Classification des données - MNIST

Objectif du TP : Classification des données MNIST

Dans ce TP, nous allons aborder la classification en utilisant le célèbre jeu de données MNIST, qui consiste en des images de chiffres manuscrits. Nous allons explorer différentes étapes du processus de classification, notamment la récupération des données, la séparation des données en ensembles d'entraînement et de test, la visualisation des images, la création de classificateurs binaires et multiclasse, ainsi que l'évaluation des performances à l'aide de différentes métriques.

Étapes du TP :

1. Récupération des données

Dans cette première étape, nous allons charger les données MNIST à l'aide de la bibliothèque `scikit-learn`. Nous allons également veiller à convertir les étiquettes en types appropriés.

```
import numpy as np
from sklearn.datasets import fetch_openml
mnist = fetch_openml('mnist_784', version=1, cache=True, as_frame=False)
mnist.target = mnist.target.astype(np.int8)
```

2. Séparation des données d'entraînement et de test

Pour évaluer nos modèles de classification, nous devons diviser nos données en ensembles d'entraînement et de test. Il est recommandé de mélanger les données pour garantir une répartition aléatoire.

Questions : - Quelle est l'importance de mélanger les données avant de les diviser en ensembles d'entraînement et de test ?

3. Visualisation d'une image

Cette étape consiste à visualiser une image à partir du jeu de données. Nous utiliserons la bibliothèque Matplotlib pour afficher une image d'un chiffre manuscrit.

```
import matplotlib as mpl
import matplotlib.pyplot as plt
some_digit = X[24000]
some_digit_image = some_digit.reshape(28, 28)
plt.imshow(some_digit_image, cmap = mpl.cm.binary,
           interpolation="nearest")
plt.axis("off")

plt.show()
```

4. Classificateur binaire

Dans cette étape, nous allons choisir un chiffre (entre 0 et 9) et utiliser le classificateur `SGDClassifier` pour prédire si une image correspond à ce chiffre. Nous devrons créer une nouvelle variable cible binaire.

Questions : - Pourquoi devons-nous créer une variable cible binaire ? - Comment entraînons-nous le modèle `SGDClassifier` pour cette tâche de classification binaire ?

5. Précision et rappel (faux-positif/faux-négatif)

Pour évaluer les performances de notre modèle binaire, nous utiliserons les métriques de précision (`precision_score`) et de rappel (`recall_score`).

Questions : - Quelle est la signification de la précision et du rappel dans le contexte de la classification ? - Comment calculons-nous la précision et le rappel ?

6. Comparaison avec RandomForestClassifier

Nous allons comparer les performances de notre modèle SGDClassifier avec celles d'un autre classificateur, le RandomForestClassifier.

Questions : - Quelle est la différence entre le `SGDClassifier` et le `RandomForestClassifier` ? - Comment comparons-nous les performances de ces deux classificateurs ?

7. Classificateur multiclasse

Dans cette dernière étape, nous allons essayer de prédire directement les chiffres de 0 à 9 en utilisant un modèle de classification multiclasse.

Questions : - Comment adaptons-nous notre modèle pour effectuer une classification multiclasse ? - Quelles métriques d'évaluation devrions-nous utiliser pour évaluer la performance de ce modèle ?

Vous utiliserez un notebook Jupyter pour effectuer ces étapes et répondre aux questions. Assurez-vous de comprendre chaque étape du TP avant de passer à la suivante.