Identificar Fronteras no Lineales

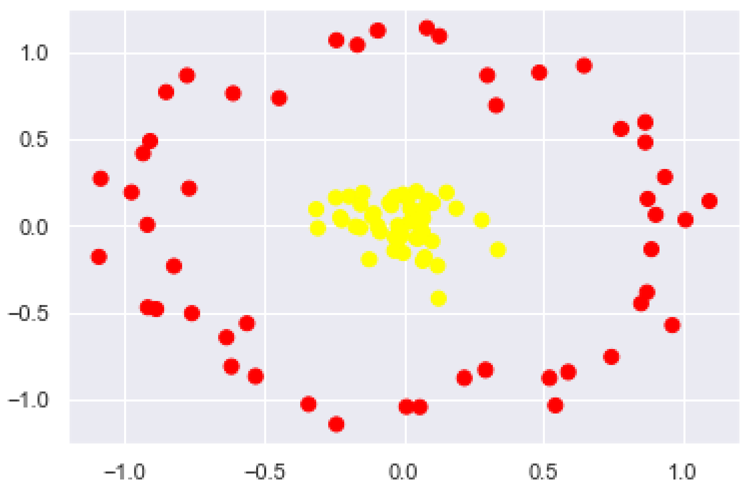
In [1]:
```python
from sklearn.datasets import make_circles, make_blobs
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

import seaborn as sns; sns.set()
from sklearn.svm import SVC
```

In [2]:
```python
X, Y = make_circles(100, factor = .1, noise= .1)
```

In [3]:
```python
def plt_svc(model, ax=None, plot_support=True):
    """Plot de la función de decisión para una clasificación en 2D con SVC"""
    if ax is None:
        ax = plt.gca()#Se grea un nuevo dibujo si ax es NONE
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    ## Generamos la parrilla de puntos para evaluar el modelo
    xx = np.linspace(xlim[0], xlim[1], 30)
    yy = np.linspace(ylim[0], ylim[1], 30)
    Y, X = np.meshgrid(yy,xx)

    xy = np.vstack([X.ravel(), Y.ravel()]).T #generamos tuple
    P = model.decision_function(xy).reshape(X.shape)

    ##Representamos las froteras y los márgenes del SVC
    ax.contour(X,Y,P, colors="k", levels=[-1,0,1], alpha=0.5, linestyles=["--", "-", "--"])

    if plot_support:
        ax.scatter(model.support_vectors_[:,0],
                   model.support_vectors_[:,1],
                   s=100, linewidth=1, facecolors="blue")
    ax.set_xlim(xlim)
    ax.set_ylim(ylim)
```

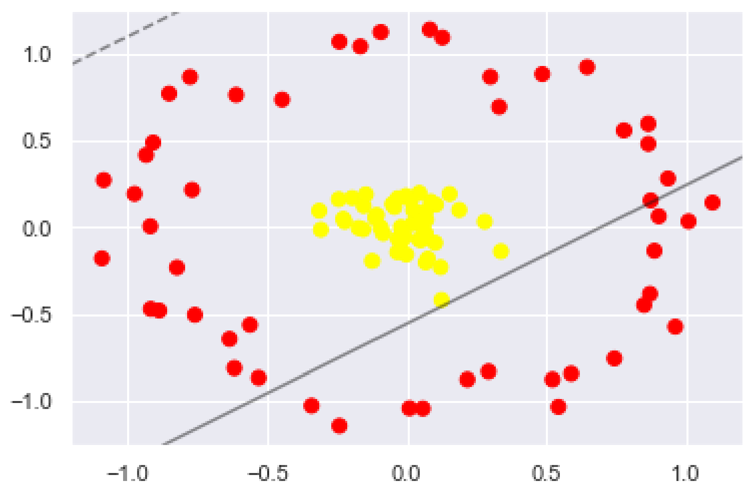
In [4]:
```python
plt.scatter(X[:,0], X[:,1], c=Y, s=50, cmap="autumn")
```

Out[4]:
```
<matplotlib.collections.PathCollection at 0x1ee5b8273d0>
```



In [5]:
```python
plt.scatter(X[:,0], X[:,1], c=Y, s=50, cmap="autumn")
```

```
plt_svc(SVC(kernel="linear").fit(X,Y), plot_support=False)
```



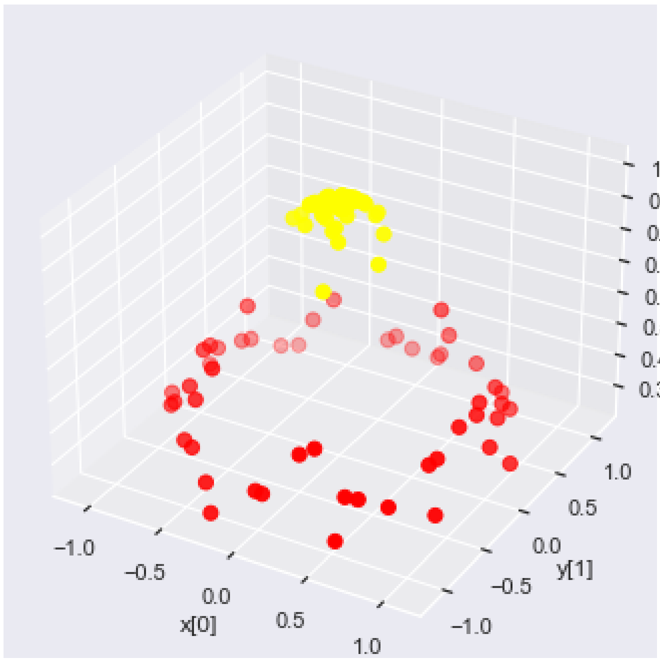
In [6]:
```
r = np.exp(-(X**2).sum(1))
r
```

Out[6]:
```
array([0.44017171, 0.29888842, 0.44687463, 0.96798826, 0.94756506,
       0.91866795, 0.9949879 , 0.29512102, 0.99119949, 0.90957344,
       0.98363353, 0.97150722, 0.99737916, 0.93597028, 0.99916096,
       0.99340273, 0.97594713, 0.35409924, 0.9727924 , 0.96156534,
       0.30824238, 0.89850521, 0.3225304 , 0.97172144, 0.25902541,
       0.99925404, 0.98729054, 0.96225004, 0.95464961, 0.9798869 ,
       0.87482822, 0.33221964, 0.36001574, 0.35191284, 0.2806796 ,
       0.4785102 , 0.94467042, 0.99265323, 0.34387399, 0.55575435,
       0.34539963, 0.37132069, 0.98471313, 0.9707681 , 0.26803854,
       0.39915508, 0.93367817, 0.52960703, 0.96013353, 0.3508059 ,
       0.99172048, 0.99133173, 0.28442907, 0.44104109, 0.82438446,
       0.32956236, 0.39631723, 0.40335882, 0.42895656, 0.97592385,
       0.43289028, 0.28596417, 0.2540268 , 0.36057458, 0.94772392,
       0.45842295, 0.98991461, 0.9941793 , 0.9914077 , 0.99688752,
       0.433807  , 0.37571896, 0.36336363, 0.52781311, 0.97788364,
       0.38564826, 0.33581694, 0.33368553, 0.95597554, 0.99645676,
       0.25377712, 0.47979792, 0.9847587 , 0.9240763 , 0.35115354,
       0.94192137, 0.30173922, 0.29358347, 0.27182885, 0.34429169,
       0.98157654, 0.43951298, 0.99423738, 0.97496252, 0.28177037,
       0.45532321, 0.99951486, 0.95114322, 0.38526834, 0.99221123])
```

In [7]:
```
from mpl_toolkits import mplot3d
from mpl_toolkits.mplot3d import Axes3D
from ipywidgets import interact, fixed
```

In [8]:
```
fig = plt.figure(figsize=(16,6))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X[:,0], X[:,1],r, c=Y, s=50, cmap = "autumn")
ax.set_xlabel("x[0]")
ax.set_ylabel("y[1]")
ax.set_zlabel("r")
```

Out[8]:
```
Text(0.5, 0, 'r')
```



In [9]:
```
def plot_3D(elev=30, azim=30, X=X, Y=Y, r=r):
    ax = plt.subplot(projection="3d")
    ax.figsize=(16,6)
    ax.scatter3D(X[:,0], X[:,1], r, c=Y, s =50, cmap = "autumn")
    ax.view_init(elev=elev, azim=azim)
    ax.set_xlabel("x[0]")
    ax.set_ylabel("y[1]")
    ax.set_zlabel("r")
```

In [10]:
```
interact(plot_3D, elev=[-90,-60,-30,0,30,60,90],
         azim=[-180,-150,-120,-90,-60,-30,0,30,60,90,120,150, 180],
         X=fixed(X), Y=fixed(Y), r=fixed(r) )
```

Out[10]:
```
<function __main__.plot_3D(elev=30, azim=30, X=array([[ 2.16943288e-01, -8.79503263e-01],
       [ 1.27135626e-01,  1.09156836e+00],
       [ 8.86929694e-01, -1.37233087e-01],
       [-3.16166358e-03,  1.80347785e-01],
       [-2.27221802e-01,  4.72221406e-02],
       [-2.43055760e-01,  1.60481871e-01],
       [ 6.34244398e-02, -3.16551070e-02],
       [ 1.09567739e+00,  1.40928432e-01],
       [-9.39806443e-02, -2.66403857e-03],
       [-3.07479656e-01, -1.53558117e-02],
       [-1.08130404e-01,  6.93519678e-02],
       [ 8.54319999e-02,  1.46996427e-01],
       [ 4.92157224e-02,  1.42160151e-02],
       [-1.95041840e-01,  1.67720714e-01],
       [-1.83307186e-02, -2.24360147e-02],
       [ 6.93146878e-02,  4.25981953e-02],
       [-1.55426844e-01, -1.37607279e-02],
       [-5.30963083e-01, -8.69629958e-01],
       [ 1.04264064e-01,  1.29281043e-01],
       [ 7.50800678e-02, -1.83182260e-01],
       [-3.40801552e-01, -1.02991415e+00],
       [-3.13343697e-01,  9.40133143e-02],
       [ 7.45213624e-01, -7.59087971e-01],
       [-3.32896450e-02,  1.66065957e-01],
       [-7.77342171e-01,  8.64041813e-01],
       [ 1.95884607e-02,  1.90403330e-02],
       [ 6.71962680e-02,  9.09702037e-02],
       [-1.53692889e-01,  1.21899315e-01],
[ 6.75911736e-02, -2.04553998e-01],
[-5.24329190e-02,  1.32547782e-01],
[ 3.37669440e-01, -1.40381929e-01],
[ 8.64835944e-01,  5.94993922e-01],
[-8.87976069e-01, -4.82810557e-01],
[-6.18124782e-01, -8.13814161e-01],
[-9.34123501e-02,  1.12330567e+00],
[-4.46506122e-01,  7.33287146e-01],
[-1.45895769e-01,  1.88768625e-01],
[ 4.46135565e-02, -7.33724550e-02],
[-9.18455610e-01, -4.73201105e-01],
[ 3.31279990e-01,  6.91145767e-01],
[ 5.89945803e-01, -8.45586855e-01],
[-9.76721329e-01,  1.91584549e-01],
[-1.12342668e-01,  5.27639918e-02],
[-1.72156788e-01, -5.45006145e-03],
[-8.52432083e-01,  7.68104184e-01],
[ 7.78943422e-01,  5.58258376e-01],
[ 1.20716823e-01, -2.32488531e-01],
[-5.62110542e-01, -5.65377524e-01],
[ 4.45779291e-02,  1.96712276e-01],
[-9.35046893e-01,  4.16184463e-01],
[ 5.11930275e-02, -7.54537132e-02],
[-2.14575312e-02, -9.08054749e-02],
[ 9.62047282e-01, -5.75965610e-01],
[ 9.02643021e-01,  6.20710369e-02],
[ 1.24311380e-01, -4.21503218e-01],
[-1.65703433e-01,  1.04044801e+00],
[ 8.49964720e-01, -4.50666477e-01],
[ 8.71068291e-01, -3.86223735e-01],
[-9.19990898e-01,  4.04727773e-03],
[ 1.37829377e-02,  1.55501611e-01],
[ 2.99620170e-01,  8.64580095e-01],
[-1.08554555e+00,  2.71071221e-01],
[ 5.44150817e-01, -1.03644363e+00],
[ 1.00944865e+00,  3.27089460e-02],
[-1.23680975e-01, -1.95946568e-01],
[ 2.93108929e-01, -8.33096760e-01],
[ 2.22134889e-02,  9.81995716e-02],
[ 6.70649069e-02, -3.66061111e-02],
[-8.50261410e-02, -3.74163129e-02],
[-7.10387836e-03, -5.53792902e-02],
[-7.59973749e-01, -5.07538621e-01],
[ 8.64991250e-01,  4.80316569e-01],
[ 4.86330471e-01,  8.80814329e-01],
[-7.70035789e-01,  2.14611034e-01],
[-3.47745314e-02, -1.45448696e-01],
[-6.11744034e-01,  7.60656817e-01],
[ 9.10105277e-03, -1.04456032e+00],
[ 5.69270196e-02, -1.04609539e+00],
[ 1.87903341e-01,  9.85661281e-02],
[ 4.42166764e-02,  3.99300980e-02],
[-2.40715607e-01, -1.14601696e+00],
[-8.24296676e-01, -2.34361388e-01],
[-4.50675748e-02,  1.15445049e-01],
[ 2.79276297e-01,  3.10705711e-02],
[ 5.22831877e-01, -8.79305712e-01],
[ 1.53181694e-01,  1.90706189e-01],
[-2.40822381e-01,  1.06779994e+00],
[-1.09213369e+00, -1.81210630e-01],
[ 8.22937950e-02,  1.13833667e+00],
[-9.10019800e-01,  4.87985675e-01],
[ 1.01222779e-01, -9.13741539e-02],
[-6.35458636e-01, -6.46745966e-01],
[-3.25560814e-02, -6.86978153e-02],
[-2.40425549e-03, -1.59218311e-01],
       [ 6.47148864e-01,  9.20793784e-01],
       [ 8.73590543e-01,  1.53581632e-01],
       [-2.20190075e-02, -6.49827674e-04],
       [-2.21632321e-01,  3.11406930e-02],
       [ 9.35547906e-01,  2.80295044e-01],
       [ 6.33998351e-02, -6.16418879e-02]]), Y=array([0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
1, 1, 1, 1, 0, 1, 1, 0, 1,
       0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0,
       0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1], dtype=int64), r=array([0.44017171, 0.29888842, 0.
44687463, 0.96798826, 0.94756506,
       0.91866795, 0.9949879 , 0.29512102, 0.99119949, 0.90957344,
       0.98363353, 0.97150722, 0.99737916, 0.93597028, 0.99916096,
       0.99340273, 0.97594713, 0.35409924, 0.9727924 , 0.96156534,
       0.30824238, 0.89850521, 0.3225304 , 0.97172144, 0.25902541,
       0.99925404, 0.98729054, 0.96225004, 0.95464961, 0.9798869 ,
       0.87482822, 0.33221964, 0.36001574, 0.35191284, 0.2806796 ,
       0.4785102 , 0.94467042, 0.99265323, 0.34387399, 0.55575435,
       0.34539963, 0.37132069, 0.98471313, 0.9707681 , 0.26803854,
       0.39915508, 0.93367817, 0.52960703, 0.96013353, 0.3508059 ,
       0.99172048, 0.99133173, 0.28442907, 0.44104109, 0.82438446,
       0.32956236, 0.39631723, 0.40335882, 0.42895656, 0.97592385,
       0.43289028, 0.28596417, 0.2540268 , 0.36057458, 0.94772392,
       0.45842295, 0.98991461, 0.9941793 , 0.9914077 , 0.99688752,
       0.433807  , 0.37571896, 0.36336363, 0.52781311, 0.97788364,
       0.38564826, 0.33581694, 0.33368553, 0.95597554, 0.99645676,
       0.25377712, 0.47979792, 0.9847587 , 0.9240763 , 0.35115354,
       0.94192137, 0.30173922, 0.29358347, 0.27182885, 0.34429169,
       0.98157654, 0.43951298, 0.99423738, 0.97496252, 0.28177037,
       0.45532321, 0.99951486, 0.95114322, 0.38526834, 0.99221123]))>
```

RBF: Radial Basis Function

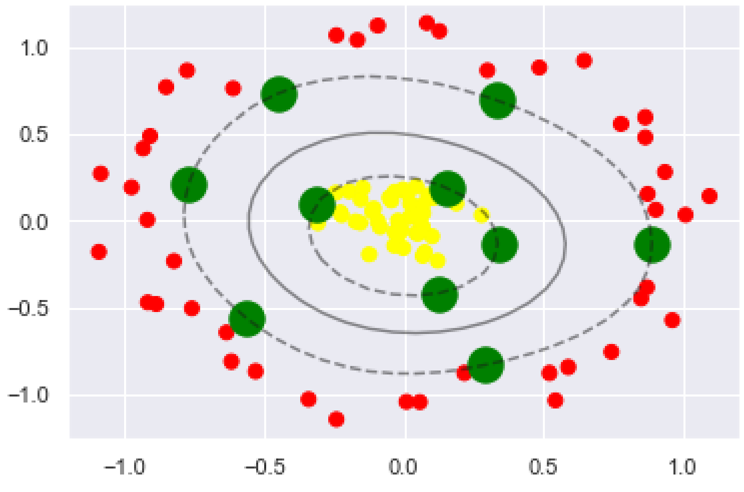
Busca forma circularas que separen correctamente los datos distribuidos en forma circular.

In [11]:
```python
rbf = SVC(kernel="rbf", C=1E6)
rbf.fit(X,Y)
```

Out[11]:
```
SVC(C=1000000.0)
```

In [12]:
```python
plt.scatter(X[:,0], X[:,1], c=Y, s=50, cmap="autumn")
plt_svc(rbf)
plt.scatter(rbf.support_vectors_[:,0], rbf.support_vectors_[:,1], s=300, lw=1, facecolors="gr
#El último scatter imprime los puntitos de los suport vector, ya los hace la función
```

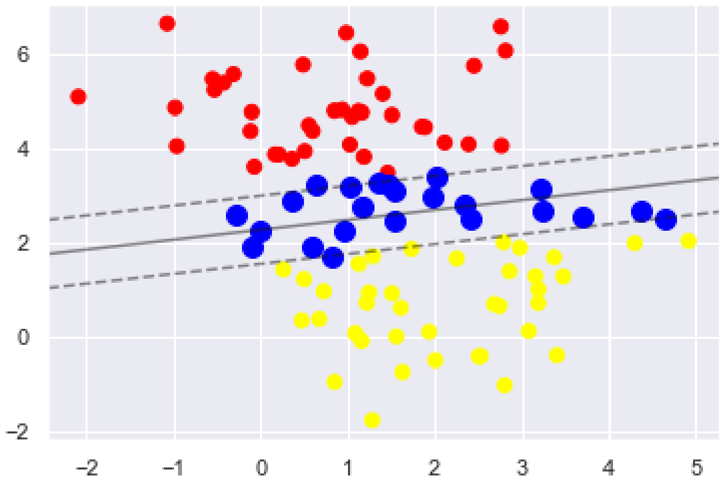
Out[12]:
```
<matplotlib.collections.PathCollection at 0x1ee5bab5b80>
```



Ajustar los parámetros de SVM

In [32]:
```
X,Y = make_blobs(n_samples=100, centers=2, random_state=0, cluster_std=1.2)
```

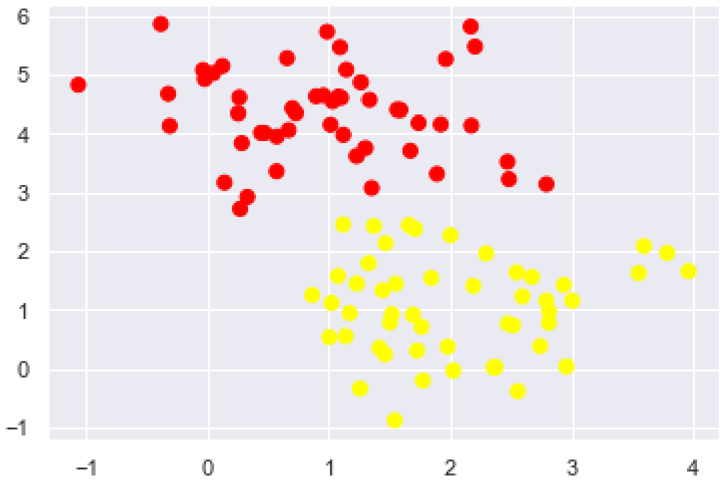
In [44]:
```
plt.scatter(X[:,0], X[:,1], c=Y, s=50, cmap="autumn")
model = SVC(kernel="linear", C=10000)
model.fit(X,Y)
plt_svc(model)
```



In [15]:
```
X,Y = make_blobs(n_samples=100, centers=2, random_state=0, cluster_std=0.8)
```

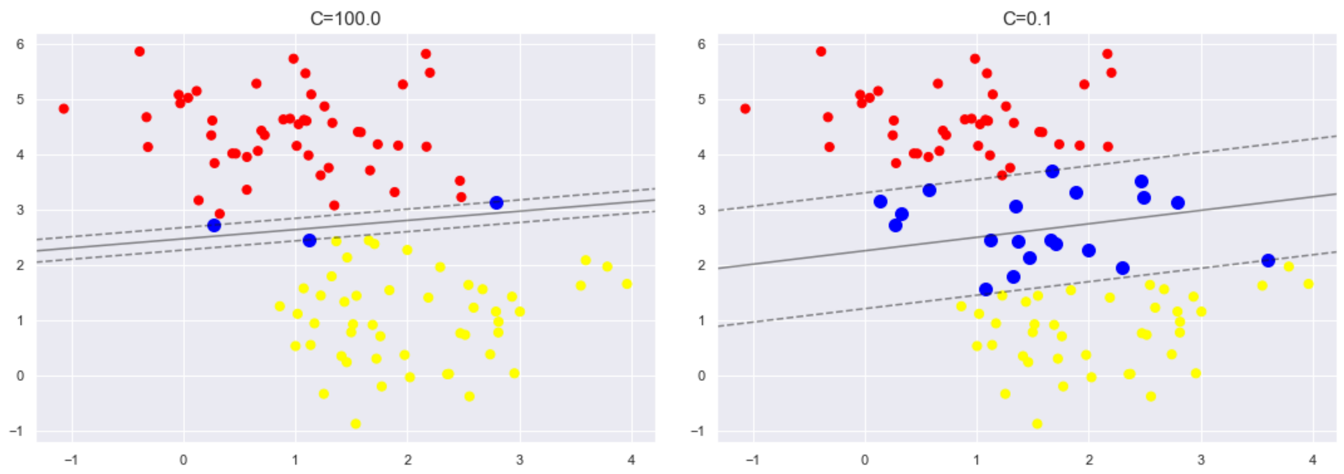
In [16]:
```
plt.scatter(X[:,0], X[:,1], c=Y, s=50, cmap="autumn")
```

Out[16]:
```
<matplotlib.collections.PathCollection at 0x1ee5bb933a0>
```



In [ ]:

In [31]:
```
fig, ax = plt.subplots(1,2, figsize=(16,6))
fig.subplots_adjust(left=0.05, right=0.95, wspace=0.1)

for ax_i, C in zip(ax,[100.0, 0.1]):
    model = SVC(kernel="linear", C=C)
    model.fit(X,Y)
    ax_i.scatter(X[:,0],X[:,1], c=Y, s=50, cmap="autumn") #c es para las clases 0 o 1
    plt_svc(model, ax_i)
    ax_i.set_title("C={0:.1f}".format(C), size = 15)
```



Se observa como para el mismo conjunto de datos, con un C mucho más grande, tendremos muchos más support vectors, que cuando el C ya vale 10, o sea que es más restrictivo el factor de penalty. Esto puede variar mucho de acuerdo a lo solapada que esté la información.

In [ ]: