Note méthodologique : preuve de concept (POC)

Prédiction du statut des rendez-vous médicaux

Étudiants : BENAOUDIA Leticia - LAMARI Azzeddine
Formation : Master Informatique
Date : 19 janvier 2026

Objectif. Mettre en place une preuve de concept de modélisation supervisée afin de prédire le **statut final** d'un rendez-vous (*attended / not attended / cancelled*) et d'en déduire des actions opérationnelles (rappels ciblés, ajustement de planning).
Contraintes. Document synthétique, traçabilité des expériences (MLflow), et explicabilité du modèle (SHAP).

Table des matières

1. Dataset retenu

1.1. Contexte et source

Le dataset utilisé est un jeu de données **synthétique** simulant l'activité d'un cabinet médical sur une longue période (2015–2024). Il modélise la planification, la réalisation et les issues possibles d'un rendez-vous.

Lien dataset :
https://www.kaggle.com/datasets/carogonzalezgaltier/medical-appointment-scheduling-system

Pourquoi ce dataset :
Ce dataset a été choisi car il présente plusieurs avantages pour une preuve de concept :

— il reproduit de manière réaliste l'organisation d'un système de prise de rendez-vous médicaux ;
— sa nature synthétique permet de travailler sans exposer de données personnelles réelles, tout en conservant des comportements plausibles ;

1.2. Fichier et cible utilisés pour l'entraînement

Dans ce projet, l'entraînement des modèles est effectué uniquement sur le fichier `appointments.csv`. La variable cible est la colonne `status`.

Conformément au script `train.py`, lorsque la cible n'est pas numérique, elle est transformée en variable binaire afin d'isoler les rendez-vous à risque. Le code considère comme positifs des états tels que `cancelled` ou `did not attend` (no-show), et négatifs les états indiquant un rendez-vous honoré.
Dans un souci de cohérence métier et de qualité des données, seules les lignes dont la valeur de la colonne `status` appartient aux catégories suivantes ont été conservées : `attended`, `did not attend` et `cancelled`. Les observations associées à des statuts non exploitables ou ambigus (`unknown`, `scheduled`) ont été supprimées, car elles ne correspondent pas à un état final du rendez-vous et ne peuvent pas être utilisées pour l'apprentissage supervisé.

1.3. Colonnes de `appointments.csv`

Colonne	Signification
appointment_id	Identifiant unique du rendez-vous.
slot_id	Référence au créneau horaire.
scheduling_date	Date de prise du rendez-vous.
appointment_date	Date du rendez-vous.
appointment_time	Heure planifiée du rendez-vous.
scheduling_interval	Délai (jours) entre prise et rendez-vous.
status	**Statut final** (cible : attended / not attended / cancelled).
check_in_time	Heure réelle d'arrivée du patient.
appointment_duration	Durée réelle (minutes).
start_time	Heure réelle de début.
end_time	Heure réelle de fin.
waiting_time	Temps d'attente (minutes).
patient_id	Identifiant patient.
sex	Sexe du patient.
age	Âge du patient.
age_group	Groupe d'âge (tranches).

Intérêt métier. Le dataset contient des variables de *planification* (délai, horaire) et de *déroulement* (arrivée, durée, attente) qui sont pertinentes pour anticiper les non-présentations.

2. Les concepts de l'algorithme récent

2.1. Gradient Boosting et XGBoost

Le modèle récent retenu est XGBoost (*eXtreme Gradient Boosting*), un algorithme d'ensemble basé sur le **Gradient Boosting**. Le principe consiste à construire une suite d'arbres de décision **séquentiels** : chaque nouvel arbre corrige les erreurs commises par l'ensemble des arbres précédents.

Une formulation simplifiée est :

$$F_t(x) = F_{t-1}(x) + \eta \cdot h_t(x)$$

où h_t est un nouvel arbre et η le taux d'apprentissage.

2.2. Pourquoi XGBoost est adapté ici

Dans un problème de rendez-vous médicaux, les relations entre variables sont souvent non linéaires (ex. effet du délai, de l'horaire et de la durée combinés). XGBoost est adapté car :

- il est performant sur données tabulaires hétérogènes ;
- il capture des interactions complexes ;
- il intègre de la régularisation (limite le surapprentissage) ;
- il est explicable grâce à SHAP, ce qui est crucial dans un contexte sensible.

