Atelier : Se familiariser avec [scikit-learn/matplotlib]

Objectif

Cet atelier vise à vous familiariser avec les bases du machine learning. Vous apprendrez à manipuler des datasets, à créer des modèles de classification et de régression, et à visualiser vos résultats avec Matplotlib. L'objectif est de vous donner une compréhension pratique des techniques fondamentales du machine learning, en mettant l'accent sur des compétences concrètes et immédiatement applicables.

Prérequis
- Python 3.7+
- Bibliothèques : [scikit-learn, matplotlib, pandas, numpy]
- Jupyter Notebook ou un environnement de développement Python

Exercice 1 :
Chargez le dataset Iris et réalisez une analyse exploratoire complète. Votre objectif est de comprendre la structure des données en utilisant des méthodes de visualisation et d'analyse descriptive. Affichez les premières lignes du dataset, générez des visualisations de la distribution des caractéristiques et identifiez les principales caractéristiques statistiques.

Exemple:
```
from sklearn.datasets import load_iris
import pandas as pd
import matplotlib.pyplot as plt

# Charger le dataset Iris
iris = load_iris()
X = iris.data
y = iris.target

# Créer un DataFrame
df = pd.DataFrame(X, columns=iris.feature_names)
df['target'] = y

# Afficher les premières lignes
print(df.head())

# Visualiser la distribution des caractéristiques
plt.figure(figsize=(10, 6))
df.boxplot()
plt.title('Distribution des caractéristiques - Dataset Iris')
plt.show()
```

Exercice 2 :

Préparez le dataset Iris pour l'entraînement d'un modèle de machine learning. Divisez les données en jeux d'entraînement et de test en utilisant la fonction `train_test_split()`. Votre tâche est de séparer correctement les features (X) et le target (y), avec 30% des données réservés pour le test, tout en garantissant la reproductibilité par l'utilisation d'une graine aléatoire.

Exemple:

```
from sklearn.model_selection import train_test_split

# Séparer les features et le target
X = iris.data
y = iris.target
# Division du dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
print(f"Taille du jeu d'entraînement : {X_train.shape}")
print(f"Taille du jeu de test : {X_test.shape}")
```

Exercice 3 :

Construisez un modèle de classification en utilisant l'algorithme Random Forest pour prédire l'espèce de fleurs dans le dataset Iris. Entraînez le modèle sur les données de training, effectuez des prédictions sur les données de test, et évaluer ses performances. Générez un rapport de classification détaillé et une matrice de confusion pour visualiser la précision de votre modèle.

Exemple:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
# Créer et entraîner le modèle
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)
# Prédictions
y_pred = clf.predict(X_test)
# Évaluation du modèle
print("Rapport de classification :")
print(classification_report(y_test, y_pred, target_names=iris.target_names))
# Matrice de confusion
plt.figure(figsize=(8, 6))
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=iris.target_names,
            yticklabels=iris.target_names)
plt.title('Matrice de confusion')
plt.xlabel('Prédictions')
plt.ylabel('Vraies valeurs')
plt.show()
```

Exercice 4 :

Utilisez le dataset Boston Housing pour réaliser une analyse de régression linéaire. Votre mission est de prédire le prix des maisons en fonction de différentes caractéristiques. Entraînez un modèle de régression linéaire, effectuez des prédictions sur les données de test, et évaluez la performance du modèle en calculant l'erreur quadratique moyenne (MSE) et le score R². Créez une visualisation comparative entre les prix réels et les prix prédits.

Exemple :

```
from sklearn.datasets import load_boston
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Charger le dataset Boston Housing
boston = load_boston()
X = boston.data
y = boston.target

# Séparation train/test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Entraînement du modèle de régression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Prédictions
y_pred = regressor.predict(X_test)

# Évaluation du modèle
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Erreur quadratique moyenne : {mse}")
print(f"Score R² : {r2}")

# Visualisation des prédictions vs réalité
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
plt.xlabel('Prix réel')
plt.ylabel('Prix prédit')
plt.title('Prédictions vs Réalité - Régression Linéaire')
plt.show()
```