

# Systèmes multi-agents : métaheuristiques à base de jeu

Touahri sara

SIW

Nadir Mahammed

# Introduction

**Les systèmes multi-agents (SMA)** impliquent plusieurs agents autonomes qui interagissent pour résoudre des problèmes complexes. Ces agents peuvent collaborer ou rivaliser, chacun prenant des décisions en fonction de l'environnement et des actions des autres.

**Les métahéuristiques à base de jeu** sont une approche inspirée des stratégies de jeu, où les agents cherchent collectivement à optimiser une solution. Dans ce projet, cette méthode est utilisée pour optimiser les paramètres d'un modèle de machine learning, en maximisant sa performance.

# La mét heuristiques choisie

MÉTAHEURISTIQUE : OPTIMISATION DE PARAMÈTRES POUR LE  
MODÈLE RANDOM FOREST

L'optimisation des modèles de machine learning implique la recherche des meilleurs paramètres afin d'améliorer les performances du modèle. Dans ce projet, nous avons choisi d'optimiser les paramètres du Random Forest Classifier en utilisant une approche mét heuristique. La mét heuristique permet de rechercher de manière efficace dans un espace de paramètres complexe, en explorant différentes configurations pour maximiser la performance du modèle.

# La métاهeuristiques choisie

**PRINCIPE DE LA MÉTAHEURISTIQUE CHOISIE : MLBO (MACHINE LEARNING BASED OPTIMIZATION)**

L'algorithme utilisé ici est inspiré des principes de la recherche naturelle. Il s'agit d'un processus itératif où une population d'agents (solutions) évolue au fil des itérations pour améliorer la fonction de fitness, qui dans ce cas est l'accuracy du modèle. Chaque agent représente un ensemble de paramètres pour le modèle, et les interactions entre les agents visent à découvrir les configurations optimales. L'optimisation est basée sur l'évolution de ces solutions grâce à des mécanismes inspirés des stratégies de recherche dans la nature, tels que la collaboration et la compétition entre agents.

# Fonction de Fitness : Accuracy du Modèle

La fonction de fitness mesure la qualité d'une solution candidate (ensemble de paramètres) dans le cadre de l'optimisation. Ici, la métrique utilisée est l'accuracy (précision) du modèle Random Forest, qui correspond à la proportion des prédictions correctes sur les données de validation.

## POURQUOI L'ACCURACY ?

métrique simple et permet de comparer facilement les performances

## OBJECTIF DE L'OPTIMISATION

Maximiser l'accuracy sur l'ensemble de validation en identifiant les paramètres optimaux du Random Forest

Plus une solution (ensemble de paramètres) améliore l'accuracy, plus elle est favorisée dans les itérations suivantes..

# Dataset utilisée

Nom du dataset : Dataset de détection de profils faussement créés sur Twitter.

Source : CSV provenant de l'analyse des comportements des utilisateurs sur Twitter.(dataset2)

Taille et Structure :

- Nombre de lignes : 1001.
- Nombre de colonnes : 18.
- Caractéristiques principales :
  - Colonnes numériques : followers\_count, friends\_count, favourites\_count, statuses\_count, etc.
  - Colonnes catégoriques : verified, default\_profile, is\_fake, etc.

# Les prétraitements appliqués

- **Suppression des Colonnes Inutiles :**

Suppression des colonnes non pertinentes pour l'analyse, telles que id et description.

Ces colonnes ne contribuent pas directement à la détection des profils faux.

- **Gestion des Valeurs Manquantes**

Colonnes Numériques : Remplacement des valeurs manquantes par 0 (exemple : followers\_count, friends\_count).

Colonnes Catégoriques : Remplacement des valeurs manquantes par 'Unknown' pour maintenir la cohérence des données (exemple : verified, default\_profile).

# Les prétraitements appliqués

- **Encodage des Variables Catégorielles :**

Utilisation de la méthode One-Hot Encoding pour convertir les variables catégorielles en variables numériques.

Par exemple, les colonnes telles que verified, default\_profile, et is\_fake sont transformées en plusieurs colonnes binaires.

