20211013_Supplier_Management_MBW7

October 13, 2021

[1]:
```
# Machine Learning applied to Supply Chain Management
```

[2]:
```
# Support Vector Machines
```

[3]:
```
# This method is used to separate datasets by distance metrics
```

[4]:
```
# Image you have data from two different processes.
# You want to know which datapoints belong to which process,
# depending on the values of the process.
```

[5]:
```
# This method separates the data and puts those data together that are closer␣
↪to each other.
```

[8]:
```
!pip install numpy # Numpy. Numerical Python
!pip install scipy # SciPy. Scientific Python
!pip install matplotlib # Matplotlib. Graphical representations
!pip install seaborn # Seaborn. Create datasets
!pip install sklearn # Sklearn. Machine Learning (learn)
```

```
Requirement already satisfied: numpy in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (1.19.5)
Requirement already satisfied: scipy in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (1.6.2)
Requirement already satisfied: numpy<1.23.0,>=1.16.5 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from scipy) (1.19.5)
Requirement already satisfied: matplotlib in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (3.3.4)
Requirement already satisfied: kiwisolver>=1.0.1 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from matplotlib) (1.3.1)
Requirement already satisfied: numpy>=1.15 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from matplotlib) (1.19.5)
Requirement already satisfied: pillow>=6.2.0 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from matplotlib) (8.3.2)
Requirement already satisfied: python-dateutil>=2.1 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from matplotlib) (2.8.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from matplotlib) (2.4.7)
Requirement already satisfied: cycler>=0.10 in
```

```
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from matplotlib) (0.10.0)
Requirement already satisfied: six in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from
cycler>=0.10->matplotlib) (1.15.0)
Requirement already satisfied: seaborn in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (0.11.1)
Requirement already satisfied: pandas>=0.23 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from seaborn) (1.2.4)
Requirement already satisfied: numpy>=1.15 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from seaborn) (1.19.5)
Requirement already satisfied: matplotlib>=2.2 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from seaborn) (3.3.4)
Requirement already satisfied: scipy>=1.0 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from seaborn) (1.6.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from
matplotlib>=2.2->seaborn) (2.4.7)
Requirement already satisfied: cycler>=0.10 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from
matplotlib>=2.2->seaborn) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from
matplotlib>=2.2->seaborn) (1.3.1)
Requirement already satisfied: pillow>=6.2.0 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from
matplotlib>=2.2->seaborn) (8.3.2)
Requirement already satisfied: python-dateutil>=2.1 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from
matplotlib>=2.2->seaborn) (2.8.1)
Requirement already satisfied: six in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from
cycler>=0.10->matplotlib>=2.2->seaborn) (1.15.0)
Requirement already satisfied: pytz>=2017.3 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from pandas>=0.23->seaborn)
(2021.1)
Collecting sklearn
  Using cached sklearn-0.0-py2.py3-none-any.whl
Requirement already satisfied: scikit-learn in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from sklearn) (0.24.1)
Requirement already satisfied: numpy>=1.13.3 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from scikit-learn->sklearn)
(1.19.5)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from scikit-learn->sklearn)
(2.1.0)
Requirement already satisfied: joblib>=0.11 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from scikit-learn->sklearn)
(1.0.1)
```

```
Requirement already satisfied: scipy>=0.19.1 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from scikit-learn->sklearn)
(1.6.2)
Installing collected packages: sklearn
Successfully installed sklearn-0.0
```

[9]:
```python
import numpy as np # everytime I call "np" the kernel understands "numpy"
import matplotlib.pyplot as plt # evertime I call "plt" the kernel understands␣
↪"matplotlib.pyplot"
from scipy import stats
import seaborn as sns; sns.set()
import sklearn
```

[14]:
```python
# Generate some dataset using SKLEARN

from sklearn.datasets import make_blobs

# the sub package datasets within sklearn has a function in which
# we can create datasets made of points (blobs)
```

