CENTRALE LYON

École Centrale Lyon

MOD 7.2
Rapport

Identification des utilisateurs de Copilote

Élèves :
Anthonio ZAVATRA
Elmehdi MARHFOUR
Youssef ISMAINI

Enseignant :
Julien Velcin

Table des matières

1 Introduction

1.1 Contexte et enjeux

Le logiciel Copilote, développé par la société Infologic, représente un outil stratégique dans le secteur agro-alimentaire. L'analyse automatique des traces d'utilisation constitue un enjeu majeur pour l'entreprise, permettant d'améliorer continuellement l'offre client et d'adapter les fonctionnalités aux besoins spécifiques de chaque utilisateur. Dans ce contexte, l'identification automatique des utilisateurs à partir de leurs comportements représente une opportunité d'optimisation significative.

Ce projet s'inscrit dans le cadre du module MOD 7.2 d'Introduction à la Science des Données et vise à développer un système de classification automatique capable d'identifier avec précision les utilisateurs du logiciel Copilote en analysant leurs séquences d'actions.

1.2 Objectifs du projet

L'objectif principal consiste à concevoir et implémenter un modèle de Machine Learning performant pour l'identification des utilisateurs distincts présents dans le dataset. Pour atteindre cet objectif, nous avons défini les sous-objectifs suivants :

- Extraire des caractéristiques discriminantes à partir des séquences d'actions brutes
- Développer un modèle de classification robuste basé sur l'algorithme Random Forest
- Analyser les patterns comportementaux et identifier les variables les plus influentes
- Optimiser les performances du modèle pour maximiser l'accuracy sur le challenge Kaggle
- Fournir une solution industrialisable avec un code modulaire et documenté

1.3 Approche méthodologique

Notre approche s'articule autour d'une méthodologie rigoureuse en cinq phases : (1) exploration et compréhension des données, (2) extraction et engineering des features, (3) sélection et optimisation du modèle Random Forest, (4) évaluation multi-critères des performances, et (5) analyse des résultats et identification des axes d'amélioration.

2 Analyse et préparation des données

2.1 Description du dataset

Les données fournies se composent de deux ensembles : un dataset d'entraînement contenant 247 utilisateurs uniques avec leurs traces d'utilisation, et un dataset de test pour l'évaluation finale. Chaque ligne représente une session utilisateur caractérisée par :

- **Identifiant utilisateur** (util) : Code unique à 3 lettres (uniquement dans train)
- **Navigateur** : Type de navigateur utilisé (Firefox, Chrome, Edge)
- **Séquence d'actions** : Suite ordonnée d'interactions avec le logiciel

Les actions enregistrées incluent diverses interactions : création d'écrans, exécution de boutons, affichage de dialogues, navigation entre onglets, ainsi que des pauses temporelles codifiées (t5, t10, t15, t20, t25) indiquant le rythme d'utilisation.

2.2 Exploration statistique

L'analyse exploratoire révèle une grande hétérogénéité dans les données :

TABLE 1 – Statistiques descriptives du dataset

Caractéristique	Valeur
Nombre d'utilisateurs	247
Sessions totales	3000+
Actions uniques	35
Longueur moyenne séquence	692 actions
Longueur minimale	0 actions
Longueur maximale	11977 actions

La distribution des navigateurs montre une prédominance de Firefox (44%), suivi de Google Chrome (40%), Microsoft Edge (13%) et enfin Opéra (0.7%). Cette répartition suggère des préférences technologiques potentiellement corrélées aux profils utilisateurs.

2.3 Nettoyage et prétraitement

Le processus de nettoyage a nécessité plusieurs étapes critiques :

1. **Gestion des valeurs manquantes** : Remplissage intelligent des colonnes d'actions vides
2. **Harmonisation des formats** : Standardisation des notations d'actions (gestion des parenthèses et paramètres)
3. **Filtrage des sessions** : Suppression des sessions avec moins de 5 actions (considérées incomplètes)
4. **Encodage cohérent** : Transformation des variables catégorielles en format numérique

3 Extraction et sélection des caractéristiques

3.1 Engineering des features

Notre stratégie d'extraction de features vise à capturer la complexité des comportements utilisateurs à travers quatre dimensions principales :

3.1.1 Features comportementales globales

Ces caractéristiques capturent les patterns généraux d'utilisation :
— **features simples** : comptages d'actions réelles, uniques, nombre de marqueurs temps
— **features de durée** : durée de session approximative en multiples de 5 sec
— **features catégorielles** : actions les plus fréquentes (selon les regex)

Après une analyse comparative de plusieurs algorithmes (Régression Logistique, SVM, XGBoost, Réseaux de Neurones), nous avons sélectionné Random Forest pour les raisons suivantes :

TABLE 2 – Comparaison des algorithmes testés

Algorithme	**Accuracy**	**F1-Score Micro**	**F1-Score Macro**
Random Forest	0.7393	0.7393	0.6645
SVM (avec Class Weights)	0.5427	0.5427	0.5120
XGBoost	0.6845	0.6845	0.6209
Logistic Regression (avec Class Weights)	0.6067	0.6067	0.5851

Random Forest offre les meilleures performances, en accuracy, F1-Score Micro et Macro.

