TP 8 – Courbe ROC (Wine Dataset)

NOM : GHODBANI

PRÉNOM : Fares

Groupe : Miage

Réalisé avec Canva et Visual studio code.

Introduction

Dans ce TP, nous avons utilisé le dataset **Wine** pour évaluer un modèle de classification multi-classes à l'aide de **courbes ROC**.
Le modèle choisi est un **DecisionTreeClassifier**, et les labels ont été binarisés pour adapter la courbe ROC aux problèmes avec plus de deux classes.
Les étapes du TP ont été réalisées dans Python et les résultats analysés pour évaluer la performance du modèle.
Les remarques étape par étape sont regroupées ci-dessous et les **codes Python correspondants** sont inclus dans le fichier **consigness.py**.
Enfin, la **figure finale** représentant la courbe ROC pour la classe 1 a été tracée et est sauvegardée sous le nom **figure_tp8ia.png**.

Étapes et analyses

1. Import des librairies

Observation :

- L'importation from scipy import interp échoue :

```
ImportError: cannot import name 'interp' from 'scipy'
```

Analyse / Remarque :

- Cette erreur est **normale dans les versions récentes de SciPy** ($\geqslant$ 1.10) car la fonction scipy.interp a été **supprimée**.
- Pour réaliser une interpolation linéaire (si nécessaire pour la ROC multi-classes), il faut utiliser **numpy.interp** :

```
# Exemple d'interpolation linéaire avec numpy
y_interp = np.interp(x_new, x_old, y_old)
```

- Cela n'empêche pas le reste du TP de fonctionner, car la ROC peut être calculée sans utiliser scipy.interp.

Conclusion :

- Tous les autres imports fonctionnent normalement et permettent de réaliser les étapes suivantes du TP.
- L'erreur sur scipy.interp peut être ignorée ou contournée avec np.interp.

2. Charger l'objet Wine

- **Observation :** L'objet wine contient 178 exemples et 13 caractéristiques. Les labels sont 0, 1 ou 2.
- **Remarque :** Les données sont prêtes pour la création de DataFrames et l'analyse.

3. Construire le DataFrame X

- **Observation :** X.shape = (178, 13)
- **Remarque :** Toutes les caractéristiques numériques sont prêtes pour l'entraînement.

4.5 Construire le DataFrame y

- **Observation :** y.shape = (178, 1)
- **Remarque :** Les labels doivent être binarisés pour permettre le calcul de la courbe ROC.

6.7 Binarisation des labels

- **Résultat (5 premiers exemples) :**

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

- **Analyse :**
  - La binarisation transforme chaque label en vecteur [classe0, classe1, classe2].
  - Ici, les cinq premiers exemples appartiennent tous à la classe 0.
  - Chaque colonne correspond à une classe : 1 si l'exemple appartient à cette classe, 0 sinon.
- **Remarque :** Ce format est **normal et attendu**, nécessaire pour la ROC multi-classes.

8-10. Nombre de classes et séparation train/test

- n_classes = 3
- n_samples = 178, n_features = 13
- X_train.shape = (124, 13), X_test.shape = (54, 13)
- y_train.shape = (124, 3), y_test.shape = (54, 3)

Remarque : La répartition train/test (70/30) est correcte et conserve la structure multi-classes.

11. Construction et entraînement du modèle

- **Observation :** Le modèle DecisionTreeClassifier est créé avec max_depth=3 et min_samples_leaf=5.
- **Remarque :** Le modèle est simple, limite le surapprentissage et convient pour le dataset Wine.

12. Classes prédites par le modèle

- **Résultat (y_score – 5 premiers exemples) :**

[[1 0 0],

[0 0 1],

[0 1 0],

[1 0 0],

[0 1 0]]

- **Analyse :** Le modèle distingue correctement les classes dans la majorité des cas.
- **Remarque :** Les prédictions binaires sont cohérentes avec la structure multi-classes.

13-14. Calcul des TFP, TVP et AUC

- **Résultats :**

Classe 0 - AUC: 0.947

Classe 1 - AUC: 0.939

Classe 2 - AUC: 0.988

- **Analyse :**
  - Le modèle est performant pour toutes les classes.
  - La classe 2 est la mieux prédite (AUC = 0.988).
  - La classe 1 est légèrement moins précise mais reste excellente (AUC = 0.939).
- **Remarque :** Les taux de faux positifs et vrais positifs sont cohérents avec les AUC.

15-21. Tracé de la courbe ROC (Classe 1)

- **Observations :**
  - La courbe ROC (verte) est au-dessus de la diagonale (AUC = 0.939), indiquant une bonne performance.
  - La ligne diagonale représente un modèle aléatoire (AUC = 0.5).
  - Les axes sont correctement labelisés : Taux Faux Positifs et Taux Vrai Positifs.
  - La figure peut être sauvegardée sous **figure_tp8ia.png**.

Remarque : La visualisation permet de voir clairement la capacité du modèle à séparer les classes.

Remarques générales

- L'import from scipy import interp échoue naturellement et peut être remplacé par np.interp pour l'interpolation si nécessaire.
- Les valeurs d'AUC montrent que le modèle DecisionTreeClassifier est performant pour toutes les classes.

Conclusion

- La binarisation des labels est essentielle pour l'utilisation de ROC dans les problèmes multi-classes.
- Le modèle DecisionTreeClassifier prédit correctement les classes du dataset Wine.
- Les courbes ROC et les valeurs d'AUC fournissent une évaluation visuelle et quantitative robuste de la performance du modèle.
- La figure finale (figure_tp8ia.png) permet de présenter visuellement la performance pour la classe 1.