Test ML

Code :

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score,mean_absolute_percentage_error
from scipy import stats
from sklearn.preprocessing import StandardScaler


#Chargement des données
data = pd.read_csv('ratings.csv')
scaler = StandardScaler()
data= pd.DataFrame(scaler.fit_transform(data),columns=data.columns)
print(data)
target = pd.DataFrame(data["userId"], columns=["userId"])
# Sélectionner la colonne "rating" comme attribut
selected_attribute = data['rating']

# Afficher les premières lignes de l'attribut sélectionné
print(selected_attribute.head())

# Sélectionner la colonne "rating" comme attribut
X = data[['userId']]  # Attribut
y = data['rating']

# 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 et entraîner le modèle de régression linéaire
model = LinearRegression()
model.fit(X_train, y_train)

# Prédictions sur l'ensemble de test
predictions = model.predict(X_test)

# Calculer l'erreur quadratique moyenne (MSE) pour évaluer les performances du modèle
mse = mean_squared_error(y_test, predictions)
print("Mean Squared Error (MSE):", mse)
# Calculer l'erreur quadratique moyenne (MSE)
mape = mean_absolute_percentage_error(y_test,predictions)
print("Mean Absolute Percentage Error:", mape)

# Calculer l'erreur absolue moyenne (MAE)
mae = mean_absolute_error(y_test, predictions)
print("Mean Absolute Error (MAE):", mae)

print("---------------------------------\n")
# Sélectionner les colonnes sauf la variable cible
```

```
features = data.drop(columns=['rating'])
# Liste pour stocker les performances de chaque attribut
attribute_performance = []
# Boucle à travers chaque attribut
for column in features.columns:
    # Sélectionner l'attribut courant comme caractéristique
    X = data[[column]]
    y = data['rating']  # Variable cible à prédire

    # 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 et entraîner le modèle de régression linéaire
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Prédictions sur l'ensemble de test
    predictions = model.predict(X_test)

    coefficient = model.coef_[0]

    # Ajouter le coefficient et le nom de l'attribut à la liste
    attribute_performance.append((column, abs(coefficient)))



    # Calculer l'erreur quadratique moyenne (MSE) pour évaluer les performances du modèle
    mse = mean_squared_error(y_test, predictions)
    print("Attribute:", column)
    print("Mean Squared Error (MSE):", mse)
    mape = mean_absolute_percentage_error(y_test,predictions)
    print("Mean Absolute Percentage Error:", mape)
    mae = mean_absolute_error(y_test, predictions)
    print("Mean Absolute Error (MAE):", mae)
    print("----------------------------------")


# Trier les attributs par ordre décroissant de leurs performances
attribute_performance.sort(key=lambda x: x[1], reverse=True)
# Afficher les attributs les plus dominants
print("Attributs les plus dominants:")
for attribute, performance in attribute_performance:
    print(attribute, ": Coefficient =", performance)

# Attribut le plus dominant
most_dominant_attribute = attribute_performance[0][0]

print("Attribut le plus dominant: " + most_dominant_attribute)
print("------------------------------------------------------------------")
#7. Entraînez un nouveau modèle de régression linéaire avec seulement les attributs les
plus dominants sélectionnés.
X = data[[most_dominant_attribute]]
```

```
Y = data["rating"]
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

model = LinearRegression()
model.fit(X_train_scaled, Y_train)

predicted_Y = model.predict(X_test_scaled)

#8.Évaluez les performances du modèle et comparez-les avec le modèle utilisant tous les
attributs.
print("Évaluez les performances du modèle et comparez-les avec le modèle utilisant tous
les attributs.")
mae = mean_absolute_error(Y_test, predicted_Y)
mse = mean_squared_error(Y_test, predicted_Y)
mape = mean_absolute_percentage_error(Y_test, predicted_Y)

print("Mean Absolute Error:", mae)
print("Mean Square Error:", mse)
print("Mean Absolute Percentage Error:", mape)
print("----------------------------------------------------------------------")
# Construction du modèle avec tous les attributs
X_all = data[["userId", "movieId", "timestamp"]]
Y_all = data["rating"]

X_train_all, X_test_all, Y_train_all, Y_test_all = train_test_split(X_all, Y_all,
test_size=0.2, random_state=42)

scaler_all = StandardScaler()
X_train_scaled_all = scaler_all.fit_transform(X_train_all)
X_test_scaled_all = scaler_all.transform(X_test_all)

model_all = LinearRegression()
model_all.fit(X_train_scaled_all, Y_train_all)

predicted_Y_all = model_all.predict(X_test_scaled_all)

# Évaluez les performances du modèle utilisant tous les attributs
mae_all = mean_absolute_error(Y_test_all, predicted_Y_all)
mse_all = mean_squared_error(Y_test_all, predicted_Y_all)
mape_all = mean_absolute_percentage_error(Y_test_all, predicted_Y_all)

print("Mean Absolute Error pour all attributes:", mae_all)
print("Mean Square Error pour all attributes:", mse_all)
print("Mean Absolute Percentage Error pour all attributes:", mape_all)
print("--------------------------------")
print(f"Performance du modèle de l'attribut le plus dominant ({ most_dominant_attribute
}):")
print("Mean Absolute Error :", mae)
```

