Ingénieurs spécialité instrumentation 2024-2025
(2e année)

TD - TP 3 Apprentissage non supervisé

Clustering, détection d'anomalie, réduction de dimensionnalité

Environnement de travail

Un compte-rendu individuel est demandé à la fin de chaque séance. Toutes les simulations seront réalisées sous l'environnement Python sur PC. Vous serez amenés à utiliser plusieurs bibliothèques scientifiques notamment numpy, scipy et sklearn, matplotlib… Vous utiliserez l'éditeur Spyder, qui précharge les modules scientifiques.
Dans Spyder, vous pouvez entrer vos commandes directement dans la console, ou bien créer un script et l'exécuter ou n'exécuter que les lignes sélectionnées. Une aide en ligne est aussi disponible via l'inspecteur d'objet, qui est automatiquement activé lorsque vous entrez des commandes dans l'éditeur ou la console. Vous disposez également d'une aide en ligne par help (nom_de_la fonction).
L'objectif de ce TP est de vous familiariser avec les algorithmes de régression et de classification du machine learning. Vous travaillerez avec les modules de la librairie Scikit Learn. La documentation est disponible via le lien suivant : https://scikit-learn.org/stable/. La plupart des modèles et algorithmes de machine learning sont déjà implémentés avec une architecture orientée objets dans laquelle chaque modèle dispose de sa propre classe. La figure ci-dessous présente un résumé des différents algorithmes implémentés dans la librairie Scikit Learn.

Exercice 1 : Clustering (K-means clustering)

Dans le contexte de l'apprentissage non supervisé, on demande à l'algorithme d'analyser la structure du jeu de données pour apprendre de lui-même à réaliser des tâches particulières par exemple le clustering. Le clustering permet de classer des documents, de classer des photos, des tweets, de segmenter la clientèle d'une société. Le clustering apprend à classer des données selon leurs ressemblances. C'est un algorithme itératif qui cherche à minimiser la distance entre les points d'un cluster (minimiser la variance des clusters). Vous utiliserez dans cet exercice l'algorithme du K-means.

1) Générez et affichez le jeu de données sur lequel vous serez amené à travailler :

```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=100, centers=3, n_features=2,random_state=0)
plt.scatter(X[:,0],X[:,1])
```

2) On souhaite utiliser le modèle ci-dessous :

a) Importez les modules nécessaires à la réalisation de cet exercice à savoir :

```
from sklearn.cluster import KMeans
```

b) Donnez les caractéristiques des hyperparamètres du modèle :

```
model = KMeans (n_clusterq = 3)
```

- la signification de n_clusterq = 3.

- le nombre d'initialisation fixé par défaut.
- le nombre d'itérations maximal utilisé par défaut.
- la méthode d'initialisation utilisée.
- comment sont choisis les points des centres ?

3) Entrainez ce modèle (model.fit(X)).

4) Visualisez la prédiction du modèle :

```
model.predict(X))
plt.scatter((X[ :,0], X[ :,1], c=model.predict(X))
```

5) Affichez les centroides en utilisant :

```
model.cluster_centers_
model.cluster_centers_[:,0], model.cluster_centers_[:,1])
plt.scatter(model.cluster_centers_[:,0], model.cluster_centers[:,1])
```

6) A quoi servent ces centroides ?

7) Calculez le coût engendré par ce modèle en utilisant model.inertia_. Commet est-il calculé ?

8) Evaluez le modèle en utilisant cette fois-ci model.score(X). Comparez le résultat obtenu à la question précédente.

9) On souhaite choisir le bon nombre de cluster en utilisant la technique « Elbow method ». Cette méthode consiste à détecter une « zone de coude » sur la courbe coût en fonction du nombre de cluster. Déterminez cette « zone de coude » en vous appuyant sur le code ci-dessous que vous commenterez :

```
inertia =[]
K_range = range (1,20)
for k in K_range:
        model = KMeans(n_clusters = k).fit(X)
        inertia.append(model.inertia_)
plt.plot(K_range,inertia)
plt.xlabel('nombre de clusters')
plt.ylabel('Cout du modele (Inertia)'
```

Remarque : D'autres méthodes de clustering existes notamment lorsque les données sont non convexes et anisotropiques : DBSCAN et AgglomeativeClustering.

Exercice 2 : Détection d'anomalie (Isolation Forest algorithm)

1) Générez et affichez le jeu de données sur lequel vous serez amené à travailler :

```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=50, centers=1, cluster_std=0.1, random_state=0)
X[-1, :]= np.array([2,255])
plt.scatter(X[:,0],X[:,1])
```

2) L'algorithme isolation forests (générateur d'arbres) réalise une série de découpe (split) aléatoire (horizontale et/ou verticale) jusqu'à isoler un point (anomalie).

a) Importez la librairie :

from sklearn.esnsemble import IsolationForest

b) Sélectionnez le modèle « IsolationForest » :

model = IsolationForest(contamination =0.01)

Que représente contamination =0.01 ?

c) Entrainez le modèle (model.fit(X)) puis affichez le résultat : plt.scatter(X[:,0], X[:,1], c = model.predict(X))

d) Modifiez l'hyperparamètre contamination, puis analysez l'impact de ce changement sur l'entrainement du modèle.

Exercice 3 : Détection des chiffres manuscrits mal écrits dans la base « digits » (Isolation Forest algorithm)

On souhaite détecter dans le jeu de données "digits" de chiffre manuscrit, retirer les chiffres mal écrits qui peuvent prêter à confusion.

1) Importez la librairie pour télécharger le jeu de données :

from sklearn.datasets import load_digits

2) Téléchargez le jeu de données correspondant à des chiffres manuscrits représentés par des images :

digits = load_digits()

Images = digits.images

3) Récupérez les images et les targets comme suit :

X = digits.data

y = digits.target

4) En vous appuyant sur print(X, shape) :

a) Déduisez la taille de X.

b) Quelle est la taille de chaque image ?

c) Visualisez quelques images en vous utilisant plt.imshow(images[10]) par exemple.

5) Sélectionnez le modèle IsolationForest :

model =IsolationForest(random_state=0, contamination = 0.02)

6) Entrainez le modèle : model.fit(X).

7) Evaluez le modèle : model.predict(X). Interprétez les résultats.

8) Filtrez toutes prédictions égales à -1 (outliers = model.prect(X)==-1), puis analysez le contenu de outliers.

9) Sélectionnez la première image en utilisant Images[outliers][0]) puis affichez la (plt.imshow(Images[outliers][0])). Interprétez la décision de l'algorithme.

10) Modifiez l'hyperparamètre contamination, puis analysez l'impact de ce changement sur l'entrainement du modèle.

Exercice 4 : Réduction de la dimensionnalité (Analyse en composantes principales, (PCA pour Principale Components Analysis))

L'objectif de cet exercice est d'étudier la réduction de la dimension du jeu de données « digits » tout en minimisant la perte de qualité dans deux cas de figures : la visualisation et la compression de données.

A) Téléchargement du jeu de données « digits »

1) Importez la librairie pour télécharger le jeu de données :

```
from sklearn.datasets import load_digits
```

2) Téléchargez le jeu de données correspondant à des chiffres manuscrits représentés par des images :

```
digits = load_digits()
images = digits.images
```

3) Récupérez les images et les targets comme suit :

```
X = digits.data
y = digits.target
```

4) Donnez les dimensions de ce jeu de données.

B) Visualisation des données

L'objectif de cette partie est de visualiser les 64 variables dans un espace à 2 dimensions.

1) Chargez le transformer PCA depuis le module "decomposition" :

```
from sklearn.decomposition import PCA
```

2) Sélectionnez le modèle PCA puis fixez à 2 le nombre de dimension sur lequel vous souhaitez projeter les données :

```
model = PCA(n_components = 2)
```

3) Entrainez le modèle :

```
X_reduced = model.fit_transform(X)
```

4) Vérifiez la dimension du tableau X_reduced.

5) Observez les composantes de X_reduced en vous appuyant sur :

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

6) Ajoutez des couleurs au graphique

```
plt.scatter(X_reduced[:,0], X_reduced[:,1], c=y)
```

Que contient y ?

7) Ajoutez une barre des couleurs pour analyser la visualization :

```
plt.colorbar()
```

8) Interprétez cette visualisation.

9) Quelle est la signification des axes ?

10) Analyser le contenu de chaque composantes :

```
model.components_.shape
```

C) Compression de données

1) Commencez par entrainer le modèle sur le même nombre de dimension (d) que X :

```
model = PCA(n_components = d)
```

2) Entrainez le model :

```
X_reduced = model.fit_transform(X)
```

3) Examinez le pourcentage de variance préserver par chacune des composantes :

```
model.explained_variance_ratio_
```

4) Analysez ensuite la somme cumulée de variance :

```
np.cumsum (model.explained_variance_ratio_)
```

5) Tracez le pourcentage de variance cumulée en fonction du nombre de composantes :

```
plt.plot(Np.cumsum (model.explained_varaince_ratio_))
```

6) En vous appuyant sur le graphique de la question précédente, déterminez à partir de combien de composantes, on atteint 90% de variance cumulée.

7) Trouvez à partir de combien de composantes, on atteint 99% de variance en vous appuyant sur :

```
np.argmax(np.cumsum (Model.explained_varaince_ratio_)>99
```

Interprétez le résultat obtenu.

8) Prenez le nombre de composante trouver à la question précédente puis entrainez le modèle :

```
model = PCA(n_components = d)
X_reduced = model.fit_transform(X)
```

9) Décompressez les images en vous appuyant sur :

```
X_recovered = Molel.inverse_transform(X_reduced)
```

10) Affichez une des images de X_recovreder

```
plt.imshow(X_recoverde[0].reshape((8,8)))
```

Analysez la qualité de l'image.

11) Il est possible de fixer le pourcentage de variance à préserver comme suit :

```
model = PCA(n_components=0.40)
```

pour cela vous devez réaliser les mises à échelle suivantes :

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
data_rescaled = scaler.fit_transform(X)
from sklearn.decomposition import PCA
pca = PCA(n_components = 0.4)
pca.fit(data_rescaled)
reduced = pca.transform(data_rescaled)
```

Vérifiez le nombre de composantes utilisées :

```
Model.n_components_
```

12) Analyser l'impact de la réduction du nombre de composante sur la qualité des images.