[26]:
```python
# now we generate the dataset

X , y = make_blobs (n_samples = 100, # first we define the number of points
                    centers = 2,      # number of classes
                    cluster_std = 0.8)# standard deviation of the data
```

[27]:
```python
print(X)
```

```
[[ -8.39297718  -0.33496523]
 [ -7.85711898   1.01005544]
 [ -8.48855104   1.72715275]
 [  1.29583396   1.56418481]
 [  0.88840294   1.77740796]
 [ -0.23741854   0.84981091]
 [ -8.49851301   0.09431891]
 [ -9.8795934    1.50279029]
 [  2.17126885   1.27992086]
 [  2.3496711    1.36364099]
 [ -8.79203411   0.84270945]
 [ -7.47174704   1.89673019]
 [ -8.9281964    0.96439594]
 [  2.56106708   1.80105966]
 [ -8.4982567    0.66157709]
 [ -7.53904084   0.41043409]
 [ -7.42685306   1.46719359]
 [  0.93092993   2.28938255]
 [ -8.301917     0.63102994]
 [  2.68069347   1.21564931]
```

```
[ -8.29235778  -1.28422   ]
[  1.58680928   1.30600376]
[  2.25551565   1.33556006]
[  2.0066601    0.05869983]
[  2.5811785    2.2646207 ]
[  2.11215996   1.76221356]
[ -8.32118632   0.9705696 ]
[  1.3894115    1.62374328]
[ -8.59667508   0.88226113]
[ -9.17615997   0.12923971]
[  1.54702384   2.68184511]
[ -8.7166899    1.18713438]
[ -7.20502537   1.62983032]
[ -8.27236967   2.178396  ]
[  0.04380205   1.24694589]
[  1.74016323   2.09298349]
[  0.85765643   1.71975366]
[  2.41042389   1.44448195]
[ -9.03042081   0.41087721]
[  2.12023237  -0.07853032]
[ -8.47418881   0.27568784]
[  0.72525466   1.50768113]
[ -0.01562176   2.15088837]
[ -9.07316276  -0.23873019]
[ -7.34750437   0.8183989 ]
[  1.49629365   1.69402541]
[  0.69299168   1.62722525]
[ -8.09375048   2.03559795]
[  1.74156237   1.45562521]
[ -8.31000615   1.2331159 ]
[ -7.64921106   1.22010546]
[  0.53601981   0.35501926]
[  0.66603062  -0.02076456]
[ -8.71315176  -0.14195039]
[ -8.10892591   0.97377296]
[  1.09260064   1.1802545 ]
[ -8.17952907   2.18112821]
[  1.84469539   2.63850358]
[  0.15870064   2.10687792]
[ -8.03893475   1.3764203 ]
[  0.5963779    0.04867026]
[  2.57782064   1.06685514]
[  3.95383742   1.15444756]
[  2.26968518   0.55742431]
[ -8.25905374   1.00170354]
[  2.65086554   0.31928087]
[ -8.22925878   2.06254716]
[ -6.84929485  -0.08844469]
```

```
[ -8.13472006   1.08572591]
[  2.10212779   0.62256938]
[  1.59057737   1.53970799]
[ -8.21573665   1.78369623]
[ -9.42390872   1.01863893]
[ -8.42527562   0.22948207]
[  0.99420322   0.51250243]
[  0.90474053   1.28796154]
[ -8.29536433   1.78693483]
[ -8.07315117   0.3814427 ]
[ -9.55377076   0.9944138 ]
[  2.95745982   2.09886931]
[ -0.34884223  -0.23867254]
[  1.42381658   1.39863499]
[ -9.59454649   1.74020796]
[-11.32644585   2.46699745]
[ -7.8874499    1.88421551]
[ -0.1619124    2.17454672]
[  1.52366127   2.36693851]
[ -9.11008465   0.15712576]
[  1.21041666   1.45426962]
[  0.67277855   3.03340399]
[  3.14330866  -0.57315537]
[ -8.28140108   1.87301047]
[  1.33427973   1.2559591 ]
[-10.63801996   1.78811681]
[ -9.43634897   1.91276351]
[ -9.12290059   2.31346692]
[ -9.6139816    0.46762008]
[ -8.02447395   0.23918307]
[  1.73186966   1.15105054]
[  0.46400489   1.99953319]]
```