3. La modélisation

3.1. Préparation et format de la variable cible

La variable cible utilisée pour l'apprentissage est la colonne `status`, qui décrit l'issue finale d'un rendez-vous médical. À l'origine, cette variable est catégorielle (valeurs textuelles telles que `attended`, `did not attend`, `cancelled`).

Conformément au script `train.py`, la variable `status` est transformée en une variable binaire numérique afin de formuler le problème comme une classification supervisée binaire. La conversion appliquée est la suivante :

— valeur **1** : rendez-vous à risque (`cancelled`, `did not attend`) ;
— valeur **0** : rendez-vous honoré (`attended`).

Ce format binaire permet une utilisation homogène de la variable cible par l'ensemble des modèles évalués.

3.2. Pourquoi ces 3 modèles (LogReg / RF / XGBoost)

Les trois modèles étudiés utilisent cette variable cible sous le même format numérique, mais l'exploitent de manière différente :

— **Régression logistique** : utilise la variable cible binaire pour estimer la probabilité qu'un rendez-vous appartienne à la classe à risque, via une fonction logistique optimisant la log-loss.
— **Random Forest** : exploite la cible binaire comme étiquette discrète afin d'apprendre des règles de décision séparant les rendez-vous honorés des rendez-vous à risque.
— **XGBoost** : utilise la cible binaire numérique dans un cadre de Gradient Boosting, en optimisant une fonction de perte de classification binaire et en corrigeant progressivement les erreurs de prédiction.

3.3. Méthodologie conforme à `train.py`

La modélisation suit un pipeline scikit-learn, garantissant la reproductibilité :

— séparation de la cible `status` et des features ;
— **split train/test stratifié** (80/20) afin de préserver la proportion des classes ;
— prétraitement via **ColumnTransformer** :
  — numériques : imputation médiane ;
  — catégorielles : imputation mode + **one-hot encoding**.
— intégration prétraitement + modèle dans un **Pipeline** (pas de fuite entre train et test) ;
— suivi complet des runs via **MLflow** (paramètres, métriques, artefacts).

3.4. Métriques et optimisation

Les métriques calculées sont : accuracy, precision, recall, F1 et ROC-AUC. Un **score métier** est défini comme :

$$metier\_score = \frac{precision + recall}{2}$$

Ce score reflète le compromis entre faux positifs et faux négatifs, avec une importance forte de la capacité à détecter les rendez-vous à risque.

L'optimisation de XGBoost est réalisée avec **GridSearchCV** et une validation croisée stratifiée (**StratifiedKFold, 5 folds**), en maximisant le score métier.

Pourquoi cette metrique :
Le choix de cette métrique est motivé par le contexte métier du problème. Un faux négatif (rendez-vous prédit comme honoré alors qu'il est en réalité manqué) entraîne une perte directe de créneau, tandis qu'un faux positif génère au plus une action préventive inutile (rappel, confirmation).

Ainsi, il est essentiel de privilégier la capacité du modèle à détecter les rendez-vous à risque (*recall*), tout en maintenant un niveau de précision acceptable afin de limiter les actions inutiles. Le score métier retenu permet de refléter ce compromis et d'orienter l'optimisation du modèle vers un usage opérationnel.

4. Une synthèse des résultats

4.1. Résultats des 3 modèles (MLflow)

Les runs MLflow montrent les performances suivantes (jeu de test) :

Modèle	Accuracy	Precision	Recall	F1	Score métier
LogReg (baseline)	0.788	1.000	0.057	0.107	0.528
Random Forest (baseline)	1.000	1.000	1.000	1.000	1.000
XGBoost (GridSearch)	0.9996	0.998	1.000	0.999	0.999

4.2. Interprétation et comparaison

LogReg présente une précision élevée mais un rappel très faible : le modèle est trop conservateur, et laisse passer de nombreux rendez-vous réellement à risque (faux négatifs). Il n'est pas adapté à une stratégie de prévention.

Random Forest obtient des scores parfaits. Toutefois, des performances à 1 sur toutes les métriques peuvent indiquer une fuite d'information ou un apprentissage sur des variables non disponibles en conditions réelles. Ce modèle sert ici à explorer le potentiel des données, mais nécessite une validation stricte avant usage opérationnel.

