Aicha Ndiaye
DIC2 IABD

I. Complement du TP 1

A. Objectif:

- **Optimiser** les algorithmes **Decision Tree** et **Random Forest** avec les méthodes **GridSearch Optuna** et **Random Search.**
- **Les comparer entre eux.**

B. Le code commenté:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, RandomizedSearchCV, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from scipy.stats import randint
import optuna

# Charger le dataset
dataset = pd.read_csv('uci_malware_detection.csv')

# Séparer les caractéristiques (X) et les labels (y)
X = dataset.drop(columns='Label')
y = dataset['Label']

# 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=0.4, random_state=3)

# Créer les modèles
decision_tree_model = DecisionTreeClassifier(random_state=2)
random_forest_model = RandomForestClassifier(n_estimators=100, random_state=2)
```

Aicha Ndiaye
DIC2 IABD

```
# ------------------------- 1. Random Search -------------------------

# Définir les hyperparamètres pour RandomizedSearchCV

param_dist_dt = {

   'criterion': ['gini', 'entropy'],

   'max_depth': [None, 1, 2, 3, 5, 10],

   'min_samples_split': randint(2, 20),

   'min_samples_leaf': randint(1, 20),

}


param_dist_rf = {

   'n_estimators': [50, 100, 150, 200],

   'max_depth': [None, 5, 10, 20],

   'min_samples_split': randint(2, 20),

   'min_samples_leaf': randint(1, 20),

   'bootstrap': [True, False]

}


# Random Search pour Decision Tree

random_search_dt = RandomizedSearchCV(decision_tree_model,
param_distributions=param_dist_dt, n_iter=10, scoring='accuracy', cv=5, random_state=42)

random_search_dt.fit(X_train, y_train)

print("\nMeilleurs paramètres (Random Search Arbre de Décision) :",
random_search_dt.best_params_)

print("Meilleure précision (Random Search Arbre de Décision) :", random_search_dt.best_score_)


# Random Search pour Random Forest

random_search_rf = RandomizedSearchCV(random_forest_model,
param_distributions=param_dist_rf, n_iter=10, scoring='accuracy', cv=5, random_state=42)

random_search_rf.fit(X_train, y_train)

print("\nMeilleurs paramètres (Random Search Random Forest) :",
random_search_rf.best_params_)
```

Aicha Ndiaye
DIC2 IABD

```
print("Meilleure précision (Random Search Random Forest) :", random_search_rf.best_score_)


# ------------------------- 2. Grid Search -------------------------
# Définir la grille des hyperparamètres pour GridSearchCV
param_grid_dt = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 1, 2, 3, 5, 10],
    'min_samples_split': [2, 5, 10, 15],
    'min_samples_leaf': [1, 2, 4],
}

param_grid_rf = {
    'n_estimators': [50, 100, 150, 200],
    'max_depth': [None, 5, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}



# Grid Search pour Decision Tree
grid_search_dt = GridSearchCV(decision_tree_model, param_grid=param_grid_dt,
scoring='accuracy', cv=5, verbose=1)
grid_search_dt.fit(X_train, y_train)
print("\nMeilleurs paramètres (Grid Search Arbre de Décision) :", grid_search_dt.best_params_)
print("Meilleure précision (Grid Search Arbre de Décision) :", grid_search_dt.best_score_)

# Grid Search pour Random Forest
grid_search_rf = GridSearchCV(random_forest_model, param_grid=param_grid_rf,
scoring='accuracy', cv=5, verbose=1)
grid_search_rf.fit(X_train, y_train)
```

Aicha Ndiaye
DIC2 IABD

```
print("\nMeilleurs paramètres (Grid Search Random Forest) :", grid_search_rf.best_params_)
print("Meilleure précision (Grid Search Random Forest) :", grid_search_rf.best_score_)



# ------------------------- 3. Optuna -------------------------

# Fonction objectif pour optimiser l'Arbre de Décision avec Optuna
def objective_dt(trial):
    dt_model = DecisionTreeClassifier(
        criterion=trial.suggest_categorical('criterion', ['gini', 'entropy']),
        max_depth=trial.suggest_int('max_depth', 1, 10),
        min_samples_split=trial.suggest_int('min_samples_split', 2, 10),
        min_samples_leaf=trial.suggest_int('min_samples_leaf', 1, 10),
        random_state=2
    )
    dt_model.fit(X_train, y_train)
    y_pred = dt_model.predict(X_test)
    return accuracy_score(y_test, y_pred)

# Créer une étude Optuna pour maximiser la précision de l'Arbre de Décision
study_dt = optuna.create_study(direction='maximize')
study_dt.optimize(objective_dt, n_trials=50)
print("\nMeilleurs paramètres (Optuna Arbre de Décision) :", study_dt.best_params)
print("Meilleure précision (Optuna Arbre de Décision) :", study_dt.best_value)

# Fonction objectif pour optimiser Random Forest avec Optuna
def objective_rf(trial):
    rf_model = RandomForestClassifier(
        n_estimators=trial.suggest_int('n_estimators', 50, 200),
        max_depth=trial.suggest_int('max_depth', 1, 10),
```

```
        min_samples_split=trial.suggest_int('min_samples_split', 2, 10),
        min_samples_leaf=trial.suggest_int('min_samples_leaf', 1, 10),
        bootstrap=trial.suggest_categorical('bootstrap', [True, False]),
        random_state=2
    )
    rf_model.fit(X_train, y_train)
    y_pred = rf_model.predict(X_test)
    return accuracy_score(y_test, y_pred)

# Créer une étude Optuna pour maximiser la précision de Random Forest
study_rf = optuna.create_study(direction='maximize')
study_rf.optimize(objective_rf, n_trials=50)
print("\nMeilleurs paramètres (Optuna Random Forest) :", study_rf.best_params)
print("Meilleure précision (Optuna Random Forest) :", study_rf.best_value)



# ---------------------------- 4. Comparaison des Modèles ----------------------------
# Comparer la précision des modèles sur le test set
y_pred_dt = grid_search_dt.best_estimator_.predict(X_test)
y_pred_rf = grid_search_rf.best_estimator_.predict(X_test)

accuracy_dt = accuracy_score(y_test, y_pred_dt)
accuracy_rf = accuracy_score(y_test, y_pred_rf)

print("\nPrécision globale (Arbre de Décision) :", accuracy_dt)
print("Précision globale (Random Forest) :", accuracy_rf)

if accuracy_dt > accuracy_rf:
    print("\nL'Arbre de Décision est plus performant.")
else:
```

```
print("\nRandom Forest est plus performant.")
```

C. Les resultats obtenus:

Meilleurs paramètres (Random Search Arbre de Décision) : {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 3, 'min_samples_split': 6}
Meilleure précision (Random Search Arbre de Décision) : 0.9731313131313133

Meilleurs paramètres (Random Search Random Forest) : {'bootstrap': True, 'max_depth': None, 'min_samples_leaf': 7, 'min_samples_split': 12, 'n_estimators': 150}
Meilleure précision (Random Search Random Forest) : 1.0
Fitting 5 folds for each of 144 candidates, totalling 720 fits

Meilleurs paramètres (Grid Search Arbre de Décision) : {'criterion': 'entropy', 'max_depth': 1, 'min_samples_leaf': 1, 'min_samples_split': 2}
Meilleure précision (Grid Search Arbre de Décision) : 0.968888888888889
Fitting 5 folds for each of 288 candidates, totalling 1440 fits

Meilleurs paramètres (Grid Search Random Forest) : {'bootstrap': True, 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}
Meilleure précision (Grid Search Random Forest) : 1.0

Meilleurs paramètres (Optuna Arbre de Décision) : {'criterion': 'gini', 'max_depth': 10, 'min_samples_split': 6, 'min_samples_leaf': 1}
Meilleure précision (Optuna Arbre de Décision) : 0.9933333333333333

Meilleurs paramètres (Optuna Random Forest) : {'n_estimators': 97, 'max_depth': 10, 'min_samples_split': 3, 'min_samples_leaf': 2, 'bootstrap': True}
Meilleure précision (Optuna Random Forest) : 0.9866666666666667

Précision globale (Arbre de Décision) : 0.9733333333333334
Précision globale (Random Forest) : 0.9866666666666667

Aicha Ndiaye
DIC2 IABD

Random Forest est plus performant.
Matrice de confusion (Arbre de décision) :
[[116 1]
[0 33]]

Matrice de confusion (Random Forest) :
[[116 1]
[1 32]]

Rapport de classification (Arbre de décision) :

	precision	recall	f1-score	support
malicious	1.00	0.99	1.00	117
non-malicious	0.97	1.00	0.99	33
accuracy			0.99	150
macro avg	0.99	1.00	0.99	150
weighted avg	0.99	0.99	0.99	150

Rapport de classification (Random Forest) :

	precision	recall	f1-score	support
malicious	0.99	0.99	0.99	117
non-malicious	0.97	0.97	0.97	33
accuracy			0.99	150
macro avg	0.98	0.98	0.98	150
weighted avg	0.99	0.99	0.99	150

Précision globale (Arbre de décision) : 0.9933333333333333
Précision globale (Random Forest) : 0.9866666666666667

L'Arbre de décision est plus performant.