```
print("Mean Square Error :", mse)
print("Mean Absolute Percentage Error :", mape)
print("Comparaison des performances:")
if mae < mae_all:
    print("mae < mae_all : L'erreur absolue moyenne est meilleure pour le modèle utilisant uniquement l'attribut le plus dominant.")
else:
    print("mae > mae_all : L'erreur absolue moyenne est meilleure pour le modèle utilisant tous les attributs.")

if mse < mse_all:
    print("mse < mse_all : L'erreur quadratique moyenne est meilleure pour le modèle utilisant uniquement l'attribut le plus dominant.")
else:
    print("mse > mse_all : L'erreur quadratique moyenne est meilleure pour le modèle utilisant tous les attributs.")

if mape < mape_all:
    print("mape < mape_all : L'erreur de pourcentage absolue moyenne est meilleure pour le modèle utilisant uniquement l'attribut le plus dominant.")
else:
    print("mape > mape_all : L'erreur de pourcentage absolue moyenne est meilleure pour le modèle utilisant tous les attributs.")
```

résulta :

```
PS C:\Users\ASUS\Desktop\GitHub\TravauxProgUniversitaires\DLSI_ADBD\DLSI_ADBD_S2\AI\TP\Test> & C:/Users/ASUS/AppData/Local/Programs/Python/Python311/python.exe c:/Users/ASUS/Desktop/GitHub/TravauxProgUniversitaires/DLSI_ADBD/DLSI_ADBD_S2/AI/TP/Test/test.py
         userId   movieId    rating  timestamp
0     -1.780374 -0.546970  0.478112  -1.114230
1     -1.780374 -0.546914  0.478112  -1.114237
2     -1.780374 -0.546830  0.478112  -1.114232
3     -1.780374 -0.545676  1.437322  -1.114225
4     -1.780374 -0.545591  1.437322  -1.114229
...         ...       ...       ...        ...
100831  1.554464  4.140032  0.478112   1.331279
100832  1.554464  4.188272  1.437322   1.331287
100833  1.554464  4.188328  1.437322   1.333242
100834  1.554464  4.188385  1.437322   1.331269
100835  1.554464  4.262208 -0.481099   1.331270

[100836 rows x 4 columns]
0    0.478112
1    0.478112
2    0.478112
3    1.437322
4    1.437322
Name: rating, dtype: float64
Mean Squared Error (MSE): 1.0103243043805215
Mean Absolute Percentage Error: 4.571457832380103
Mean Absolute Error (MAE): 0.8010722432202865
--------------------------------

Attribute: userId
Mean Squared Error (MSE): 1.0103243043805215
Mean Absolute Percentage Error: 4.571457832380103
Mean Absolute Error (MAE): 0.8010722432202865
```

```
Attribute: movieId
Mean Squared Error (MSE): 1.012109412891674
Mean Absolute Percentage Error: 1.1427353524387305
Mean Absolute Error (MAE): 0.7977758065057393
---------------------------------
Attribute: timestamp
Mean Squared Error (MSE): 1.01208315817474
Mean Absolute Percentage Error: 1.138746351230023
Mean Absolute Error (MAE): 0.7978363093504454
---------------------------------
Attributs les plus dominants:
userId : Coefficient = 0.050786702144713806
timestamp : Coefficient = 0.0047948047404820435
movieId : Coefficient = 0.002536225450453234
Attribut le plus dominant: userId
-----------------------------------------------------------------
Évaluez les performances du modèle et comparez-les avec le modèle utilisant tous les attributs.
Mean Absolute Error: 0.8010722432202865
Mean Square Error: 1.0103243043805215
Mean Absolute Percentage Error: 4.571457832380111
-----------------------------------------------------------------
Mean Absolute Error pour all attributes: 0.8010772932307403
Mean Square Error pour all attributes: 1.010290719654976
Mean Absolute Percentage Error pour all attributes: 4.582188484703133
---------------------------------
Performance du modèle de l'attribut le plus dominant (userId):
Mean Absolute Error : 0.8010722432202865
Mean Square Error : 1.0103243043805215
Mean Absolute Percentage Error : 4.571457832380111
Comparaison des performances:
mae < mae_all : L'erreur absolue moyenne est meilleure pour le modèle utilisant uniquement l'attribut le plus dominant.
mse > mse_all : L'erreur quadratique moyenne est meilleure pour le modèle utilisant tous les attributs.
mape < mape_all : L'erreur de pourcentage absolue moyenne est meilleure pour le modèle utilisant uniquement l'attribut le plus dominant.
PS C:\Users\ASUS\Desktop\GitHub\TravauxProgUniversitaires\DLSI_ADBD\DLSI_ADBD_S2\AI\TP\Test>
```