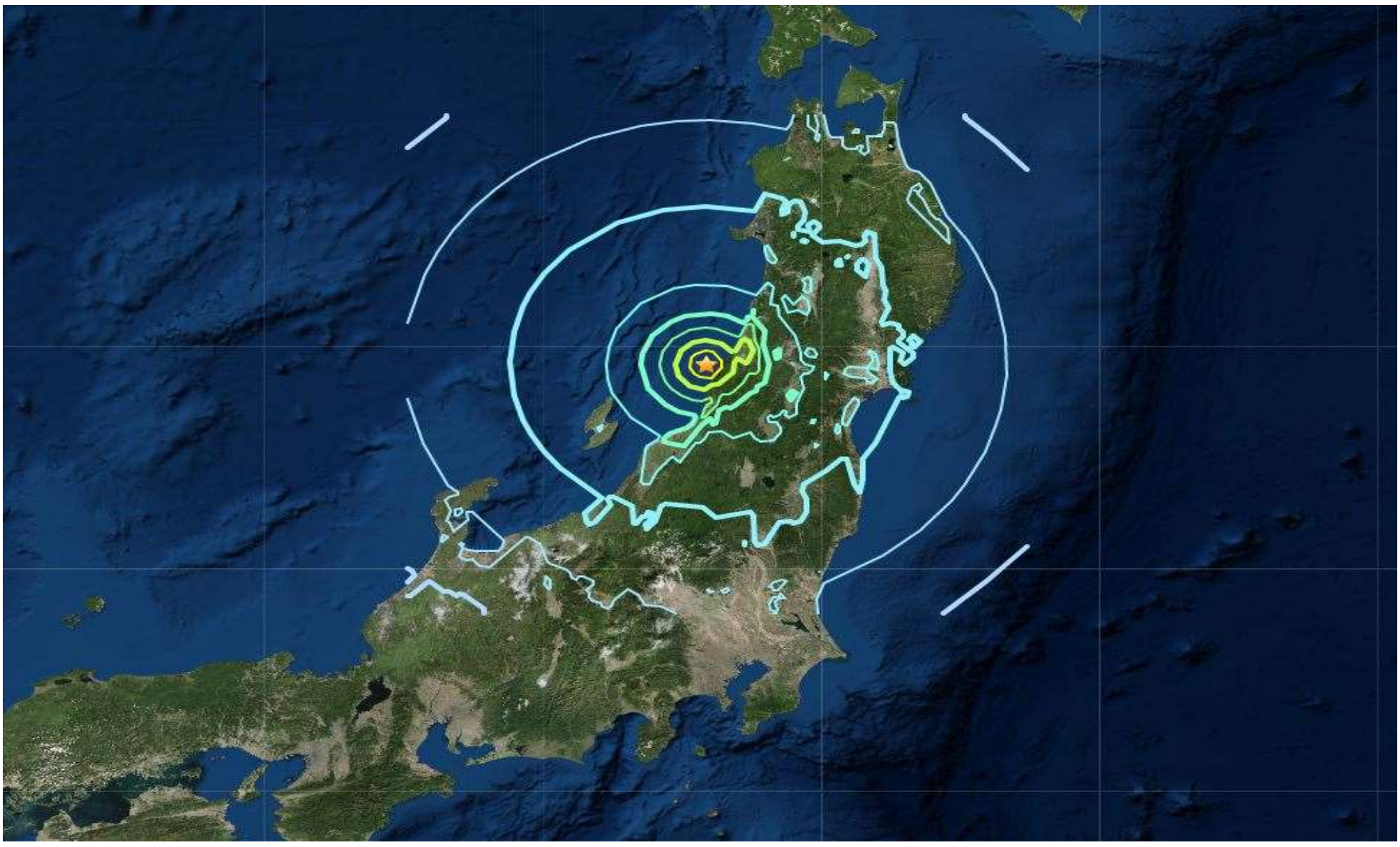
Clustering Significant Earthquakes in Japan with DBSCAN



Picture Source: bloomberg

Data Description

Context

The National Earthquake Information Center (NEIC) determines the location and size of all significant earthquakes that occur worldwide and disseminates this information immediately to national and international agencies, scientists, critical facilities, and the general public. The NEIC compiles and provides to scientists and to the public an extensive seismic database that serves as a foundation for scientific research through the operation of modern digital national and global seismograph networks and cooperative international agreements. The NEIC is the national data center and archive for earthquake information.

Content

This dataset includes a record of the date, time, location, depth, magnitude, and source of every earthquake with a reported **magnitude 5.5 or higher since 1965**.

