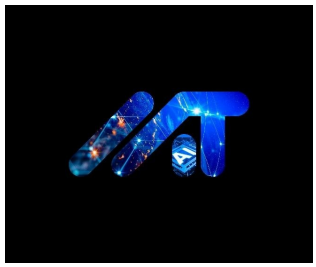


Density Clustering

Most of the traditional clustering techniques, such as k-means, hierarchical and fuzzy clustering, can be used to group data without supervision.

However, when applied to tasks with arbitrary shape clusters, or clusters within a cluster, the traditional techniques might be unable to achieve good results. That is, elements in the same cluster might not share enough similarity or the performance may be poor. Additionally, Density-based clustering locates regions of high density that are separated from one another by regions of low density. Density, in this context, is defined as the number of points within a specified radius.

Weather Station Clustering using DBSCAN & scikit-learn

DBSCAN is especially very good for tasks like class identification in a spatial context. The wonderful attribute of DBSCAN algorithm is that it can find out any arbitrary shape cluster without getting affected by noise. For example, this following example cluster the location of weather stations in Canada.

DBSCAN can be used here, for instance, to find the group of stations which show the same weather condition. As you can see, it not only finds different arbitrary shaped clusters, can find the denser part of data-centered samples by ignoring less-dense areas or noises.

Let's start playing with the data. We will be working according to the following workflow:

1. Loading data

- Overview data
- Data cleaning
- Data selection
- Clusteing

Importing required packages

Set before importing Basemap

In [23]:
```
import os
os.environ['PROJ_LIB'] = 'C:/Users/Meer Moazzam/Anaconda3/share/proj'
```

In [24]:
```
import numpy as np
from sklearn.cluster import DBSCAN
from mpl_toolkits.basemap import Basemap
from pylab import rcParams
from sklearn.datasets import make_blobs
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pandas as pd
import sklearn.utils
%matplotlib inline
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

Let's download and import the data on weather station using *pandas* `read_csv()` method.

Download Dataset

Dataset Description

Environment Canada Monthly Values for July - 2015

Name in the table	Meaning
Stn_Name	**Station Name**

Lat	**Latitude (North+, degrees)**
Long	**Longitude (West - , degrees)**
Prov	Province
Tm	Mean Temperature (°C)
DwTm	Days without Valid Mean Temperature
D	Mean Temperature difference from Normal (1981-2010) (°C)
Tx	Highest Monthly Maximum Temperature (°C)
DwTx	Days without Valid Maximum Temperature
Tn	Lowest Monthly Minimum Temperature (°C)
DwTn	Days without Valid Minimum Temperature
S	Snowfall (cm)
DwS	Days without Valid Snowfall
S%N	Percent of Normal (1981-2010) Snowfall
P	**Total Precipitation (mm)**
DwP	Days without Valid Precipitation
P%N	Percent of Normal (1981-2010) Precipitation
S_G	Snow on the ground at the end of the month (cm)
Pd	Number of days with Precipitation 1.0 mm or more
BS	Bright Sunshine (hours)
DwBS	Days without Valid Bright Sunshine
BS%	Percent of Normal (1981-2010) Bright Sunshine
HDD	Degree Days below 18 °C
CDD	Degree Days above 18 °C
Stn_No	Climate station identifier (first 3 digits indicate drainage basin, last 4 characters are for sorting alphabetically).
NA	Not Available

Reading the data

In [25]:
```
df = pd.read_csv("weather-stations.csv")