3.2 Optimisation des hyperparamètres

L'optimisation des hyperparamètres du modèle *Random Forest* a été réalisée à l'aide de la méthode **RandomizedSearchCV** avec une validation croisée à 3 plis. Cette approche aléatoire permet d'explorer efficacement un large espace de recherche tout en réduisant le temps de calcul comparé à une recherche exhaustive.

Les principaux hyperparamètres étudiés sont les suivants :

— `n_estimators` : nombre d'arbres dans la forêt (100 → 300)
— `max_depth` : profondeur maximale des arbres (`None`, 10, 20, 30)
— `min_samples_split` : nombre minimal d'échantillons requis pour diviser un nœud (2 → 10)
— `min_samples_leaf` : nombre minimal d'échantillons par feuille (1 → 5)
— `max_features` : stratégie de sélection des variables (`'sqrt'`, `'log2'`)
— `class_weight` : gestion du déséquilibre des classes (`None`, `'balanced'`)

La recherche a été effectuée sur 20 combinaisons aléatoires (`n_iter=20`), en utilisant le score *F1-macro* comme métrique d'optimisation. Cette configuration a permis d'identifier un ensemble d'hyperparamètres offrant un bon compromis entre biais et variance, tout en améliorant la robustesse du modèle.

3.3 Validation croisée et robustesse

La validation croisée 3-fold confirme la stabilité du modèle :
— Accuracy : 75.15%
— Performances cohérentes sur tous les folds

4 Discussion et perspectives

4.1 Limitations identifiées

Plusieurs limitations méritent attention :
— **Sessions courtes** : Performance dégradée pour les séquences < 20 actions
— **Utilisateurs similaires** : Confusion entre profils aux comportements proches

— **Évolution temporelle** : Modèle statique ne capturant pas l'évolution des comportements
— **Données déséquilibrées** : Certains utilisateurs sous-représentés

4.2 Axes d'amélioration

Pour adresser ces limitations, nous proposons :

4.2.1 Améliorations court terme

— **Augmentation de données** : Techniques de SMOTE pour les classes minoritaires
— **Features spécifiques** : Caractéristiques adaptées aux sessions courtes

5 Justification du Choix des Modèles de Classification

Nous avons sélectionné quatre classifieurs distincts pour évaluer différentes stratégies d'apprentissage. L'objectif est de trouver l'approche qui maximise le **F1 Macro Score**, garantissant une performance équilibrée pour toutes les classes.

5.1 Présentation et Justification des Modèles

— **Random Forest Classifier (Optimisé par Grid Search) :** Modèle d'ensemble de type *Bagging*. Choisi pour sa **robustesse** et sa capacité à réduire la variance. **GridSearchCV** a été utilisé pour trouver l'ensemble d'hyperparamètres (θ) qui maximise la performance générale.
— **Gradient Boosting Classifier :** Modèle d'ensemble de type *Boosting*. Privilégié pour son potentiel de **haute précision** en corrigeant les erreurs des modèles précédents par une descente de gradient.
— **Support Vector Machine (SVM) :** Classifieur basé sur la **maximisation de la marge** dans l'espace des caractéristiques. Il est très efficace dans les espaces de grande dimension et permet d'explorer des séparations non linéaires via la méthode du noyau.
— **Logistic Regression :** Modèle **linéaire** simple. Il sert de **modèle de référence** (*baseline*) pour évaluer si la complexité des modèles d'ensemble est justifiée par un gain significatif de performance.

6 Résultats et Justification du Modèle Retenu

Le **Random Forest Classifier** optimisé a produit la **meilleure performance** sur l'ensemble de validation, comme en témoigne le plus haut F1 Macro Score.

6.1 Justification de la Rétention du Random Forest

La rétention de ce modèle est justifiée par le succès de l'optimisation des hyperparamètres, qui a permis de trouver le jeu de paramètres θ^* donnant le score le plus élevé :

$$\theta^* = \underset{\theta \in \text{param}}{\text{argmax}} \left(\text{F1 Macro ScoreCV}(\theta) \right)$$

Où le F1 Macro Score est la moyenne non pondérée de l'équilibre entre Précision et Rappel pour chaque classe :

$$\text{F1 Macro Score} = \frac{1}{N} \sum_{i=1}^{N} \text{F1}i = \frac{1}{N} \sum i = 1^{N} 2 \cdot \frac{\text{Précision}_i \cdot \text{Rappel}_i}{\text{Précision}_i + \text{Rappel}_i}$$

Conclusion : L'approche de **Bagging** combinée à l'optimisation ciblée a permis de construire un modèle à la fois **robuste** et **performant**, capable de fournir une performance homogène sur l'ensemble des classes, justifiant son choix pour la classification finale.