Dataset Link

You can download or take a look at original website of the dataset: Kaggle

License

CC0: Public Domain

Keywords

- Geology

- Earth Science
- Earthquake
- Japan

License

CC BY-SA 4.0

Sources

- Kaggle
- Bloomberg

Objective for This Notebook

Within the scope of this project, a clustering task was done. With density based clustering, we were trying to classify or make cluster of points (based of their location, earthquake depth and magnitude).

Estimated Time Needed: **25 min**

Importing Libraries

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

Load the Dataset

You can download the data from Kaggle. After downloading and importing process, we can unzip.

```
!unzip archive.zip
```

```
Archive:  archive.zip
  inflating: database.csv
```

```
filename='database.csv'
```

Let's take a look at our dataset

```
pdf = pd.read_csv(filename)
```

```
pdf.dropna(subset=['Latitude', 'Longitude', 'Depth', 'Magnitude', 'Source'])
```

	Date	Time	Latitude	Longitude	Type	Depth	Depth Error	Depth Seismic Stations	Magnitude	Magnitu Ty
0	01/02/1965	13:44:18	19.2460	145.6160	Earthquake	131.60	NaN	NaN	6.0	M
1	01/04/1965	11:29:49	1.8630	127.3520	Earthquake	80.00	NaN	NaN	5.8	M
2	01/05/1965	18:05:58	-20.5790	-173.9720	Earthquake	20.00	NaN	NaN	6.2	M
3	01/08/1965	18:49:43	-59.0760	-23.5570	Earthquake	15.00	NaN	NaN	5.8	M
4	01/09/1965	13:32:50	11.9380	126.4270	Earthquake	15.00	NaN	NaN	5.8	M
...	...	...	...	...	...	...	...	...	...	
23407	12/28/2016	08:22:12	38.3917	-118.8941	Earthquake	12.30	1.2	40.0	5.6	M

```
pdf.head(10)
```

	Date	Time	Latitude	Longitude	Type	Depth	Depth Error	Depth Seismic Stations	Magnitude	Magnitude Type	.
0	01/02/1965	13:44:18	19.246	145.616	Earthquake	131.6	NaN	NaN	6.0	MW	
1	01/04/1965	11:29:49	1.863	127.352	Earthquake	80.0	NaN	NaN	5.8	MW	
2	01/05/1965	18:05:58	-20.579	-173.972	Earthquake	20.0	NaN	NaN	6.2	MW	
3	01/08/1965	18:49:43	-59.076	-23.557	Earthquake	15.0	NaN	NaN	5.8	MW	
4	01/09/1965	13:32:50	11.938	126.427	Earthquake	15.0	NaN	NaN	5.8	MW	
5	01/10/1965	13:36:32	-13.405	166.629	Earthquake	35.0	NaN	NaN	6.7	MW	
6	01/12/1965	13:32:25	27.357	87.867	Earthquake	20.0	NaN	NaN	5.9	MW	
7	01/15/1965	23:17:42	-13.309	166.212	Earthquake	35.0	NaN	NaN	6.0	MW	
8	01/16/1965	11:32:37	-56.452	-27.043	Earthquake	95.0	NaN	NaN	6.0	MW	
9	01/17/1965	10:43:17	-24.563	178.487	Earthquake	565.0	NaN	NaN	5.8	MW	

10 rows × 21 columns

```
pdf.tail(10)
```

	Date	Time	Latitude	Longitude	Type	Depth	Depth Error	Depth Seismic Stations	Magnitude	Magnitud Typ
23402	12/24/2016	03:58:55	-5.1460	153.5166	Earthquake	30.00	1.8	NaN	5.8	MW
23403	12/25/2016	14:22:27	-43.4029	-73.9395	Earthquake	38.00	1.9	NaN	7.6	MW
23404	12/25/2016	14:32:13	-43.4810	-74.4771	Earthquake	14.93	3.3	NaN	5.6	M
23405	12/27/2016	23:20:56	45.7192	26.5230	Earthquake	97.00	1.8	NaN	5.6	MW
23406	12/28/2016	08:18:01	38.3754	-118.8977	Earthquake	10.80	1.3	34.0	5.6	M
23407	12/28/2016	08:22:12	38.3917	-118.8941	Earthquake	12.30	1.2	40.0	5.6	M
23408	12/28/2016	09:13:47	38.3777	-118.8957	Earthquake	8.80	2.0	33.0	5.5	M
23409	12/28/2016	12:38:51	36.9179	140.4262	Earthquake	10.00	1.8	NaN	5.9	MW
23410	12/29/2016	22:30:19	-9.0283	118.6639	Earthquake	79.00	1.8	NaN	6.3	MW
23411	12/30/2016	20:08:28	37.3973	141.4103	Earthquake	11.94	2.2	NaN	5.5	M

