0.0
4	K means self implemenation	2.659216	0.779103	0.562926	0.956349	0.0

Result for Spiral data set

In [10]:

```
from matplotlib import pyplot as plt
Spiral_Eva.set_index('Clustring Algorithm').plot(kind = 'bar')
plt.title("Clustering Evaluation Metrics Bar Graph on Spiral Data Set")
plt.xlabel("Evaluation Metrics of Spiral Data Set ")
plt.ylabel("Cost Range")
plt.rcParams["figure.figsize"] = (15,6)
plt.show()
Spiral_Eva.head()
```



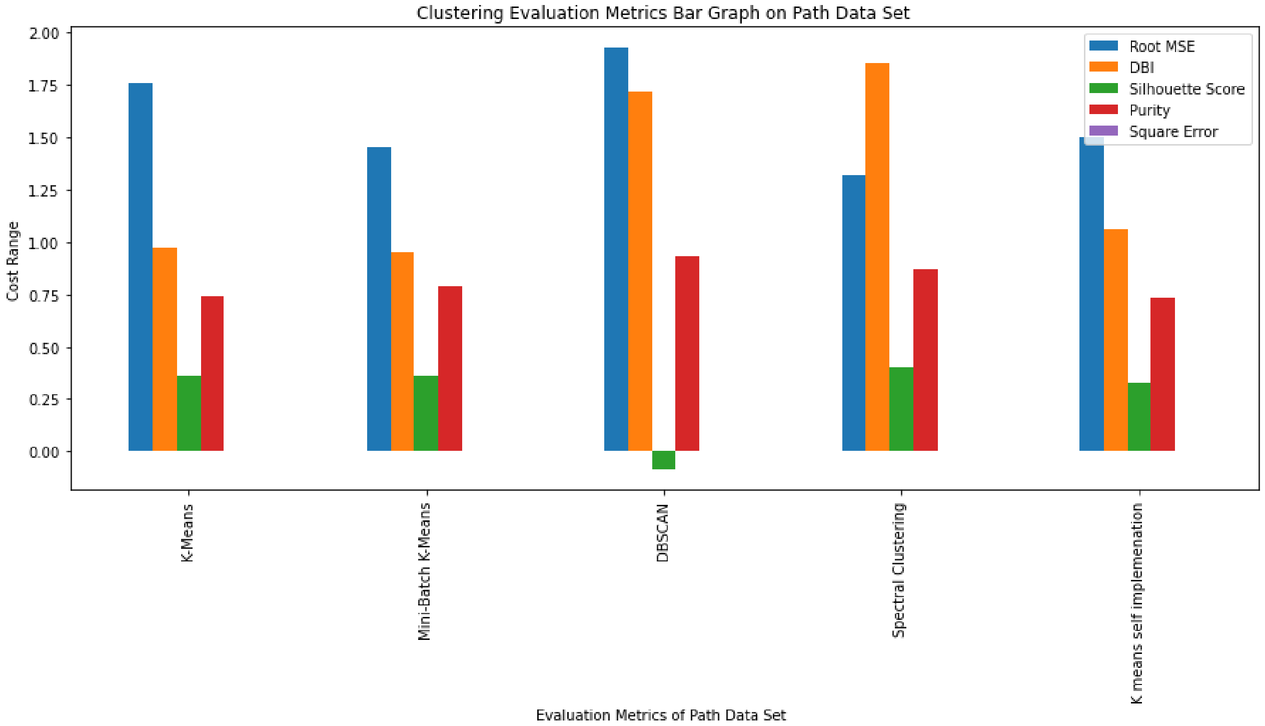
Out[10]:

	Clustring Algorithm	Root MSE	DBI	Silhouette Score	Purity	Square Error
0	K-Means	1.563243	0.886698	0.361981	0.350482	0.0
1	Mini-Batch K-Means	1.565299	0.879688	0.359801	0.344051	0.0
2	DBSCAN	4.436765	2.378795	-0.156053	0.996785	0.0
3	Spectral Clustering	1.294311	5.899294	0.000785	1.000000	0.0
4	K means self implemenation	1.515196	0.885007	0.361372	0.347267	0.0

Result for Path data set

In [11]:

```
from matplotlib import pyplot as plt
Path_Eva.set_index('Clustring Algorithm').plot(kind = 'bar')
plt.title("Clustering Evaluation Metrics Bar Graph on Path Data Set")
plt.xlabel("Evaluation Metrics of Path Data Set ")
plt.ylabel("Cost Range")
plt.rcParams["figure.figsize"] = (15,6)
plt.show()
Path_Eva.head()
```



Out[11]:

	Clustring Algorithm	Root MSE	DBI	Silhouette Score	Purity	Square Error
0	K-Means	1.759827	0.971655	0.363569	0.742475	0.0
1	Mini-Batch K-Means	1.450406	0.951018	0.359900	0.789298	0.0
2	DBSCAN	1.926752	1.718659	-0.083031	0.933110	0.0
3	Spectral Clustering	1.318761	1.849967	0.401102	0.869565	0.0
4	K means self implemenation	1.502506	1.061061	0.329675	0.732441	0.0

In [ ]: