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

In [ ]:
```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

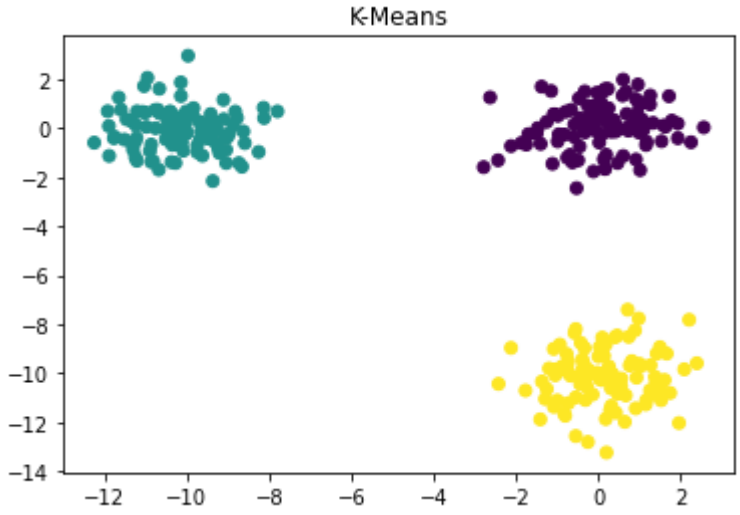
In [ ]:
```
np.random.seed(123)
```

In [ ]:
```
X1 = np.random.normal(loc=[0,-10], size=(100,2))
X2 = np.random.normal(loc=[-10, 0], size=(100,2))
X3 = np.random.normal(loc=[0, 0], size=(100,2))
X = np.vstack((X1,X2,X3))
y = np.array([1]*100 + [2]*100 + [3]*100)
```

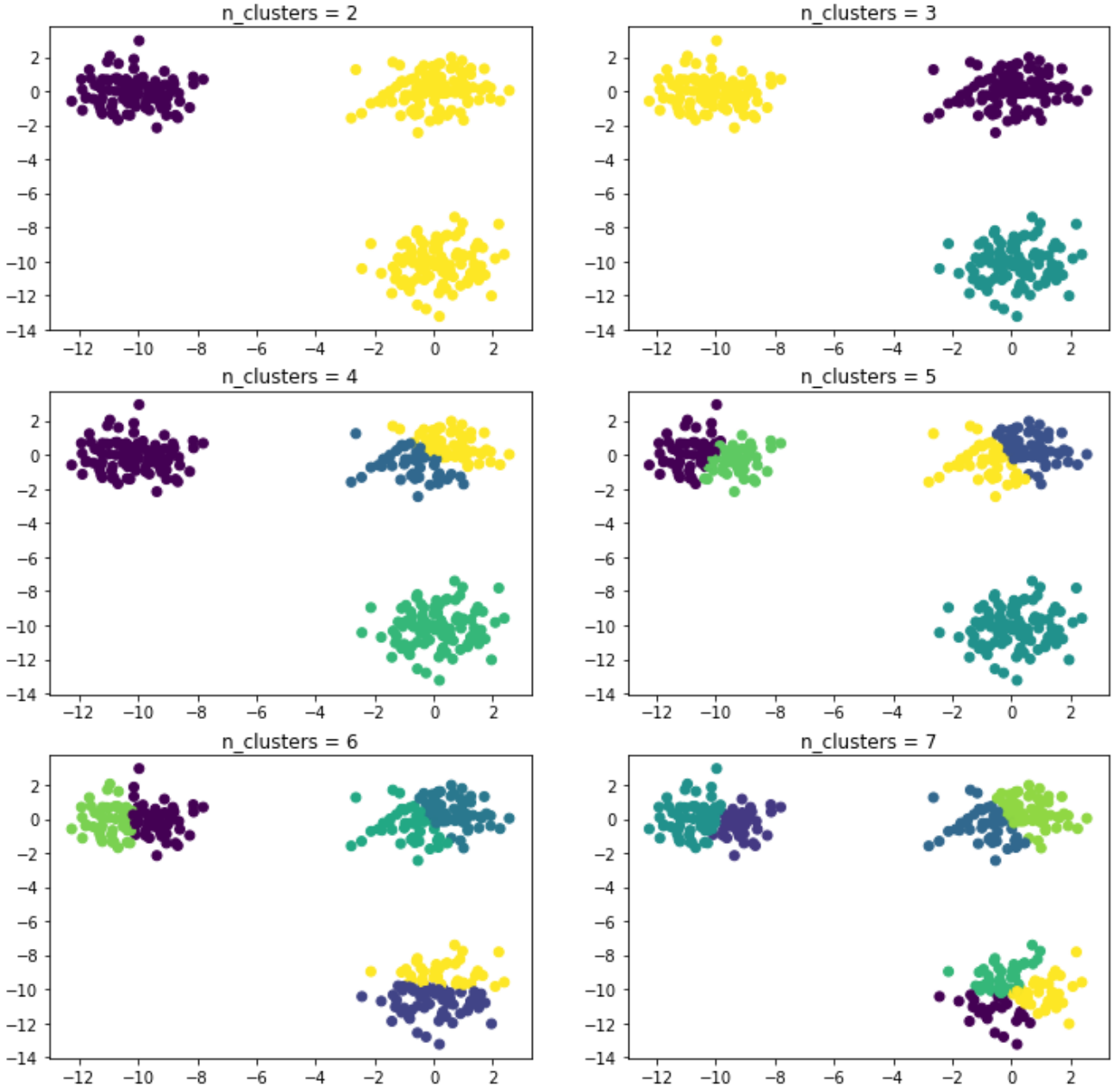
In [ ]:
```
k_means = KMeans(n_clusters=3)
```

In [ ]:
```
clusters = k_means.fit_predict(X)
```

In [ ]:
```
plt.scatter(X[:, 0], X[:, 1], c=clusters)
plt.title('K-Means')
plt.show()
```

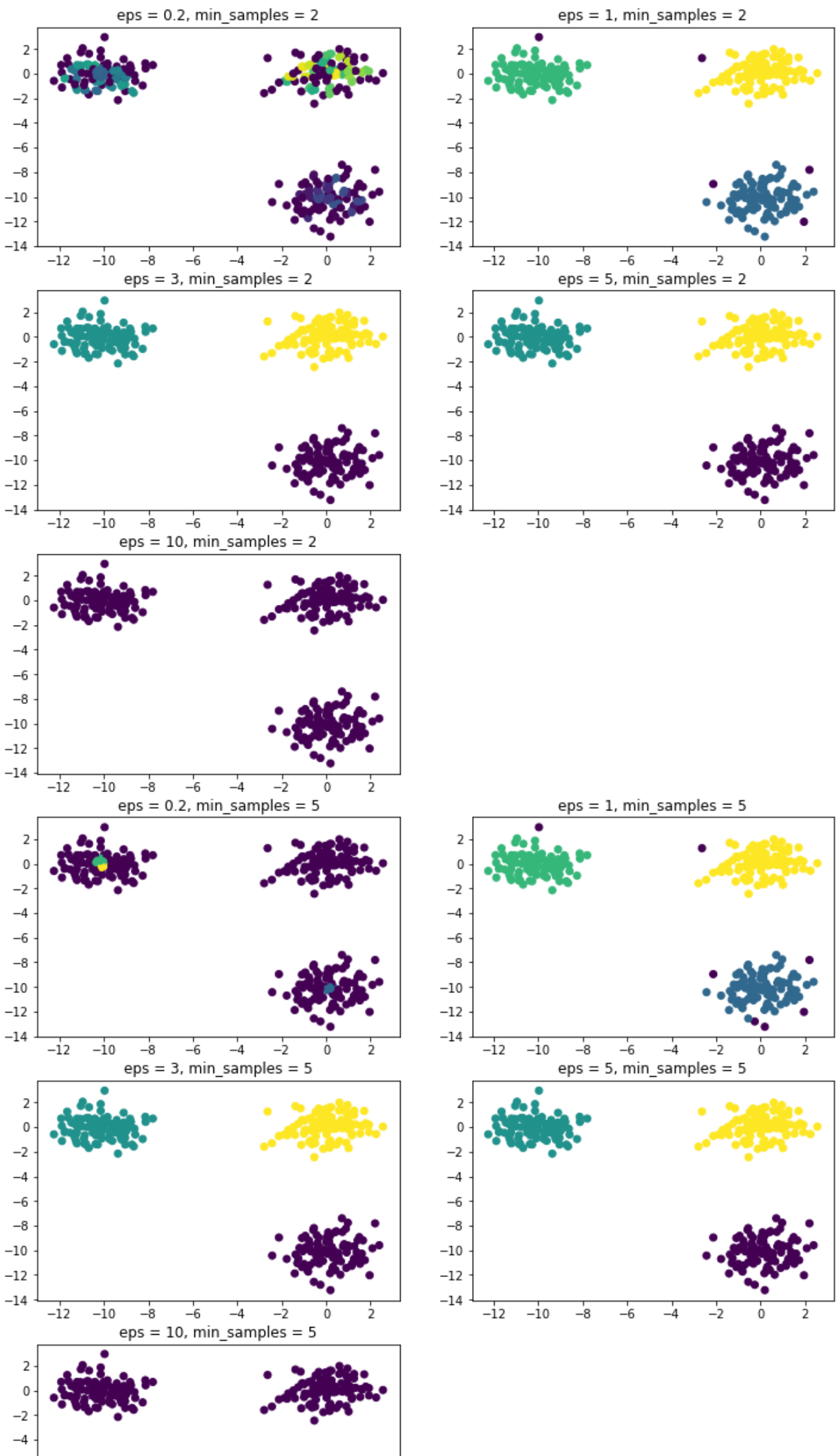


In [ ]:
```
plt.figure(figsize=(12, 12))
for n_c in range(2, 8):
  k_means = KMeans(n_clusters=n_c)
  clusters = k_means.fit_predict(X)
  plt.subplot(3, 2, n_c-1)
  plt.scatter(X[:, 0], X[:, 1], c=clusters)
  plt.title('n_clusters = {}'.format(n_c))
plt.show()
```



In [ ]:

```
from sklearn.cluster import DBSCAN
plt.figure(figsize=(12, 36))
i = 1
for sample in [2, 5, 10]:
  for e in [0.2, 1, 3, 5, 10]:
    dbscan = DBSCAN(eps=e, min_samples=sample)
    clusters = dbscan.fit_predict(X)
    plt.subplot(9, 2, i)
    plt.scatter(X[:, 0], X[:, 1], c=clusters)
    plt.title('eps = {}, min_samples = {}'.format(e, sample))
    i += 1
  i += 1
plt.show()
```


eps = 0.2, min_samples = 2
eps = 1, min_samples = 2
eps = 3, min_samples = 2
eps = 5, min_samples = 2
eps = 10, min_samples = 2
eps = 0.2, min_samples = 5
eps = 1, min_samples = 5
eps = 3, min_samples = 5
eps = 5, min_samples = 5
eps = 10, min_samples = 5

In [ ]: from

In [ ]: n_sa

In [ ]: X = datasets.make_circles(n_samples=n_samples, f

In [ ]: X =

In [ ]: X.sh

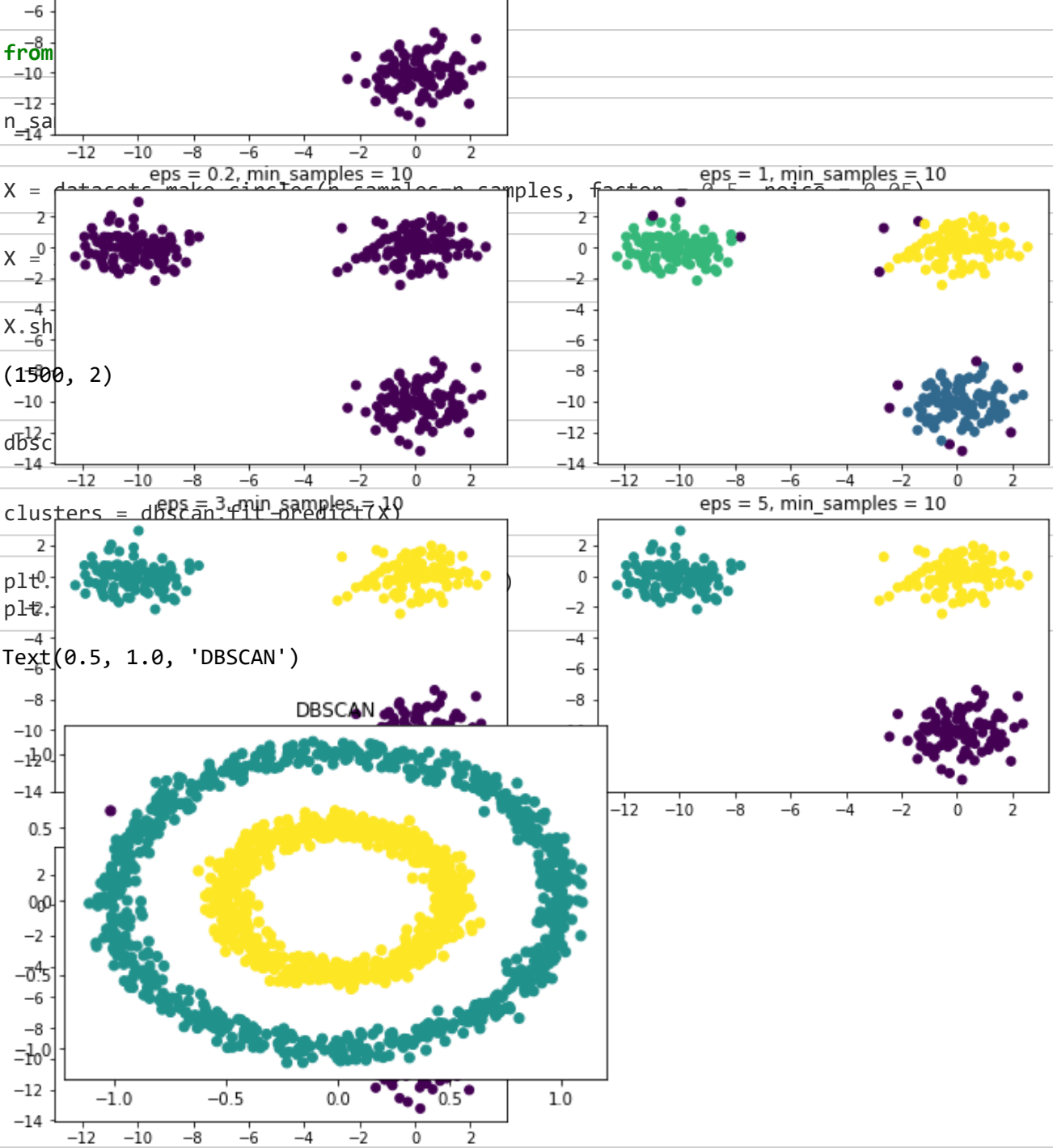
Out[ ]: (1500, 2)

In [ ]: dbsc

In [ ]: clusters = dbscan.fit_predict(X)

In [ ]: plt.
plt.

Out[ ]: Text(0.5, 1.0, 'DBSCAN')

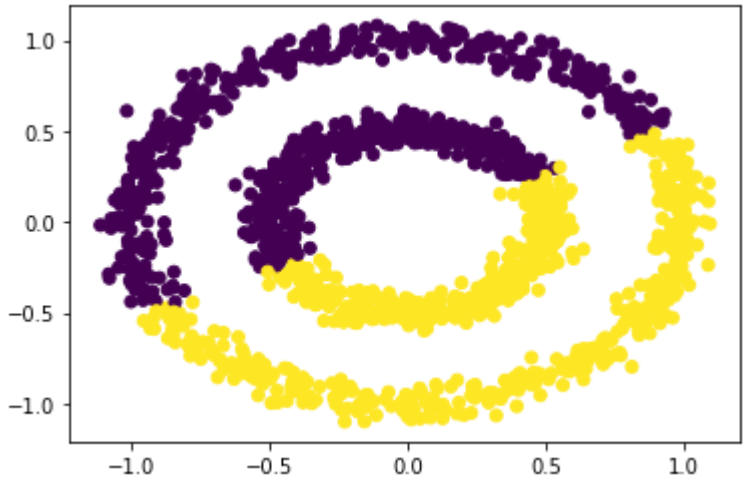


In [ ]: k_means = KMeans(n_clusters=2)

In [ ]: clusters = k_means.fit_predict(X)

In [ ]: plt.scatter(X[:, 0], X[:, 1], c = clusters)

Out[ ]: <matplotlib.collections.PathCollection at 0x7f8b5d06c390>



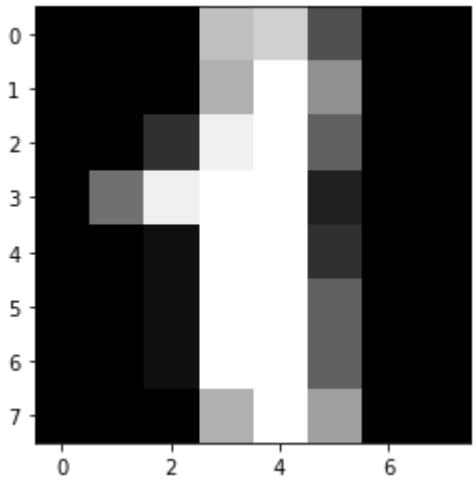
In [ ]:
```
from sklearn.datasets import load_digits
```

In [ ]:
```
digits = load_digits()
```

In [ ]:
```
X, y = digits['data'], digits['target']
```

In [ ]:
```
plt.imshow(X[1].reshape(8, 8), cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7f8b5d2545c0>
```



In [ ]:
```
km = KMeans(n_clusters=10)
```

In [ ]:
```
clusters = km.fit_predict(X)
```

In [ ]:
```
pred = np.zeros(X.shape[0])
```

In [ ]:
```
for i in range(10):
  bc = np.bincount(y[clusters == i])
  pred[clusters == i] = bc.argmax()
```

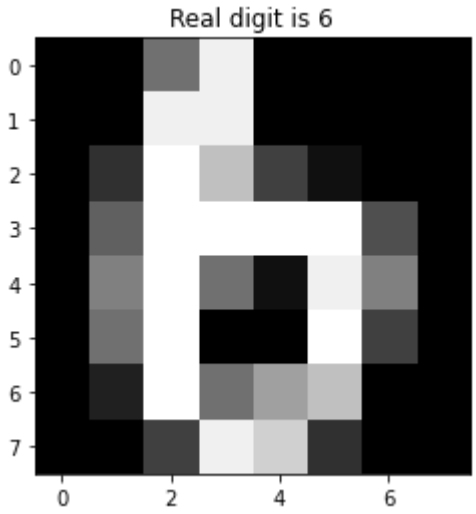
In [ ]:
```
from sklearn.metrics import accuracy_score
```

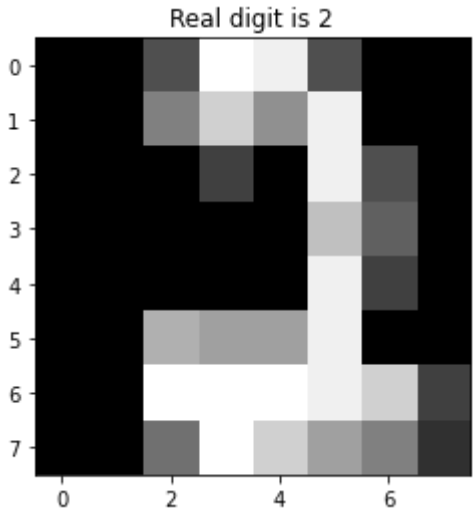
In [ ]:
```
accuracy_score(y, pred)
```

Out[ ]:
```
0.7957707289927657
```

In [ ]:
```
incorrect_indeces = np.where(np.logical_and(pred == 0, y!=0))[0]
```

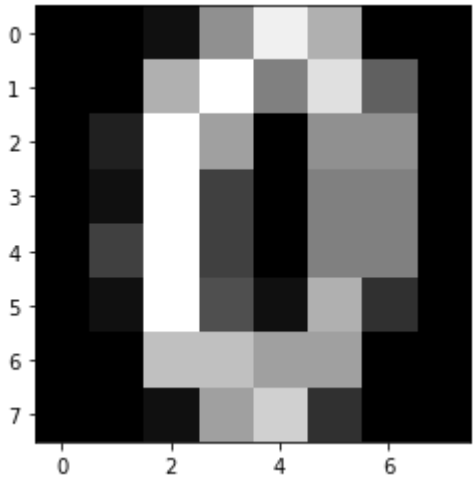
In [ ]:
```
for i in range(2):
  plt.imshow(X[incorrect_indeces[i]].reshape(8, 8), cmap='gray')
  plt.title('Real digit is {}'.format(y[incorrect_indeces[i]]))
  plt.show()
```





In [ ]:
```
correct_indeces = np.where(np.logical_and(pred == 0, y == 0))[0]
```

In [ ]:
```
for i in range(2):
  plt.imshow(X[correct_indeces[i]].reshape(8, 8), cmap='gray')
```



In [ ]:
```
from sklearn.decomposition import PCA
```

In [ ]:
```
pca = PCA(n_components=2)
```

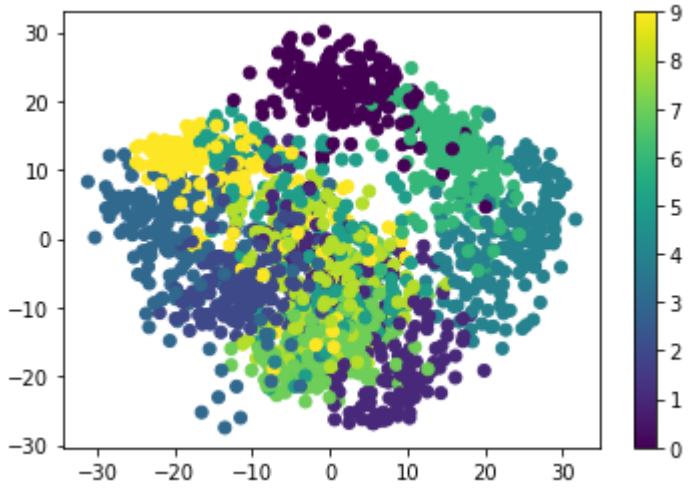
In [ ]:
```
digits_2d = pca.fit_transform(digits['data'])
```

In [ ]:
```
plt.figure(figsize=(13, 10))
```

Out[ ]:
```
<Figure size 936x720 with 0 Axes>
```

```
<Figure size 936x720 with 0 Axes>
```

In [ ]:
```
plt.scatter(digits_2d[:, 0], digits_2d[:, 1], c=y)
plt.colorbar()
plt.show()
```

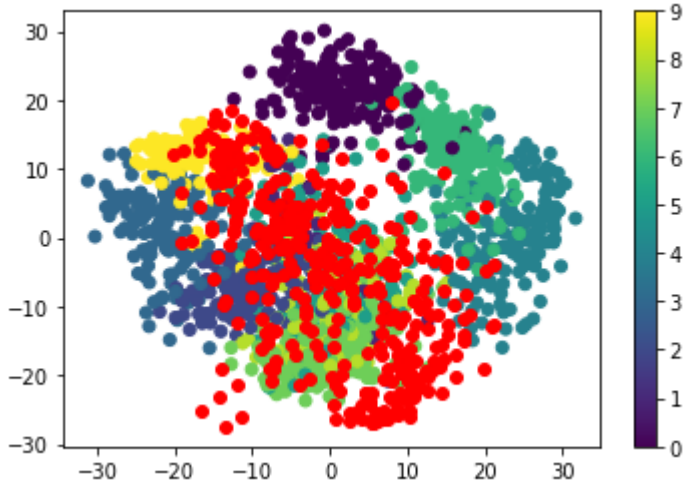


In [ ]:
```
plt.figure(figsize = (13, 10))
```

Out[ ]:
```
<Figure size 936x720 with 0 Axes>
```

```
<Figure size 936x720 with 0 Axes>
```

In [ ]:
```
plt.scatter(digits_2d[y==pred, 0], digits_2d[y == pred, 1], c = y[y == pred])
plt.colorbar()
plt.scatter(digits_2d[y!=pred, 0], digits_2d[y!=pred, 1], c='red')
plt.show()
```



In [ ]:
```
from sklearn.manifold import TSNE
```

In [ ]:
```
tsne = TSNE(n_components=2)
```

In [ ]:
```
digits_2d = tsne.fit_transform(digits['data'])
```

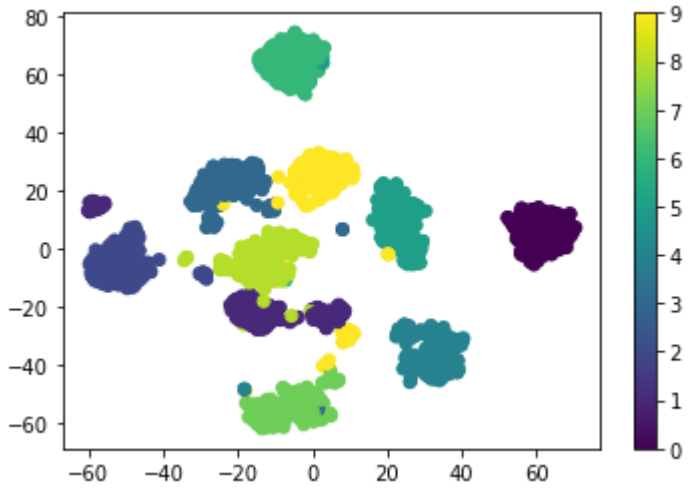
In [ ]:
```
plt.figure(figsize = (13, 10))
```

Out[ ]:
```
<Figure size 936x720 with 0 Axes>
```

```
<Figure size 936x720 with 0 Axes>
```

In [ ]:
```
plt.scatter(digits_2d[:, 0], digits_2d[:, 1], c = y)
plt.colorbar()
plt.show()
```

```
ERROR! Session/line number was not unique in database. History logging moved to new
session 59
```



In [ ]:
```
tsne = TSNE(n_components=3)
```

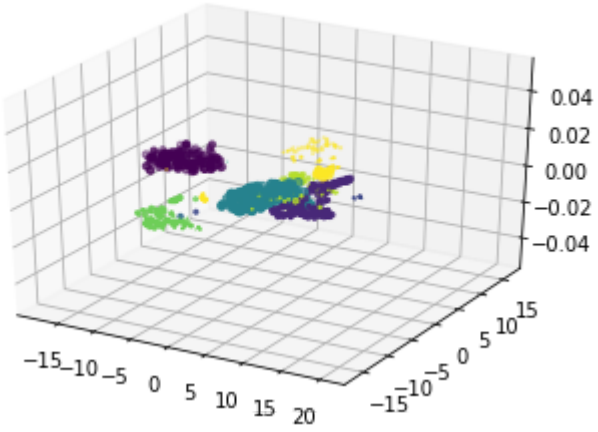
In [ ]:
```
digits_3d = tsne.fit_transform(digits['data'])
```

In [ ]:
```
from mpl_toolkits.mplot3d import Axes3D
```

In [ ]:
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
plt.scatter(digits_3d[:, 0], digits_3d[:,1], digits_3d[:, 2], c=y)
```

```
/usr/local/lib/python3.6/dist-packages/matplotlib/collections.py:886: RuntimeWarnin
g: invalid value encountered in sqrt
  scale = np.sqrt(self._sizes) * dpi / 72.0 * self._factor
```

Out[ ]:
```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x7f8b5bc67320>
```



In [ ]: