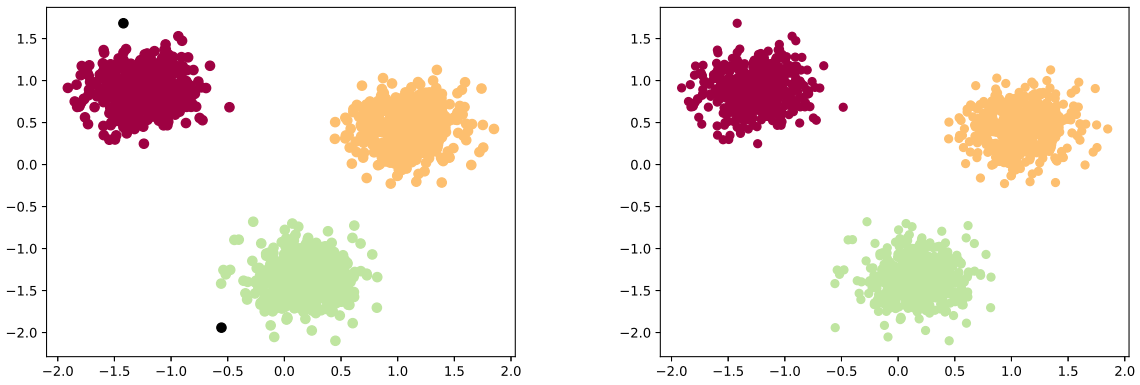
DBSCAN Algorithm - General randomly generated datapoints

```
# Using DBSCAN for random generated data points

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler
from sklearn.datasets.samples_generator import make_blobs

# Data generation function #

def datapoints(centroid,samples,clus_deviation):
    X,y=make_blobs(n_samples=samples,centers=centroid,cluster_std=clus_deviation)
    X=StandardScaler().fit_transform(X)
    return X,y

X, y = datapoints([[4,3], [2,-1], [-1,4]] , 1500, 0.5)

# Modeling #
# DBSCAN works based on two parameters: Epsilon and Minimum Points
# Epsilon determine a specified radius that if includes enough number of points
# within, we call it dense area
# minimumSamples determine the minimum number of data points we want in a
# neighborhood to define a cluster.

eps=0.3
min_samples=7
db=DBSCAN(eps=eps,min_samples=min_samples).fit(X)
labels=db.labels_
with open('DBSCAN_general.txt','a') as f:
    print(labels,file=f)

## Distinguish Outliers ##

samples_mask=np.zeros_like(db.labels_,dtype=bool) # array creation of booleans
samples_mask[db.core_sample_indices_]=True

n_clusters_=len(set(labels))-(1 if -1 in labels else 0) # number of clusters in labels

unique_labels=set(labels) # remove repetition in labels by turning into a set

with open('DBSCAN_general.txt','a') as f:
```

```
    print(samples_mask,file=f)
    print(n_clusters_,file=f)
    print(unique_labels,file=f)

## Data visualization ##
colors=plt.cm.Spectral(np.linspace(0,1,len(unique_labels))) # Cluster color creation

for k,col in zip(unique_labels,colors): # color of points
    if k==-1:
        col='k' # Black for noise
    class_mask=(labels==k)

    xy=X[class_mask & samples_mask]
    plt.scatter(xy[:,0],xy[:,1],s=50,c=[col],marker=u'o',alpha=0.5) # Data point plot

    xy=X[class_mask & ~samples_mask]
    plt.scatter(xy[:,0],xy[:,1],s=50,c=[col],marker=u'o',alpha=0.5) # Outliers plot

# Using K-Means #
from sklearn.cluster import KMeans
k = 3
k_means3 = KMeans(init = "k-means++", n_clusters = k, n_init = 12)
k_means3.fit(X)
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
for k, col in zip(range(k), colors):
    my_members = (k_means3.labels_ == k)
    plt.scatter(X[my_members, 0], X[my_members, 1],  c=col, marker=u'o', alpha=0.5)

#Display plot
plt.show()
```

Solution:

```
[0 0 1 ... 0 0 1]
[ True  True  True ...  True  True  True]
3
{0, 1, 2, -1}
```