XGBoost fournit le meilleur compromis pour une mise en pratique : rappel parfait, très peu de faux positifs, et optimisation explicite sur un score métier.

4.3. Figures MLflow (preuves)

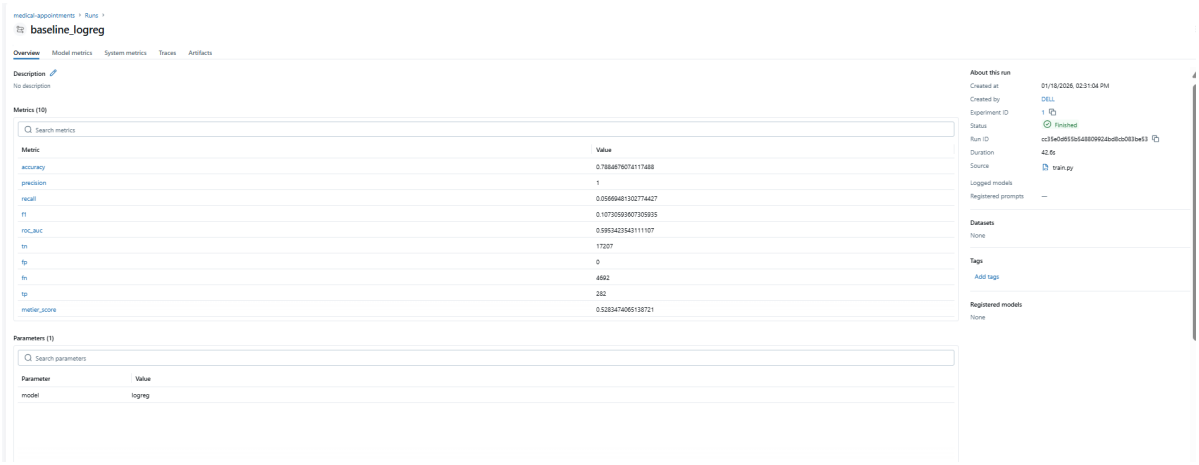

FIGURE 1 – Run MLflow — LogReg baseline : métriques et matrice de confusion.

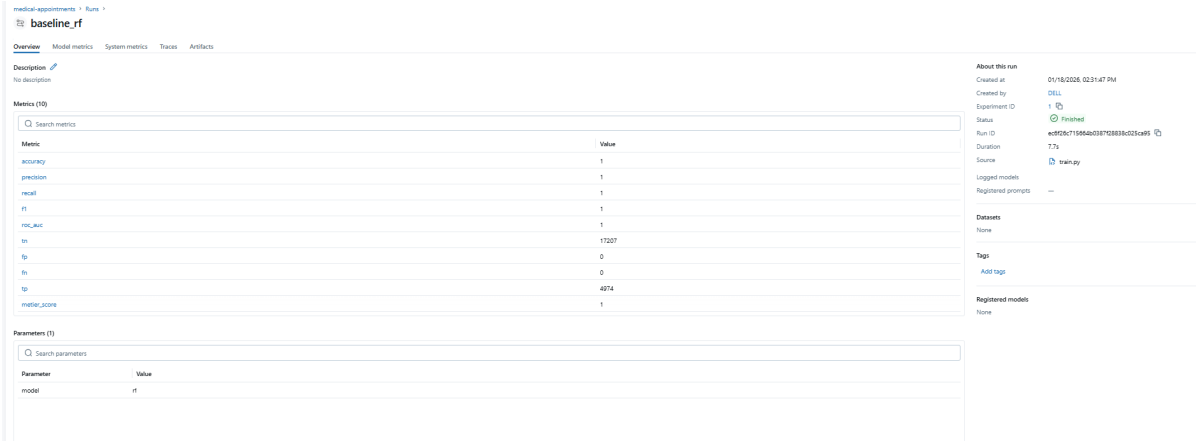

FIGURE 2 – Run MLflow — Random Forest baseline : métriques et matrice de confusion.

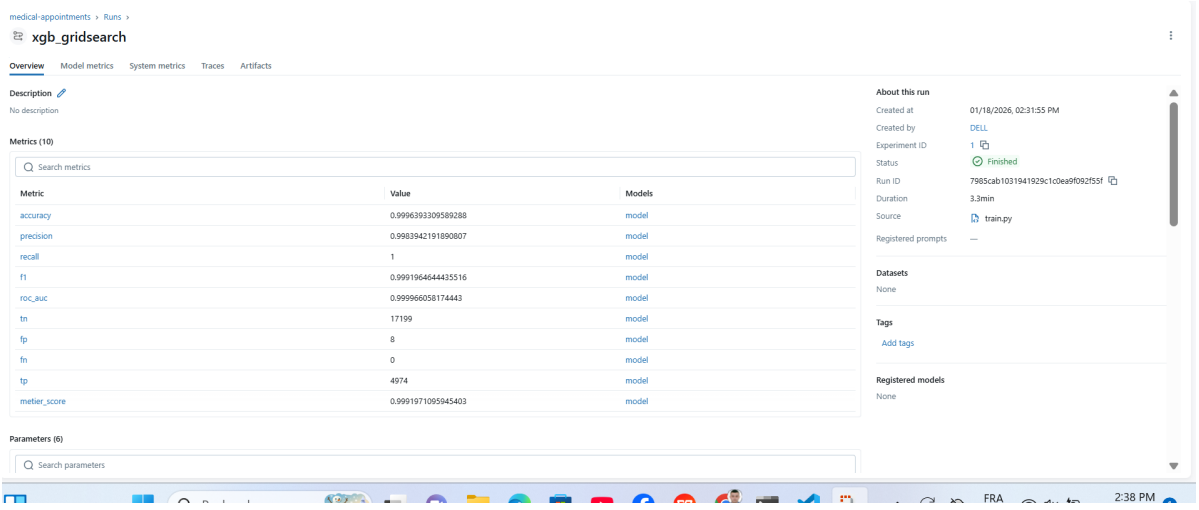

FIGURE 3 – Run MLflow — XGBoost GridSearch : métriques et matrice de confusion.

4.4. Conclusion

Les résultats confirment l'intérêt de la technique récente (XGBoost) par rapport aux baselines, en particulier pour détecter les rendez-vous à risque. La combinaison performance + explicabilité rend l'approche exploitable dans un contexte opérationnel, sous réserve de validations complémentaires.

5. Analyse de la feature importance globale et locale

5.1. Pourquoi l'explicabilité est utile pour prédire et décider

La performance globale d'un modèle ne suffit pas dans un contexte opérationnel : il est nécessaire de comprendre **quelles variables** influencent les prédictions et **pourquoi** un rendez-vous est considéré à risque. SHAP permet :

— de vérifier que le modèle s'appuie sur des variables pertinentes et exploitables ;
— d'identifier les facteurs dominants (pour orienter des actions métiers) ;
— de transformer une prédiction en **aide à la décision** (rappel ciblé, replanification, priorisation).

5.2. Importance globale (SHAP summary)

L'analyse globale SHAP met en évidence les variables les plus influentes dans l'entraînement du modèle. Dans les résultats, les variables temporelles liées à la durée et aux horaires jouent un rôle majeur, indiquant que l'organisation du planning est un déterminant fort du statut final.

