TP2
Linear / Logistic Regression

Régression Linéaire

Nous allons utiliser l'ensemble de données sur les prix des logements à Boston, disponible dans la bibliothèque Scikit-learn.
Cet ensemble de données contient des informations sur diverses caractéristiques des logements à Boston et les prix médians des logements dans ces quartiers.
"""

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
#from sklearn.datasets import load_boston
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Charger l'ensemble de données Boston Housing
data_url = "http://lib.stat.cmu.edu/datasets/boston"
raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
X = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
y = raw_df.values[1::2, 2]

# Diviser les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Créer un modèle de régression linéaire
model = LinearRegression()

# Entraîner le modèle sur l'ensemble d'entraînement
model.fit(X_train, y_train)

# Prédire sur l'ensemble de test
y_pred = model.predict(X_test)

# Calculer la précision du modèle (score R2)
accuracy = r2_score(y_test, y_pred)
print(f"Précision du modèle (score R2) : {accuracy}")

# Calculer l'erreur quadratique moyenne (MSE)
mse = mean_squared_error(y_test, y_pred)
print(f"Erreur quadratique moyenne (MSE) : {mse}")

# Visualiser les prédictions par rapport aux valeurs réelles
plt.scatter(y_test, y_pred)
plt.xlabel('Valeurs réelles')
plt.ylabel('Prédictions')
plt.title('Régression Linéaire - Boston Housing Prices')
```

```
plt.show()
```

Régression logistique

Dans l'exemple précédent de régression logistique sur l'ensemble de données Iris, les classes correspondent aux différentes catégories de fleurs d'Iris que le modèle essaie de classifier. Dans l'ensemble de données Iris, il y a trois classes de fleurs d'Iris :

1. **Setosa :** C'est l'une des espèces d'Iris. C'est une des classes d'iris que le modèle essaie de prédire.

2. **Versicolor :** Une autre espèce d'Iris. C'est la deuxième classe que le modèle essaie de prédire.

3. **Virginica :** La troisième espèce d'Iris. C'est la troisième classe que le modèle essaie de prédire.
Dans l'ensemble de données Iris, les classes sont généralement représentées par des valeurs entières 0, 1 et 2, correspondant aux trois espèces d'Iris. Voici la correspondance entre les valeurs de classe et les noms des espèces d'Iris :

- Classe 0 : Setosa
- Classe 1 : Versicolor
- Classe 2 : Virginica

Ainsi, dans l'ensemble de données Iris, les valeurs de classe 0, 1 et 2 correspondent respectivement aux trois espèces d'Iris : Setosa, Versicolor et Virginica. Lorsque vous effectuez des opérations de classification, ces valeurs de classe sont utilisées pour représenter les différentes catégories d'espèces d'Iris.

Lorsque le modèle de régression logistique effectue des prédictions sur de nouvelles données, il attribuera l'une de ces trois classes aux observations en fonction des caractéristiques fournies en entrée. Le modèle utilise les caractéristiques des fleurs:
- longueur du sépale
- largeur du sépale
- longueur du pétale
- largeur du pétale
 pour effectuer ces prédictions.

Ainsi, les classes dans cet exemple sont des catégories de fleurs d'Iris que le modèle tente de classifier en fonction de leurs caractéristiques. Lorsque vous visualisez les prédictions par rapport aux valeurs réelles, vous pouvez voir comment le modèle a réussi à classer les fleurs en fonction des classes réelles (les trois espèces d'Iris) et des classes prédites par le modèle.
"""

```
import numpy as np
import matplotlib.pyplot as plt
```

```
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score

# Charger l'ensemble de données Iris
iris = load_iris()
X = iris.data
y = iris.target
print(X)
print(y)

# Diviser les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Créer un modèle de régression logistique
model = LogisticRegression(max_iter=1000)

# Entraîner le modèle sur l'ensemble d'entraînement
model.fit(X_train, y_train)

# Prédire sur l'ensemble de test
y_pred = model.predict(X_test)

# Calculer la précision du modèle
accuracy = accuracy_score(y_test, y_pred)
print(f"Précision du modèle : {accuracy}")

# Afficher le rapport de classification pour obtenir plus d'informations
report = classification_report(y_test, y_pred)
print("Rapport de classification :")
print(report)
# Visualiser les prédictions par rapport aux valeurs réelles
plt.figure(figsize=(8, 6))
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_pred, cmap='viridis', s=50,
label='Prédiction')
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap='Set1',
marker='X', s=30, label='Réel')
plt.xlabel('Caractéristique 1')
plt.ylabel('Caractéristique 2')
plt.legend(['Prédiction', 'Réel'])
plt.title('Visualisation des prédictions par rapport aux valeurs
réelles')
plt.show()
```