RAPPORT DE PROJET : Intelligence Artificielle

SYSTÈME INTELLIGENT DE CLASSIFICATION BOTANIQUE

Analyse prédictive du Dataset "Iris de Fisher"

INFORMATIONS GÉNÉRALES

DOMAINE : Apprentissage Automatique Supervisé (Machine Learning)

AUTEUR : [Boukais Ines]

TECHNOLOGIES UTILISÉES : Python | Scikit-Learn | Pandas | Matplotlib

---

Année Universitaire 2023 - 2024

1. Introduction et Problématique

Ce projet vise à automatiser l'identification des espèces de fleurs d'Iris à partir de mesures morphologiques. L'enjeu est de construire un modèle capable de distinguer les subtilités entre trois classes : *Setosa*, *Versicolor*, et *Virginica*. Contrairement à une programmation classique basée sur des règles manuelles, nous utilisons ici l'apprentissage automatique, où l'algorithme apprend les frontières de décision directement à partir des données historiques.

2. Exploration et Structuration des Données

Le jeu de données est la pierre angulaire du projet. Il se compose de quatre variables explicatives (caractéristiques) et d'une variable cible (l'espèce).

Chargement et Analyse Descriptive

Nous utilisons **Pandas** pour transformer les données brutes en un objet structuré appelé DataFrame.

```
import pandas as pd

# Chargement des données
df = pd.read_csv("Iris.csv")

# Sélection des caractéristiques (X) et de la cible (Y)
# On exclut la colonne d'index (0) pour garder les mesures physiques
x = df.iloc[:, 1:5]
y = df.iloc[:, 5]

print("Aperçu des caractéristiques :\n", x.head())
```

Index	Longueur Sépale	Largeur Sépale	Longueur Pétale	Largeur Pétale	Espèce
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
...	...	...	...	...	...

3. Visualisation de l'Espace des Données

Pour valider la séparabilité des classes, nous réalisons une projection en deux dimensions (Longueur Sépale vs Longueur Pétale).

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6)) # Définition de la taille de la figure [cite: 16]

# Visualisation par espèce avec marqueurs distincts [cite: 18, 21]
plt.scatter(x[y=='Iris-setosa'].iloc[:, 0], x[y=='Iris-setosa'].iloc[:, 2], label='Iris-setosa', c='red', marker='*')
plt.scatter(x[y=='Iris-versicolor'].iloc[:, 0], x[y=='Iris-versicolor'].iloc[:, 2], label='Iris-versicolor')
plt.scatter(x[y=='Iris-virginica'].iloc[:, 0], x[y=='Iris-virginica'].iloc[:, 2], label='Iris-virginica')

plt.xlabel('Longueur Sépale') [cite: 29]
plt.ylabel('Longueur Pétale') [cite: 29]
plt.legend() # Affichage de la légende pour interpréter les couleurs [cite: 27]
plt.show()
```

Analyse visuelle : L'espèce *Setosa* (étoiles rouges) forme un groupe isolé, facilitant sa classification. Les espèces *Versicolor* et *Virginica* sont plus proches, nécessitant des algorithmes capables de définir des frontières de décision plus fines.

4. Stratégie d'Apprentissage et de Test

Pour garantir la fiabilité du projet, nous appliquons une méthode de validation rigoureuse.

- **Partitionnement :** Les données sont divisées en un ensemble d'entraînement (70%) et un ensemble de test (30%).
- **Contrôle de l'aléatoire :** Le paramètre *random_state=1* assure que les résultats sont reproductibles à chaque exécution.

Ensemble	**Nombre d'échantillons**	**Rôle**
Entraînement	105	Ajustement des poids et paramètres du modèle.
Test	45	Évaluation de la performance sur données inconnues.

5. Modélisation et Comparaison des Performances

Nous avons implémenté trois types d'architectures pour comparer leur efficacité.

A. K-Nearest Neighbors (KNN)

L'algorithme KNN classe un point selon la classe majoritaire de ses plus proches voisins.

- **Configuration :** k=7 voisins.
- **Performance :** Score d'exactitude de **0.977** (soit 97.7%).

B. Réseau de Neurones (MLP - Multi-Layer Perceptron)

Modèle simulant des couches de neurones pour capturer des relations complexes.

- **Architecture :** Deux couches cachées de 5 neurones chacune.
- **Optimisation :** Utilisation de l'algorithme de descente de gradient 'adam' et d'un paramètre de régularisation alpha pour éviter le surapprentissage.

C. Forêt Aléatoire (Random Forest)

Algorithme d'ensemble créant une multitude d'arbres de décision.

- **Configuration :** Profondeur maximale de l'arbre limitée à 2 pour maintenir la simplicité du modèle.

6. Phase de Prédiction et Validation Finale

Un modèle est jugé utile s'il peut prédire des cas réels. Nous avons testé le modèle final avec deux fleurs fictives.

```
# Données de test (Mesures de deux nouvelles fleurs)
nouveau_jeu = [[5.1, 3.5, 1.4, 0.2], [6.2, 3.4, 5.4, 2.3]]

# Prédiction avec le modèle entraîné
preds = modele.predict(nouveau_jeu)

print("Résultats de la prédiction :", preds)
# Sortie : ['Iris-setosa', 'Iris-virginica']
```

7. Conclusion

Le projet démontre la puissance de l'apprentissage supervisé. Le modèle **KNN** s'est avéré le plus performant avec une précision de **97.7%**. L'intégration des réseaux de neurones ouvre des perspectives pour des jeux de données plus massifs, tandis que la visualisation avec **Matplotlib** confirme que les caractéristiques morphologiques choisies sont de bons indicateurs discriminants pour la classification botanique.