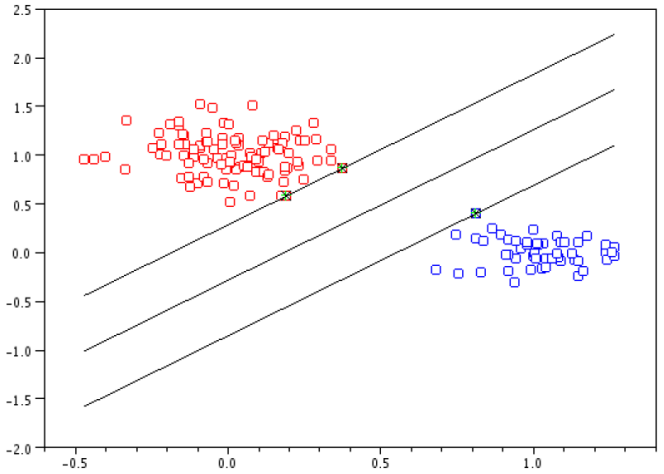
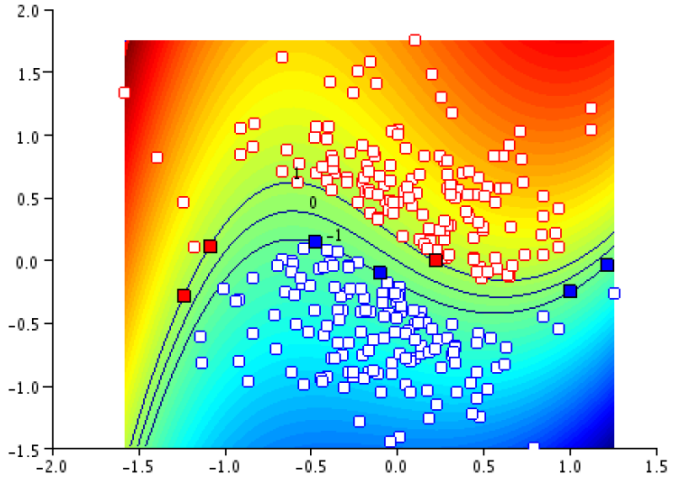




Stéphane GAZUT

Ingénieur-Chercheur

CEA, LIST, Laboratoire Analyse de Données et Intelligence des Systèmes

CEA Saclay – Gif-sur-Yvette

stephane.gazut@cea.fr

TP N°3 - SVM

Université Paris-Saclay

Master SETI - INSTN

2021-2022

TP N°3 - SVM

Objectifs du TP

On se propose de faire de la modélisation par SVM sur des problématiques de discrimination, en utilisant la bibliothèque scikit-learn.

- Traiter un problème de discrimination linéairement séparable
- Traiter un problème non linéairement séparable (SVM à noyaux)
- Traiter le problème de la discrimination de chiffres manuscrits

Problème linéairement séparable

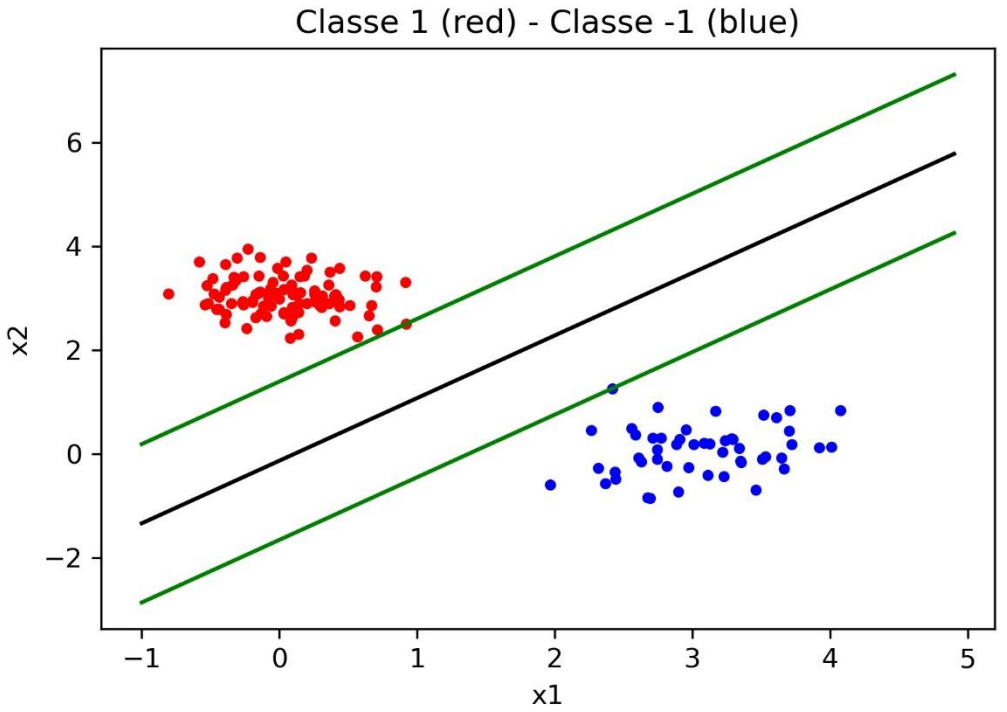
PROBLÈME LINÉAIREMENT SÉPARABLE

- **Vous devez créer un script dans lequel il y aura:**

  - Une étape de création des données du problème linéairement séparable (création de deux nuages de points gaussiens distants)

  - L'étape de création du modèle SVM linéaire

  - Des fonctions de visualisation qui font apparaitre
    - Les deux classes
    - La séparatrice
    - La marge



GÉNÉRATION DES POINTS

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.svm import SVC


def genere_ex_1(n1=100, n2=50, mu1=[0,3], mu2=[3,0], sd1=0.15, sd2=0.2):
    X = np.concatenate((np.random.multivariate_normal(mu1, np.diagflat(sd1*np.ones(2)), n1),
                        np.random.multivariate_normal(mu2, np.diagflat(sd2*np.ones(2)), n2)))

    Y = np.concatenate((np.ones((n1,1)), -1*np.ones((n2,1))))[:,0]

    return X, Y
```

- **Proposition de fonction pour générer le premier set d'exemples**
  - Deux nuages gaussiens dans R^2 avec pour vecteur de moyenne mu1 et mu2
  - Des matrices de covariance isovariées d'éléments diagonaux sd1 et sd2
  - Les deux nuages contiennent respectivement n1 et n2 points

CRÉATION DU SVM LINÉAIRE

```
def main(X, Y):
    classifier = SVC(kernel='linear', probability=True)
    classifier = classifier.fit(X, Y)

#     w = classifier.coef_[0]
#     b = classifier.intercept_[0]

    plot_data_hyperplan(X, Y, classifier, 'Graph_SVM_linear')
```

On peut spécifier que l'on souhaite l'estimation des probabilités. C'est la méthode de Platt vue en cours qui est utilisée.

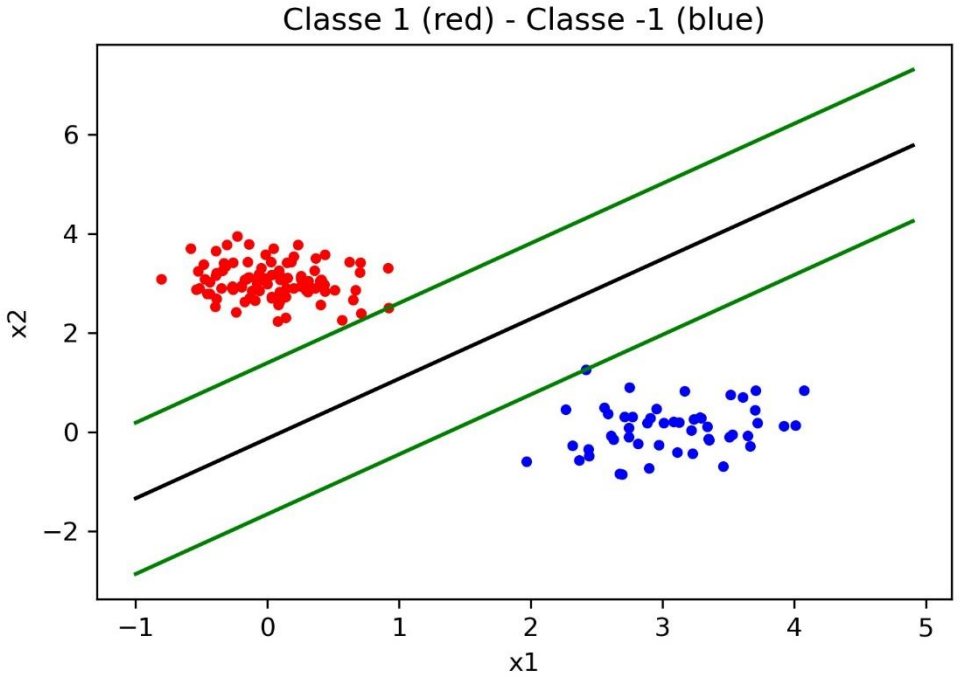
Doit être spécifier à la création du classifieur

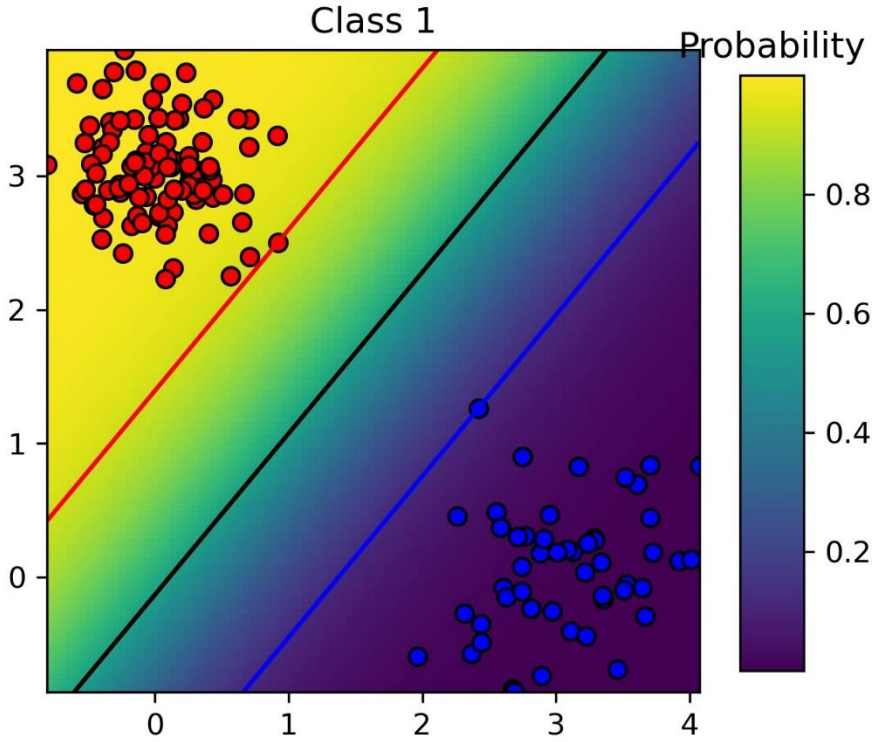
Pour info, les paramètres w et b vus en cours sont coef_ et intercept_ dans l'objet classifieur

- Dans ce premier cas simple du TP il faut juste créer le classifieur SVM en spécifiant le noyau linéaire
- Déterminer les paramètres du modèles avec la fonction fit de l'objet classifieur
- Faire la visualisation

VISUALISATION

- À partir du classifieur obtenu et des informations du cours sur les équations de l'hyperplan et de la marge, vous devez faire une fonction de visualisation qui permet de voir les points, la frontière de décision et la marge.





Visualisation en exploitant la probabilité.

Problème non linéairement séparable

PROBLÈME NON LINÉAIREMENT SÉPARABLE

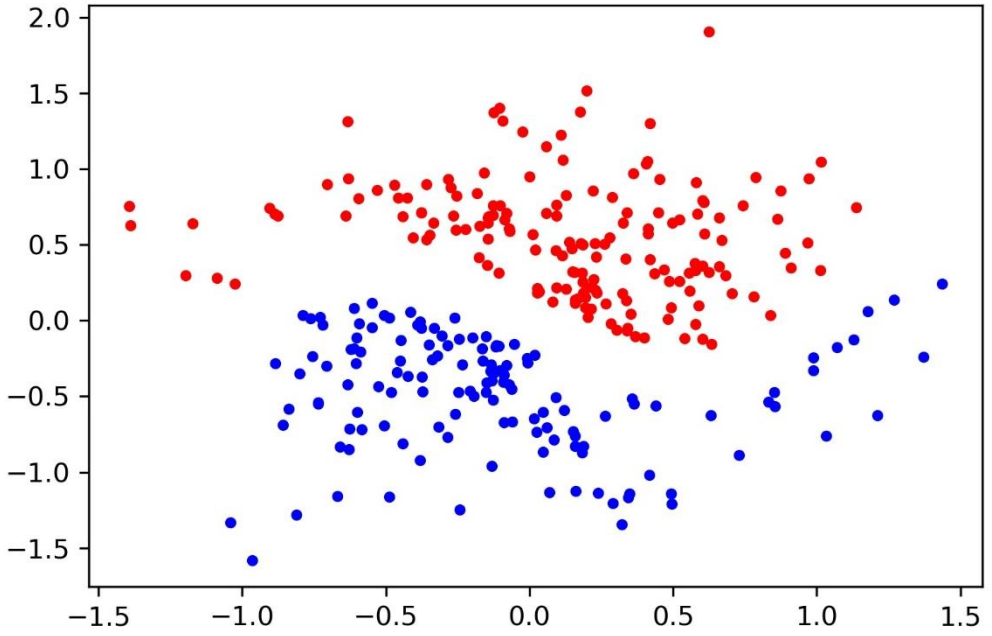
- Pour cet exemple un peu plus complexe, vous devrez avoir une démarche de recherche de paramètre de noyau via, par exemple, l'utilisation de **GridSearchCV** de **sklearn.model_selection**

- **Même structure de script:**
  - Création des données
  - Création du modèle SVM non linéaire
  - Visualisation

LES DONNÉES