# take a look at the dataset
df.head()
```

Out[25]:

	Stn_Name	Lat	Long	Prov	Tm	DwTm	D	Tx	DwTx	Tn	...	DwP	P%N	S_G	Pd	BS	DwBS	BS%	HDD	CDD	Stn_No
0	CHEMAINUS	48.935	-123.742	BC	8.2	0.0	NaN	13.5	0.0	1.0	...	0.0	NaN	0.0	12.0	NaN	NaN	NaN	273.3	0.0	1011500
1	COWICHAN LAKE FORESTRY	48.824	-124.133	BC	7.0	0.0	3.0	15.0	0.0	-3.0	...	0.0	104.0	0.0	12.0	NaN	NaN	NaN	307.0	0.0	1012040
2	LAKE COWICHAN	48.829	-124.052	BC	6.8	13.0	2.8	16.0	9.0	-2.5	...	9.0	NaN	NaN	11.0	NaN	NaN	NaN	168.1	0.0	1012055
3	DISCOVERY ISLAND	48.425	-123.226	BC	NaN	NaN	NaN	12.5	0.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1012475
4	DUNCAN KELVIN CREEK	48.735	-123.728	BC	7.7	2.0	3.4	14.5	2.0	-1.0	...	2.0	NaN	NaN	11.0	NaN	NaN	NaN	267.7	0.0	1012573

5 rows × 25 columns

In [26]:
```
df.isnull().any()
```

Out[26]:
```
Stn_Name    False
Lat         False
Long        False
Prov        False
Tm           True
DwTm         True
D            True
Tx           True
DwTx         True
Tn           True
DwTn         True
S            True
DwS          True
S%N          True
P            True
DwP          True
P%N          True
S_G          True
Pd           True
BS           True
DwBS         True
BS%          True
HDD          True
CDD          True
Stn_No      False
dtype: bool
```

Cleaning the data

Let's remove rows that don't have any value in the **Tm** field.

In [27]:
```
df = df[pd.notnull(df["Tm"])]
df = df.reset_index(drop=True)
df.head(5)
```

Out[27]:

	Stn_Name	Lat	Long	Prov	Tm	DwTm	D	Tx	DwTx	Tn	...	DwP	P%N	S_G	Pd	BS	DwBS	BS%	HDD	CDD	Stn_No
0	CHEMAINUS	48.935	-123.742	BC	8.2	0.0	NaN	13.5	0.0	1.0	...	0.0	NaN	0.0	12.0	NaN	NaN	NaN	273.3	0.0	1011500
1	COWICHAN LAKE FORESTRY	48.824	-124.133	BC	7.0	0.0	3.0	15.0	0.0	-3.0	...	0.0	104.0	0.0	12.0	NaN	NaN	NaN	307.0	0.0	1012040
2	LAKE COWICHAN	48.829	-124.052	BC	6.8	13.0	2.8	16.0	9.0	-2.5	...	9.0	NaN	NaN	11.0	NaN	NaN	NaN	168.1	0.0	1012055
3	DUNCAN KELVIN CREEK	48.735	-123.728	BC	7.7	2.0	3.4	14.5	2.0	-1.0	...	2.0	NaN	NaN	11.0	NaN	NaN	NaN	267.7	0.0	1012573
4	ESQUIMALT HARBOUR	48.432	-123.439	BC	8.8	0.0	NaN	13.1	0.0	1.9	...	8.0	NaN	NaN	12.0	NaN	NaN	NaN	258.6	0.0	1012710

5 rows × 25 columns

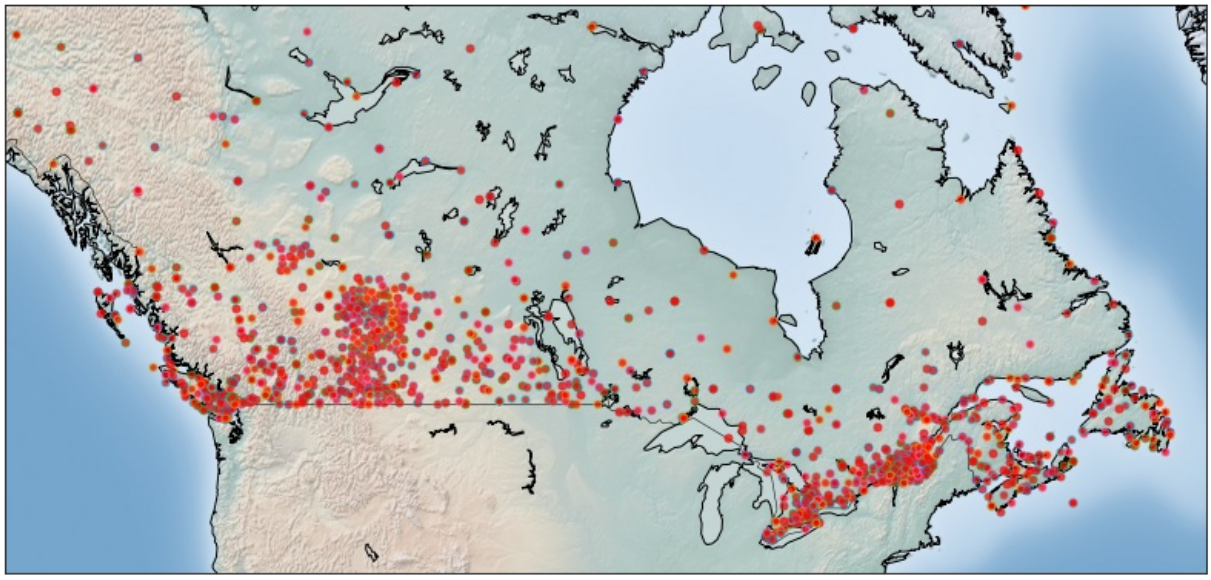
Getting insights from the data

Visualization of stations on map using basemap package. 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.

Please notice that the size of each data points represents the average of maximum temperature for each station in a year.

In [28]:
```
rcParams['figure.figsize'] = (14,10)