10 rows × 21 columns

```
pdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23412 entries, 0 to 23411
Data columns (total 21 columns):
 #   Column                      Non-Null Count  Dtype
```

```
---  ------                     --------------  -----
 0   Date                       23412 non-null  object
 1   Time                       23412 non-null  object
 2   Latitude                   23412 non-null  float64
 3   Longitude                  23412 non-null  float64
 4   Type                       23412 non-null  object
 5   Depth                      23412 non-null  float64
 6   Depth Error                4461 non-null   float64
 7   Depth Seismic Stations     7097 non-null   float64
 8   Magnitude                  23412 non-null  float64
 9   Magnitude Type             23409 non-null  object
 10  Magnitude Error            327 non-null    float64
 11  Magnitude Seismic Stations  2564 non-null   float64
 12  Azimuthal Gap              7299 non-null   float64
 13  Horizontal Distance        1604 non-null   float64
 14  Horizontal Error           1156 non-null   float64
 15  Root Mean Square           17352 non-null  float64
 16  ID                         23412 non-null  object
 17  Source                     23412 non-null  object
 18  Location Source            23412 non-null  object
 19  Magnitude Source           23412 non-null  object
 20  Status                     23412 non-null  object
dtypes: float64(12), object(9)
memory usage: 3.8+ MB
```

Visualization

The matplotlib basemap toolkit is a library for plotting 2D data on maps in Python. Basemap does not do any plotting on it’s own, but provides the facilities to transform coordinates to a map projections.

```
!pip3 install basemap
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: basemap in /usr/local/lib/python3.7/dist-packages (1.3.4)
Requirement already satisfied: pyproj<3.4.0,>=1.9.3 in /usr/local/lib/python3.7/dist-packages (from basemap) (3.2.1)
Requirement already satisfied: basemap-data<1.4,>=1.3.2 in /usr/local/lib/python3.7/dist-packages (from basemap) (1.3.2)
Requirement already satisfied: matplotlib<3.6,>=1.5 in /usr/local/lib/python3.7/dist-packages (from basemap) (3.2.2)
Requirement already satisfied: pyshp<2.4,>=1.2 in /usr/local/lib/python3.7/dist-packages (from basemap) (2.3.1)
Requirement already satisfied: numpy<1.22,>=1.21 in /usr/local/lib/python3.7/dist-packages (from basemap) (1.21.6)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib<3.6,>=1.5->basemap)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib<3.6,>=1.5->basemap) (0.1
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib<3.6,>=1.5->basem
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplot
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from kiwisolver>=1.0.1->matplotlib<
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from pyproj<3.4.0,>=1.9.3->basemap) (2022.6.1
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1->matplotlib<3.6,>=
```

```
from mpl_toolkits.basemap import Basemap

import matplotlib.pyplot as plt
from pylab import rcParams
# %matplotlib inline
rcParams['figure.figsize'] = (20, 20)
```

Approximate coordinates of Japan

```
llon=122.77
ulon=153.14
llat=23.03
ulat=50.54
```

```
print(f"llon: {llon}")
print(f"ulon: {ulon}")
print(f"llat: {llat}")
print(f"ulat: {ulat}")
```

```
llon: 122.77
ulon: 153.14
llat: 23.03
ulat: 50.54
```

```
pdf = pdf[(pdf['Longitude'] > llon) & (pdf['Longitude'] < ulon) & (pdf['Latitude'] > llat) &(pdf['Latitude'] < ulat)]

my_map = Basemap(projection='merc',
            resolution = 'l', area_thresh = 1000.0,
```

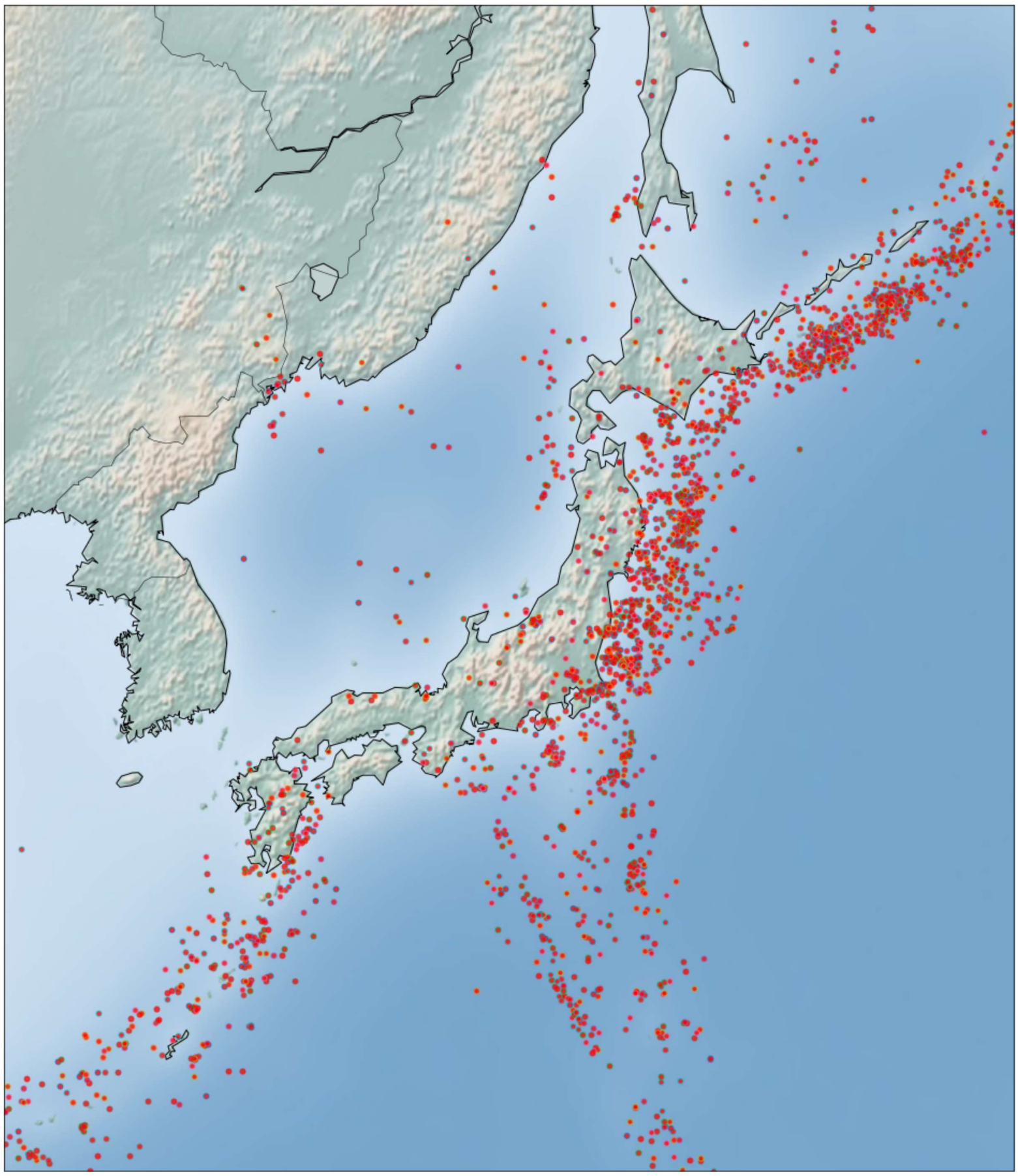
```
            llcrnrlon=llon, llcrnrlat=llat, #min longitude (llcrnrlon) and latitude (llcrnrlat)
            urcrnrlon=ulon, urcrnrlat=ulat) #max longitude (urcrnrlon) and latitude (urcrnrlat)

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

```
<matplotlib.image.AxesImage at 0x7fd91a508a50>
```

```
xs,ys = my_map(np.asarray(pdf.Longitude), np.asarray(pdf.Latitude))
pdf['xm']= xs.tolist()
pdf['ym'] =ys.tolist()
```

```
my_map.drawcoastlines()
my_map.drawcountries()
my_map.fillcontinents(color = 'white', alpha = 0.3)
my_map.shadedrelief()
for index, row in pdf.iterrows():
```

```
   my_map.plot(row.xm, row.ym, markerfacecolor =([1, 0, 0]),  marker='o', markersize= 5, alpha = 0.75)
plt.show()
```



Clustering of Stations Based on Their Location i.e. Lat & Lon

DBSCAN form sklearn library can run *DBSCAN* clustering from vector array or distance matrix. In our case, we pass it the Numpy array Clus_dataSet to find core samples of high density and expands clusters from them.

```
from sklearn.cluster import DBSCAN
import sklearn.utils
from sklearn.preprocessing import StandardScaler
```

```
sklearn.utils.check_random_state(1000)
Clus_dataSet = pdf[['xm', 'ym']]
Clus_dataSet = np.nan_to_num(Clus_dataSet)
Clus_dataSet = StandardScaler().fit_transform(Clus_dataSet)
```

Computing DBSCAN.

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

```
labels
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

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

A sample of clusters:

```
pdf[["Source", "Depth", "Clus_Db"]].head(5)
```

	Source	Depth	Clus_Db
55	ISCGEM	53.5	0
89	ISCGEM	32.1	0
101	ISCGEM	30.0	0
110	ISCGEM	39.2	0
111	ISCGEM	50.0	0

```
set(labels)
```

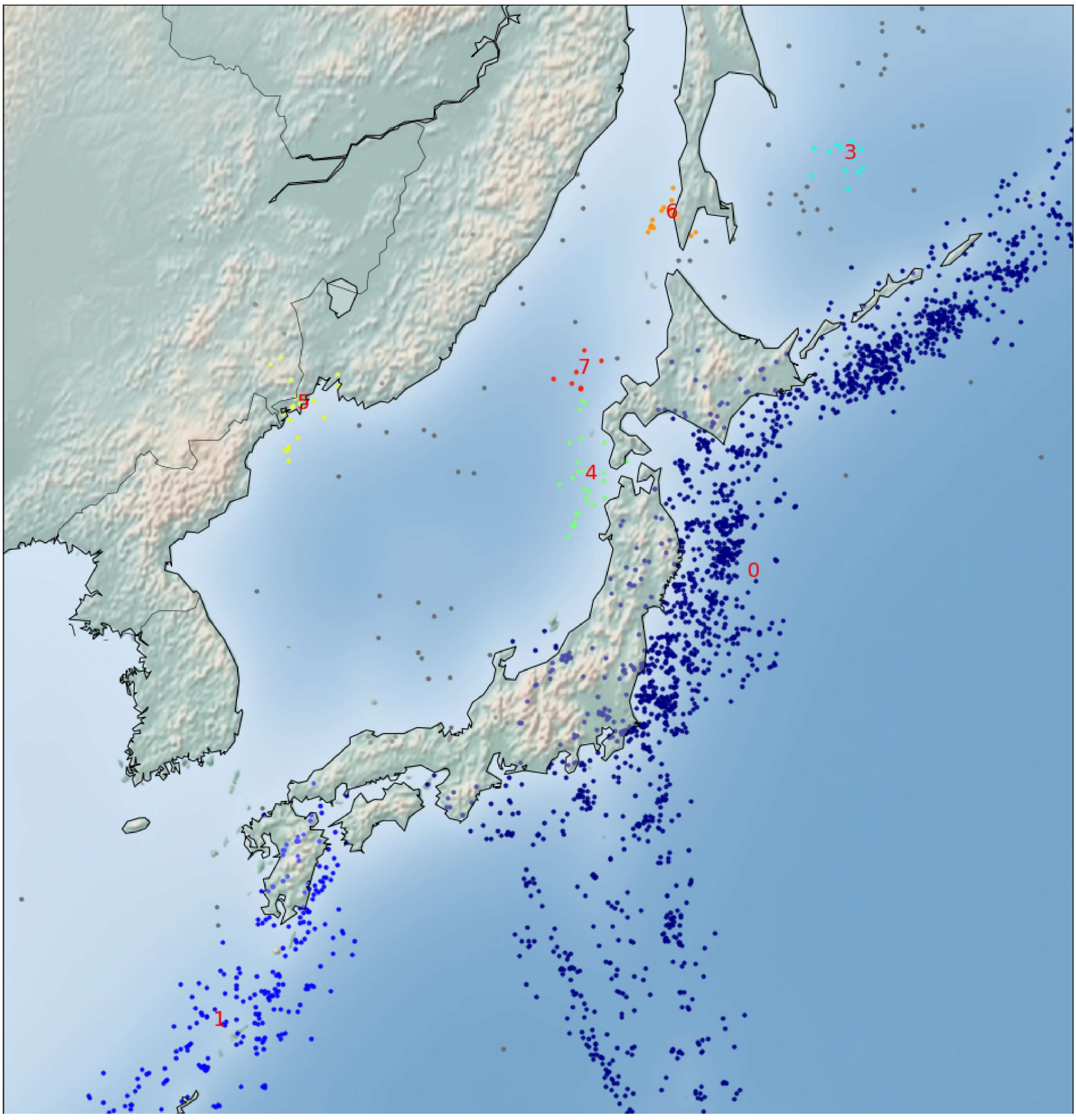
```
{-1, 0, 1, 2, 3, 4, 5, 6, 7}
```

Visualization of Clusters Based on Location

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
from pylab import rcParams
# %matplotlib inline
rcParams['figure.figsize'] = (20, 20)
```

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

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

colors = plt.get_cmap('jet')(np.linspace(0.0, 1.0, clusterNum))

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 = pdf[pdf.Clus_Db == clust_number]
    my_map.scatter(clust_set.xm, clust_set.ym, color=c,  marker='o', s=10, 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=18, color='red',)
        print ("Cluster "+str(clust_number)+', Avg Depth: '+ str(np.mean(clust_set.Depth)))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:14: DeprecationWarning: `np.int` is a deprecated alias for the bui
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations

Cluster 0, Avg Depth: 63.55876483279396
Cluster 1, Avg Depth: 52.31665529010238
Cluster 2, Avg Depth: 138.70689655172413
Cluster 3, Avg Depth: 397.0846153846153
Cluster 4, Avg Depth: 66.68888888888888
Cluster 5, Avg Depth: 546.7666666666665
Cluster 6, Avg Depth: 56.60666666666666
Cluster 7, Avg Depth: 109.25714285714287
```



