Clustering

K-Means

In [1]:

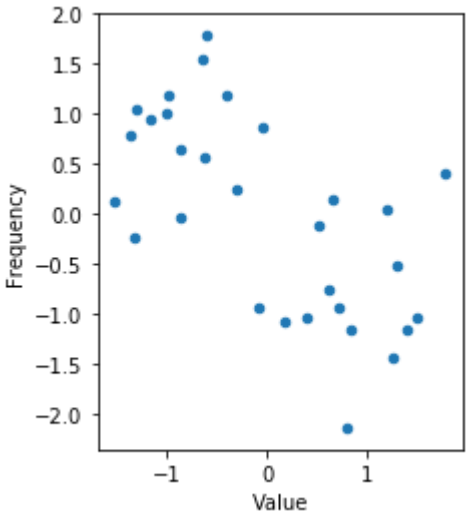
```
# Import numpy and pandas packages and K-Means algorithm
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

In [2]:

```
# Load data
df = pd.read_csv('clustering_example.csv')
```

In [3]:

```
# Plot data
df.plot.scatter(x='Value', y='Frequency').set_aspect('equal');
```



In [4]:

```
initial_centers = np.array([[0.2, 0.1],
                            [-0.1, -0.1]])
initial_centers
```

Out[4]:

```
array([[ 0.2,  0.1],
       [-0.1, -0.1]])
```

In [5]:

```
# Apply K-Means algorithm with predefined initial cluster centers
kmeans = KMeans(n_clusters=2,
                init=initial_centers,
                n_init=1,
                random_state=12345)
kmeans.fit(df.values[:, 1:])
```

Out[5]:

```
KMeans(algorithm='auto', copy_x=True,
    init=array([[ 0.2,  0.1],
       [-0.1, -0.1]]), max_iter=300,
    n_clusters=2, n_init=1, n_jobs=1, precompute_distances='auto',
    random_state=12345, tol=0.0001, verbose=0)
```

In [6]:

```
# Display cluster memberships
kmeans.labels_
```

Out[6]:

```
array([0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0,
       1, 0, 1, 1, 0, 0, 0, 1])
```

In [7]:

```
# Get objective function value
kmeans.inertia_
```

Out[7]:

```
17.297133333333335
```

In [8]:

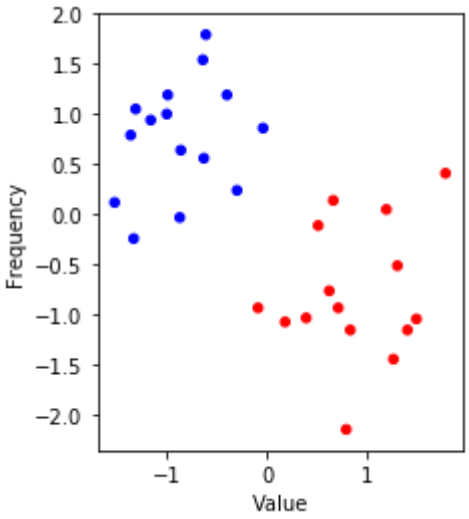
```
# Get final cluster centers
kmeans.cluster_centers_
```

Out[8]:

```
array([[ 0.868     , -0.77733333],
       [-0.868     ,  0.776     ]])
```

In [9]:

```
# Plot result
cols = np.array(['red', 'blue'])
ax = df.plot.scatter(x='Value', y='Frequency', c=cols[kmeans.labels_])
ax.set_aspect('equal');
```



In [10]:

```
# Apply K-Means algorithm without predefined initial cluster centers
kmeans = KMeans(n_clusters=2, n_init=1, random_state=123)
kmeans.fit(df.values[:, 1:])
```

Out[10]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=2, n_init=1, n_jobs=1, precompute_distances='auto',
    random_state=123, tol=0.0001, verbose=0)
```

In [11]:

```
# Display cluster memberships
kmeans.labels_
```

Out[11]:

```
array([1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1,
       0, 1, 0, 0, 1, 1, 1, 0])
```

In [12]:

```
# Apply K-Means algorithm with multistart
kmeans = KMeans(n_clusters=2, n_init=100, random_state=123)
kmeans.fit(df.values[:, 1:])
```

Out[12]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=2, n_init=100, n_jobs=1, precompute_distances='auto',
    random_state=123, tol=0.0001, verbose=0)
```

Mini Batch K-Means

In [13]:

```
# Import Mini Batch K-Means algorithm
from sklearn.cluster import MiniBatchKMeans
```

In [14]:

```
# Apply Mini Batch K-Means algorithm with multistart
mb_kmeans = MiniBatchKMeans(n_clusters=2, n_init=100, random_state=123)
mb_kmeans.fit(df.values[:, 1:])
```

Out[14]:

```
MiniBatchKMeans(batch_size=100, compute_labels=True, init='k-means++',
        init_size=None, max_iter=100, max_no_improvement=10, n_clusters=2,
        n_init=100, random_state=123, reassignment_ratio=0.01, tol=0.0,
        verbose=0)
```

In [15]:

```
# Display cluster memberships
mb_kmeans.labels_
```

Out[15]:

```
array([1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1,
       0, 1, 0, 0, 1, 1, 1, 0])
```

Comparing K-Means and Mini Batch K-Means on large data set

In [16]:

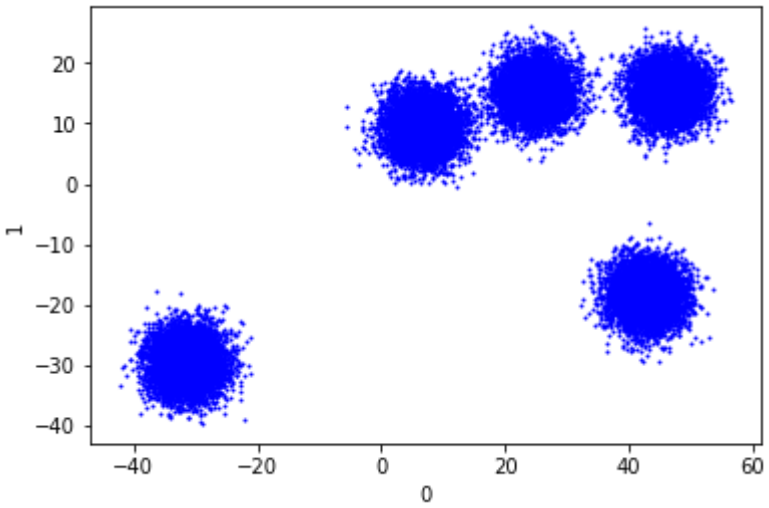
```
# Import packages
import time
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
```

In [17]:

```
# Create a large data set
X, y = make_blobs(n_samples=30000,
                  n_features=2,
                  centers=5,
                  random_state=12345,
                  cluster_std=3,
                  center_box=(-50, 50))
df = pd.DataFrame(X)
```

In [18]:

```
# Visualize data set
df.plot.scatter(x=0, y=1, c='blue', s=1);
```



In [19]:

```
# Apply K-Means algorithm
tic = time.clock()
kmeans = KMeans(n_clusters=5, n_init=100, random_state=12345)
kmeans.fit(df.values)
print('Running time K-Means: {:.2f} seconds'.format(time.clock() - tic))
print('Objective function value: {:,.2f}'.format(kmeans.inertia_))
```

```
Running time K-Means: 1.76 seconds
Objective function value: 541,278.32
```

In [20]:

```
# Apply Mini Batch K-Means algorithm
tic = time.clock()
mb_kmeans = MiniBatchKMeans(n_clusters=5, n_init=100, random_state=12345)
mb_kmeans.fit(df.values)
print('Running time Mini Batch K-Means: {:.2f} seconds'.format(time.clock() - tic))
print('Objective function value: {:,.2f}'.format(mb_kmeans.inertia_))
```

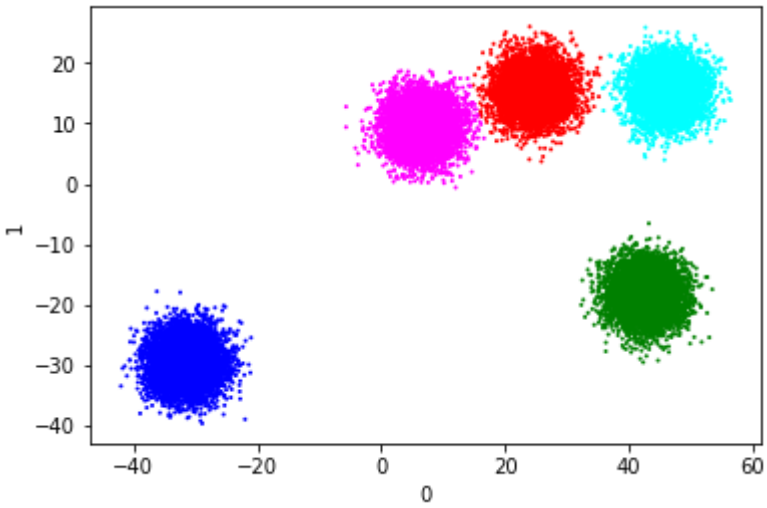
```
Running time Mini Batch K-Means: 0.21 seconds
Objective function value: 542,364.73
```

In [21]:

```
# Define colors
cols = np.array(['red', 'blue', 'green', 'cyan', 'magenta', 'yellow'])
```

In [22]:

```
# Visualize K-Means clustering result
df.plot.scatter(x=0, y=1, c=cols[kmeans.labels_], s=1);
```



In [23]:

```
# Visualize Mini Batch K-Means clustering result
df.plot.scatter(x=0, y=1, c=cols[mb_kmeans.labels_], s=1);
```