Cela permet au modèle de comprendre les relations entre ces catégories sans introduire d'ordinalité.

- **Standardisation des Variables Numériques :**

Application de StandardScaler pour mettre à l'échelle les variables numériques (followers\_count, friends\_count, etc.).

Cette étape transforme les données de manière à ce qu'elles aient une moyenne de 0 et un écart-type de 1, facilitant l'entraînement du modèle.

# Méthode de Feature Selection : SelectKBest avec f\_classif

La méthode SelectKBest sélectionne les caractéristiques les plus pertinentes en utilisant un critère de score. Le critère `f_classif` calcule la statistique F pour chaque caractéristique, mesurant sa dépendance avec la cible. Les `k` meilleures caractéristiques sont ensuite retenues. Cela permet de réduire la dimensionnalité et d'améliorer les performances du modèle.

# Méthode d'Optimisation des Paramètres

- Paramètres Optimisés :
- Les paramètres du modèle `RandomForestClassifier` sont optimisés à l'aide de l'algorithme métaheuristique.
- Les principaux paramètres optimisés sont :
  - `n_estimators` : Le nombre d'arbres dans la forêt (entre 50 et 200).
  - `max_depth` : La profondeur maximale des arbres (entre 3 et 20).
  - `min_samples_split` : Le nombre minimum d'échantillons requis pour diviser un nœud (entre 2 et 10).
  - `min_samples_leaf` : Le nombre minimum d'échantillons nécessaires pour être à une feuille (entre 1 et 5).
- Plages des Paramètres Définies pour l'Optimisation :
- Chaque paramètre a une plage spécifique de valeurs à explorer lors de l'optimisation.
- L'algorithme métaheuristique ajustera ces paramètres pour maximiser la précision du modèle.

# Algorithmes de Machine Learning Utilisés

- RandomForestClassifier :

Il s'agit d'un modèle d'arbres de décision qui combine plusieurs arbres pour améliorer la précision et la robustesse du modèle.

C'est un algorithme puissant pour la classification binaire, utilisé ici pour distinguer les profils faussement créés des vrais profils sur Twitter.

- Optimisation des Paramètres :

L'algorithme est optimisé à l'aide d'une métaheuristique (basée sur l'optimisation de paramètres), permettant de trouver les meilleures configurations pour les paramètres clés du modèle, tels que n\_estimators, max\_depth, min\_samples\_split, et min\_samples\_leaf.

# Outils et Librairies Utilisées

## **PYTHON :**

Langage de programmation principal utilisé pour développer l'ensemble du projet.

## **PANDAS ET NUMPY**

Pour le traitement des données, la manipulation des DataFrames et des tableaux numériques.

## **SCIKIT-LEARN**

Utilisé pour l'entraînement du modèle (RandomForestClassifier), l'évaluation des performances (précision, rappel, etc.), et l'optimisation des paramètres.

## **MATPLOTLIB**

Pour la visualisation des résultats, notamment l'affichage de l'importance des caractéristiques et la progression de l'optimisation.

## **STANDARDSCALER ET ONEHOTENCODER**

Utilisés pour le prétraitement des données, respectivement pour la standardisation des variables numériques et l'encodage des variables catégorielles.