Clustering of Sources Based on Their Location i.e. Lat & Lon and Depth

```
from sklearn.cluster import DBSCAN
import sklearn.utils
from sklearn.preprocessing import StandardScaler
```

```
sklearn.utils.check_random_state(1000)
Clus_dataSet = pdf[['xm', 'ym', 'Depth']]
Clus_dataSet = np.nan_to_num(Clus_dataSet)
Clus_dataSet = StandardScaler().fit_transform(Clus_dataSet)
```

```
db = DBSCAN(eps=0.15, min_samples=10).fit(Clus_dataSet)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
```

```
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_
pdf["Clus_Db"]=labels

labels
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

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

```
pdf[["Source", "Depth", "Clus_Db"]].head(5)
```

	Source	Depth	Clus_Db
55	ISCGEM	53.5	0
89	ISCGEM	32.1	0
101	ISCGEM	30.0	0
110	ISCGEM	39.2	0
111	ISCGEM	50.0	0

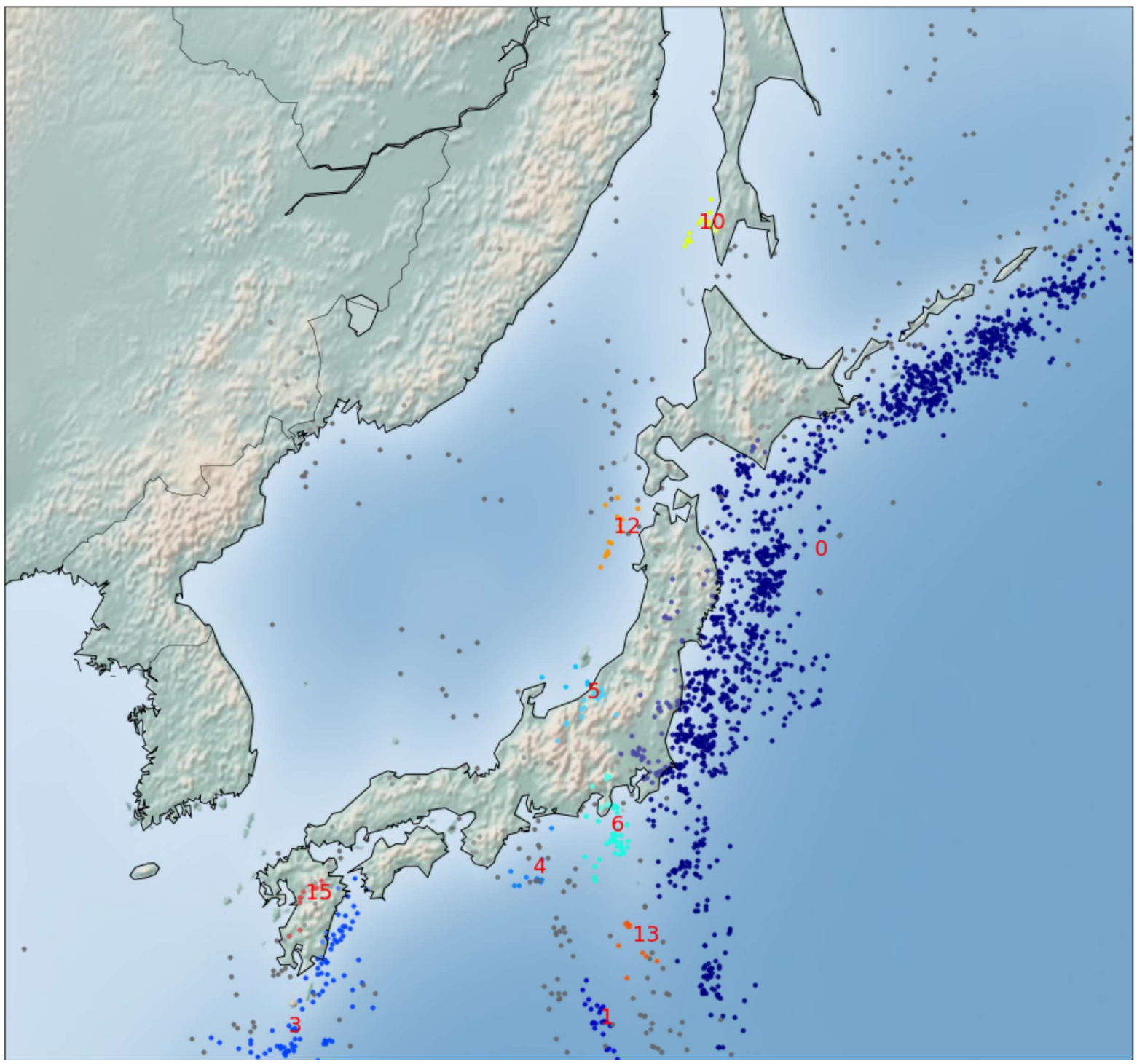
Visualization of Clusters Based on Location and Depth

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
from pylab import rcParams
# %matplotlib inline
rcParams['figure.figsize'] = (20, 20)
```

```
my_map = Basemap(projection='merc',
            resolution = 'l', area_thresh = 1000.0,
            llcrnrlon=llon, llcrnrlat=llat,
            urcrnrlon=ulon, urcrnrlat=ulat)

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

colors = plt.get_cmap('jet')(np.linspace(0.0, 1.0, clusterNum))

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 = pdf[pdf.Clus_Db == clust_number]
    my_map.scatter(clust_set.xm, clust_set.ym, color=c,  marker='o', s=10, 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=18, color='red')
        print ("Cluster " + str(clust_number)+', Avg Depth: '+ str(np.mean(clust_set.Depth)))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:14: DeprecationWarning: `np.int` is a depr
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-note
```

```
Cluster 0, Avg Depth: 35.21208852005533
Cluster 1, Avg Depth: 425.332
Cluster 2, Avg Depth: 33.23529411764706
Cluster 3, Avg Depth: 36.37571428571429
Cluster 4, Avg Depth: 12.736363636363638
Cluster 5, Avg Depth: 12.399999999999999
Cluster 6, Avg Depth: 14.634146341463415
Cluster 7, Avg Depth: 25.42941176470588
Cluster 8, Avg Depth: 25.353846153846153
Cluster 9, Avg Depth: 30.166666666666668
Cluster 10, Avg Depth: 14.976923076923079
Cluster 11, Avg Depth: 477.3016666666666
Cluster 12, Avg Depth: 25.321428571428566
Cluster 13, Avg Depth: 32.46
Cluster 14, Avg Depth: 13.557142857142855
Cluster 15, Avg Depth: 9.768
```