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

pdf = 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 (llcrnrlat)
            urcrnrlon=ulon, urcrnrlat=ulat) #max longitude (urcrnrlon) and latitude (urcrnrlat)

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():
#   x,y = my_map(row.Long, row.Lat)
   my_map.plot(row.xm, row.ym,markerfacecolor =([1,0,0]),  marker='o', markersize= 5, alpha = 0.75)
#plt.text(x,y,stn)
plt.show()
```



Clustering of stations based on their location (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.

In [29]:
```
sklearn.utils.check_random_state(1000)
Clus_dataSet = df[['xm','ym']]
Clus_dataSet = np.nan_to_num(Clus_dataSet)
Clus_dataSet = StandardScaler().fit_transform(Clus_dataSet)

# Compute 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_
df["Clus_Db"]=labels

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


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

Out[29]:

	Stn_Name	Tx	Tm	Clus_Db
0	CHEMAINUS	13.5	8.2	0
1	COWICHAN LAKE FORESTRY	15.0	7.0	0
2	LAKE COWICHAN	16.0	6.8	0
3	DUNCAN KELVIN CREEK	14.5	7.7	0
4	ESQUIMALT HARBOUR	13.1	8.8	0

As you can see for outliers, the cluster label is -1

In [30]:
```
set(labels)
```

Out[30]:
```
{-1, 0, 1}
```

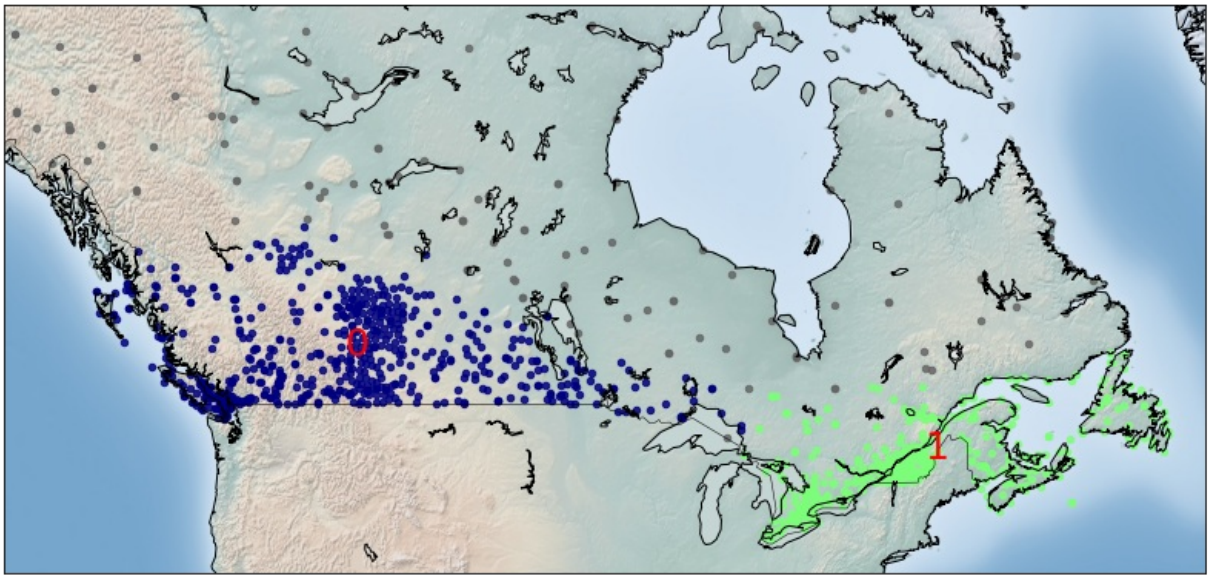
Visualization of clusters based on location