DBSCAN Algorithm - Weather Stations Canada using Basemap via Python 2.7

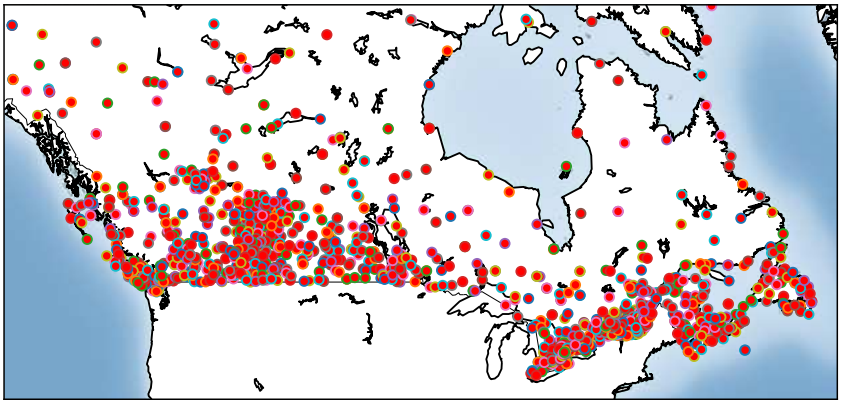
```
# Using DBSCAN for weather station in Canada

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN


# Loading Data #
df=pd.read_csv('D:\Python\edx\Machine Learning\Clustering\weather_Canada.csv')
# with open('DBSCAN_Weather_Station.txt','a') as f:
#     print(df.head(),file=f)
#     print(df.shape,file=f)

# Data Cleaning #

df=df[pd.notnull(df['Tm'])]
df=df.reset_index(drop=True)
# with open('DBSCAN_Weather_Station.txt','a') as f:
#     print(df.head(5),file=f)

# Data Visualization - using basemap package #
from mpl_toolkits.basemap import Basemap
from pylab import rcParams
```