Visualization of Clusters Based on Location and Magnitude

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
from pylab import rcParams
# %matplotlib inline
rcParams['figure.figsize'] = (20, 20)
```

```
sklearn.utils.check_random_state(1000)
Clus_dataSet = pdf[['xm', 'ym', 'Magnitude']]
Clus_dataSet = np.nan_to_num(Clus_dataSet)
Clus_dataSet = StandardScaler().fit_transform(Clus_dataSet)
```

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

```
labels
```

```
array([ 0, -1, -1, ...,  6,  6, 18])
```

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

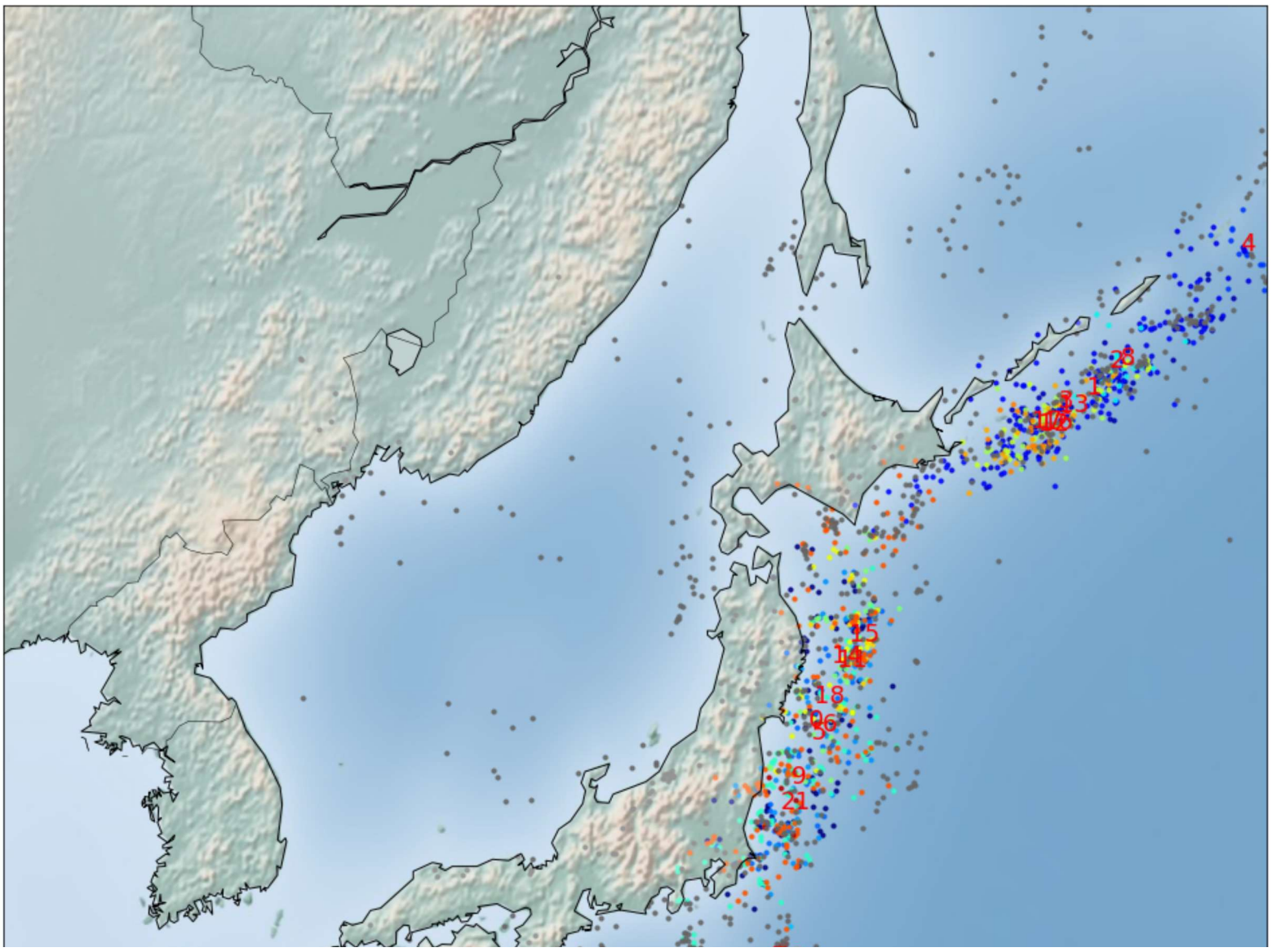
```
pdf[["Source", "Magnitude", "Clus_Db"]].head(5)
```

	Source	Magnitude	Clus_Db
55	ISCGEM	5.7	0
89	ISCGEM	6.4	-1
101	ISCGEM	6.4	-1
110	ISCGEM	5.7	1
111	ISCGEM	5.7	0

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
from pylab import rcParams
# %matplotlib inline
rcParams['figure.figsize'] = (20, 20)
```

```
my_map = Basemap(projection='merc',
            resolution = 'l', area_thresh = 1000.0,
            llcrnrlon=llon, llcrnrlat=llat,
            urcrnrlon=ulon, urcrnrlat=ulat)

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

colors = plt.get_cmap('jet')(np.linspace(0.0, 1.0, clusterNum))

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 = pdf[pdf.Clus_Db == clust_number]