[28]:

```python
print(y)
```

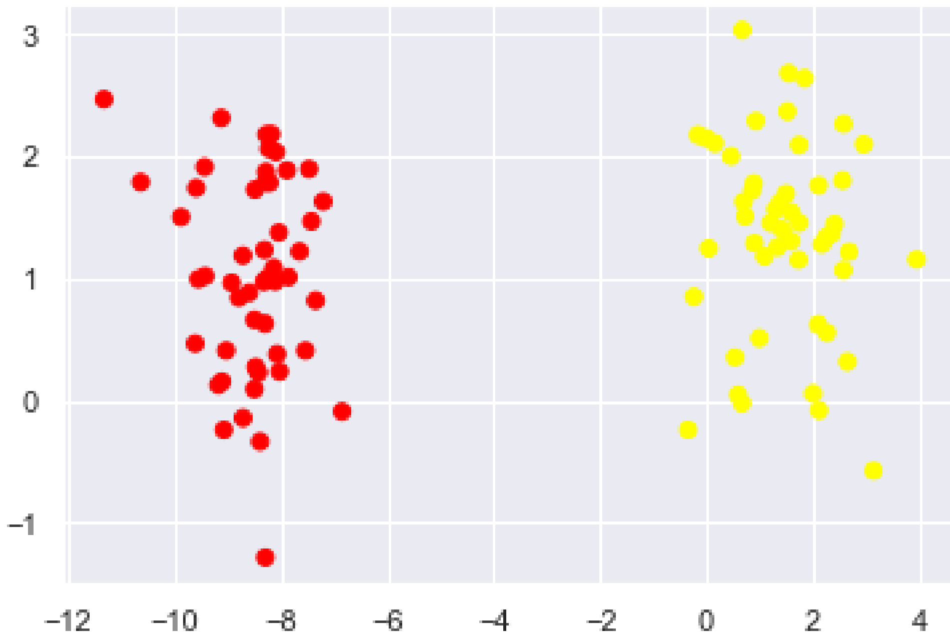
```
[0 0 0 1 1 1 0 0 1 1 0 0 0 1 0 0 0 1 0 1 0 1 1 1 1 1 0 1 0 0 1 0 0 0 1 1 1
 1 0 1 0 1 1 0 0 1 1 0 1 0 0 1 1 0 0 1 0 1 1 0 1 1 1 1 0 1 0 0 0 1 1 0 0 0
 1 1 0 0 0 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 0 0 0 0 1 1]
```

[31]:

```python
# graphical representation with "plt"

# plotting on the horizontal axis the first column of X "X[:,0]"
# and in the vertical axis the second column of X "X[:,1]"

# the color "c" is given by the value of "y"

# we can also choose a color palette "autumn"
```

```
plt.scatter(X[:,0], X[:,1], c=y, cmap='autumn')
```

