Machine Learning

Exercice 4

Pour prédire avec la plus grande certitude la quantité d'électricité produite par une ville, nous disposons de deux ensembles de données : le fichier inputs.npy contenant les données collectées par des capteurs physiques tels que la vitesse du vent, la température, etc. Ces données ont déjà été transformées pour correspondre à nos modèles. Nous avons également le fichier labels.npy contenant les quantités d'électricité produites, qui servira à entraîner notre intelligence artificielle

Dans un premier temps, nous allons diviser notre ensemble de données en échantillons de test de la manière suivante : 90% sera consacré à l'entraînement, ce qui correspond à 180 lignes, et 10% sera réservé aux tests, soit 20 lignes. Pour effectuer cette division, nous utiliserons la fonction "train_test_split()" avec le paramètre "random_state=25" afin d'avoir une sélection aléatoire des données..

Maintenant, nous sommes prêts à effectuer des tests sur différents modèles et à ajuster les hyperparamètres afin d'obtenir la meilleure précision possible.

1. Linear Regression

Dans le cas du modèle "Régression linéaire", nous n'allons pas modifier directement les hyperparamètres, mais plutôt utiliser différentes variantes de la régression linéaire. Les modèles que nous allons tester sont les suivants :

- Linear Regression
- Ridge Regression
- Lasso Regression

Ces modèles diffèrent principalement par les coefficients utilisés dans la régression linéaire. Cependant, les résultats fournis par ces modèles ne sont clairement pas suffisants, comme on peut le constater dans l'exemple ci-contre.

```
→  Exercice 4 git:(main) × python3 exercice4.py
===== Linear Regression model =====

Linear model accuracy :
        Train accuracy 0.95626
        Test accuracy 0.5497
        R2 score 0.5497
Ridge model accuracy :
        Train accuracy 0.82652
        Test accuracy 0.40893
        R2 score 0.40893
Lasso model accuracy :
        Train accuracy 0.0
        Test accuracy -0.12744
        R2 score -0.12744
```

2. Decision Tree

Pour le modèle "Arbre de décision" (Decision Tree), nous devons tout d'abord adapter les données au modèle en utilisant la fonction "LabelEncoder()". En ce qui concerne les hyperparamètres, nous allons principalement nous concentrer sur l'option "criterion", qui comporte deux valeurs principales : Gini et Entropy. La différence entre ces deux modes réside dans la formule utilisée pour calculer le modèle.

```
→  Exercice 4 git:(main) ✗ python3 exercice4.py
===== Decision Tree model =====

Criterion parameter used: Gini
F1_score: 0.0
R2 score -10.90827
Criterion parameter used: Entropy
F1_score: 0.09524
R2 score -0.90977
```

Cependant, il convient de noter que ce modèle n'est pas du tout adapté à la structure de nos données, ce qui explique pourquoi les résultats obtenus sont erronés.

Random Forest

Pour le modèle "Random Forest", nous allons utiliser la méthode "random_grid". Cette approche implique la définition d'une grille contenant différents hyperparamètres et une plage de valeurs pour chaque hyperparamètre. Ensuite, nous utiliserons la fonction "RandomizedSearchCV()" pour explorer différentes combinaisons d'hyperparamètres et sélectionner celle qui offre les meilleures performances.

Cependant, malgré ces efforts, les résultats obtenus restent largement en deçà du score souhaité.

```
→  Exercice 4 git:(main) ✗ python3 exercice4.py
===== Random Forest model =====

Train accuracy 0.90055
Test accuracy 0.21243
R2 score 0.21243
```