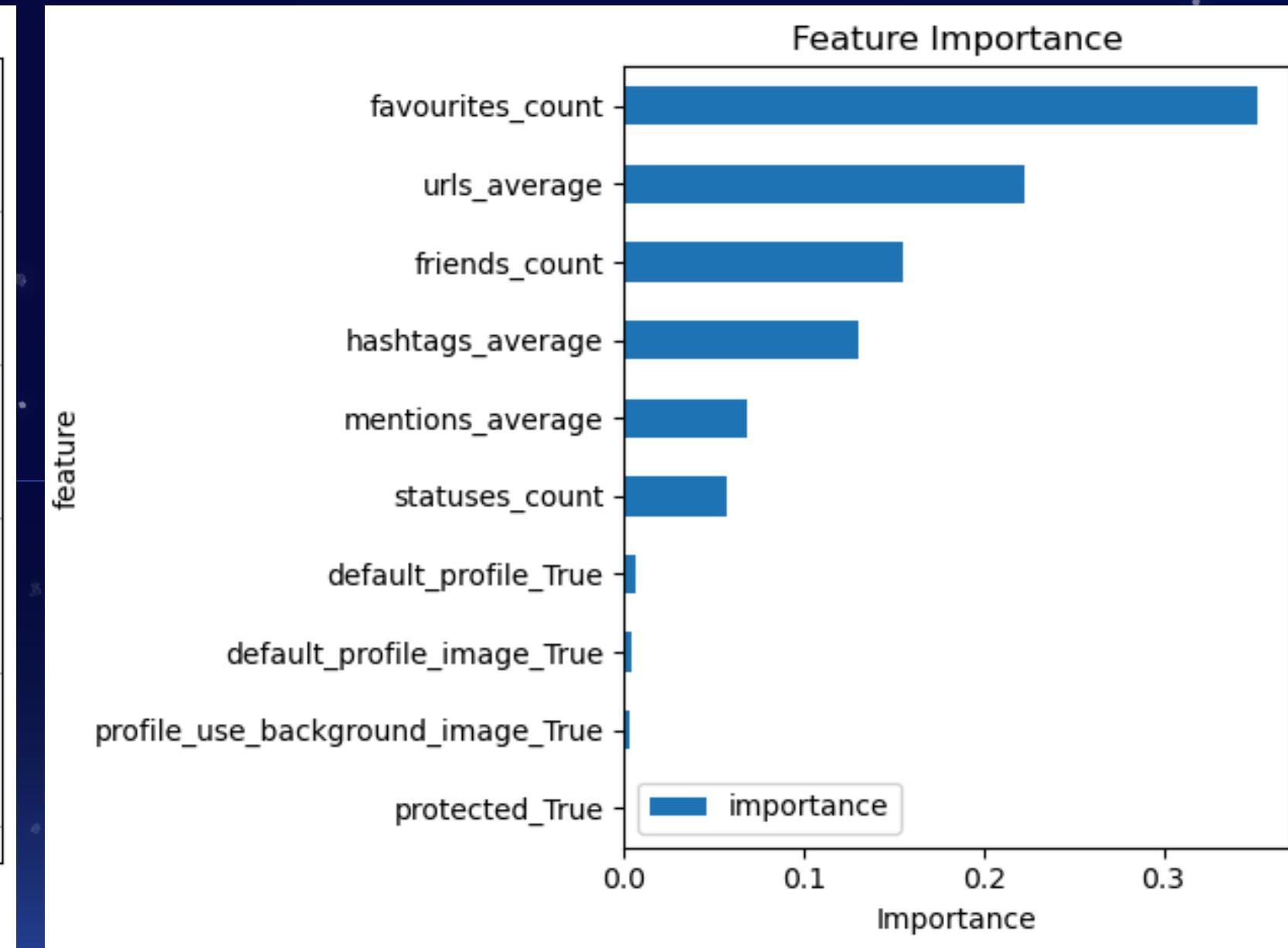
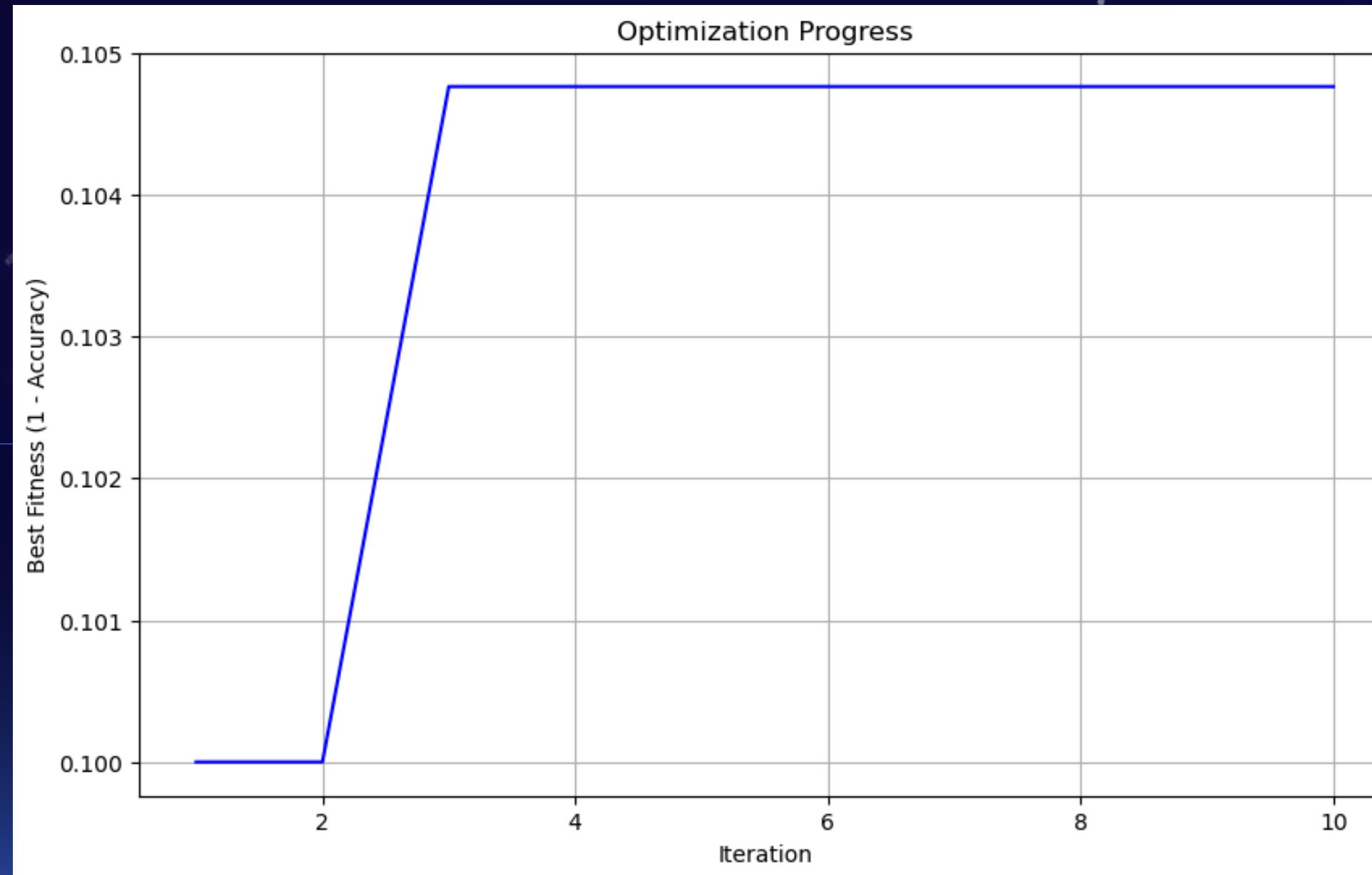
## **ENVIRONNEMENT DE DÉVELOPPEMENT :**

Le projet est développé dans Jupyter Notebook,

# Résultats Obtenus - Évaluation

Paramètres	Valeur	Métriques (Validation)	Valeur	Métriques (Test Final)	Valeur
n_estimators	113	Accuracy	0.8857	Accuracy	0.9133
max_depth	8	Precision	0.8785	Precision	0.9156
min_samples_split	6	Recall	0.8952	Recall	0.9156
min_samples_leaf	3	F1-score	0.8868	F1-score	0.9156
		ROC-AUC	0.9617	ROC-AUC	0.9706

# Résultats Obtenus - Évaluation



# Conclusion

Le modèle Random Forest optimisé avec une approche métaheuristique a donné d'excellents résultats, atteignant une précision de 91,33 % sur l'ensemble de test. Les paramètres optimaux ont été déterminés pour maximiser les performances du modèle. La sélection des caractéristiques et l'optimisation des hyperparamètres ont significativement contribué à améliorer l'efficacité du modèle. Les métriques élevées telles que le F1-score et le ROC-AUC montrent une bonne capacité de classification et une robustesse face aux données.

Merci !