In [1]:

```python
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
```

In [2]:

```python
data=pd.read_csv('housing.csv')
data
```

Out[2]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY
...	...	...	...	...	...	...	...	...	...	...
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	330.0	1.5603	78100.0	INLAND
20636	-121.21	39.49	18.0	697.0	150.0	356.0	114.0	2.5568	77100.0	INLAND
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	433.0	1.7000	92300.0	INLAND
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	349.0	1.8672	84700.0	INLAND
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	530.0	2.3886	89400.0	INLAND

20640 rows × 10 columns

In [3]:

```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object 
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

In [4]:

```python
data.describe()
```

Out[4]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671	206855.816909
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	115395.615874
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	14999.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	119600.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	179700.000000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	264725.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000

In [5]:

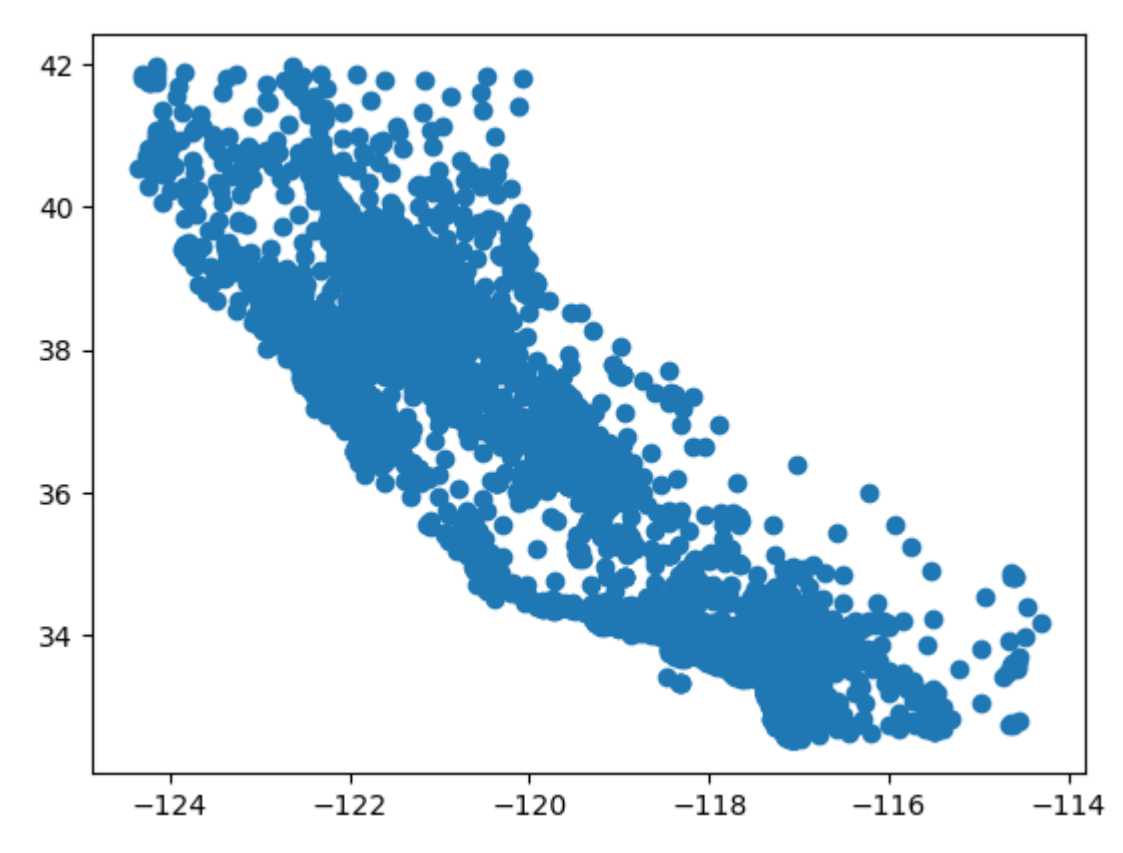
```python
lat_long=data[['latitude','longitude']]
lat,long=data['latitude'],data['longitude']
```

In [6]:

```python
plt.scatter(long,lat)
```

Out[6]:

```
<matplotlib.collections.PathCollection at 0x1f76df3caf0>
```



CLUSTERING WITH DBSCAN

In [7]:

```python
from sklearn.cluster import DBSCAN
X=lat_long.to_numpy()
X.shape
dbscan=DBSCAN(eps=0.2,min_samples=15).fit(X)
dbscan
```

Out[7]:

```
DBSCAN(eps=0.2, min_samples=15)
```

In [8]:

```python
data['cluster']=dbscan.labels_
data['cluster'].value_counts()
#-1:outliers
```

Out[8]:

```
 6    11249
 0     8545
-1      443
 2      109
 1       98
 3       98
 5       27
 7       22
 8       18
 9       16
 4       15
Name: cluster, dtype: int64
```

In [9]:

```python
data
```

Out[9]:

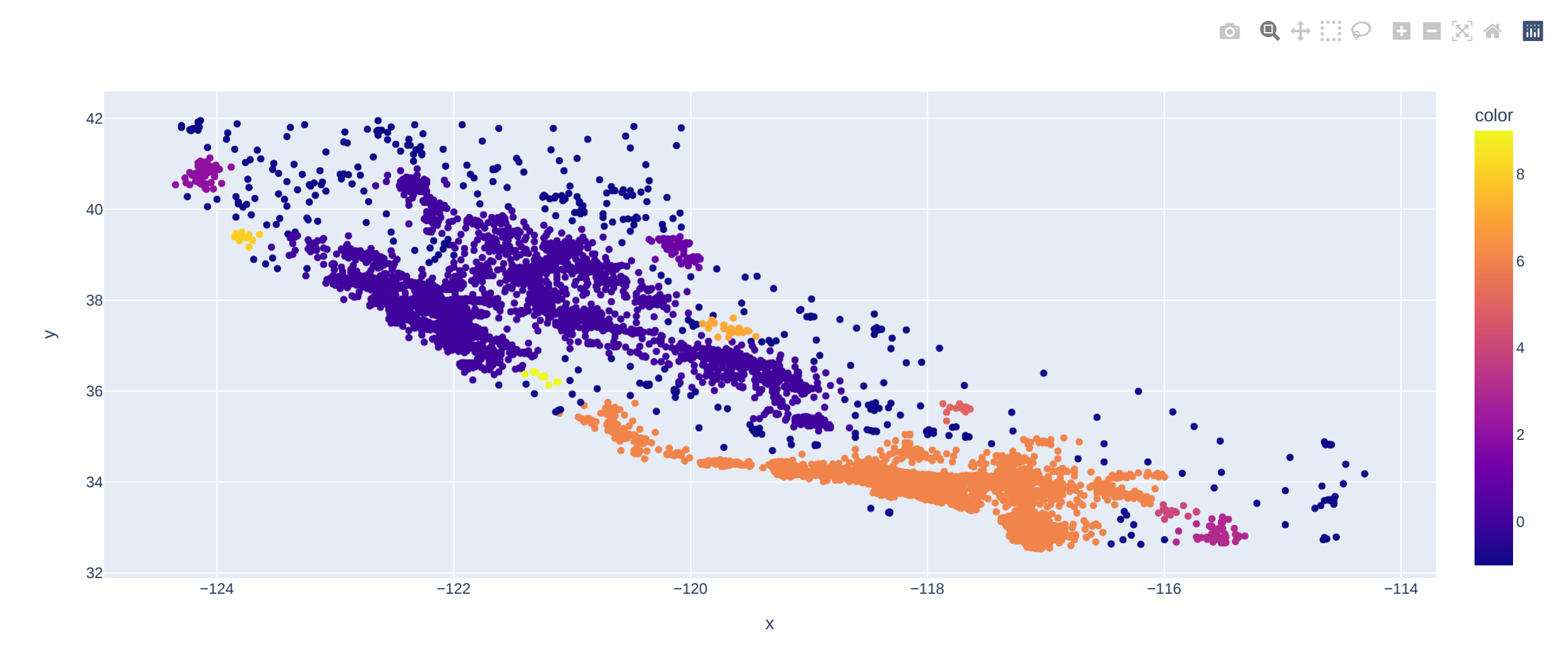
	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity	cluster
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY	0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY	0
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY	0
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY	0
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY	0
...	...	...	...	...	...	...	...	...	...	...	...
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	330.0	1.5603	78100.0	INLAND	0
20636	-121.21	39.49	18.0	697.0	150.0	356.0	114.0	2.5568	77100.0	INLAND	0
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	433.0	1.7000	92300.0	INLAND	0
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	349.0	1.8672	84700.0	INLAND	0
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	530.0	2.3886	89400.0	INLAND	0

20640 rows × 11 columns

In [10]:

```python
import plotly.express as px

fig=px.scatter(x=long,y=lat,color=data['cluster'])
fig.show()
#so we had 1 cluster
```



In [11]:

```python
#evaluate
from sklearn.metrics import silhouette_score as ss
ss(X,data['cluster'])
```

Out[11]:

```
0.21921057255397758
```

In [12]:

```python
#identify epsilon
epsilons=np.linspace(0.01,1,num=15)
epsilons
```

Out[12]:

```
array([0.01      , 0.08071429, 0.15142857, 0.22214286, 0.29285714,
       0.36357143, 0.43428571, 0.505     , 0.57571429, 0.64642857,
       0.71714286, 0.78785714, 0.85857143, 0.92928571, 1.        ])
```

In [13]:

```python
min_samples=np.arange(2,20,step=3)
min_samples
```

Out[13]:

```
array([ 2,  5,  8, 11, 14, 17])
```

In [14]:

```python
import itertools
comb=list(itertools.product(epsilons,min_samples))
comb
```

Out[14]:

```
[(0.01, 2),
 (0.01, 5),
 (0.01, 8),
 (0.01, 11),
 (0.01, 14),
 (0.01, 17),
 (0.08071428571428571, 2),
 (0.08071428571428571, 5),
 (0.08071428571428571, 8),
 (0.08071428571428571, 11),
 (0.08071428571428571, 14),
 (0.08071428571428571, 17),
 (0.15142857142857144, 2),
 (0.15142857142857144, 5),
 (0.15142857142857144, 8),
 (0.15142857142857144, 11),
 (0.15142857142857144, 14),
 (0.15142857142857144, 17),
 (0.22214285714285714, 2),
 (0.22214285714285714, 5),
 (0.22214285714285714, 8),
 (0.22214285714285714, 11),
 (0.22214285714285714, 14),
 (0.22214285714285714, 17),
 (0.29285714285714287, 2),
 (0.29285714285714287, 5),
 (0.29285714285714287, 8),
 (0.29285714285714287, 11),
 (0.29285714285714287, 14),
 (0.29285714285714287, 17),
 (0.3635714285714286, 2),
 (0.3635714285714286, 5),
 (0.3635714285714286, 8),
 (0.3635714285714286, 11),
 (0.3635714285714286, 14),
 (0.3635714285714286, 17),
 (0.4342857142857143, 2),
 (0.4342857142857143, 5),
 (0.4342857142857143, 8),
 (0.4342857142857143, 11),
 (0.4342857142857143, 14),
 (0.4342857142857143, 17),
 (0.505, 2),
 (0.505, 5),
 (0.505, 8),
 (0.505, 11),
 (0.505, 14),
 (0.505, 17),
 (0.5757142857142857, 2),
 (0.5757142857142857, 5),
 (0.5757142857142857, 8),
 (0.5757142857142857, 11),
 (0.5757142857142857, 14),
 (0.5757142857142857, 17),
 (0.6464285714285715, 2),
 (0.6464285714285715, 5),
 (0.6464285714285715, 8),
 (0.6464285714285715, 11),
 (0.6464285714285715, 14),
 (0.6464285714285715, 17),
 (0.7171428571428572, 2),
 (0.7171428571428572, 5),
 (0.7171428571428572, 8),
 (0.7171428571428572, 11),
 (0.7171428571428572, 14),
 (0.7171428571428572, 17),
 (0.7878571428571429, 2),
 (0.7878571428571429, 5),
 (0.7878571428571429, 8),
 (0.7878571428571429, 11),
 (0.7878571428571429, 14),
 (0.7878571428571429, 17),
 (0.8585714285714285, 2),
 (0.8585714285714285, 5),
 (0.8585714285714285, 8),
 (0.8585714285714285, 11),
 (0.8585714285714285, 14),
 (0.8585714285714285, 17),
 (0.9292857142857143, 2),
 (0.9292857142857143, 5),
 (0.9292857142857143, 8),
 (0.9292857142857143, 11),
 (0.9292857142857143, 14),
 (0.9292857142857143, 17),
 (1.0, 2),
 (1.0, 5),
 (1.0, 8),
 (1.0, 11),
 (1.0, 14),
 (1.0, 17)]
```

In [15]:

```python
N=len(comb)
N
```

Out[15]:

```
90
```

In [16]:

```python
def get_scores_and_labels(comb, X):
    scores = []
    all_labels = []

    for i, (eps, num_samples) in enumerate(comb):
        dbscan = DBSCAN(eps=eps, min_samples=num_samples).fit(X)
        labels = dbscan.labels_
        labels_set = set(labels)
        num_clusters = len(labels_set)
        if -1 in labels_set:
            num_clusters -= 1
        if (num_clusters < 2) or (num_clusters > 50):
            scores.append(-10)
            all_labels.append('bad')
            c = (eps, num_samples)
            print(f"combination {c} on iteration {i+1} of {N}, has {num_clusters} clusters. Moving on")
            continue
        scores.append(ss(X, labels))
        all_labels.append(labels)
        print(f"Index: {i}, score: {scores[-1]}, labels: {all_labels[-1]}, Num clusters: {num_clusters}")
    best_index = np.argmax(scores)
    best_parametrs = comb[best_index]
    best_labels = all_labels[best_index]
    best_score = scores[best_index]

    return {
        'best_epsilon': best_parametrs[0],
        'best_min_sample': best_parametrs[1],
        'best_labels': best_labels,
        'best_score': best_score
    }

best_dict = get_scores_and_labels(comb, X)
```

```
combination (0.01, 2) on iteration 1 of 90, has 2391 clusters. Moving on
combination (0.01, 5) on iteration 2 of 90, has 1114 clusters. Moving on
combination (0.01, 8) on iteration 3 of 90, has 543 clusters. Moving on
combination (0.01, 11) on iteration 4 of 90, has 262 clusters. Moving on
combination (0.01, 14) on iteration 5 of 90, has 128 clusters. Moving on
combination (0.01, 17) on iteration 6 of 90, has 65 clusters. Moving on
combination (0.08071428571428571, 2) on iteration 7 of 90, has 124 clusters. Moving on
combination (0.08071428571428571, 5) on iteration 8 of 90, has 72 clusters. Moving on
combination (0.08071428571428571, 8) on iteration 9 of 90, has 58 clusters. Moving on
combination (0.08071428571428571, 11) on iteration 10 of 90, has 54 clusters. Moving on
Index: 10, score: 0.23660136566339013, labels: [ 0  0  0 ... -1 -1 -1], Num clusters: 49
Index: 11, score: 0.2320816966670719, labels: [ 0  0  0 ... -1 -1 -1], Num clusters: 47
combination (0.15142857142857144, 2) on iteration 13 of 90, has 53 clusters. Moving on
Index: 13, score: 0.1285884819753053, labels: [0 0 0 ... 0 0 0], Num clusters: 28
Index: 14, score: 0.17410428128256125, labels: [0 0 0 ... 0 0 0], Num clusters: 25
Index: 15, score: 0.16844787092019656, labels: [0 0 0 ... 0 0 0], Num clusters: 16
Index: 16, score: 0.1945446136991663, labels: [0 0 0 ... 0 0 0], Num clusters: 11
Index: 17, score: 0.18408680371736383, labels: [0 0 0 ... 0 0 0], Num clusters: 12
Index: 18, score: 0.014769889027282565, labels: [0 0 0 ... 0 0 0], Num clusters: 21
Index: 19, score: 0.07994412530600874, labels: [0 0 0 ... 0 0 0], Num clusters: 15
Index: 20, score: 0.32086158405428405, labels: [0 0 0 ... 0 0 0], Num clusters: 15
Index: 21, score: 0.3185951044681995, labels: [0 0 0 ... 0 0 0], Num clusters: 15
Index: 22, score: 0.2784116740302822, labels: [0 0 0 ... 0 0 0], Num clusters: 11
Index: 23, score: 0.25969735738026917, labels: [0 0 0 ... 0 0 0], Num clusters: 8
Index: 24, score: -0.5704726749990954, labels: [0 0 0 ... 0 0 0], Num clusters: 12
Index: 25, score: -0.4712987842954499, labels: [0 0 0 ... 0 0 0], Num clusters: 6
Index: 26, score: -0.052307464729374364, labels: [0 0 0 ... 0 0 0], Num clusters: 4
Index: 27, score: -0.08788219682835671, labels: [0 0 0 ... 0 0 0], Num clusters: 6
Index: 28, score: -0.2836570250976929, labels: [0 0 0 ... 0 0 0], Num clusters: 7
Index: 29, score: -0.33039161693573954, labels: [0 0 0 ... 0 0 0], Num clusters: 4
Index: 30, score: -0.06146628086730184, labels: [0 0 0 ... 0 0 0], Num clusters: 8
Index: 31, score: -0.037315382925954814, labels: [0 0 0 ... 0 0 0], Num clusters: 4
Index: 32, score: -0.03908535945468379, labels: [0 0 0 ... 0 0 0], Num clusters: 3
Index: 33, score: -0.05267112406875294, labels: [0 0 0 ... 0 0 0], Num clusters: 4
Index: 34, score: -0.057657757336925075, labels: [0 0 0 ... 0 0 0], Num clusters: 5
Index: 35, score: 0.26366540150914014, labels: [0 0 0 ... 0 0 0], Num clusters: 2
Index: 36, score: 0.05781738090797295, labels: [0 0 0 ... 0 0 0], Num clusters: 3
Index: 37, score: 0.19425988994286514, labels: [0 0 0 ... 0 0 0], Num clusters: 4
Index: 38, score: 0.23466006468918613, labels: [0 0 0 ... 0 0 0], Num clusters: 2
Index: 39, score: 0.23164622865549284, labels: [0 0 0 ... 0 0 0], Num clusters: 2
Index: 40, score: 0.20986810799158925, labels: [0 0 0 ... 0 0 0], Num clusters: 2
Index: 41, score: -0.04292633289853943, labels: [0 0 0 ... 0 0 0], Num clusters: 3
Index: 42, score: 0.05753198535235856, labels: [0 0 0 ... 0 0 0], Num clusters: 2
combination (0.505, 5) on iteration 44 of 90, has 1 clusters. Moving on
combination (0.505, 8) on iteration 45 of 90, has 1 clusters. Moving on
combination (0.505, 11) on iteration 46 of 90, has 1 clusters. Moving on
combination (0.505, 14) on iteration 47 of 90, has 1 clusters. Moving on
Index: 47, score: -0.020402143102849667, labels: [0 0 0 ... 0 0 0], Num clusters: 3
Index: 48, score: 0.03746247081758117, labels: [0 0 0 ... 0 0 0], Num clusters: 2
combination (0.5757142857142857, 5) on iteration 50 of 90, has 1 clusters. Moving on
combination (0.5757142857142857, 8) on iteration 51 of 90, has 1 clusters. Moving on
combination (0.5757142857142857, 11) on iteration 52 of 90, has 1 clusters. Moving on
combination (0.5757142857142857, 14) on iteration 53 of 90, has 1 clusters. Moving on
combination (0.5757142857142857, 17) on iteration 54 of 90, has 1 clusters. Moving on
Index: 54, score: 0.03746247081758117, labels: [0 0 0 ... 0 0 0], Num clusters: 2
combination (0.6464285714285715, 5) on iteration 56 of 90, has 1 clusters. Moving on
combination (0.6464285714285715, 8) on iteration 57 of 90, has 1 clusters. Moving on
combination (0.6464285714285715, 11) on iteration 58 of 90, has 1 clusters. Moving on
combination (0.6464285714285715, 14) on iteration 59 of 90, has 1 clusters. Moving on
combination (0.6464285714285715, 17) on iteration 60 of 90, has 1 clusters. Moving on
combination (0.7171428571428572, 2) on iteration 61 of 90, has 1 clusters. Moving on
combination (0.7171428571428572, 5) on iteration 62 of 90, has 1 clusters. Moving on
combination (0.7171428571428572, 8) on iteration 63 of 90, has 1 clusters. Moving on
combination (0.7171428571428572, 11) on iteration 64 of 90, has 1 clusters. Moving on
combination (0.7171428571428572, 14) on iteration 65 of 90, has 1 clusters. Moving on
combination (0.7171428571428572, 17) on iteration 66 of 90, has 1 clusters. Moving on
combination (0.7878571428571429, 2) on iteration 67 of 90, has 1 clusters. Moving on
combination (0.7878571428571429, 5) on iteration 68 of 90, has 1 clusters. Moving on
combination (0.7878571428571429, 8) on iteration 69 of 90, has 1 clusters. Moving on
combination (0.7878571428571429, 11) on iteration 70 of 90, has 1 clusters. Moving on
combination (0.7878571428571429, 14) on iteration 71 of 90, has 1 clusters. Moving on
combination (0.7878571428571429, 17) on iteration 72 of 90, has 1 clusters. Moving on
combination (0.8585714285714285, 2) on iteration 73 of 90, has 1 clusters. Moving on
combination (0.8585714285714285, 5) on iteration 74 of 90, has 1 clusters. Moving on
combination (0.8585714285714285, 8) on iteration 75 of 90, has 1 clusters. Moving on
combination (0.8585714285714285, 11) on iteration 76 of 90, has 1 clusters. Moving on
combination (0.8585714285714285, 14) on iteration 77 of 90, has 1 clusters. Moving on
combination (0.8585714285714285, 17) on iteration 78 of 90, has 1 clusters. Moving on
combination (0.9292857142857143, 2) on iteration 79 of 90, has 1 clusters. Moving on
combination (0.9292857142857143, 5) on iteration 80 of 90, has 1 clusters. Moving on
combination (0.9292857142857143, 8) on iteration 81 of 90, has 1 clusters. Moving on
combination (0.9292857142857143, 11) on iteration 82 of 90, has 1 clusters. Moving on
combination (0.9292857142857143, 14) on iteration 83 of 90, has 1 clusters. Moving on
combination (0.9292857142857143, 17) on iteration 84 of 90, has 1 clusters. Moving on
combination (1.0, 2) on iteration 85 of 90, has 1 clusters. Moving on
combination (1.0, 5) on iteration 86 of 90, has 1 clusters. Moving on
combination (1.0, 8) on iteration 87 of 90, has 1 clusters. Moving on
combination (1.0, 11) on iteration 88 of 90, has 1 clusters. Moving on
combination (1.0, 14) on iteration 89 of 90, has 1 clusters. Moving on
combination (1.0, 17) on iteration 90 of 90, has 1 clusters. Moving on
```

In [17]:

```python
best_dict
```

Out[17]:

```
{'best_epsilon': 0.22214285714285714,
 'best_min_sample': 8,
 'best_labels': array([0, 0, 0, ..., 0, 0, 0], dtype=int64),
 'best_score': 0.32086158405428405}
```

In [18]:

```python
data['cluster']=best_dict['best_labels']
data['cluster'].value_counts()
```

Out[18]:

```
 4     11384
 0      8670
-1       166
 3       110
 1       103
 8        37
 13       28
 7        27
 10       19
 14       19
 9        15
 12       14
 2        13
 6        13
 5        11
 11       11
Name: cluster, dtype: int64
```

In [19]:

```python
import plotly.express as px

fig=px.scatter(x=long,y=lat,color=data['cluster'])
fig.show()
```