[31]: <matplotlib.collections.PathCollection at 0x7fd6eed92ac0>

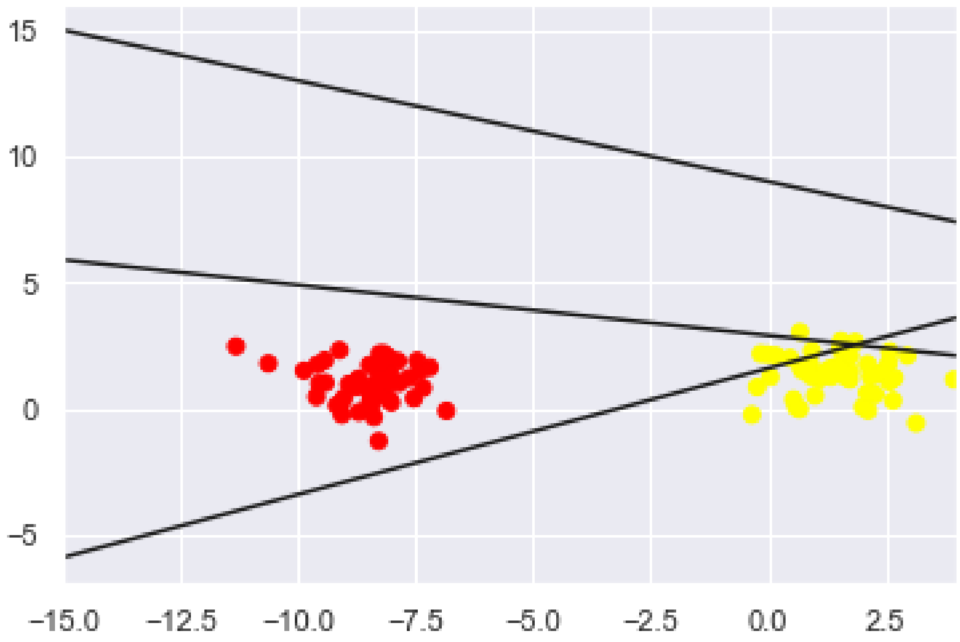


[32]:
```
# in reality we do not know what are the colors yet

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

[32]: <matplotlib.collections.PathCollection at 0x7fd6eef0a8b0>

[36]:
```
# Linear Suppor Vector Machines
# This algorithm is goint to try to draw a separation line
# between the two datasets which is optimal (maximal distance to all points)

# for a "2D" dataset we can do it by hand, but for many dimensions, it is more␣
↪difficult

xfit = np.linspace(-15,4)
plt.scatter(X[:,0], X[:,1], c=y, cmap='autumn')

for m, b in [(-0.4,9), (0.5, 1.6), (-0.2, 2.9)]:
    plt.plot(xfit, m*xfit+b, "-k")

plt.xlim(-15, 4)
```

[36]: (-15.0, 4.0)



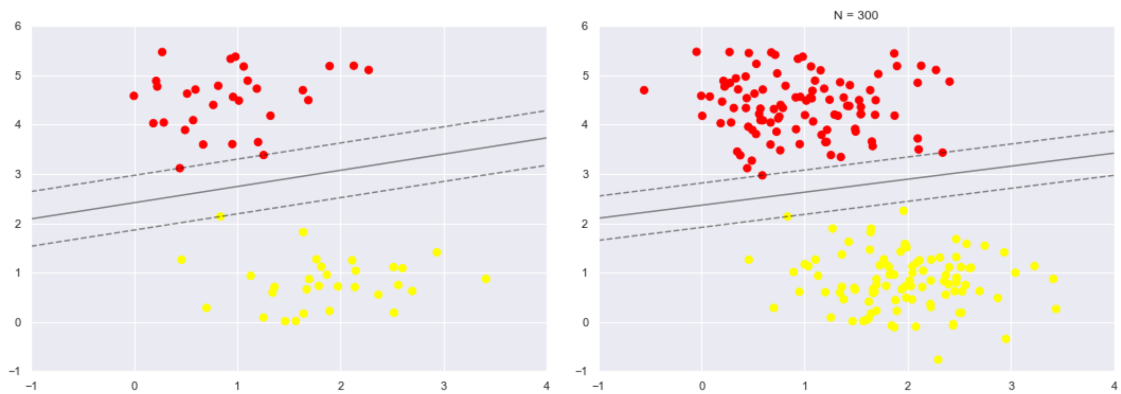
[37]:
```
# Support Vector Machines to calculate the line that best separates the dataset

from sklearn.svm import SVC # "SVC" means support vector classifier

model = SVC(kernel='linear')

model.fit(X,y)
```

[37]:
```
SVC(kernel='linear')
```

[43]:
```
# Plot the results of the model (not relevant for the exam)

def plot_svc_decision_function(model, ax=None, plot_support=True):
    if ax is None:
        ax=plt.gca()
    xlim=ax.get_xlim()
    ylim=ax.get_ylim()
    x=np.linspace(xlim[0],xlim[1],30)
    y=np.linspace(ylim[0],ylim[1],30)
    Y,X = np.meshgrid(y,x)
    xy=np.vstack([X.ravel(),Y.ravel()]).T
    P=model.decision_function(xy).reshape(X.shape)
    ax.contour(X, Y, P, colors='k',
               levels=[-1, 0, 1], alpha=0.5,
               linestyles=['--', '-', '--'])
```

```python
    if plot_support:
        ax:scatter(model.support_vectors_[:,0],
              model.support_vectors_[:,1],
              s=300, linewidth=1, facecolors='none');
    ax.set_xlim(xlim)
    ax.set_ylim(ylim)

def plot_svm(N=10, ax=None):
    X,y =make_blobs(n_samples=200, centers=2,
                   random_state=0, cluster_std=0.6)
    X=X[:N]
    y=y[:N]
    model=SVC(kernel='linear', C=1E10)
    model.fit(X,y)
    ax=ax or plt.gca()
    ax.scatter(X[:,0], X[:,1], c=y, s=50, cmap='autumn')
    ax.set_xlim(-1, 4)
    ax.set_ylim(-1, 6)
    plot_svc_decision_function(model,ax)

fig,ax=plt.subplots(1,2,figsize=(16,6))
fig.subplots_adjust(left=0.0625, right=0.95, wspace=0.1)
for axi, N in zip(ax, [60, 300]): plot_svm(N, axi)
axi.set_title('N = {0}'.format(N))
```

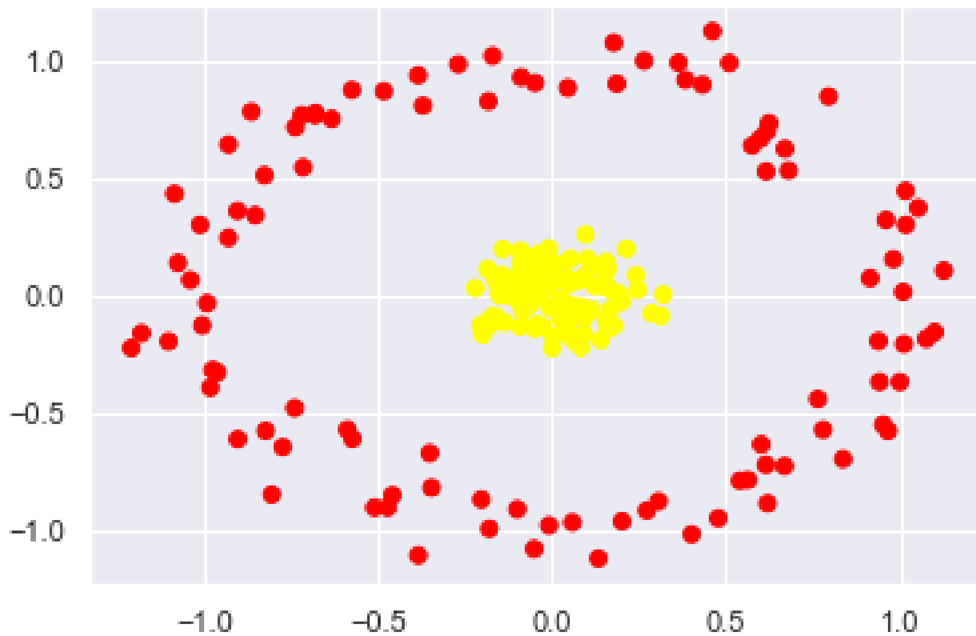
[43]: Text(0.5, 1.0, 'N = 300')



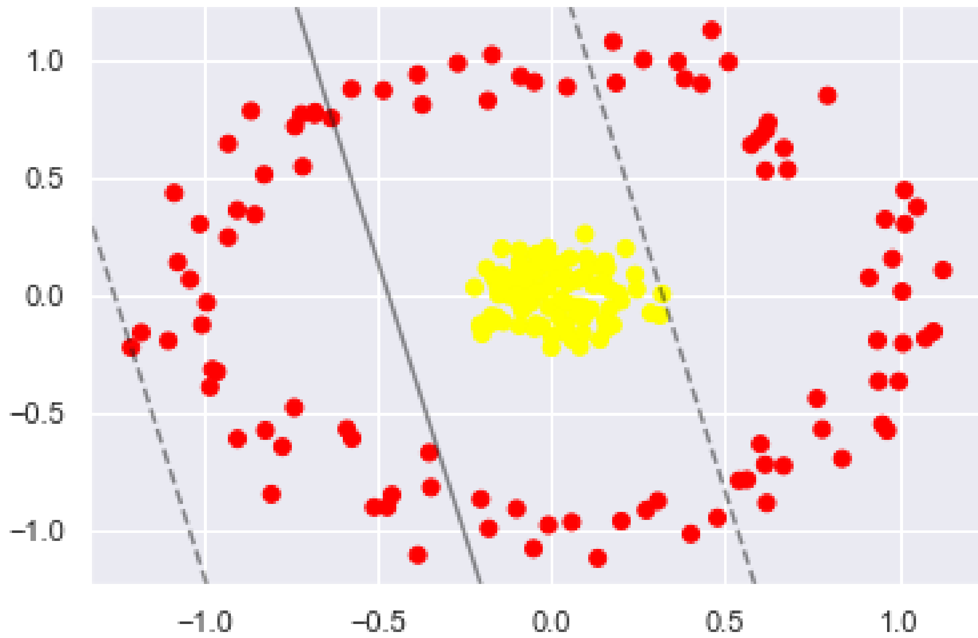
[44]:
```python
# what happens if the dataset cannot be separated with lines!?
# what happens if the dataset has a circular (non linear) shape?!

# then we need to use other kernel
```

[75]:
```
# lets create some non--linear dataset

from sklearn.datasets import make_circles

X , y = make_circles(200, factor =.1, noise=.1)

plt.scatter(X[:,0], X[:,1], c=y, cmap='autumn')
```

[75]:
```
<matplotlib.collections.PathCollection at 0x7fd6f06c2a60>
```



[78]:
```
# now we try to separate these dataset with a linear kernel

clf = SVC(kernel='linear').fit(X,y) # clf means "classifier linear function"

plt.scatter(X[:,0], X[:,1], c=y, cmap='autumn')
plot_svc_decision_function(clf, plot_support=False);
```

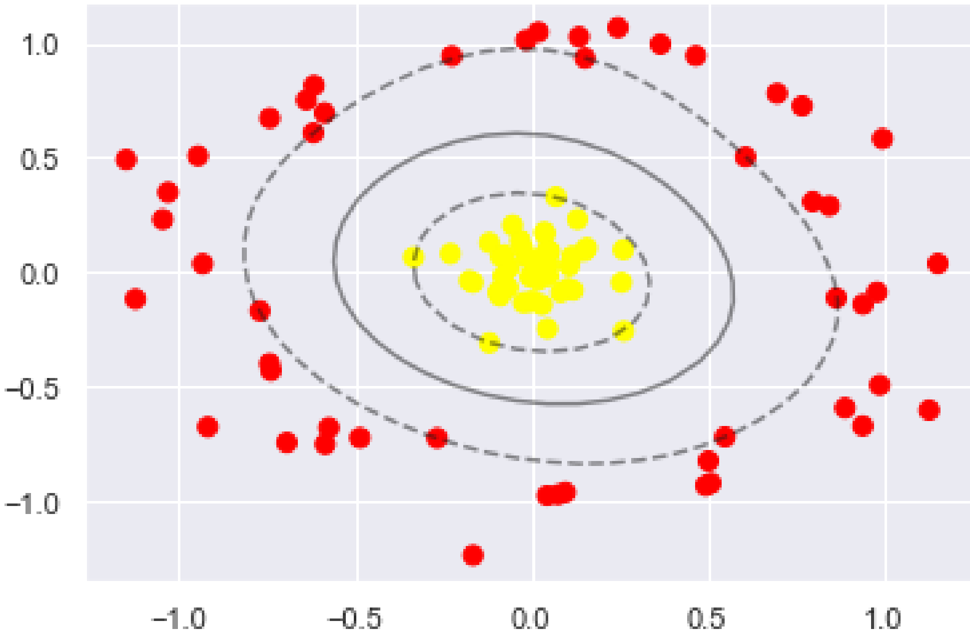


[72]:
```
# our linear classifier is not succeeding in separating the data

# we need to define another kernel

# Radial basis function (circles)

r = np.exp(-(X**2).sum(1))
```

[73]:
```
# now we create another kernel based on the RBF

rbf = SVC(kernel='rbf', C=1E6)
rbf.fit(X,y)
```

[73]:
```
SVC(C=1000000.0)
```

[74]:
```
plt.scatter(X[:,0], X[:,1], c=y, s=50, cmap='autumn')
plot_svc_decision_function(rbf)
```



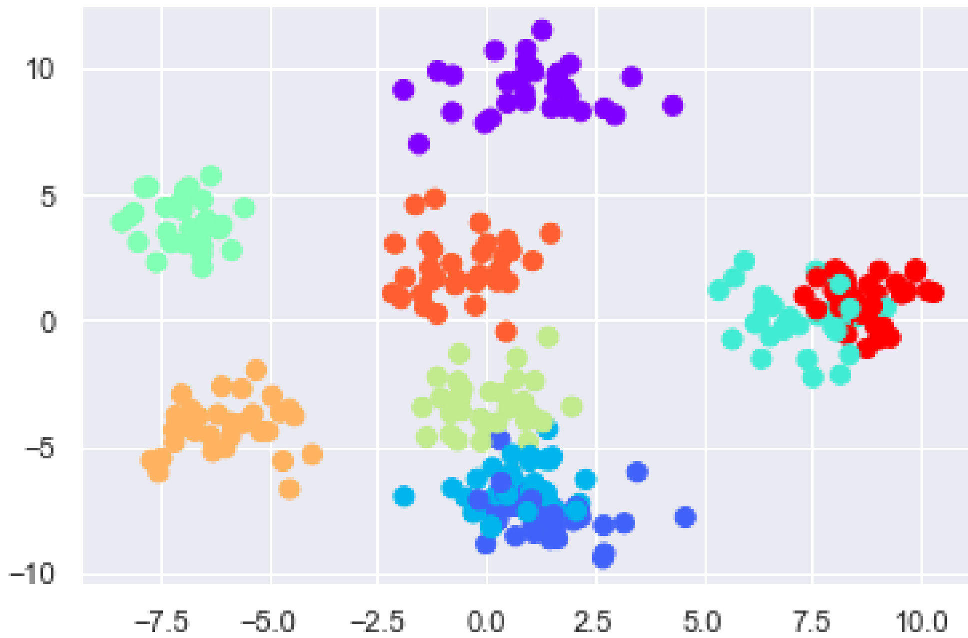
[79]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
```

[80]:
```
# Decission Trees. Separation of many dimensional datasets
```

[81]:
```
# dataset with more than 2 centers
```

[89]:
```
from sklearn.datasets import make_blobs

X, y = make_blobs(n_samples=300,
                  centers=9,
                  cluster_std=1)

plt.scatter(X[:,0], X[:,1], c=y, s=50, cmap='rainbow')
```

[89]: <matplotlib.collections.PathCollection at 0x7fd6ef0fea90>



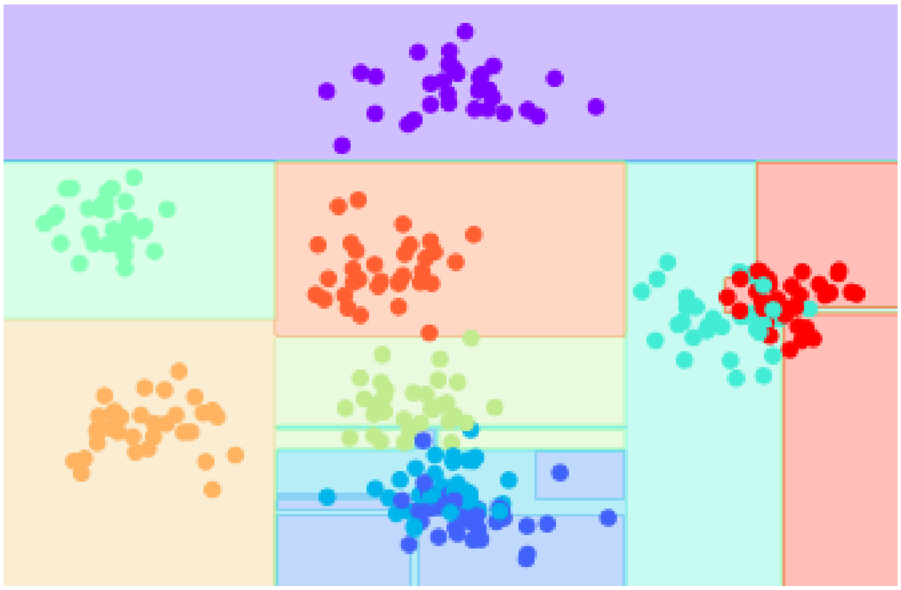
[90]:
```python
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier().fit(X,y)
```

[91]:
```python
# visualization is not relevant for the exam

def visualize_classifier(model, X, y, ax=None, cmap='rainbow'):
    ax=ax or plt.gca()
    # Plot the training points
    ax.scatter(X[:, 0], X[:, 1], c=y, s=30, cmap=cmap,
               clim=(y.min(), y.max()), zorder=3)
    ax.axis('tight')
    ax.axis('off')
    xlim=ax.get_xlim()
    ylim=ax.get_ylim()
    # fit the estimator
    model.fit(X,y)
    xx, yy = np.meshgrid(np.linspace(*xlim, num=200),
                         np.linspace(*ylim, num=200))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)
    # create a color plot with the results
    n_classes = len(np.unique(y))
    contours=ax.contourf(xx,yy,Z, alpha=0.3,
                         levels=np.arange(n_classes+1)-0.5,
                         cmap=cmap, clim=(y.min(),y.max()),
                         zorder=1)
```

```
    ax.set(xlim=xlim, ylim=ylim)
```

[92]:
```
visualize_classifier(DecisionTreeClassifier(),X,y)
```

```
<ipython-input-91-da4776368e8b>:19: UserWarning: The following kwargs were not
used by contour: 'clim'
  contours=ax.contourf(xx,yy,Z, alpha=0.3,
```



[88]:
```
# ABC - XYZ Analysis -- here we described several strategies depending on
# the amount of value of products and the frequency of these products
```

[93]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.tree.
 ↪DecisionTreeClassifier.html
```

[94]:
```
# homework: find or create three dimensional dataset X (three columns) and y␣
 ↪(classes)
# that is distributed linearly.
# apply decission tree classifier to it and visualize the results
```

[ ]:
```

```