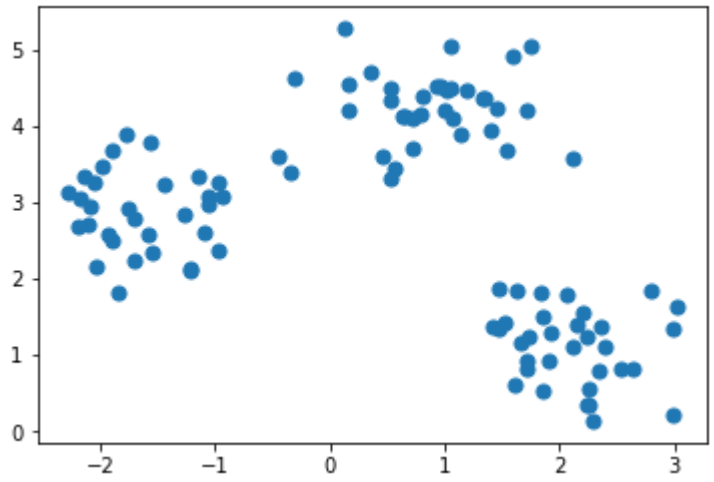
In [2]:
```
import matplotlib.pyplot as plt
```

In [3]:
```
from sklearn.datasets.samples_generator import make_blobs
```

In [4]:
```
X, y = make_blobs(n_samples=100,n_features=2,centers=3, cluster_std=0.5,shuffl
e=True, random_state=0)
```

In [5]:
```
plt.scatter(X[:, 0], X[:, 1], s=50)
plt.show()
```



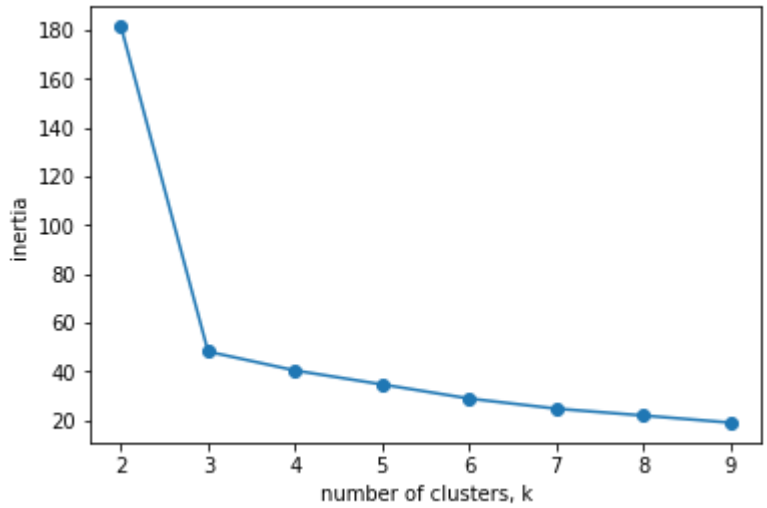
In [6]:
```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)
```

Out[6]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=2, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

In [7]:
```
y_kmeans = kmeans.predict(X)
```

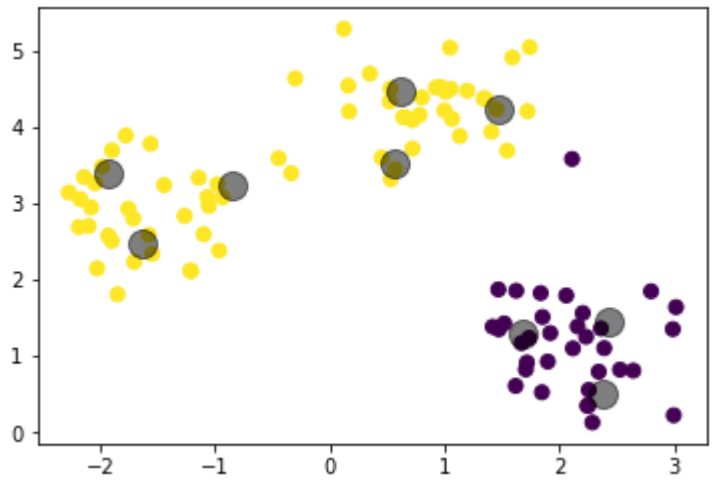
In [8]:
```
inertia = kmeans.inertia_
labels = kmeans.labels_
centroids = kmeans.cluster_centers_
```

In [9]:
```
from matplotlib import pyplot
import numpy as np

ks = range(2,10)
inertias = []

for k in ks:
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(X)
    inertias.append(kmeans.inertia_)

plt.plot(ks, inertias, '-o')
plt.xlabel('number of clusters, k')
plt.ylabel('inertia')
plt.xticks(ks)
plt.show()
```