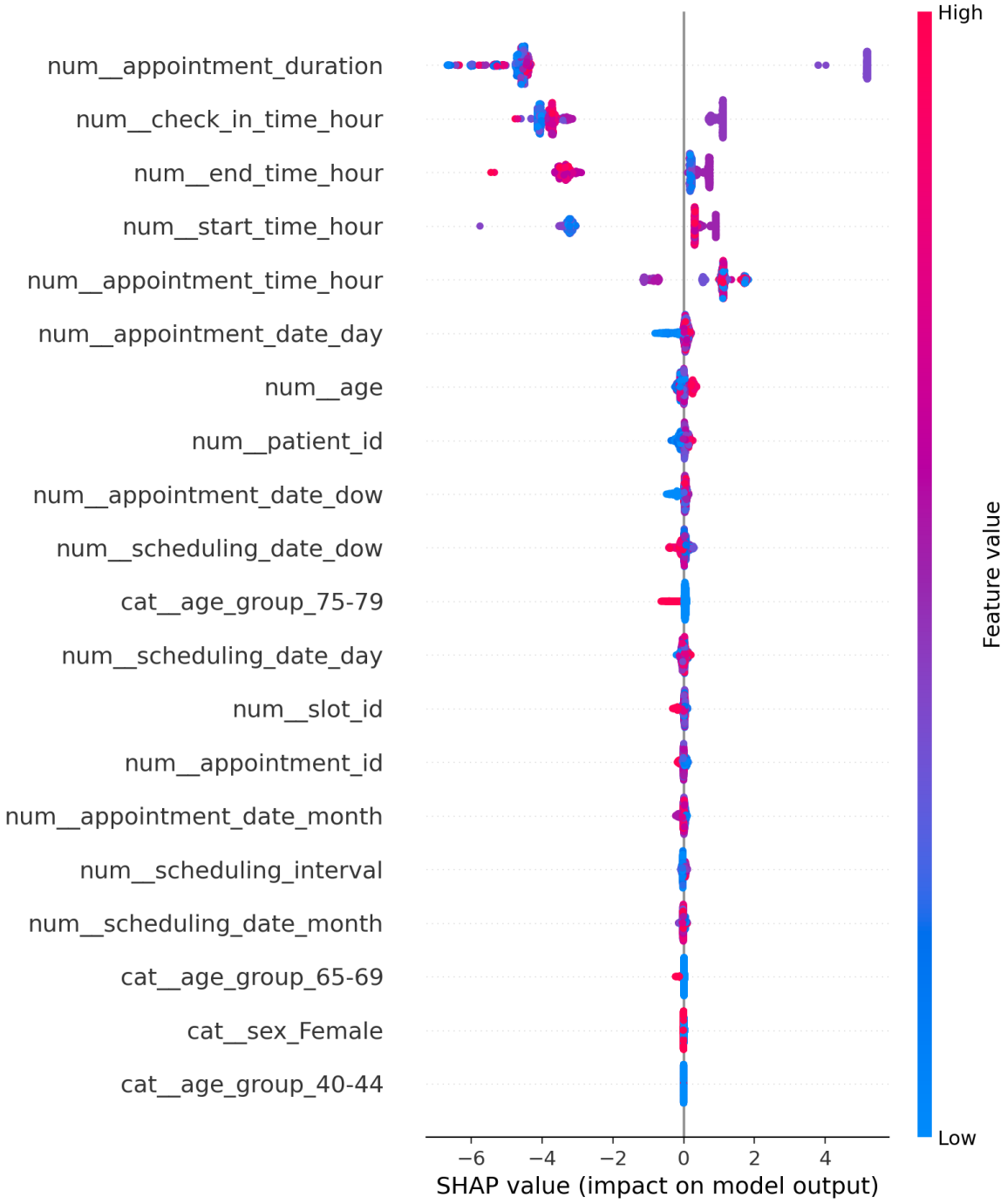

FIGURE 4 – SHAP summary plot — importance globale des variables.

5.3. Importance locale (SHAP waterfall : explication d'un rendez-vous)

Le graphique SHAP *waterfall* explique une prédiction individuelle.

La valeur moyenne du modèle est $E[f(X)] = -1.426$ (sortie moyenne en log-odds). La sortie finale pour ce rendez-vous est $f(x) = -9.948$, indiquant une très faible probabilité de no-show.

Les barres bleues diminuent la probabilité de no-show, les barres rouges l'augmentent.

— `num__appointment_duration = 17.4` : impact ≈ -5.34. Les rendez-vous longs sont nettement

plus respectés.

— `num__end_time_hour = 14` : impact ≈ −3.65. Les RDV se terminant vers 14h sont globalement mieux honorés.

— `num__appointment_time_hour = 12` : impact ≈ −1.01. Les RDV programmés à midi sont plus stables.

— `num__check_in_time_hour = 12` : impact ≈ +0.77. Arrivée tardive → légère augmentation du risque.

— `num__start_time_hour = 14` : impact ≈ +0.46. Début à 14h → légèrement plus risqué.

— Les autres variables (id, jour, âge, etc.) ont un impact faible dans cet exemple.

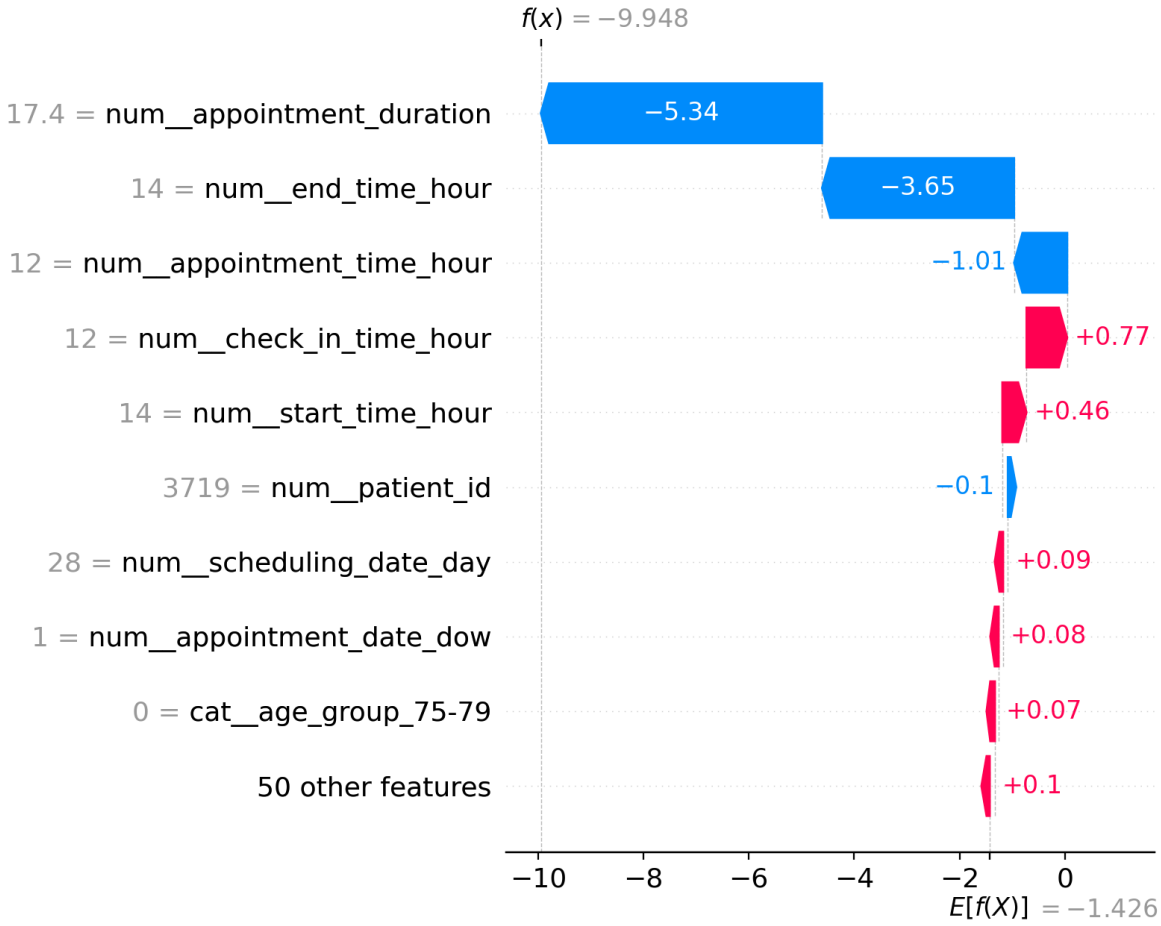

FIGURE 5 – SHAP waterfall — explication locale d'une prédiction individuelle.

Conclusion locale. Pour ce rendez-vous, le modèle prédit une très faible probabilité de no-show, principalement grâce à une durée longue et un créneau stable autour de midi / début d'après-midi. Cette explicabilité permet de justifier la prédiction et d'envisager des actions ciblées sur les cas réellement à risque.

6. Limites et améliorations possibles

6.1. Limites

— **Données synthétiques :** la généralisation à un contexte réel doit être validée.
— **Risque de fuite d'information :** certaines variables de déroulement (ex. heure réelle de début/fin, durée) peuvent ne pas être disponibles au moment où l'on souhaite prédire (ex. lors de la planification).
— **Validation :** une validation temporelle (entraîner sur le passé, tester sur une période future) renforcerait la crédibilité des résultats.

6.2. Améliorations

— Restreindre les features aux variables disponibles avant le rendez-vous pour un usage réel.
— Ajuster le seuil de décision en fonction d'un coût métier (FN plus coûteux que FP).
— Ajouter un monitoring (dérive des données, recalibrage) et un suivi de performance en continu.
— Étendre l'explicabilité (analyse par tranche horaire, profils patients) pour guider des actions ciblées.