Rapport de Projet IA N°2 DS : Détection de Malware basée sur le Machine Learning suite Optimisation des Hyperparamètres pour la Classification.

Etudiants :

- Assane BA Master 1 Intelligence Artificielle et Big Data
- Cheikh Mouhamed Tidiane THIAM Master 1 Intelligence Artificielle et Big Data

1. Sélection des Algorithmes de Classification :

Choisissez judicieusement quatre algorithmes de machine learning dédiés à la classification. Recherche des Meilleurs Hyperparamètres :

Notre choix c'est port sur les algorithmes de classification ci-dessous :

- KNeighborsClassifier() ;
- DecisionTreeClassifier() ;
- RandomForestClassifier() ;
- LogisticRegression()

```python
# Choix et Définition des modèles à évaluer
models = {
    "LogisticRegression": LogisticRegression(),
    "DecisionTreeClassifier": DecisionTreeClassifier(),
    "RandomForestClassifier": RandomForestClassifier(),
    "KNN": KNeighborsClassifier()
}
```

2. Pour chaque algorithme sélectionné, effectuez une recherche sur le web pour identifier les trois hyperparamètres les plus influents sur la précision du modèle :

Pour chaque algorithme, des valeurs ont été proposées pour chaque élément de paramètre. Ce faisant la meilleure combinaison nous sera retournée et sera utilisé afin d'améliorer le résultat de prédiction de chaque algorithme.

```python
# Définition des hyperparamètres d'optimiser pour chaque algorithme
param_distributions = {
    "LogisticRegression": {'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]},
    "DecisionTreeClassifier": {'max_depth': [1, 5, 10, 20],
                               'min_samples_split': [2, 5, 10],
                               'min_samples_leaf': [1, 2, 5, 10]},
    "RandomForestClassifier": {'criterion': ['gini', 'entropy'],
                               'n_estimators': [10, 50, 100, 150],
                               'max_depth': [1, 5, 10, 20],
                               'min_samples_split': [2, 5, 10],
                               'min_samples_leaf': [1, 2, 5, 10]},
    "KNN": {'n_neighbors': [3, 5, 7, 10],
            'weights': ['uniform', 'distance'],
            'p': [1, 2]}
}
```

3. Optimisation par Grid Search, Random Search, ou Optuna :

Application, pour chaque algorithme, la méthode d'optimisation avancée telle que Random Search afin de déterminer les meilleures valeurs pour les hyperparamètres. Ces valeurs correspondent à celles qui rendent le modèle optimal en termes de performance.

```
# Utiliser de RandomizedSearchCV pour optimiser les modèles
for model_name, model in models.items():
    random_search = RandomizedSearchCV(model, param_distributions[model_name], n_iter=7, cv=5, scoring='accuracy',
                                       random_state=42)

    random_search.fit(X_train_scaled, y_train)

    # Obtenir les meilleurs hyperparamètres après l'optimisation
    best_params = random_search.best_params_

    # Évaluation du modèle avec les meilleurs paramètres
    y_pred = random_search.predict(X_test_scaled)
    accuracy = accuracy_score(y_test, y_pred)

    results_after_optimization.append({
        'Modèle': model_name,
        'Meilleurs hyperparamètres': best_params,
        'Précision après optimisation': accuracy
    })

    # Évaluation du modèle avec les paramètres par défaut
    model.fit(X_train_scaled, y_train)
    y_pred_default = model.predict(X_test_scaled)
    accuracy_default = accuracy_score(y_test, y_pred_default)

    results_before_optimization.append({
        'Modèle': model_name,
        'Précision avant optimisation': accuracy_default
    })
```

4. **Comparaison Avant et Après l'Optimisation :** élaborez un tableau comparatif présentant, pour chaque algorithme, les résultats avant et après l'optimisation des hyperparamètres. Incluez des métriques de performance telles que la précision du modèle et la matrice de confusion. Conclusions et Recommandations :

Comparaison avant et après optimisation:

	Modèle	Précision avant optimisation	Meilleurs hyperparamètres	Précision après optimisation
3	KNN	0.980744	{'weights': 'distance', 'p': 1, 'n_neighbors':...	0.985982

Avec l'algorithme **K Neighbors Classifier** (Voisins les plus proches), suite optimisation des hyperparamètres avec la méthode **Randomized Search CV**, nous avons gagné **0,005238** en précision par rapport à celle du même modèle avec les paramètres par défaut. Voir ci-dessus.

Comparaison avant et après optimisation:

	Modèle	Précision avant optimisation	Meilleurs hyperparamètres	Précision après optimisation
1	DecisionTreeClassifier	0.987413	{'min_samples_split': 5, 'min_samples_leaf': 1...	0.987632

De même qu'avec l'algorithme **Decision Tree Classifier** (Arbres de décision), suite optimisation des hyperparamètres avec la méthode **Grid Search CV**, nous avons gagné **0,000219** en précision par rapport à celle du même modèle avec les paramètres par défaut.

- **Conclusion**
  En somme nous pouvons retenir que l'optimisation des hyperparamètres d'un algorithme donné contribue à une amélioration sur la précision/prédiction du modèle à déployer.

- **Recommandation**
  Il est d'une importance capitale de toujours chercher à optimiser voire améliorer la précision de nos modèles via les méthodes d'optimisation des hyperparamètres.

5. <u>Analyse des résultats obtenus suite optimisation et conclusion sur l'impact de cette démarche sur les performances des modèles :</u>

L'analyse des résultats obtenus après optimisation des hyperparamètres, ont données une révélés sur la possibilité de rendre un modèle performant de manière continue tout en agissant sur ses paramètres. Autrement dit la recherche de paramètre adéquat au jeu de données(dataset) correspondants permettra toujours d'améliorer la prédiction d'un modèle donné.

En conclusion la performance du modèle sera toujours garantie avec un choix judicieux de la méthode d'optimisation (Grid Search, Random Search, Optuna …) et de l'algorithme de classification le tout étant en adéquation avec les données à traiter(dataset).

6. <u>Recommandations quant au choix final des modèles et de leurs hyperparamètres pour la classification efficace des logiciels malveillants :</u>

Le choix du modèle final **(K Neighbors Classifier)** est motivé par les résultat obtenu suite optimisation par la méthode **Randomized Search CV**. Ce modèle une fois déployé, permettra de faire des prédictions et ainsi garantir une classification efficace des logiciels malveillants.