```python
import matplotlib.pyplot as plt
import numpy as np
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV


def genere_ex_2(n=300, mu=[0,0], std=0.25, delta=0.2):
    X = np.random.multivariate_normal(mu, np.diagflat(std*np.ones(2)), n)
    Y = np.zeros((X.shape[0]))
    for i in range(X.shape[0]):
        x = X[i,0]
        y = X[i,1]
        if y < x*(x-1)*(x+1):
            Y[i] = -1
            X[i,1] = X[i,1] - delta
        else:
            Y[i] = 1
            X[i,1] = X[i,1] + delta
    return X,Y
```

- Pour les données, nous générons un nuage gaussien centré en 0, traversé par une fonction polynomiale de degré 3 $x(x-1)(x+1)$:
  - On ajoute delta à l'ordonnée des points au-dessus de cette frontière.
  - On retranche delta à l'ordonnée des points en-dessous de cette frontière.



CRÉATION DU MODÈLE

1. Utilisez la fonction GridSearchCV pour rechercher le meilleur paramétrage (noyau/paramètre):
   - De la documentation est disponible sur le site de scikit-learn.
   - Vous pouvez vous limiter aux noyaux de type ‘poly’ et ne chercher que « degree » et « C »

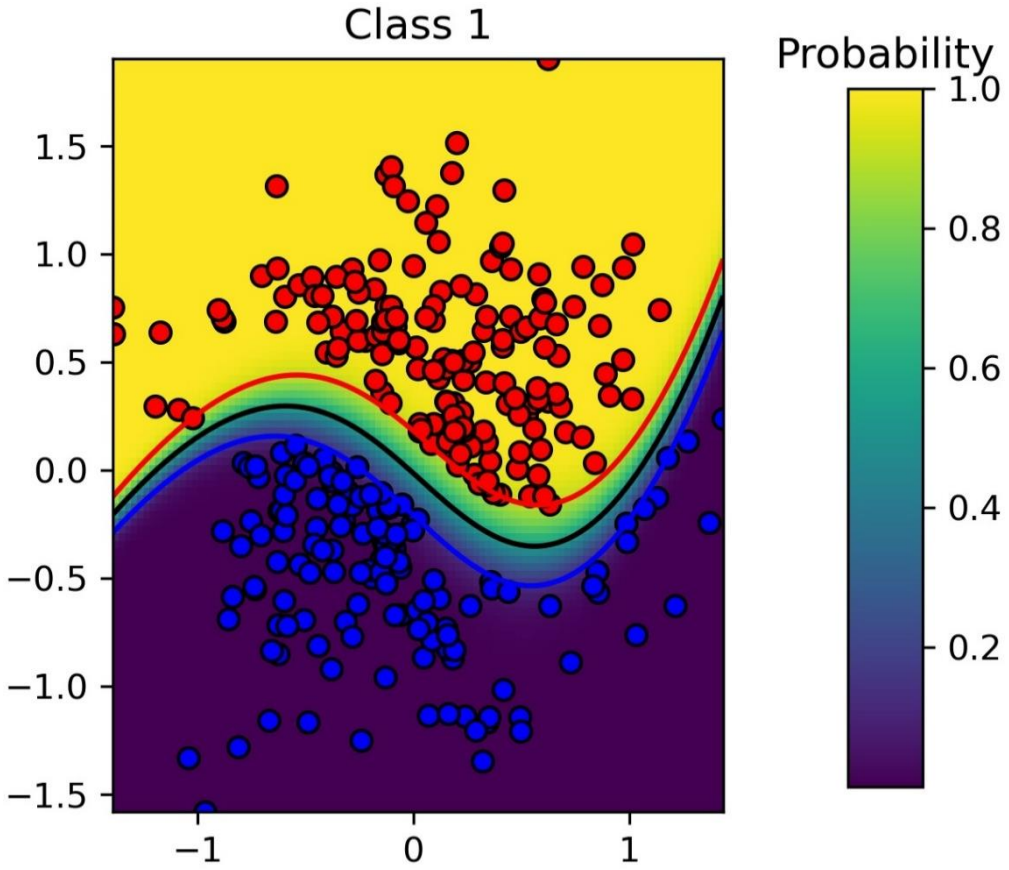
   - **Remarque**: Prenez garde aux paramètres de la fonction SVC de scikit-learn. Certains paramètres sont exclusifs à certains noyaux. Par exemple le paramètre coef0 n’est utile que pour les noyaux de type polynôme et sigmoïde. Pour le noyau polynomial, il correspond à la valeur de seuil fixé à 1 dans la formulation générale:

$$\text{polynomial } k(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x}^T \mathbf{y})^d$$

   - Ce terme, nommé « independent term » est fixé par défaut à 0 (cas particulier des noyaux dits homogènes). Dans notre cas, il devra être fixé à 1.

2. Créer le modèle correspondant au paramétrage renvoyé par la fonction GridSearchCV

VISUALISATION



- **Faire apparaitre:**
  - Les points des deux classes
  - La séparatrice
  - La marge
  - Le code couleur de la probabilité si probability=True a été utilisé dans le script.

VISUALISATION

- Ce type de graphique est documenté dans le site de scikit-learn

- Pour cela nous devons créer un maillage de points (comme pour la surface sinus cardinal du TP n°2) et estimer la prédiction du SVM et la probabilité d'appartenance à l'une des deux classes en chacun des points du maillage.
  - La frontière et les marges correspondent aux lignes de niveaux [-1, 0, 1] de la surface d'estimation
  - Le fond de couleur de la figure est indexé sur les valeurs de la surface de probabilités.

```
...
minx1 = min(X[:,0])
maxx1 = max(X[:,0])
minx2 = min(X[:,1])
maxx2 = max(X[:,1])

xx = np.linspace(minx1, maxx1, 100)
yy = np.linspace(minx2, maxx2, 100).T
xx, yy = np.meshgrid(xx, yy)
Xfull = np.c_[xx.ravel(), yy.ravel()]

probas = classifier.predict_proba(Xfull)
Z = classifier.decision_function(Xfull)
...
```

Vous pourrez utiliser plt.imshow(…) pour le fond et plt.contour pour la frontière et la marge correspondant aux iso-contours de niveau -1, 0 et 1.

Reconnaissance de chiffres manuscrits

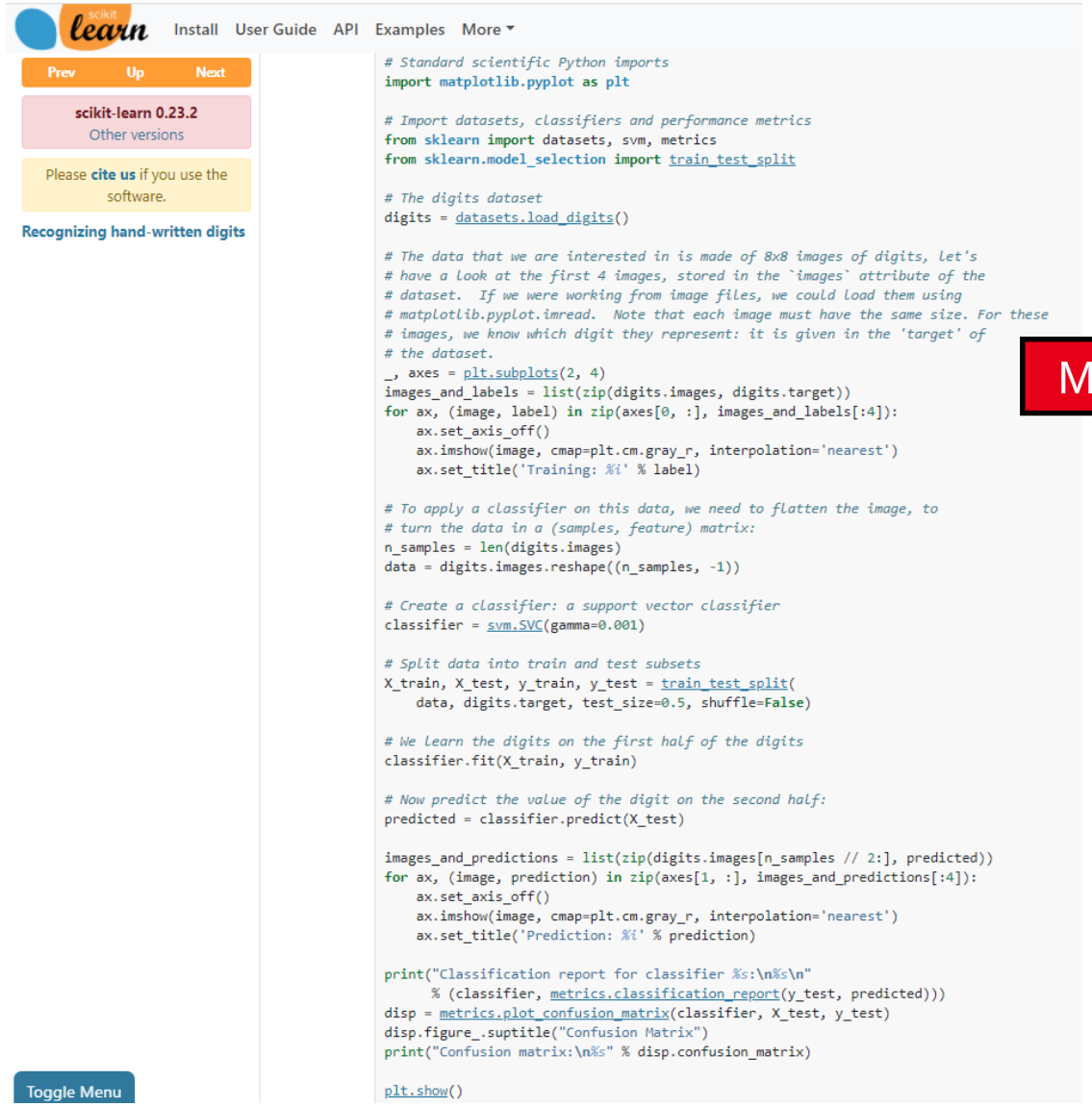
RECONNAISSANCE DE CHIFFRES MANUSCRITS

- Les données de cet exemple que nous utilisons depuis longtemps en TP avec M. Martinez (données « mnist handwritten digits dataset » de Yann LeCun est très bien décrit et documenté dans le site de scikit-learn avec l'utilisation des SVM. Les sources sont disponibles:

  https://scikit-learn.org/stable/auto_examples/classification/plot_digits_classification.html

- Je vous demanderai donc, d'utiliser ces codes mais en ajoutant certaines étapes qui ne sont pas implémentées et qui correspondent aux étapes attendues lorsque l'on traite d'un problème de machine learning:
  - Choix du bon modèle (recherche du noyau adapté et de la valeur de C adaptée). Le modèle utilisé dans l'exemple de scikit-learn est le noyau par défaut de SVC (le noyau RBF).
  - Interprétation des résultats

RECONNAISSANCE DE CHIFFRES MANUSCRITS




```python
# Standard scientific Python imports
import matplotlib.pyplot as plt

# Import datasets, classifiers and performance metrics
from sklearn import datasets, svm, metrics
from sklearn.model_selection import train_test_split

# The digits dataset
digits = datasets.load_digits()

# The data that we are interested in is made of 8x8 images of digits, let's
# have a look at the first 4 images, stored in the `images` attribute of the
# dataset.  If we were working from image files, we could load them using
# matplotlib.pyplot.imread.  Note that each image must have the same size. For these
# images, we know which digit they represent: it is given in the 'target' of
# the dataset.
_, axes = plt.subplots(2, 4)
images_and_labels = list(zip(digits.images, digits.target))
for ax, (image, label) in zip(axes[0, :], images_and_labels[:4]):
    ax.set_axis_off()
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
    ax.set_title('Training: %i' % label)

# To apply a classifier on this data, we need to flatten the image, to
# turn the data in a (samples, feature) matrix:
n_samples = len(digits.images)
data = digits.images.reshape((n_samples, -1))

# Create a classifier: a support vector classifier
classifier = svm.SVC(gamma=0.001)

# Split data into train and test subsets
X_train, X_test, y_train, y_test = train_test_split(
    data, digits.target, test_size=0.5, shuffle=False)

# We learn the digits on the first half of the digits
classifier.fit(X_train, y_train)

# Now predict the value of the digit on the second half:
predicted = classifier.predict(X_test)

images_and_predictions = list(zip(digits.images[n_samples // 2:], predicted))
for ax, (image, prediction) in zip(axes[1, :], images_and_predictions[:4]):
    ax.set_axis_off()
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
    ax.set_title('Prediction: %i' % prediction)

print("Classification report for classifier %s:\n%s\n"
      % (classifier, metrics.classification_report(y_test, predicted)))
disp = metrics.plot_confusion_matrix(classifier, X_test, y_test)
disp.figure_.suptitle("Confusion Matrix")
print("Confusion matrix:\n%s" % disp.confusion_matrix)

plt.show()
```

MAIN

Il suffit de mettre les images en ligne et les pixels en colonne, et le problème se résout comme les autres exemples vus avant dans le TP.

```python
import matplotlib.pyplot as plt
from sklearn import datasets, svm, metrics
from sklearn.model_selection import train_test_split

digits = datasets.load_digits()

#... VISUALISATION DE QUELQUES EXEMPLES DE CHIFFRES

# To apply a classifier on this data, we need to flatten the image, to
# turn the data in a (samples, feature) matrix:
n_samples = len(digits.images)
data = digits.images.reshape((n_samples, -1))

# Create a classifier: a support vector classifier
classifier = svm.SVC(gamma=0.001)

# Split data into train and test subsets
X_train, X_test, y_train, y_test = train_test_split(
    data, digits.target, test_size=0.5, shuffle=False)

# We learn the digits on the first half of the digits
classifier.fit(X_train, y_train)

# Now predict the value of the digit on the second half:
predicted = classifier.predict(X_test)

#... VISUALISATION ET RAPPORT - MATRICE DE CONFUSION
```

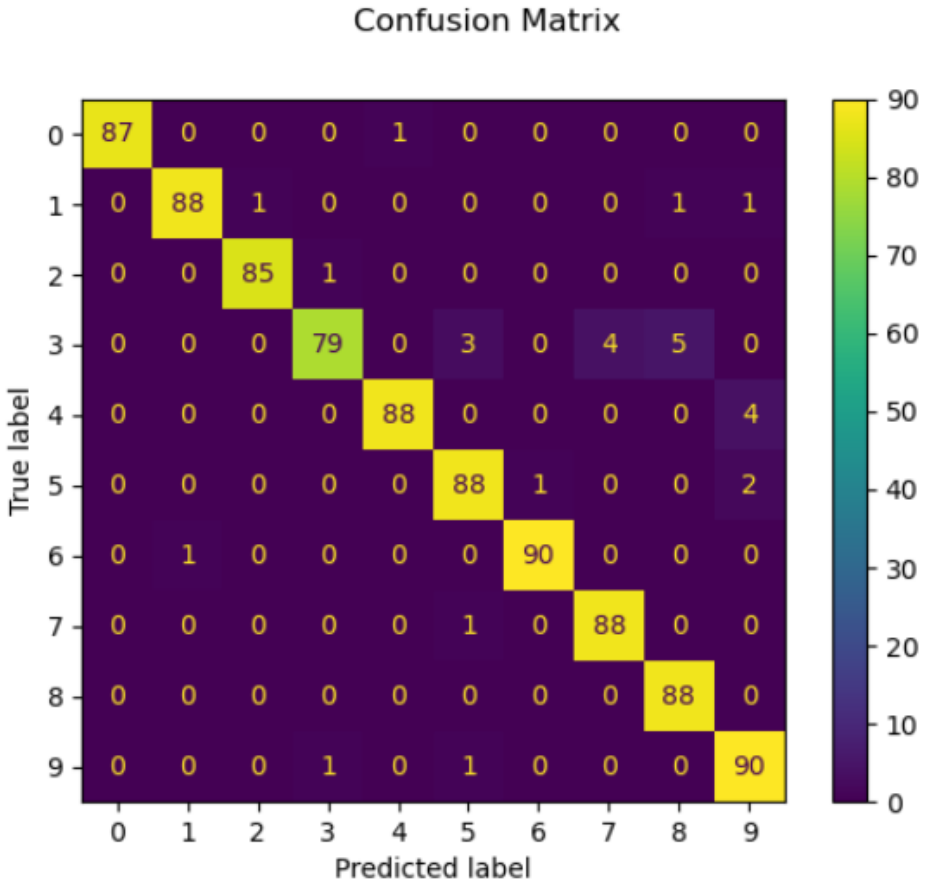
RECONNAISSANCE DE CHIFFRES MANUSCRITS

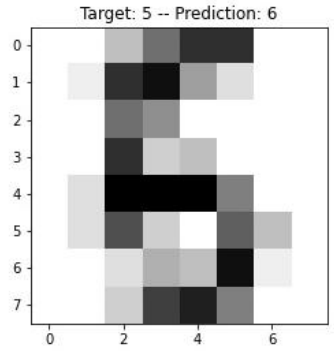
```
import matplotlib.pyplot as plt
from sklearn import datasets, svm, metrics
from sklearn.model_selection import train_test_split

digits = datasets.load_digits()

#... VISUALISATION DE QUELQUES EXEMPLES DE CHIFFRES

# To apply a classifier on this data, we need to flatten the image, to
# turn the data in a (samples, feature) matrix:
n_samples = len(digits.images)
data = digits.images.reshape((n_samples, -1))

# Create a classifier: a support vector classifier
classifier = svm.SVC(gamma=0.001)

# Split data into train and test subsets
X_train, X_test, y_train, y_test = train_test_split(
    data, digits.target, test_size=0.5, shuffle=False)

# We learn the digits on the first half of the digits
classifier.fit(X_train, y_train)

# Now predict the value of the digit on the second half:
predicted = classifier.predict(X_test)

#... VISUALISATION ET RAPPORT - MATRICE DE CONFUSION
```

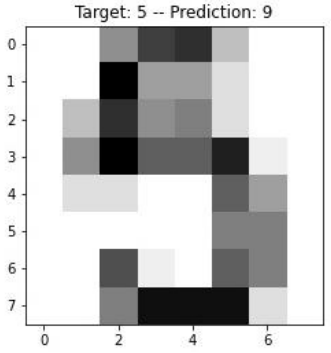
- Ajoutez au bon endroit du script la fonction GridSearchCV qui vous permettra de déterminer le bon noyau (poly ? RBF ? Sigmoid ?) et la bonne valeur de C.

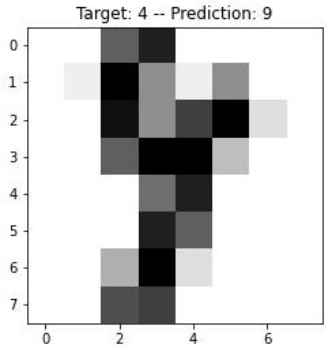
INTERPRÉTATION DES RÉSULTATS

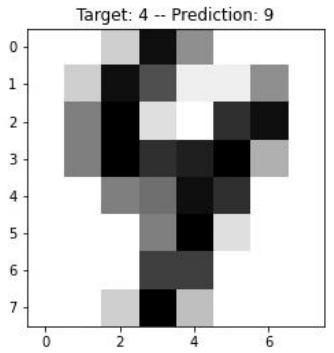
- Les SVM sont très performants sur ce problème.
- Dans une démarche réelle de modélisation, il faut interpréter les résultats.
- Vous pouvez identifier dans les données les cas mal classés (attention avec les index; première moitié de la base en apprentissage et l'autre en test avec shuffle à False).
- Vous pourrez donc visualiser les cas mal classés en reprenant les fonctions de visualisation des images de chiffres manuscrits contenus dans le script. Exemple:











Commissariat à l'énergie atomique et aux énergies alternatives
Institut List | CEA SACLAY NANO-INNOV | BAT. 861 – PC142
91191 Gif-sur-Yvette Cedex - FRANCE
www-list.cea.fr

Établissement public à caractère industriel et commercial | RCS Paris B 775 685 019