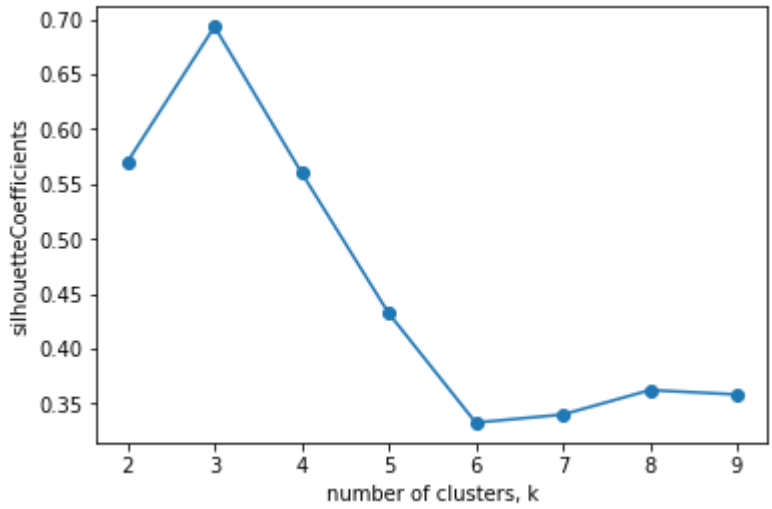
In [10]:
```
#  Determine the best K from the plot => K = 9
```

In [11]:
```
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5)
plt.show()
```



In [12]:
```
from matplotlib import pyplot
from sklearn.metrics import silhouette_samples, silhouette_score
import numpy as np

ks = range(2,10)
silhouettecoefficients = []
i=2
for k in ks:
    clusterer = KMeans(n_clusters=k, random_state=10)
    cluster_labels = clusterer.fit_predict(X)
    silhouette_avg = silhouette_score(X, cluster_labels)
    print("The average silhouette_score k =",i,"is :", silhouette_avg)
    silhouettecoefficients.append(silhouette_avg)
    i=i+1

plt.plot(ks, silhouettecoefficients, '-o')
plt.xlabel('number of clusters, k')
plt.ylabel('silhouetteCoefficients')
plt.xticks(ks)
plt.show()
```

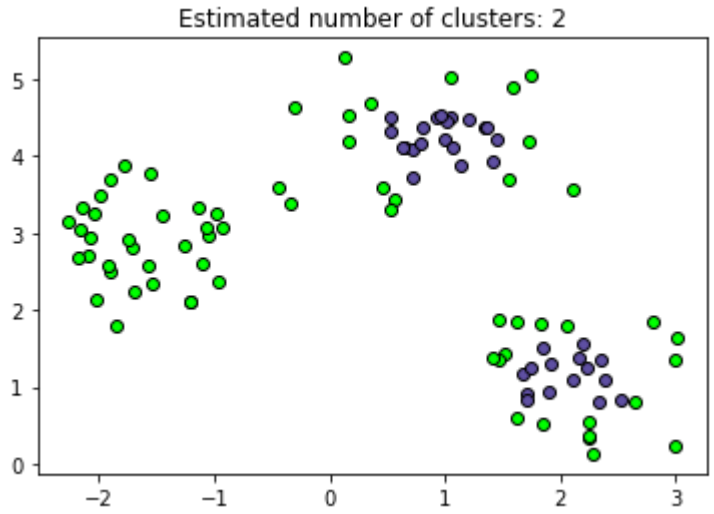
```
The average silhouette_score k = 2 is : 0.570297867185
The average silhouette_score k = 3 is : 0.693301679732
The average silhouette_score k = 4 is : 0.560165779005
The average silhouette_score k = 5 is : 0.432358320141
The average silhouette_score k = 6 is : 0.332824343812
The average silhouette_score k = 7 is : 0.340196074708
The average silhouette_score k = 8 is : 0.362423801125
The average silhouette_score k = 9 is : 0.358478746138
```



In [13]:
```
# Determine the K corresponding to the lowest values: K = 6
```

In [14]:
```
from sklearn.cluster import DBSCAN
```

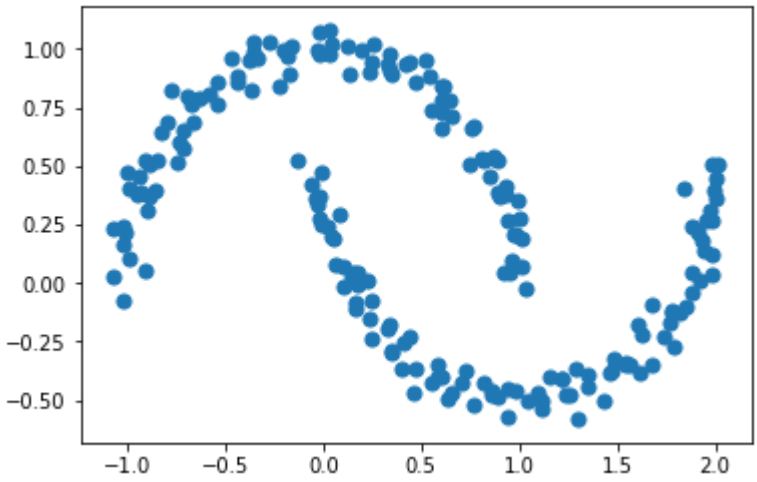
In [15]:
```
dbsc = DBSCAN(eps = .5, min_samples = 15).fit(X)
labels = dbsc.labels_
core_samples = np.zeros_like(labels, dtype = bool)
core_samples[dbsc.core_sample_indices_] = True
core_samples_mask = np.zeros_like(dbsc.labels_, dtype=bool)

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

unique_labels = set(labels)
colors = [plt.cm.Spectral(each)
          for each in np.linspace(1, 1, len(unique_labels))]
for k, col in zip(unique_labels, colors):
    if k == -1:
        col = [0, 1, 0, 1]

    class_member_mask = (labels == k)

    xy = X[class_member_mask & core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
             markeredgecolor='k', markersize=14)

    xy = X[class_member_mask & ~core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
             markeredgecolor='k', markersize=6)

plt.title('Estimated number of clusters: %d' % n_clusters_)
plt.show()
```



In [17]:
```
from sklearn.datasets import make_moons
```

In [18]:
```
X, y = make_moons(n_samples=200,noise=0.05, random_state=0)
```

In [19]:
```
plt.scatter(X[:, 0], X[:, 1], s=50)
plt.show()
```



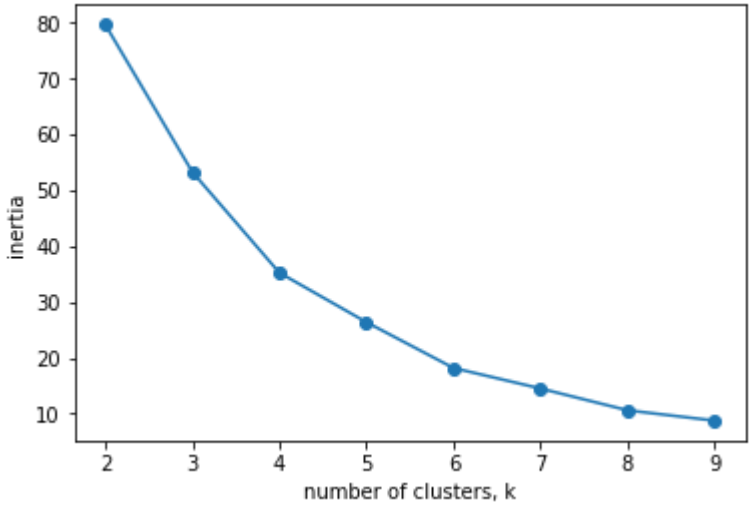
In [20]:
```
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)
```

Out[20]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=2, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

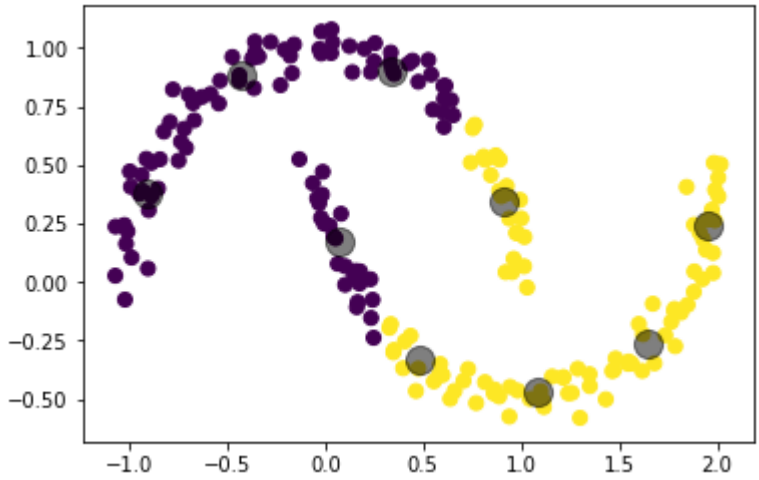
In [21]:
```
y_kmeans = kmeans.predict(X)
```

In [22]:
```
inertia = kmeans.inertia_
labels = kmeans.labels_
centroids = kmeans.cluster_centers_
```

In [23]:
```
ks = range(2,10)
inertias = []

for k in ks:
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(X)
    inertias.append(kmeans.inertia_)

plt.plot(ks, inertias, '-o')
plt.xlabel('number of clusters, k')
plt.ylabel('inertia')
plt.xticks(ks)
plt.show()
```

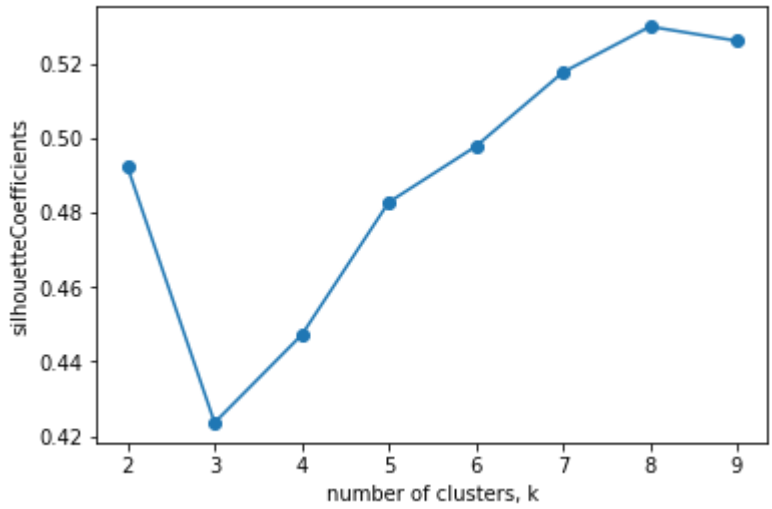


In [24]:
```
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5)
plt.show()
```

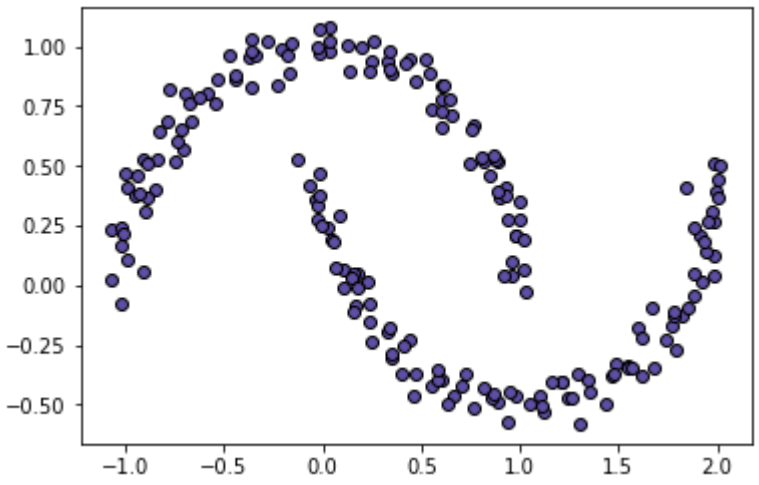


In [25]:
```
from matplotlib import pyplot
from sklearn.metrics import silhouette_samples, silhouette_score
import numpy as np

ks = range(2,10)
silhouettecoefficients = []
i=2
for k in ks:
    clusterer = KMeans(n_clusters=k, random_state=10)
    cluster_labels = clusterer.fit_predict(X)
    silhouette_avg = silhouette_score(X, cluster_labels)
    print("The average silhouette_score k =",i,"is :", silhouette_avg)
    silhouettecoefficients.append(silhouette_avg)
    i=i+1

plt.plot(ks, silhouettecoefficients, '-o')
plt.xlabel('number of clusters, k')
plt.ylabel('silhouetteCoefficients')
plt.xticks(ks)
plt.show()
```

```
The average silhouette_score k = 2 is : 0.492156482683
The average silhouette_score k = 3 is : 0.42358855872
The average silhouette_score k = 4 is : 0.447092578216
The average silhouette_score k = 5 is : 0.482688809906
The average silhouette_score k = 6 is : 0.497679369395
The average silhouette_score k = 7 is : 0.517534557056
The average silhouette_score k = 8 is : 0.529816616517
The average silhouette_score k = 9 is : 0.525992364999
```



In [26]:
```
dbsc = DBSCAN(eps = .5, min_samples = 15).fit(X)
labels = dbsc.labels_
core_samples = np.zeros_like(labels, dtype = bool)
core_samples[dbsc.core_sample_indices_] = True
core_samples_mask = np.zeros_like(dbsc.labels_, dtype=bool)

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

unique_labels = set(labels)
colors = [plt.cm.Spectral(each)
          for each in np.linspace(1, 1, len(unique_labels))]
for k, col in zip(unique_labels, colors):
    if k == -1:
        col = [0, 1, 0, 1]

    class_member_mask = (labels == k)

    xy = X[class_member_mask & core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
             markeredgecolor='k', markersize=14)

    xy = X[class_member_mask & ~core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
             markeredgecolor='k', markersize=6)

plt.show()
```



In [27]:
```
# Run KMeans and DBSCAN and determine which algorithm works better on this dat
aset (by identifying two moon shaped clusters
# DBSCAN turns out to be better
```