    my_map.scatter(clust_set.xm, clust_set.ym, color=c,  marker='o', s=10, 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=18, color='red')
        print ("Cluster " + str(clust_number)+', Avg Magnitude: '+ str(np.mean(clust_set.Magnitude)))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:14: DeprecationWarning: `np.int` is a depr
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-note

Cluster 0, Avg Magnitude: 5.700000000000002
Cluster 1, Avg Magnitude: 5.7
Cluster 2, Avg Magnitude: 5.5
Cluster 3, Avg Magnitude: 5.600000000000001
Cluster 4, Avg Magnitude: 5.6000000000000005
Cluster 5, Avg Magnitude: 5.799999999999999
Cluster 6, Avg Magnitude: 5.8999999999999995
Cluster 7, Avg Magnitude: 5.599999999999999
Cluster 8, Avg Magnitude: 6.0
Cluster 9, Avg Magnitude: 5.6000000000000005
Cluster 10, Avg Magnitude: 5.700000000000001
Cluster 11, Avg Magnitude: 6.0
Cluster 12, Avg Magnitude: 6.099999999999998
Cluster 13, Avg Magnitude: 5.900000000000001
Cluster 14, Avg Magnitude: 6.1
Cluster 15, Avg Magnitude: 6.200000000000001
Cluster 16, Avg Magnitude: 5.800000000000001
Cluster 17, Avg Magnitude: 6.0
Cluster 18, Avg Magnitude: 5.5
Cluster 19, Avg Magnitude: 5.900000000000002
Cluster 20, Avg Magnitude: 5.700000000000001
Cluster 21, Avg Magnitude: 6.1000000000000005
```



Contact Me

If you have something to say to me please contact me:

- Twitter: https://twitter.com/Doguilmak
- Mail address: doguilmak@gmail.com