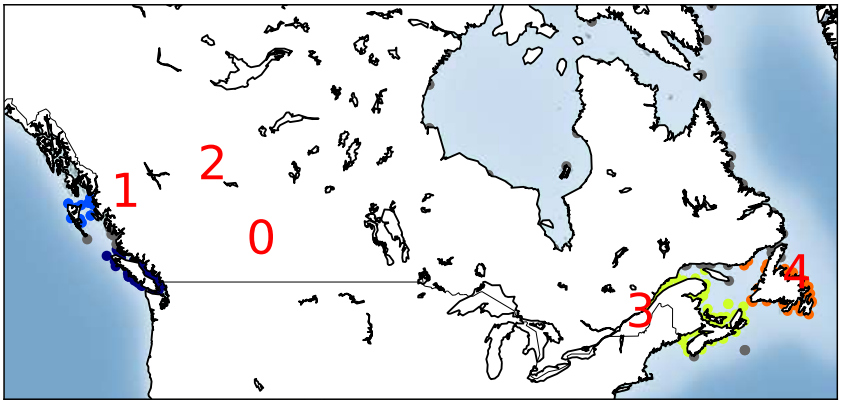
```
rcParams['figure.figsize']=(8,6)

llon=-140
ulon=-50
llat=40
ulat=65

df=df[(df['Long'] > llon) & (df['Long'] < ulon) & (df['Lat'] > llat) & (df['Lat'] < ulat)
my_map = Basemap(projection='merc',
            resolution = 'l', area_thresh = 1000.0,
            llcrnrlon=llon, llcrnrlat=llat, #min longitude (llcrnrlon) and latitude (llcr
            urcrnrlon=ulon, urcrnrlat=ulat) #max longitude (urcrnrlon) and latitude (urcr

my_map.drawcoastlines()
my_map.drawcountries()
#my_map.drawmapboundary()
my_map.fillcontinents(color = 'white', alpha = 0.3)
my_map.shadedrelief()

# To collect data based on stations

xs,ys = my_map(np.asarray(df.Long), np.asarray(df.Lat))
df['xm']= xs.tolist()
df['ym'] =ys.tolist()

#Visualization1
for index,row in df.iterrows():
   my_map.plot(row.xm, row.ym,markerfacecolor =([1,0,0]),  marker='o', markersize= 5, alp

## Clustering of stations based on their location

from sklearn.preprocessing import StandardScaler
import sklearn.utils
sklearn.utils.check_random_state(1000)

c_dataset=df[['xm','ym']]
c_dataset=np.nan_to_num(c_dataset)
c_dataset=StandardScaler().fit_transform(c_dataset)

# Compute DBSCAN

db = DBSCAN(eps=0.15, min_samples=10).fit(c_dataset)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_
df["Clus_Db"]=labels
```

```
realClusterNum=len(set(labels)) - (1 if -1 in labels else 0)
clusterNum = len(set(labels))

# A sample of clusters
e=df[["Stn_Name","Tx","Tm","Clus_Db"]].head(5)
set(labels)


## Visualization
plt.figure()
rcParams['figure.figsize'] = (8,6)

my_map = Basemap(projection='merc',
            resolution = 'l', area_thresh = 1000.0,
            llcrnrlon=llon, llcrnrlat=llat, #min longitude (llcrnrlon) and latitude (llc
            urcrnrlon=ulon, urcrnrlat=ulat) #max longitude (urcrnrlon) and latitude (urc

my_map.drawcoastlines()
my_map.drawcountries()
#my_map.drawmapboundary()
my_map.fillcontinents(color = 'white', alpha = 0.3)
my_map.shadedrelief()

# To create a color map
colors = plt.get_cmap('jet')(np.linspace(0.0, 1.0, clusterNum))

#Visualization1
for clust_number in set(labels):
    c=(([0.4,0.4,0.4]) if clust_number == -1 else colors[np.int(clust_number)])
    clust_set = df[df.Clus_Db == clust_number]
    my_map.scatter(clust_set.xm, clust_set.ym, color =c,  marker='o', s= 20, alpha = 0.85
    if clust_number != -1:
        cenx=np.mean(clust_set.xm)
        ceny=np.mean(clust_set.ym)
        plt.text(cenx,ceny,str(clust_number), fontsize=25, color='red',)
        print ("Cluster "+str(clust_number)+', Avg Temp: '+ str(np.mean(clust_set.Tm)))

## Clustering of stations based on their location,mean,max and min Temp

Clus_dataSet = df[['xm','ym','Tx','Tm','Tn']]
Clus_dataSet = np.nan_to_num(Clus_dataSet)
Clus_dataSet = StandardScaler().fit_transform(Clus_dataSet)

# Compute DBSCAN
db1 = DBSCAN(eps=0.3, min_samples=10).fit(Clus_dataSet)
```

```
core_samples_mask1 = np.zeros_like(db1.labels_, dtype=bool)
core_samples_mask1[db1.core_sample_indices_] = True
labels1 = db1.labels_
df["Clus_Db"]=labels1

realClusterNum1=len(set(labels1)) - (1 if -1 in labels1 else 0)
clusterNum1 = len(set(labels1))


# A sample of clusters
f=df[["Stn_Name","Tx","Tm","Clus_Db"]].head(5)

# Visulaization based on temp and location

plt.figure()
rcParams['figure.figsize'] = (8,6)

my_map = Basemap(projection='merc',
            resolution = 'l', area_thresh = 1000.0,
            llcrnrlon=llon, llcrnrlat=llat, #min longitude (llcrnrlon) and latitude (llc
            urcrnrlon=ulon, urcrnrlat=ulat) #max longitude (urcrnrlon) and latitude (urc

my_map.drawcoastlines()
my_map.drawcountries()
#my_map.drawmapboundary()
my_map.fillcontinents(color = 'white', alpha = 0.3)
my_map.shadedrelief()

# To create a color map
colors = plt.get_cmap('jet')(np.linspace(0.0, 1.0, clusterNum))



#Visualization1
for clust_number1 in set(labels):
    c=(([0.4,0.4,0.4]) if clust_number1 == -1 else colors[np.int(clust_number1)])
    clust_set1 = df[df.Clus_Db == clust_number1]
    my_map.scatter(clust_set1.xm, clust_set1.ym, color =c,  marker='o', s= 20, alpha = 0
    if clust_number1 != -1:
        cenx1=np.mean(clust_set1.xm)
        ceny1=np.mean(clust_set1.ym)
        plt.text(cenx1,ceny1,str(clust_number1), fontsize=25, color='red',)
        print ("Cluster "+str(clust_number1)+', Avg Temp: '+ str(np.mean(clust_set1.Tm)))

#Display plot
plt.show()
```

Solution:




1
2
0
3
4