Now, we can visualize the clusters using basemap:

In [31]:
```
rcParams['figure.figsize'] = (14,10)

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.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)))
```

```
Cluster 0, Avg Temp: -5.590609555189461
Cluster 1, Avg Temp: -14.763990825688072
```



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

In this section we re-run DBSCAN, but this time on a 5-dimensional dataset:

In [32]:
```
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
db = DBSCAN(eps=0.3, 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_
df["Clus_Db"]=labels

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


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

Out[32]:

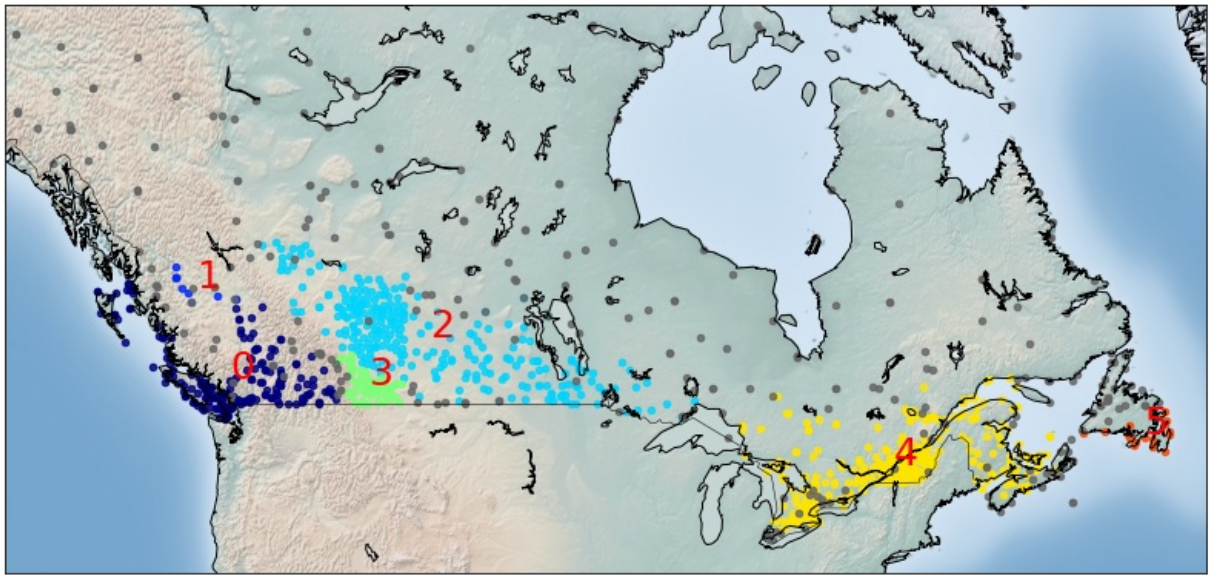
	Stn_Name	Tx	Tm	Clus_Db
0	CHEMAINUS	13.5	8.2	0
1	COWICHAN LAKE FORESTRY	15.0	7.0	0
2	LAKE COWICHAN	16.0	6.8	0
3	DUNCAN KELVIN CREEK	14.5	7.7	0
4	ESQUIMALT HARBOUR	13.1	8.8	0

Visualization of clusters based on location and Temperture

Visualization of clusters based on location and Temperature

In [33]:
```
rcParams['figure.figsize'] = (14,10)

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.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)))
```

```
Cluster 0, Avg Temp: 5.374634146341463
Cluster 1, Avg Temp: -3.2300000000000004
Cluster 2, Avg Temp: -13.68409090909091
Cluster 3, Avg Temp: -4.153703703703704
Cluster 4, Avg Temp: -16.065060240963852
Cluster 5, Avg Temp: -4.70625
```



Thank you

Author

Moazzam Ali
