DUT STID, Université de la Côte d'Azur

Data mining

Partitionnement (Clustering)

Prof. Dario Malchiodi

UNIVERSITÀ DEGLI STUDI DI MILANO
DIPARTIMENTO DI INFORMATICA

UNIVERSITÉ CÔTE D'AZUR

In [1]:
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
iris = sns.load_dataset('iris')
iris.head()
```

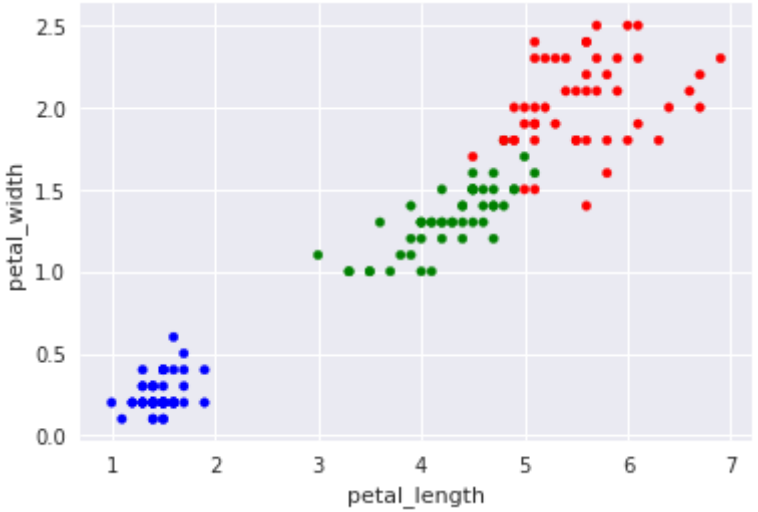
Out[1]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [2]:
```
X_iris = iris.drop(['sepal_length', 'sepal_width', 'species'], axis=1)
y_iris = iris['species']
```

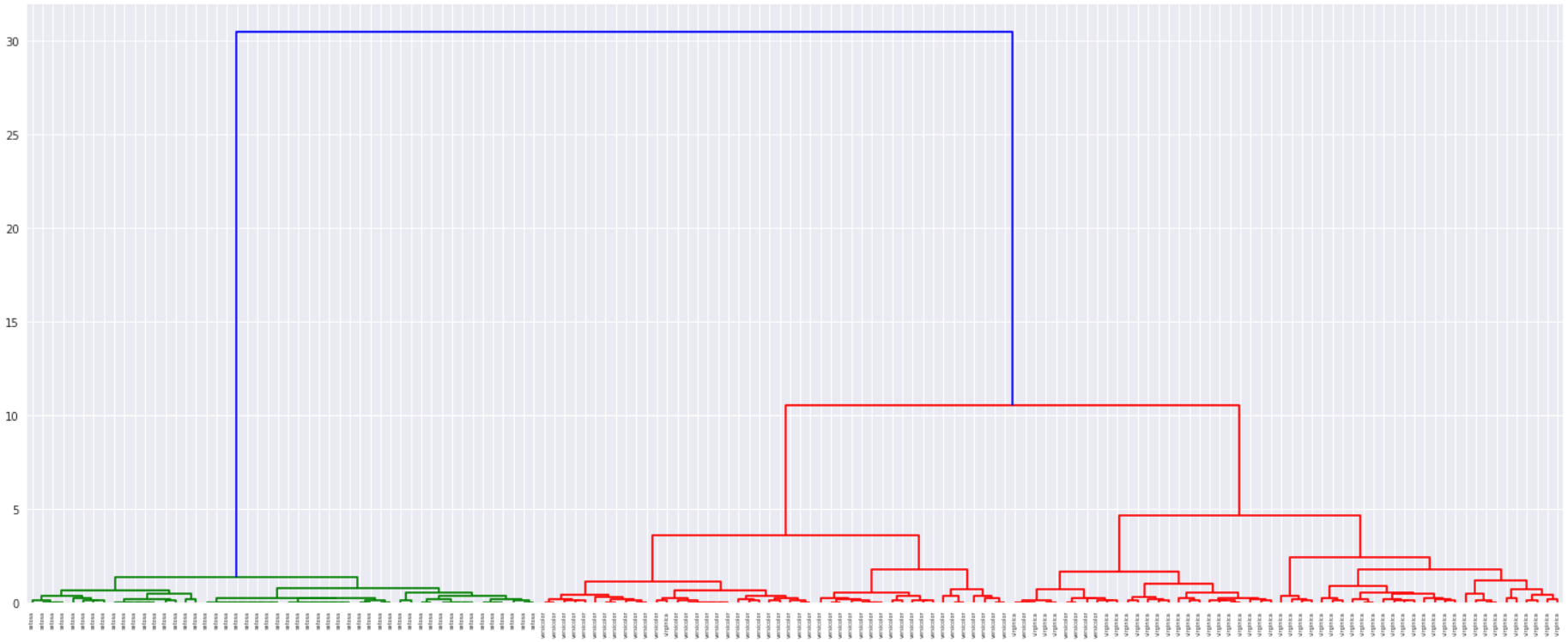
In [3]:
```
def get_color(p):
    cols = {'setosa': 'b', 'virginica': 'r', 'versicolor': 'g'}
    return cols[p]

colors = [get_color(p) for p in y_iris]
```

In [4]:
```
X_iris.plot.scatter('petal_length', 'petal_width', c=colors)
plt.show()
```



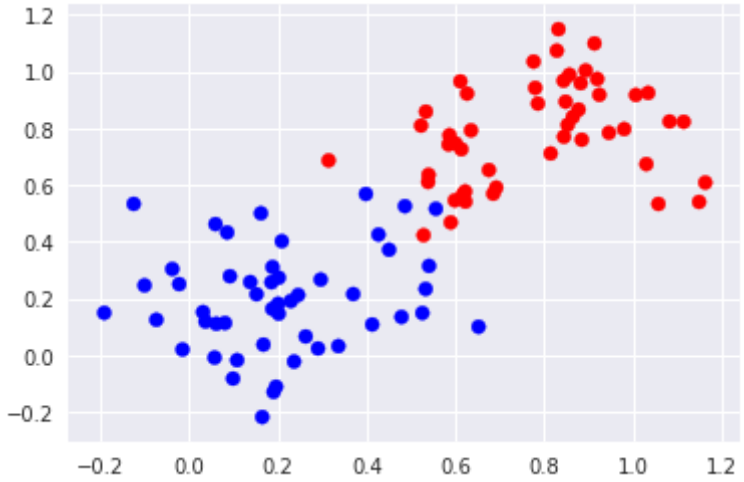
In [5]:
```
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
```

In [6]:
```
Z = linkage(X_iris, 'ward')
fig = plt.figure(figsize=(25, 10))
dn = dendrogram(Z, labels=y_iris.values)
```

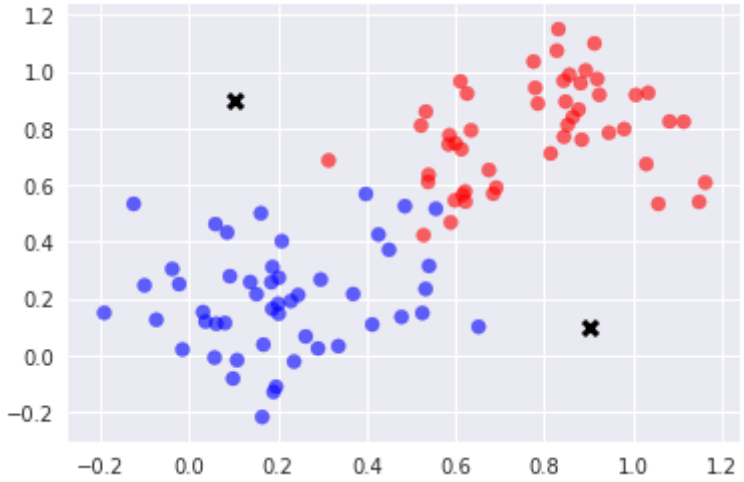


In [ ]:

In [7]:
```
from sklearn.datasets import make_blobs

X, y = make_blobs(n_samples = 100, n_features=2, centers=[(.2, .2), (.8, .8)], cluster_std=0.2)
colors = ['r' if y_el else 'b' for y_el in y]
plt.scatter(X[:,0], X[:,1], c=colors)
plt.show()
```

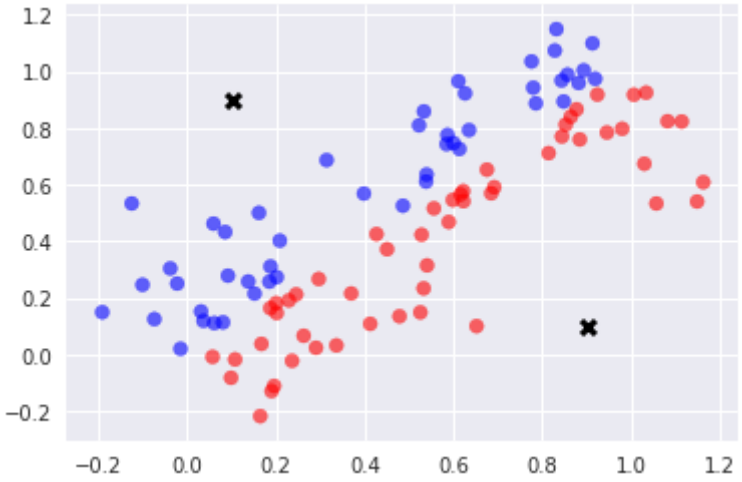


In [8]:
```
import numpy as np

centers = np.array([(0.1, 0.9), (0.9, 0.1)])
plt.scatter(X[:,0], X[:,1], c=colors, alpha=0.6)
plt.scatter(centers[:, 0], centers[:, 1], c='k', marker='x', linewidth=3)
plt.show()
```



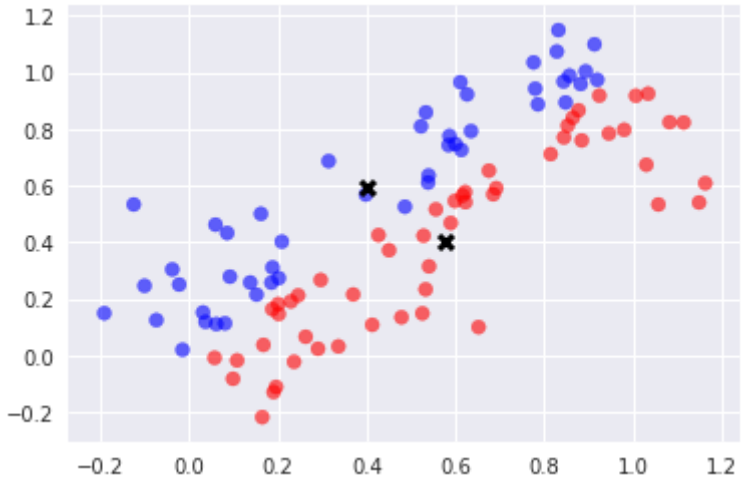
In [9]:
```
assigned_clusters = np.array([np.argmin([np.linalg.norm(x - c)  for c in centers]) for x in X])
```

In [10]:
```
colors = ['r' if y_el else 'b' for y_el in assigned_clusters]
plt.scatter(X[:,0], X[:,1], c=colors, alpha=0.6)
plt.scatter(centers[:, 0], centers[:, 1], c='k', marker='x', linewidth=3)
plt.show()
```



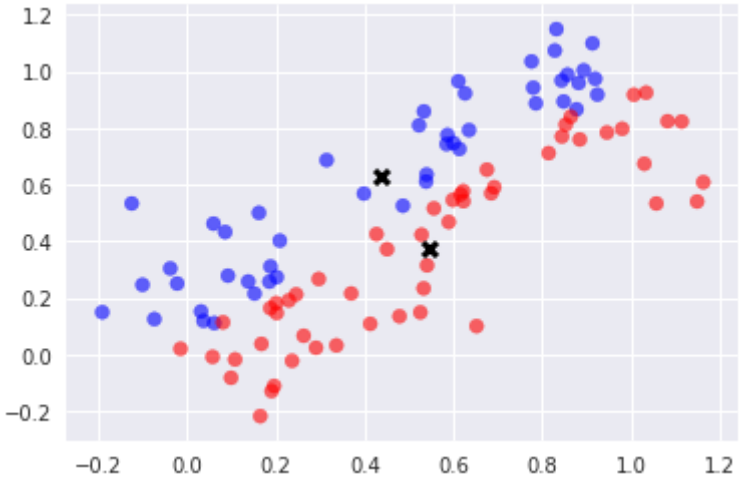
In [11]:
```
centers = np.array([np.average(X[assigned_clusters == 0], axis=0),
                    np.average(X[assigned_clusters == 1], axis=0)])
```

In [12]:
```
plt.scatter(X[:,0], X[:,1], c=colors, alpha=0.6)
plt.scatter(centers[:, 0], centers[:, 1], c='k', marker='x', linewidth=3)
plt.show()
```

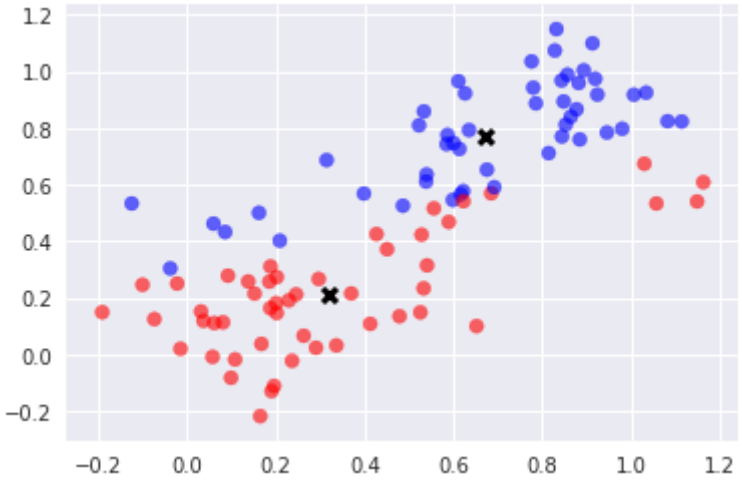


In [13]:
```
def kmeans_step(centers, plot=True):
    assigned_clusters = np.array([np.argmin([np.linalg.norm(x - c)  for c in centers]) for x in X])
    centers = np.array([np.average(X[assigned_clusters == 0], axis=0),
                        np.average(X[assigned_clusters == 1], axis=0)])
    if plot:
        colors = ['r' if y_el else 'b' for y_el in assigned_clusters]
        plt.scatter(X[:,0], X[:,1], c=colors, alpha=0.6)
        plt.scatter(centers[:, 0], centers[:, 1], c='k', marker='x', linewidth=3)
    return centers
```

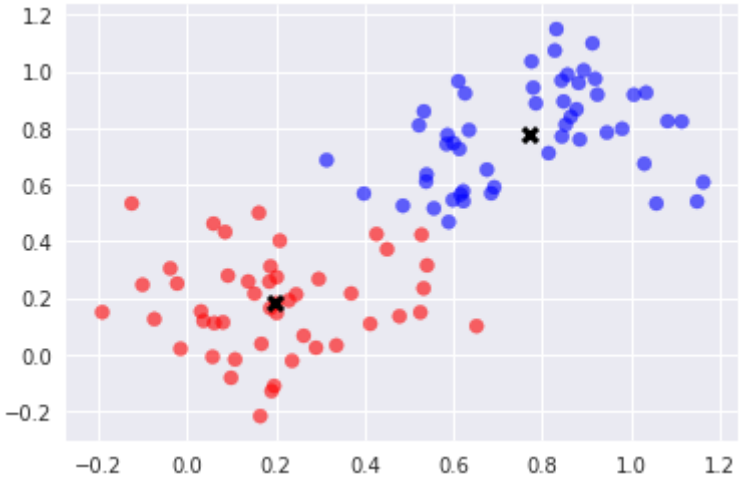
In [14]:
```
centers = kmeans_step(centers)
plt.show()
```



In [15]:
```
centers = kmeans_step(centers)
plt.show()
```



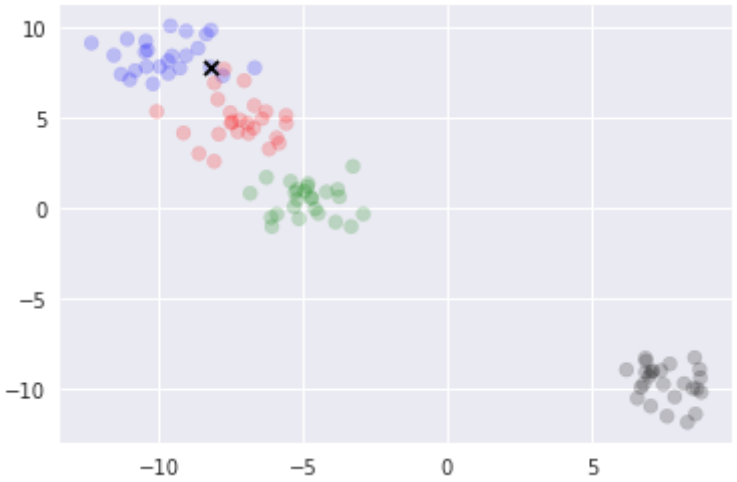
In [16]:
```
centers = kmeans_step(centers)
plt.show()
```



In [ ]:

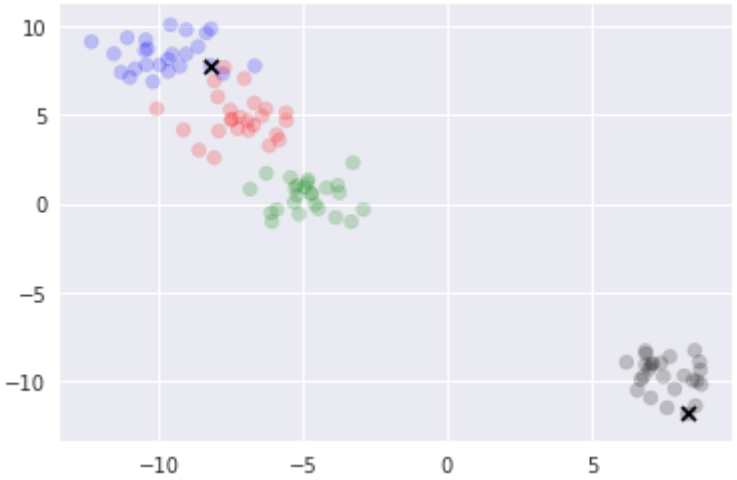
In [17]:
```
X, y = make_blobs(n_samples = 100, n_features=2, centers=4, cluster_std=1, random_state=12)
colors = [['r', 'g', 'b', 'k'][y_el] for y_el in y]
plt.scatter(X[:,0], X[:,1], c=colors, alpha=0.6)
plt.show()
```

In [18]:
```
first_index = np.argmin([np.linalg.norm(x - np.array([-8, 8])) for x in X])
found_centers = np.array([X[first_index]])
plt.scatter(X[:,0], X[:,1], c=colors, alpha=0.2)
plt.scatter(found_centers[:, 0], found_centers[:, 1], marker='x', c='k', linewidths=4)
plt.show()
```

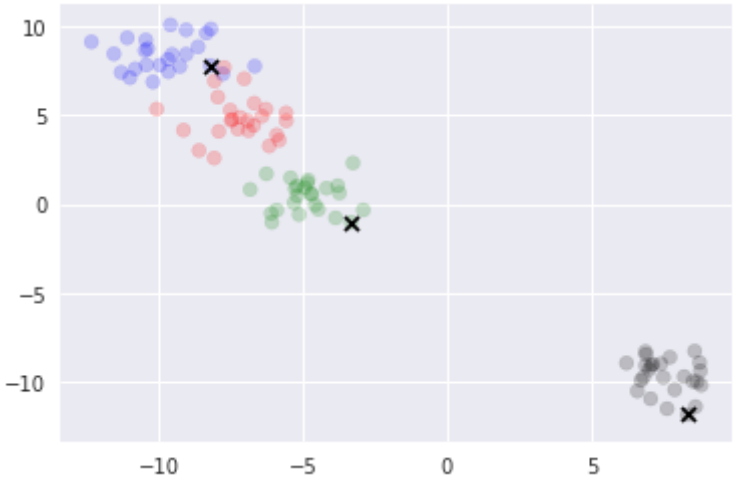


In [19]:
```
second_index = np.argmax([min([np.linalg.norm(x - c) for c in found_centers]) for x in X])
```

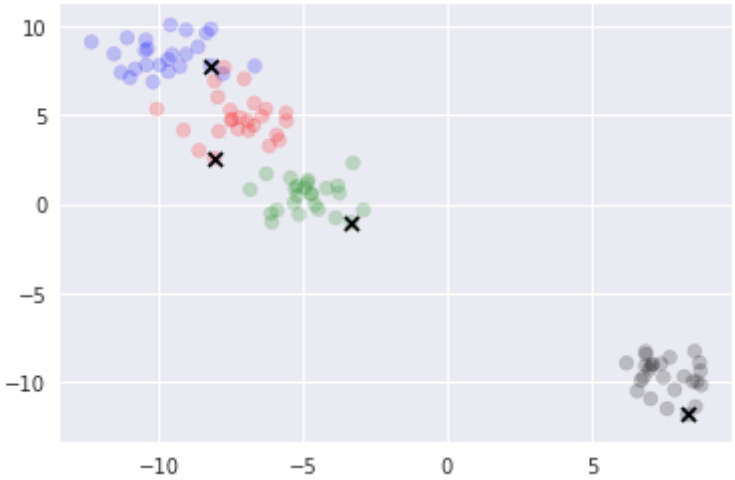
In [20]:
```
found_centers = np.vstack([found_centers, X[second_index]])
plt.scatter(X[:,0], X[:,1], c=colors, alpha=0.2)
plt.scatter(found_centers[:, 0], found_centers[:, 1], marker='x', c='k', linewidths=4)
plt.show()
```



In [21]:
```
third_index = np.argmax([min([np.linalg.norm(x - c) for c in found_centers]) for x in X])

found_centers = np.vstack([found_centers, X[third_index]])
plt.scatter(X[:,0], X[:,1], c=colors, alpha=0.2)
plt.scatter(found_centers[:, 0], found_centers[:, 1], marker='x', c='k', linewidths=4)
plt.show()
```



In [22]:
```
fourth_index = np.argmax([min([np.linalg.norm(x - c) for c in found_centers]) for x in X])

found_centers = np.vstack([found_centers, X[fourth_index]])
plt.scatter(X[:,0], X[:,1], c=colors, alpha=0.2)
plt.scatter(found_centers[:, 0], found_centers[:, 1], marker='x', c='k', linewidths=4)
plt.show()
```



In [23]:
```
def kmeans_step(centers, plot=True):
    assigned_clusters = np.array([np.argmin([np.linalg.norm(x - c)  for c in centers]) for x in X])
    centers = np.array([np.average(X[assigned_clusters == i], axis=0) for i in range(4)])
    if plot:
        colors = [['r', 'g', 'b', 'k'][y_el] for y_el in assigned_clusters]
        plt.scatter(X[:,0], X[:,1], c=colors, alpha=0.6)
        plt.scatter(centers[:, 0], centers[:, 1], c='k', marker='x', linewidth=3)
    return centers
```

In [24]:
```
new_centers = kmeans_step(found_centers)
plt.show()
print(found_centers)
print(new_centers)
print(np.linalg.norm(found_centers-new_centers))
```


```
[[ -8.20259984   7.78760407]
 [  8.32880403 -11.81957591]
 [ -3.32597193  -1.01290846]
 [ -8.07110339   2.58766045]]
[[-9.23697021  7.82985759]
 [ 7.62646261 -9.66838641]
 [-4.64798564  0.31868486]
 [-6.87708672  3.75444649]]
3.5356093714149694
```

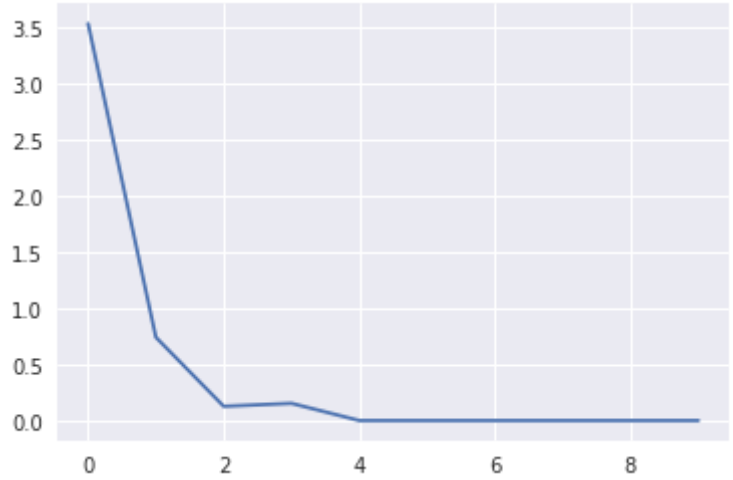
In [25]:
```
curr_centers = new_centers
new_centers = kmeans_step(curr_centers)
plt.show()
print(np.linalg.norm(curr_centers-new_centers))
```


```
0.74174588053625
```

In [26]:
```
curr_centers = new_centers
new_centers = kmeans_step(curr_centers)
plt.show()
print(np.linalg.norm(curr_centers-new_centers))
```


```
0.12686073898808137
```

In [27]:
```
ts = range(10)
errors = []
curr_centers = found_centers
for t in ts:
    new_centers = kmeans_step(curr_centers, plot=False)
    errors.append(np.linalg.norm(curr_centers-new_centers))
    curr_centers= new_centers

plt.plot(ts, errors)
plt.show()
```



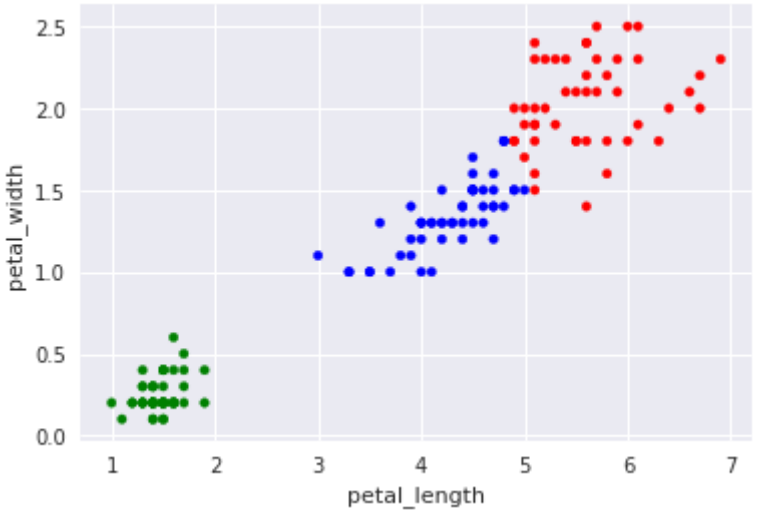
In [28]:
```
from sklearn.cluster import KMeans

y_pred = KMeans(n_clusters=3).fit_predict(X_iris)
```

In [29]:
```
y_pred
```

Out[29]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int32)
```

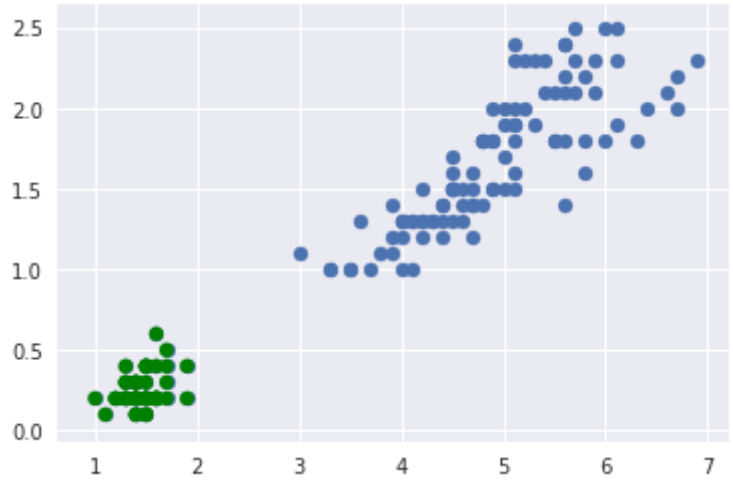
In [30]:
```
def get_col(p):
    cols = {0: 'r', 1: 'g', 2: 'b'}
    return cols[p]

colors = [get_col(y) for y in y_pred]
```

In [31]:
```
X_iris.plot.scatter('petal_length', 'petal_width', c=colors)
plt.show()
```



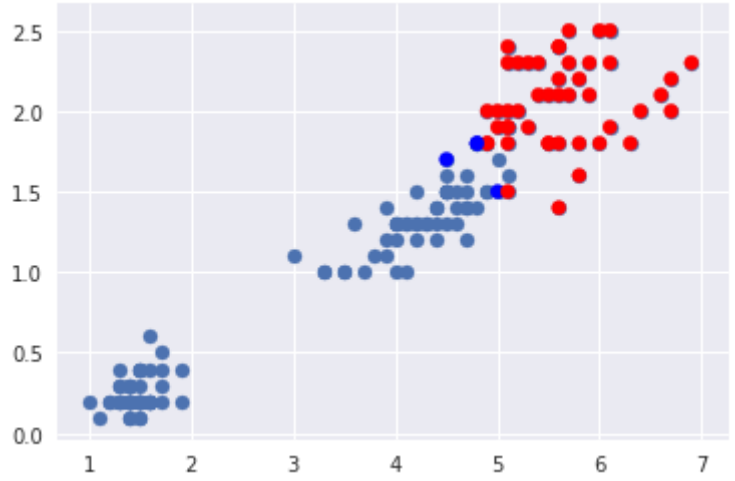
In [32]:
```
def show_clustering(left, right):
    plt.scatter(X_iris['petal_length'],
                X_iris['petal_width'])
    plt.scatter(X_iris['petal_length'][left:right],
                X_iris['petal_width'][left:right],
                c=colors[left:right])
    plt.show()
```

In [33]:
```
show_clustering(0, 50)
```



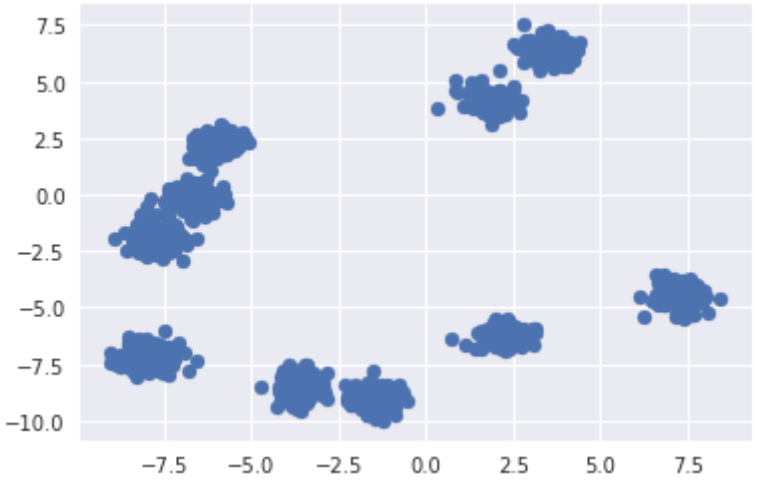
In [34]:
```
show_clustering(50, 100)
```

In [35]:
```
show_clustering(100, 150)
```



In [36]:
```
X, y = make_blobs(n_samples = 1000, n_features=2, centers=10, cluster_std=0.4)
```

In [37]:
```
plt.scatter(X[:, 0], X[:, 1])
plt.show()
```

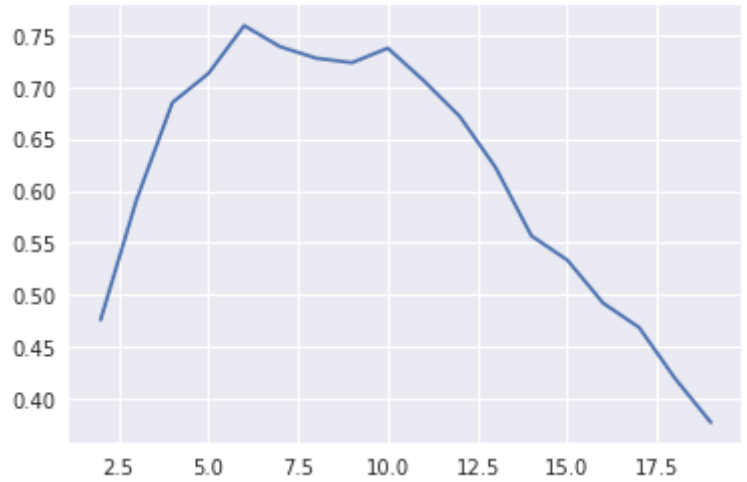


In [38]:
```
from sklearn import metrics

kmeans_model = KMeans(n_clusters=3, random_state=1).fit(X)

labels = kmeans_model.labels_
metrics.silhouette_score(X, labels, metric='euclidean')
```

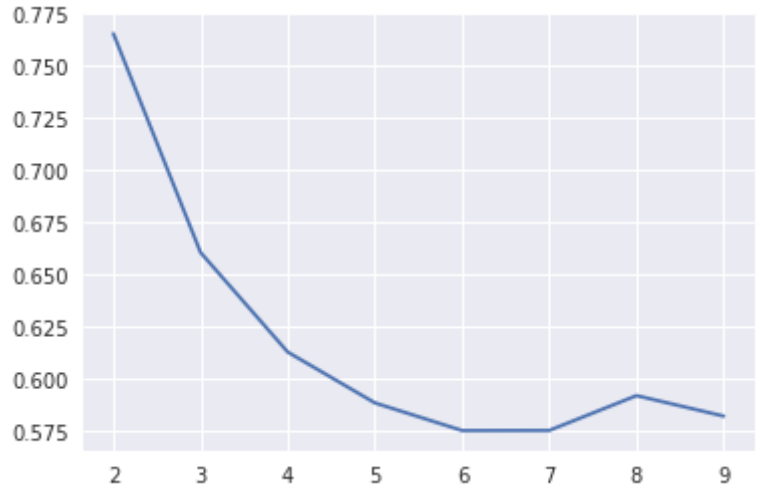
Out[38]:
```
0.5913044451165838
```

In [39]:
```
def get_silhouette_score(X, d):
    kmeans_model = KMeans(n_clusters=d).fit(X)

    labels = kmeans_model.labels_
    return metrics.silhouette_score(X, labels)
```

In [40]:
```
x = range(2, 20)
s = [get_silhouette_score(X, d) for d in x]

plt.plot(x, s)
plt.show()
```

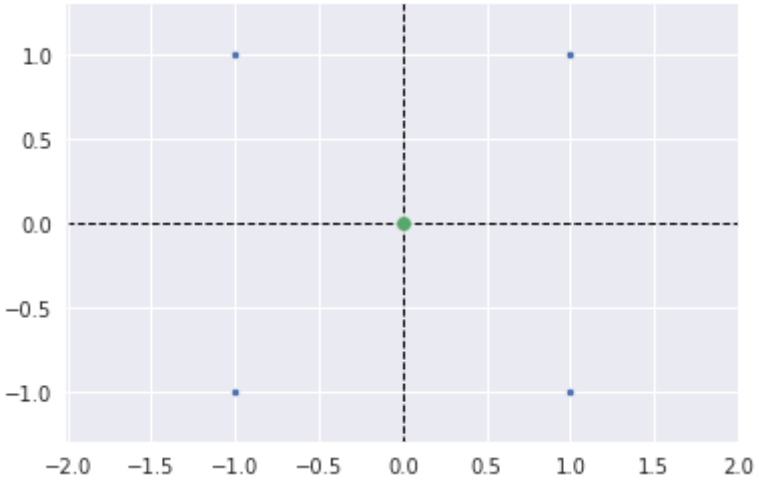


In [41]:
```
x = range(2, 10)
s = [get_silhouette_score(X_iris, d) for d in x]

plt.plot(x, s)
plt.show()
```

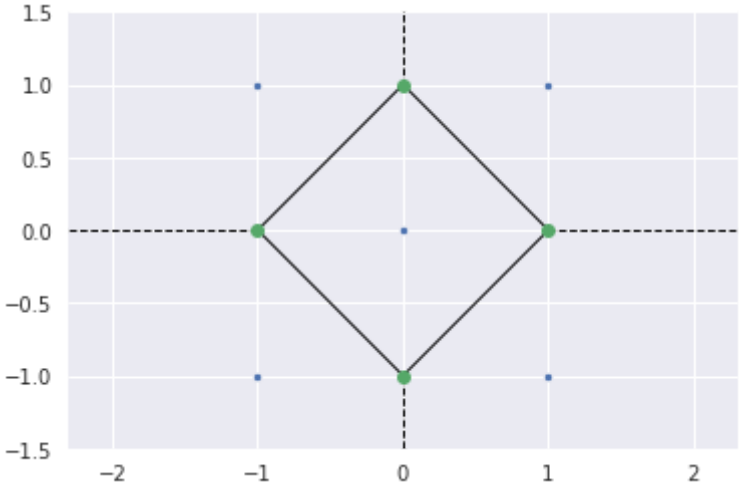


In [42]:
```
kmeans_model = KMeans(n_clusters=10).fit(X)
```

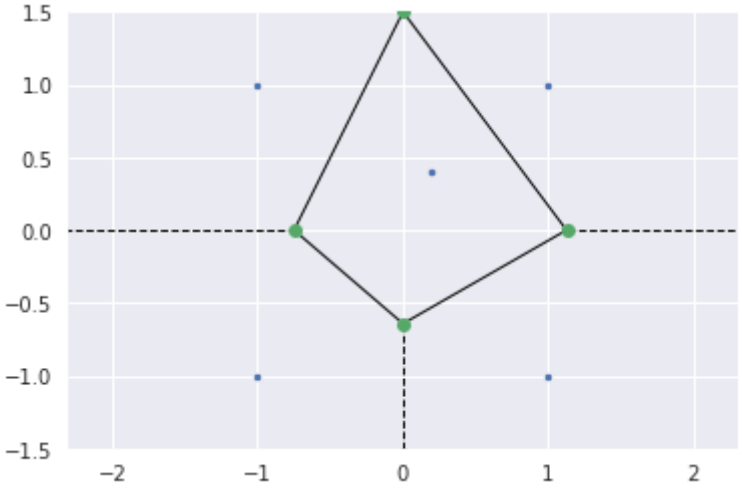
In [43]:
```
from scipy.spatial import Voronoi, voronoi_plot_2d
vor = Voronoi([(-1, -1), (-1, 1), (1, -1), (1, 1)])
fig = voronoi_plot_2d(vor)
plt.axis('equal')
plt.xlim((-1.3, 1.3))
plt.ylim((-1.3, 1.3))
plt.show()
```



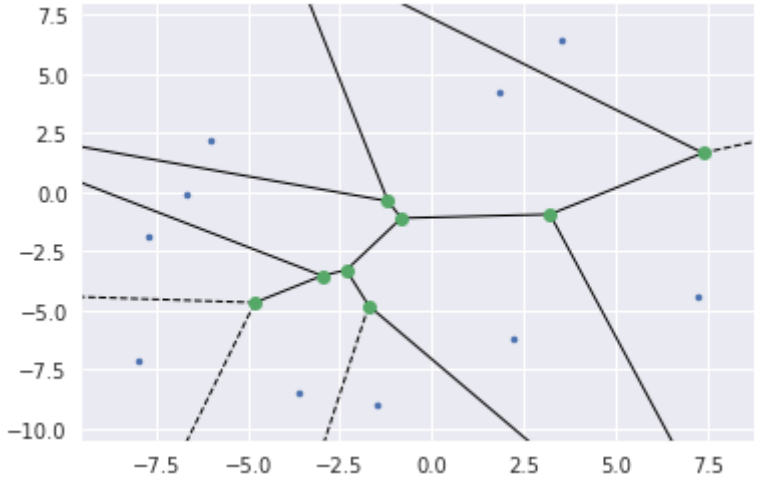
In [44]:
```
from scipy.spatial import Voronoi, voronoi_plot_2d
vor = Voronoi([(-1, -1), (-1, 1), (1, -1), (1, 1), (0, 0)])
fig = voronoi_plot_2d(vor)
plt.axis('equal')
plt.xlim((-1.5, 1.5))
plt.ylim((-1.5, 1.5))
plt.show()
```



In [45]:
```
from scipy.spatial import Voronoi, voronoi_plot_2d
vor = Voronoi([(-1, -1), (-1, 1), (1, -1), (1, 1), (0.2, 0.4)])
fig = voronoi_plot_2d(vor)
plt.axis('equal')
plt.xlim((-1.5, 1.5))
plt.ylim((-1.5, 1.5))
plt.show()
```

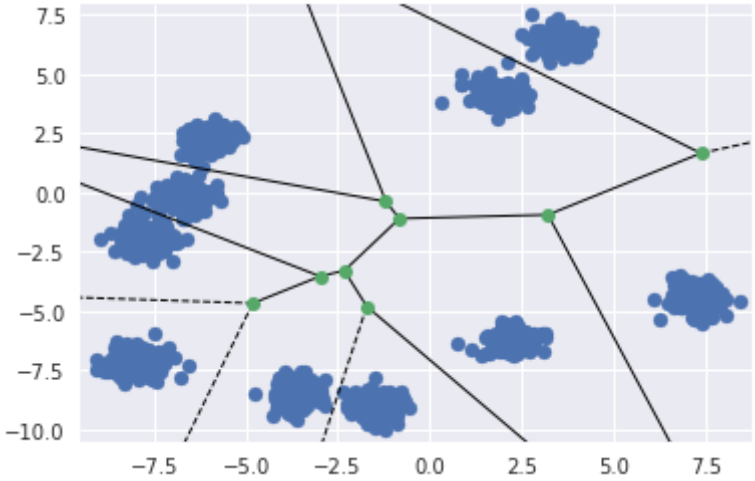


In [46]:
```
from scipy.spatial import Voronoi, voronoi_plot_2d
vor = Voronoi(kmeans_model.cluster_centers_)
fig = voronoi_plot_2d(vor)
plt.show()
```

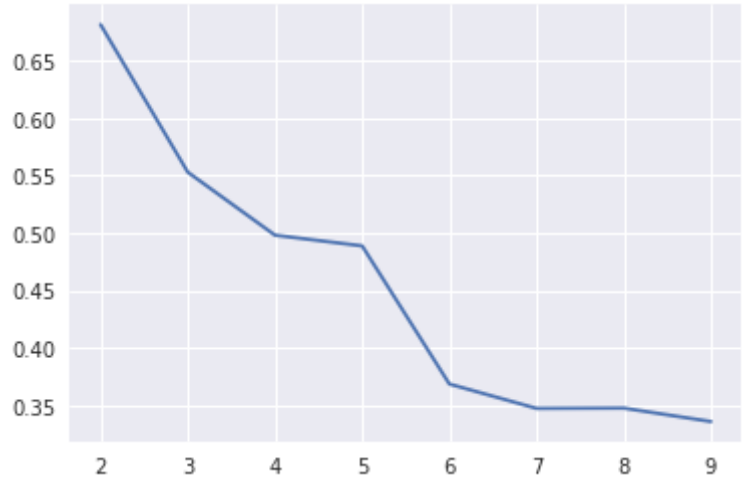


In [47]:
```
fig, ax = plt.subplots()
ax.scatter(X[:, 0], X[:, 1])

voronoi_plot_2d(vor, ax=ax)
plt.show()
```

```
/home/malchiodi/anaconda3/lib/python3.6/site-packages/scipy/spatial/_plotutils.py:20: MatplotlibDeprecationW
arning: The ishold function was deprecated in version 2.0.
  was_held = ax.ishold()
```



In [48]:
```
X_iris_4d = iris.drop(['species'], axis=1)
x = range(2, 10)
s = [get_silhouette_score(X_iris_4d, d) for d in x]

plt.plot(x, s)
plt.show()
```



In [49]:
```
from sklearn.datasets import fetch_olivetti_faces
faces_dataset = fetch_olivetti_faces(shuffle=True, random_state=0)
faces = faces_dataset.images.reshape(len(faces_dataset.images), -1)
```

In [50]:
```
def plot_faces(faces):
    fig = plt.figure(figsize=(10, 8))
    for i in range(20):
        sub = plt.subplot(4, 5, i + 1)
        sub.imshow(faces[i].reshape(64, 64), cmap='Greys_r')
        plt.xticks(())
        plt.yticks(())

    plt.tight_layout()
    plt.show()

plot_faces(faces)
```

In [51]:
```
faces[0]
```

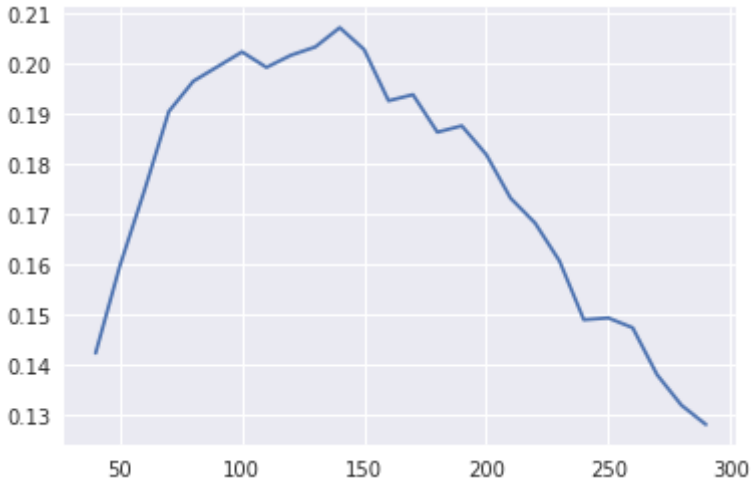
Out[51]:
```
array([0.6694215 , 0.6363636 , 0.6487603 , ..., 0.08677686, 0.08264463,
       0.07438017], dtype=float32)
```

In [52]:
```
len(faces)
```

Out[52]:
```
400
```

In [53]:
```
x = np.arange(40, 300, 10)
s = [get_silhouette_score(faces, d) for d in x]

plt.plot(x, s)
plt.show()
```

```
/home/malchiodi/anaconda3/lib/python3.6/site-packages/sklearn/metrics/pairwise.py:257: RuntimeWarning: inval
id value encountered in sqrt
  return distances if squared else np.sqrt(distances, out=distances)
```



In [54]:
```
kmeans_model = KMeans(n_clusters=150).fit(faces)
```

```
/home/malchiodi/anaconda3/lib/python3.6/site-packages/sklearn/metrics/pairwise.py:257: RuntimeWarning: inval
id value encountered in sqrt
  return distances if squared else np.sqrt(distances, out=distances)
```

In [55]:
```
for f in faces[kmeans_model.labels_== 1]:
    fig = plt.figure(figsize=(2, 2))
    plt.imshow(f.reshape(64, 64), cmap='Greys_r')
    plt.xticks(())
    plt.yticks(())
```

In [56]:
```
def show_cluster(c):
    for f in faces[kmeans_model.labels_== c]:
        fig = plt.figure(figsize=(2, 2))
        plt.imshow(f.reshape(64, 64), cmap='Greys_r')
        plt.xticks(())
        plt.yticks(())
```

In [57]:
```
show_cluster(0)
```

In [58]:
```
show_cluster(9)
```

In [59]:
```
show_cluster(93)
```

In [60]:
```
kmeans_model = KMeans(n_clusters=50).fit(faces)
```

In [61]:
```
show_cluster(2)
```

In [62]:
```
len(faces)
```

Out[62]: 400

In [64]:
```
set(faces_dataset.target)
```

Out[64]:
```
{0,
 1,
 2,
 3,
 4,
 5,
 6,
 7,
 8,
 9,
 10,
 11,
 12,
 13,
 14,
 15,
 16,
 17,
 18,
 19,
 20,
 21,
 22,
 23,
 24,
 25,
 26,
 27,
 28,
 29,
 30,
 31,
 32,
 33,
 34,
 35,
 36,
 37,
```

```
 38,
 39}
```

In [65]:
```
train, test = faces[:300], faces[300:]
kmeans_model = KMeans(n_clusters=150).fit(train)
```

```
/home/malchiodi/anaconda3/lib/python3.6/site-packages/sklearn/metrics/pairwise.py:257: RuntimeWarning: inval
id value encountered in sqrt
  return distances if squared else np.sqrt(distances, out=distances)
```

In [66]:
```
def show_image(i):
    plt.figure(figsize=(2, 2))
    plt.imshow(test[i].reshape(64, 64), cmap='Greys_r')
    plt.xticks(())
    plt.yticks(())

def show_cluster(c):
    for f in train[kmeans_model.labels_== c]:
        fig = plt.figure(figsize=(2, 2))
        plt.imshow(f.reshape(64, 64), cmap='Greys_r')
        plt.xticks(())
        plt.yticks(())
```

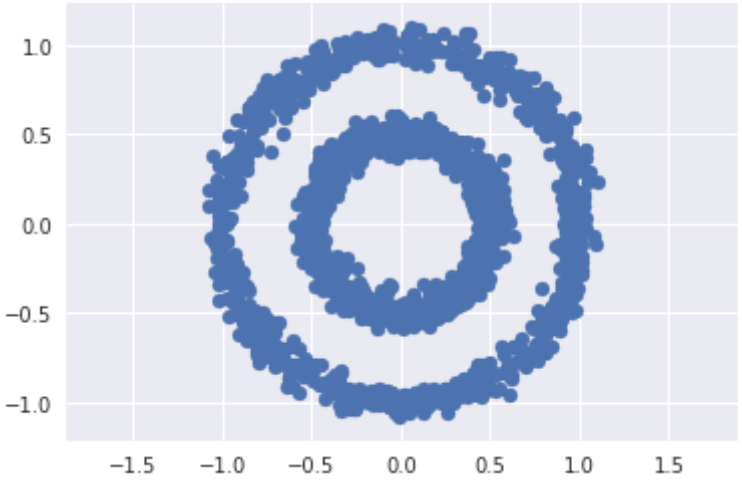
In [67]:
```
cluster = kmeans_model.predict([test[3]])[0]

show_cluster(cluster)
```

In [68]:
```
show_image(3)
```

In [69]:
```
from sklearn.datasets import make_circles

n_samples = 1500
noisy_circles = make_circles(n_samples=n_samples, factor=.5,
                                          noise=.05)
```

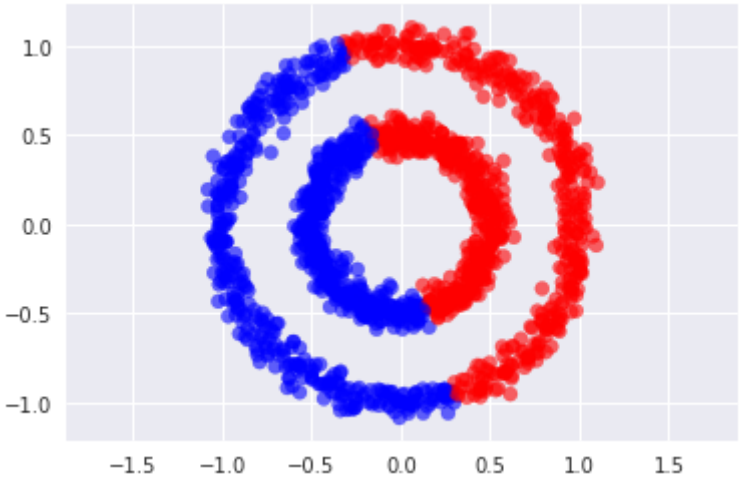
In [70]:
```
X, y = noisy_circles
```

In [71]:
```
plt.scatter(X[:, 0], X[:, 1])
plt.axis('equal')
plt.show()
```

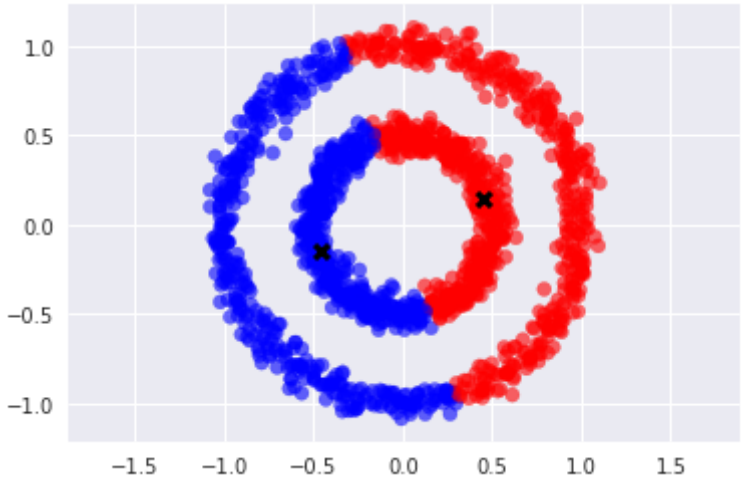


In [72]:
```
kmeans_model = KMeans(n_clusters=2, random_state=1).fit(X)

labels = kmeans_model.labels_
```

In [73]:
```
colors = ['r' if y else 'b' for y in labels]
plt.scatter(X[:, 0], X[:, 1], c=colors, alpha=0.6)
plt.axis('equal')
plt.show()
```

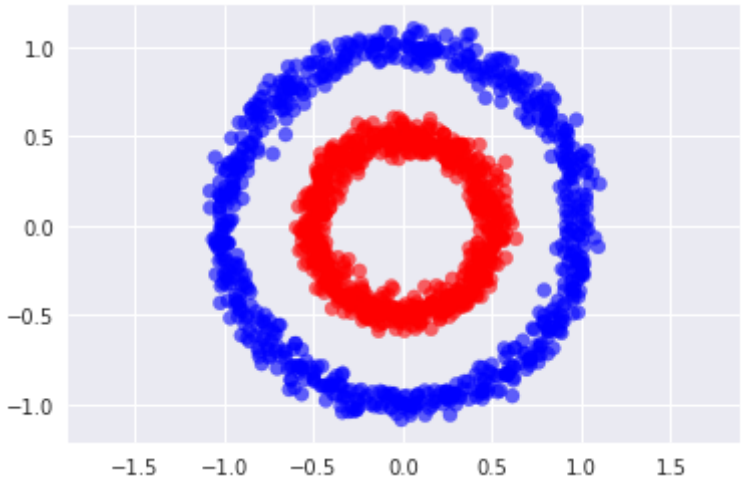


In [74]:
```
centers = kmeans_model.cluster_centers_

plt.scatter(X[:, 0], X[:, 1], c=colors, alpha=0.6)
plt.scatter(centers[:, 0], centers[:, 1], c='k', marker='x', linewidth=3)
plt.axis('equal')
plt.show()
```



In [75]:
```
from sklearn.cluster import DBSCAN

sc_model = DBSCAN(eps=.1).fit(X)

labels = sc_model.labels_
```

In [76]:
```
colors = ['r' if y else 'b' for y in labels]

plt.scatter(X[:, 0], X[:, 1], c=colors, alpha=0.6)
plt.axis('equal')
plt.show()
```



In [77]:
```
sc_model = DBSCAN(eps=6, min_samples=2).fit(faces)
```

In [78]:
```
sc_model.labels_
```

Out[78]:
```
array([ 0,  1, -1,  2,  3, -1, -1,  4,  5,  6,  7,  8, -1,  9, -1, 10, 11,
       12, 13, 14, 15, -1, 16,  9, 15, -1, -1, 17, 18, 19, 15,  1, 20, 21,
       22, -1, -1, 15, 15, 23, 19, 24, -1, -1, 11, 10, 25, -1, -1, 26, -1,
        5, 11, 15, 27, -1, 28, 29, -1, 30, 31, -1, -1, -1, 14, 25, 12, -1,
       32, 33, -1,  4, 22, 34, 19, -1, 35,  1, 35, 24, 20, -1, 10, 19, 36,
       -1, 37, 25, 38, -1, 39, 40, 10, -1, 14,  5,  0, 38, -1, 35, 41, -1,
       -1, 42, 18, -1, 21, -1, 32, 11, -1, -1, 11, 34, 35,  6, -1, 43, 26,
       -1, 44, 21, -1, 26, 25, 12, 34, -1, -1, 45, -1, 23, -1,  3,  6, 19,
       -1,  2, 24, 29, 46, 47,  7, 48,  4, 49, -1,  3, 50, 48, 51, 52, 27,
       -1, -1, -1, 53, -1,  2, 11, 27,  8, 54, 55, 56, 54, 50, -1, 21, -1,
       -1, 57, 58, 43, 50, -1, 49, 59, -1, 60, 21, -1, -1, 39, 54,  6, 61,
       62, 63, -1, 28, 24, 64, -1, -1,  6, -1,  6, 25, 65,  0, 50,  6, 66,
       -1, 36, 26, -1,  5, 43, -1, 67, 51, 30,  3, 16, 44, 48, 45, 39, 39,
       -1, 59, 17, -1, -1,  3, 43, -1, -1, -1, -1, 39, 39, -1, 46, -1, -1,
       21, 62, 38, -1, 40, 47,  5, 55, -1,  5, -1, 42, 29,  4, -1, 19, 24,
        5, 33,  0, 50, -1,  3, 36, 29, 21, 36, -1, 66, -1, 31, 47,  1, 25,
       68, 62, 69, 21, 41, -1, 11, 21,  4, 54, 39, 10, 59, 53, 28, 13, -1,
       -1, 51, 37, 21, 22,  1,  8, 47, 67, 59, -1, 58, 51, -1, 39, 15, 54,
       -1, -1, -1, 69, -1, -1, 36, 30, 13, -1, 65,  5, 12, 62, -1, -1, 60,
        4, 24,  5, 21, 52, 68, -1, -1, 21,  4, -1, 11, 10, 24, 40, 69, 59,
       19, 63, -1, 64, -1, 10, 68, -1, 11, 61, 47, 56, 42,  3, 58, -1, -1,
       -1, -1, 25, 57, 63, -1, 18, -1, 15, -1, 51, 51, -1,  3, -1, -1, 31,
       -1, -1, 45, 51, 39, 15, 39, -1, 70, -1, 21, -1, 21, 11,  3, -1,  6,
       -1, -1, 70, -1, 24, 14, -1, 30, 15])
```

In [79]:
```
def show_cluster(c):
    for f in faces[sc_model.labels_== c]:
        fig = plt.figure(figsize=(2, 2))
        plt.imshow(f.reshape(64, 64), cmap='Greys_r')
        plt.xticks(())
        plt.yticks(())
```

In [80]:
```
show_cluster(30)
```

In [81]:
```
show_cluster(-1)
```

```
/home/malchiodi/anaconda3/lib/python3.6/site-packages/matplotlib/pyplot.py:537: RuntimeWarning: More than
20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) ar
e retained until explicitly closed and may consume too much memory. (To control this warning, see the rcP
aram `figure.max_open_warning`).
  max_open_warning, RuntimeWarning)
```

In [ ]: