# Défi IA 2024/2025 : Prédiction de réachats sur la base d'un historique

BALLU Samuel, FRANCOIS Gautier, LE BRETON Paul

January 9, 2025

# Contents

# 1 Introduction

## 1.1 Contexte du projet

Dans un contexte où la personnalisation et l'efficacité des services numériques deviennent des facteurs essentiels de compétitivité, le Défi IA 2024-2025 constitue une opportunité unique d'explorer l'impact de l'intelligence artificielle dans le domaine du e-commerce alimentaire. Organisé par Carrefour en collaboration avec l'Université de Bordeaux, ce défi s'inscrit dans le cadre du plan stratégique Carrefour 2027 visant à transformer l'entreprise en leader de la data-driven economy.

Les courses alimentaires, en raison de leur fréquence et de leur récurrence, offrent un terrain fertile pour le développement d'algorithmes prédictifs. Cependant, la diversité des comportements d'achat, couplée à la complexité des catalogues produits, pose des défis analytiques uniques.

Ce projet réalisé pendant notre seconde année de Master MAS parcours Science des Données, Intelligence Artificielle nous ambitionne car nous devons optimiser la pertinence des recommandations affichées sur le carrousel d'achats fréquents de Carrefour.fr, un levier stratégique pour fidéliser les clients tout en augmentant les ventes.

## 1.2 Objectif du projet

À travers ce projet, l'objectif est de développer un système de recommandation avancé capable de prédire les produits les plus susceptibles d'être rachetés par un client, améliorant ainsi la satisfaction client tout en maximisant les performances commerciales.

Cette démarche vise à **améliorer la satisfaction et la fidélité** des clients en leur proposant des recommandations adaptées à leurs habitudes d'achats. Elle permet aussi de **maximiser les performances commerciales** grâce à une meilleure anticipation des comportements d'achats.

Ce rapport se concentre sur l'élaboration, la mise en œuvre et l'évaluation d'un modèle de machine learning capable de prédire les 10 produits les plus probables d'être rachetés en priorité pour un ensemble de clients inconnus, sur la base de leur historique d'achat et des informations produits disponibles.

## 1.3 Plan

Ce rapport est structuré en plusieurs sections pour détailler les différents étapes du projet :

- **Analyse exploratoire des données (Section 2)** : Cette section se concentre sur l'exploration des données brutes, leur structure et leurs principales caractéristiques, complétées par des visualisations pour une meilleure compréhension des tendances et des particularités.
- **Méthodologie (Section 3)** : Nous détaillons ici la démarche méthodologique adoptée incluant :
  - Le prétraitement des données, pour assurer leur qualité et leur pertinence.
  - Le choix des modèles et des techniques de machine learning employés pour répondre à la problématique.
  - La définition des métriques d'évaluations, essentielles pour juger la performance des modèles.
- **Résultats (Section 4)** : Cette partie présente les résultats obtenus, notamment :
  - Les performances du modèle développé selon les métriques choisies.
  - Une comparaison avec d'autres approches envisagées.
  - Une analyse critique des limites observées et des points d'amélioration potentiels.
- **Conclusion et perspective (Section 5)** : Cette section offre une synthèse globale du projet en récapitulant les principales étapes, les résultats obtenus, et en mettant en avant les enseignements tirés de l'étude.

# 2 Analyse exploratoire des données

## 2.1 Exploration des données brutes

L'objectif principal est de concevoir un système de recommandation performant pour prédire les produits les plus susceptibles d'être achetés par des clients, en utilisant trois jeux de données distincts :

- Les informations sur les produits.
- L'historique des transactions de clients sur les années précédentes.
- L'historique partiel de nouveaux clients pour lesquels il faut prédire la première transaction.

### 2.1.1 Description générale des données

Les fichiers de données utilisés dans cette analyse sont les suivants :

- **products_data.csv** : Contient des informations sur l'ensemble des produits proposés par l'entreprise. Les détails incluent des catégories de produit, des descriptions, etc.
- **train_data.csv** : Ensemble de données d'apprentissage contenant l'historique des transactions de 100 000 clients entre 2022 et 2023. Ce fichier est divisé en 10 parties, chacune représentant 10 000 clients (e.g., `train_data.csv_part_1` pour les clients 1 à 10 000).
- **test_data.csv** : Ensemble de données de test contenant l'historique des transactions de 80 000 clients en 2024. Les transactions des 20 000 clients suivants (clients 80 001 à 100 000) sont masquées et doivent être prédites.

### 2.1.2 Description du jeu de données `products_data.csv`

- Le jeu de données comprend environ 82 966 produits alimentaires, décrits à travers 49 colonnes(8.1).
- Chaque produit possède un ID unique et une description détaillée.
- Les produits sont organisés selon une hiérarchie structurée : 36 départements principaux, 1003 classes et 3452 sous-classes.
- Ils sont caractérisés par différents attributs, incluant notamment les labels bio, local/français, végétarien, etc.

Ce graphique montre la "Répartition des produits par niveau de rayon 1", qui semble représenter la distribution des produits. On observe une forte disparité entre les catégories les plus importantes et les plus petites. Les trois premières catégories comptent chacune plus de 6000 produits, tandis que les dernières catégories comme "E-cartes et coffrets cadeaux" ou "Repas livrés" comptent moins de 500 produits.

La distribution suit une courbe décroissante assez régulière, avec une forte concentration sur les produits de première nécessité et d'usage quotidien (épicerie, hygiène, boissons), puis une décroissance progressive vers des catégories plus spécialisées ou moins courantes. Les catégories du milieu de graphique (entre 1000 et 3000 produits) concernent principalement l'équipement de la maison, les produits pour bébé, et le jardin.

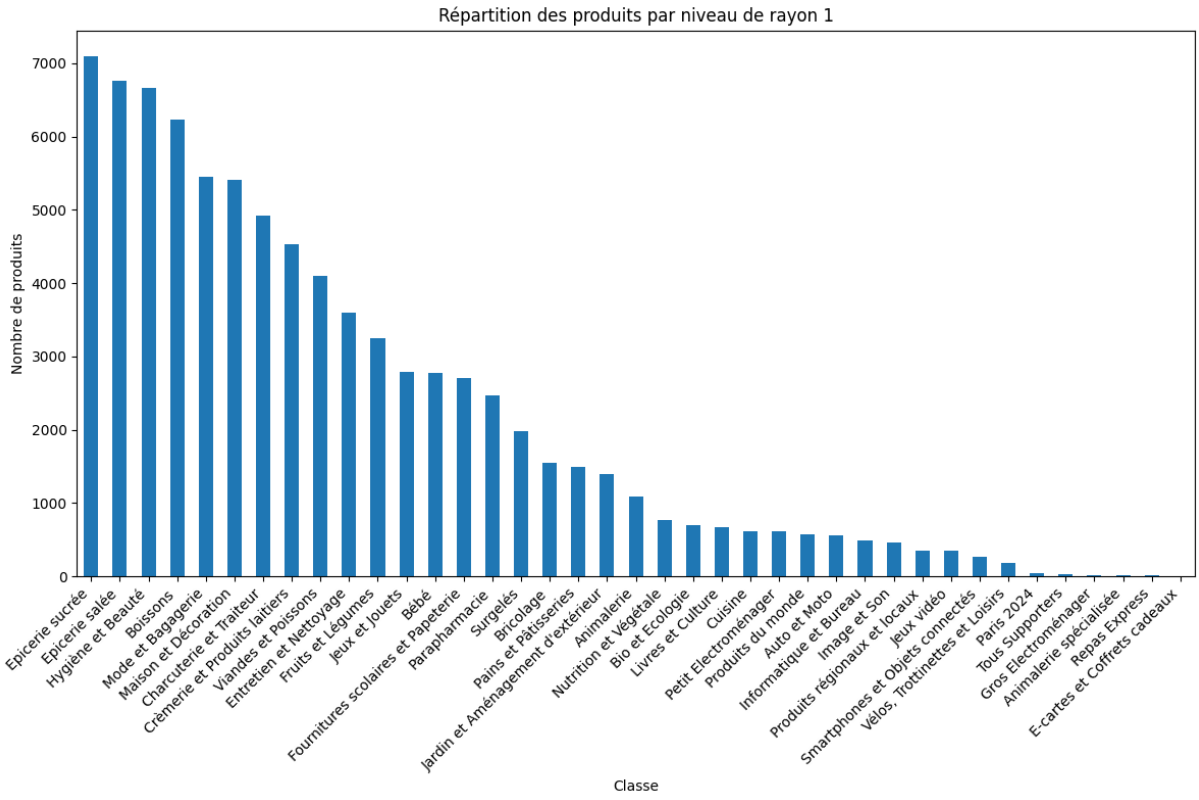


Figure 1: Répartition des produits par niveau de rayon 1

# 3 Méthodologie

Pour répondre à la problématique, nous avons opté pour un modèle de Machine Learning basé sur un réseau de neurones récurrent, plus précisément un modèle **LSTM (Long Short-Term Memory)**. Les modèles LSTM sont particulièrement adaptés aux données séquentielles, ce qui en fait une solution idéale pour la prédiction des achats futurs des clients, en tenant compte de leurs comportements passés.

Afin d'optimiser les performances et de réduire la complexité du modèle, nous avons choisi de travailler sur les classes de produits (colonne `class_key`), plutôt que sur les produits eux-mêmes. Les classes de produits sont définies par la variable `class_key` du jeu descriptif des données et permettent de réduire l'espace de recherche tout en conservant la richesse des informations comportementales.

Le modèle a été conçu pour être entraîné sur des séquences d'achats historiques de chaque client, prenant en compte l'ordre et la fréquence des achats pour identifier des patterns récurrents. En apprenant à partir de ces séquences, le modèle est capable de prédire, pour chaque client, les classes de produits qu'il pourrait être susceptible d'acheter dans un futur proche. Cette approche permet de fournir des recommandations personnalisées et pertinentes, adaptées au profil d'achat spécifique de chaque utilisateur.

# 4 Prétraitement des données

Le prétraitement des données constitue une étape essentielle dans la construction de notre modèle de prédiction des achats futurs. Nous avons appliqué plusieurs techniques de préparation pour garantir que les données soient dans un format optimal avant de les introduire dans le modèle LSTM. Ces étapes permettent de rendre les données compatibles avec les exigences de l'apprentissage profond et de maximiser les performances du modèle.

## 4.1 Etape 1 - Ordonnancement

Les données ont d'abord été triées par `customer_id` et `date`. Cette étape est cruciale pour assurer la bonne chronologie des transactions de chaque utilisateur, ce qui permet au modèle de prendre en compte l'ordre des achats dans le temps.

## 4.2 Etape 2 - Encodage

Les labels présents dans la colonne `class_key`, représentant les catégories de produits, ont été encodés en valeurs numériques à l'aide d'un *LabelEncoder*. Cette étape est nécessaire car les modèles d'apprentissage profond, tels que les réseaux LSTM, ne peuvent traiter directement que des valeurs numériques. L'encodage permet ainsi de transformer les catégories en un format adapté à l'entrée du modèle.

## 4.3 Etape 3 - Création des séquences

Nous avons créé des séquences de produits achetés pour chaque utilisateur, en groupant les transactions par client. Chaque séquence contient les achats effectués dans un ordre temporel donné, avec une longueur fixe de **10** éléments, ce qui permet de prédire l'achat suivant. Les séquences sont ensuite découpées en entrées (séquences de produits) et sorties (le produit suivant à prédire). Cette étape est cruciale car elle transforme les données brutes en une forme temporelle exploitable pour le modèle LSTM. 8.2

## 4.4 Etape 4 - Préparation des produits les plus achetés par client par classe

Après l'entraînement du modèle, l'objectif va être de faire le lien entre les classes prédites et les produits à prédire qui représentent l'objet du problème. Pour cela, nous avons créé un jeu de données à partir de l'historique des achats qui identifie, pour chaque client, le produit qu'il a le plus acheté dans chaque classe de produit.

# 5 Approche de modélisation

## 5.1 Théorie du modèle choisi

Pour répondre à la problématique, nous avons donc décidé d'utiliser un LSTM.
Les LSTM sont une variante des réseaux de neurones récurrents (RNN) qui sont capables de :

- Capturer des relations à long terme dans des séquences de données.
- Résoudre les problèmes des RNN classiques, notamment le *vanishing gradient* et l'*exploding gradient* qui rendent l'apprentissage des dépendances à long terme difficile.

Les LSTM sont utilisés pour traiter des données séquentielles comme des séries temporelles, des textes ou des vidéos, en apprenant à capturer les dépendances temporelles entre les éléments d'une séquence.

### 5.1.1 Architecture

L'architecture d'un LSTM repose sur 3 portes principales : la **porte d'entrée**, la **porte d'oubli** et la **porte de sortie**. L'objectif est de réguler l'information dans l'état interne du réseau, d'éviter la disparition du gradient et d'ajuster l'importance de chaque information au fil du temps.
1. La porte d'entrée (Input Gate)

$$i_t = \phi(W_i \times [h_{t-1}, x_t] + b_i)$$

2. La porte d'oubli (Forget Gate)

$$f_t = \phi(W_f \times [h_{t-1}, x_t] + b_f)$$

3. La porte de sortie (Output Gate)

$$o_t = \phi(W_o \times [h_{t-1}, x_t] + b_o)$$

L'**état cellulaire** est un vecteur qui sert à stocker les informations à long terme. Il est mis à jour à chaque étape de temps en fonction de la porte d'oubli et de la porte d'entrée :

$$C_t = f_t * C_{t-1} + i_t \times \tanh(W_c \times [h_{t-1}, x_t] + b_c)$$

L'**état caché** est la sortie finale de la cellule LSTM et est utilisé pour prédire l'élément suivant de la séquence :

$$h_t = o_t \times \tanh(C_t)$$

### 5.1.2 Fonctionnement

L'idée principale derrière un LSTM est de permettre au réseau de se souvenir de certaines informations pendant une période prolongée, tout en oubliant des informations inutiles. En conséquence, ces réseaux sont capables de capturer des dépendances temporelles complexes même sur de longues séquences.

### 5.1.3 Pratique

Dans cette partie, nous allons voir comment nous avons utilisé un modèle LSTM pour répondre à la problématique et mettre en place un système de recommandation basé sur les historiques d'achats des classes de produits.
Pour ce faire, nous avons utilisé la bibliothèque `Keras`, une bibliothèque permettant de construire des modèles d'apprentissage profond.

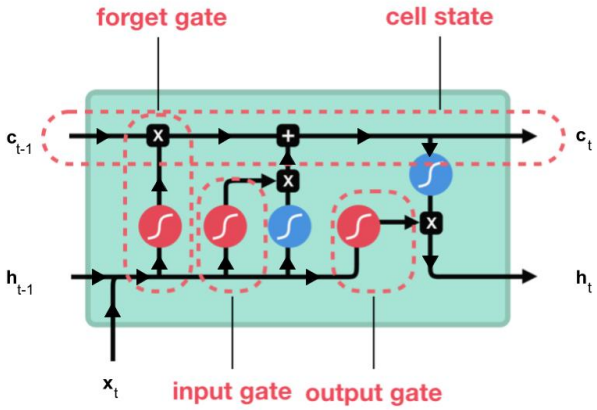


Figure 2: Schéma de l'architecture LSTM

#### Construction du modèle

1. **Embedding Layer** : elle permet de transformer les indices des classes en vecteurs denses de taille fixe (ici 150) dans l'objectif d'aider le modèle à apprendre une représentation plus dense et plus riche des classes.
2. **LSTM Layers** : deux couches LSTM de 128 unités sont utilisées dans le modèle :
   - La première retourne une séquence complète pour chaque élément de la séquence d'entrée.
   - La seconde retourne seulement l'état final de la séquence.
3. **Dropout** : des couches Dropout sont utilisées pour éviter le sur-apprentissage en supprimant aléatoirement une fraction des neurones pendant l'entraînement.
4. **Dense Layers** : après les couches LSTM, des couches denses sont utilisées :
   - La première a 64 unités et utilise une activation `relu`.
   - La dernière est une couche de sortie avec une fonction d'activation `softmax`, produisant les probabilités pour chaque classe de produit.

#### Compilation et entraînement

Le modèle est compilé en utilisant l'optimiseur `Adam` et la fonction de perte `sparse_categorical_crossentropy`, adaptée à un problème de classification multi-classe. Le modèle est ensuite entraîné pendant 5 époques, avec une taille de lot de 1024, en utilisant une validation croisée sur 20% des données.

### 5.2 Prédiction et sélection des produits recommandés

Une fois le modèle entraîné, il est utilisé pour prédire les classes de produits les plus pertinentes pour chaque client.
Les 10 classes de produits les plus recommandées sont sélectionnées en fonction des probabilités prédites par le modèle. Ce processus garantit que chaque client bénéficie d'une sélection diversifiée, avec 10 classes distinctes afin d'éviter les doublons parmi les produits recommandés.
Enfin, les 10 produits à recommander à chaque client sont déterminés en sélectionnant ceux qui ont été les plus fréquemment achetés dans les classes prédictives correspondantes.

### 5.3 Métrique d'évaluation

L'objectif de ce critère est d'évaluer la pertinence des recommandations en mesurant le pourcentage de produits recommandés qui ont effectivement été achetés par le client. Le score imposé par la compétition

est **HitRate@10**. Il permet de quantifier la précision en vérifiant combien de produits parmi les 10 recommandés ont été achetés par le client.
La formule utilisée pour calculer le score HitRate@10 est la suivante, avec l'introduction de deux vecteurs :

- $(a_i)_{1 \leq i \leq N}$ représentant les produits achetés par un client lors d'une commande, où $N$ est le nombre total de produits achetés (avec $N \in \mathbb{N}^*$),
- $(y_i)_{1 \leq i \leq 10}$ représentant les 10 produits prédits par le modèle.

Il convient de noter que $N$ n'est pas nécessairement égal à 10. La formule est alors la suivante :

$$HitRate@K(a, y) = \frac{1}{\min(N, 10)} \sum_{i=1}^{10} \mathbb{1}_{y_i \in \{a_1, \ldots, a_N\}}$$

Quelques précisions importantes :

- Le terme $\min(N, 10)$ est utilisé pour normaliser le score, garantissant ainsi un score parfait de 1 si un client achète moins de 10 produits et si tous ces produits sont correctement prédits.
- Cette définition suppose que tous les produits $y_1, \ldots, y_{10}$ sont distincts. En cas de doublon dans les recommandations, cela pourrait fausser le score. Par exemple, si le même produit apparaît 10 fois parmi les recommandations et que ce produit est effectivement acheté, le score serait alors de 1. Afin d'éviter ce problème, il est demandé que les 10 produits recommandés soient distincts.
- Concernant le score privé et public sur Kaggle, il est à noter que le classement des produits n'est pas pris en compte. Toutefois, après la compétition, il sera possible d'utiliser le score HitRate@K pour des valeurs de $K < 10$ afin de départager des groupes ayant des scores proches. Il est donc demandé d'attribuer un rang unique entre 1 et 10 à chaque produit, ce qui correspond à une permutation de l'ensemble $\{1, \ldots, 10\}$.

# 6 Résultats

## 6.1 Performance du modèle choisi et évaluation

Voici l'architecture de notre modèle :

| Layer (type) | Output Shape | Param # |
|---|---|---|
| embedding_1 | (None, 10, 150) | 98,250 |
| lstm_2 | (None, 10, 128) | 142,848 |
| dropout_2 | (None, 10, 128) | 0 |
| lstm_3 | (None, 128) | 131,584 |
| dropout_3 | (None, 128) | 0 |
| dense_2 | (None, 64) | 8,256 |
| dense_3 | (None, 655) | 42,575 |

Table 1: Architecture du modèle

Le modèle présente des performances limitées avec une précision d'environ 20 %, ce qui suggère qu'il parvient à prédire correctement seulement un cinquième des cas. L'entraînement prend environ 5 heures, ce qui reflète une complexité computationnelle non négligeable, probablement due à la taille du modèle et au nombre élevé de paramètres (plus de 420 000).

Cette faible précision peut indiquer une inadéquation entre la complexité du modèle et la nature des données, un surapprentissage ou un sous-apprentissage, ou encore un problème lié à la qualité ou à la quantité des données d'entraînement. Des améliorations au niveau des données, du fine-tuning des hyperparamètres, ou l'exploration de modèles alternatifs pourraient permettre d'améliorer ces résultats.

Pour que le modèle atteigne de meilleures performances, nous pouvons réaliser une segmentation client et appliquer notre réseau à chaque catégorie. Ainsi, nous gagnerons en temps mais aussi en performance, car le modèle serait mieux entraîné en fonction du type de client. Pour réduire le temps de calcul, nous pouvons aussi entraîner notre modèle sur une sous-population.

## 6.2 Segmentation RFM

Étant donné le nombre d'individus conséquent, nous pensons tout d'abord à essayer de les classifier. Pour ce faire, nous nous sommes dit qu'il pouvait exister des profils types de consommateurs. Effectivement, certaines personnes préfèrent faire les courses une seule fois par semaine en achetant beaucoup de produits d'un coup. Or, il se peut que certaines personnes viennent de façon plus récurrente en achetant des produits en quantité plus faible, ou encore que certaines aient fait leurs courses chez Carrefour de façon exceptionnelle.

Pour essayer de classifier les individus, nous avons donc commencé par réaliser une segmentation RFM. La segmentation RFM (Récence, Fréquence, Montant) est une méthode analytique utilisée en marketing pour classer les clients en fonction de leur comportement d'achat. Elle repose sur trois dimensions principales :

- **Récence (R)** : Temps écoulé depuis le dernier achat d'un client.
- **Fréquence (F)** : Nombre total de transactions effectuées par un client sur une période donnée.
- **Montant (M)** : Valeur monétaire totale dépensée par un client (ou valeur moyenne par transaction) (pour nous, ce sera le nombre d'articles achetés par commande).

Ainsi, nous pouvons identifier les clients prioritaires, personnaliser les recommandations, réduire le bruit des données et les intégrer dans le système de recommandation. Au final, nous obtenons donc 5 classes différentes représentant les différents profils de consommateurs.
Cela nous a permis de mieux comprendre les profils des consommateurs et d'améliorer les recommandations.

## 6.3 Augmentation de la séquence pour le modèle LSTM

Dans le cadre de l'amélioration des performances de notre modèle LSTM, nous avons expérimenté avec une augmentation de la longueur des séquences d'entrée. Initialement, les séquences utilisées pour l'entraînement et la prédiction avaient une longueur de 10, ce qui correspondait à l'utilisation des 10 transactions les plus récentes pour chaque client. Afin de capturer davantage de contexte dans les comportements d'achat, nous avons étendu cette longueur à 30, intégrant ainsi les 30 dernières transactions dans chaque séquence.

Cette augmentation de la séquence a permis au modèle de bénéficier d'une vision plus large des habitudes d'achat des clients. En incorporant davantage de données temporelles, le modèle est mieux équipé pour identifier des schémas complexes et des relations à long terme entre les produits achetés. Cependant, cette modification a également entraîné une augmentation significative de la taille des données d'entrée, ce qui a nécessité des ajustements dans la gestion de la mémoire et le temps d'entraînement.

Bien que les résultats finaux de cette approche n'aient pas encore été obtenus, les premières évaluations montrent une amélioration notable de l'accuracy (0,052) par rapport au modèle basé sur des séquences de longueur 10 (0,047). Cette augmentation suggère que l'utilisation de séquences plus longues permet au modèle de mieux généraliser et de produire des prédictions plus précises.

## 6.4 Comparaison d'autres approches utilisées

### 6.4.1 Random Forest

Afin d'évaluer la pertinence de notre modèle basé sur LSTM, nous avons exploré une approche alternative en utilisant un algorithme classique de machine learning, le Random Forest. Ce choix s'est porté sur sa simplicité d'utilisation et sa capacité à gérer des données tabulaires complexes. L'objectif était de comparer ses performances à celles du modèle LSTM dans le cadre de la prédiction des classes de produits les plus susceptibles d'être achetées par un client.

Dans cette approche, les données ont été prétraitées pour s'adapter au format attendu par le modèle Random Forest. Contrairement à l'utilisation des séquences temporelles dans le LSTM, les données d'entrée ont été agrégées pour représenter des caractéristiques statiques. Ces caractéristiques incluaient la fréquence d'achat des produits par classe et d'autres attributs descriptifs des produits, tels que leur prix moyen ou leur popularité globale. Ainsi, chaque observation dans l'ensemble de données représentait un client avec des informations agrégées sur ses comportements d'achat, sans prendre en compte l'ordre chronologique des transactions.

Le modèle Random Forest a ensuite été entraîné sur ces données, avec pour objectif de prédire les 10 classes de produits les plus susceptibles d'être achetées par chaque client. Bien que cette méthode soit efficace pour capturer des relations non linéaires entre les caractéristiques, elle s'est avérée limitée dans ce contexte. Les résultats obtenus n'ont pas été convaincants, avec des performances significativement inférieures à celles du modèle LSTM. Cela peut être attribué à l'incapacité du Random Forest à exploiter les dépendances temporelles et séquentielles présentes dans les données, qui sont essentielles pour comprendre les comportements d'achat des clients.

En comparaison, le modèle LSTM, en intégrant directement les séquences temporelles, a démontré une capacité supérieure à identifier les schémas complexes dans les données et à fournir des recommandations pertinentes. Cette expérience a permis de confirmer que, bien que les modèles comme Random Forest soient adaptés à des données tabulaires statiques, ils ne conviennent pas aux problématiques nécessitant une prise en compte explicite des dynamiques temporelles.

### 6.4.2 Utilisation de la sous classe de produits

Dans cette section, nous avons exploré l'impact de l'utilisation de la sous-classe des produits comme caractéristique dans notre modèle. L'objectif était d'évaluer si une granularité plus fine dans la catégorisation

des produits pouvait améliorer les performances du modèle.

Pour intégrer cette information, la sous-classe des produits a été encodée et ajoutée en tant qu'entrée dans notre pipeline de traitement. Cependant, les résultats obtenus n'ont pas répondu aux attentes. Le modèle a atteint un taux d'hitrate de seulement 0,15, ce qui est inférieur aux résultats obtenus avec d'autres configurations. Cela suggère que l'ajout de cette caractéristique n'a pas permis d'améliorer la capacité du modèle à prédire efficacement les séquences cibles.

Dans cette expérimentation, une séquence fixe de 10 a été utilisée pour l'ensemble des modèles, y compris lors de l'intégration de la sous-classe de produits comme caractéristique supplémentaire. Cette approche, bien que cohérente, peut avoir limité la capacité du modèle à capturer des dépendances à plus long terme entre les produits. L'utilisation d'une séquence plus longue, comme 20 ou 30, pourrait potentiellement améliorer les performances en offrant une meilleure contextualisation des relations temporelles ou des interactions entre les produits.

Cela permettrait également de mieux exploiter les informations supplémentaires fournies par la sous-classe de produits. Cependant, il est important de noter qu'une augmentation de la longueur des séquences entraînerait une complexité accrue du modèle, augmentant ainsi le temps d'entraînement et le risque de surapprentissage, en particulier si les données disponibles sont limitées. Par conséquent, l'évaluation de séquences plus longues pourrait constituer une piste intéressante pour les travaux futurs, tout en nécessitant une attention particulière à l'équilibre entre performance et complexité.

# 7 Conclusion et perspectives

Dans cette étude, nous avons exploré l'utilisation de différents modèles d'apprentissage automatique, notamment les modèles LSTM, Random Forest, et la prise en compte des sous-classes de produits, pour prédire des comportements ou des tendances à partir de données. Bien que certaines approches, comme l'intégration de la sous-classe de produits, n'aient pas donné de résultats significatifs, nous avons observé que l'augmentation de la longueur des séquences, de 10 à 30, a permis d'améliorer la précision du modèle LSTM, avec une meilleure capacité à capturer des relations à plus long terme entre les produits. Ces résultats suggèrent qu'une plus grande profondeur temporelle pourrait potentiellement améliorer les performances de prédiction, bien que des ajustements supplémentaires soient nécessaires pour éviter le surapprentissage.

Les résultats obtenus montrent également que les modèles basés sur des arbres décisionnels comme Random Forest ne sont pas aussi performants dans ce contexte spécifique, notamment en raison de la complexité des relations temporelles entre les données, que les réseaux de neurones récurrents comme les LSTM peuvent mieux modéliser. Cependant, ces derniers présentent un coût computationnel plus élevé et nécessitent une plus grande quantité de données pour atteindre des performances optimales.

En termes de perspectives, plusieurs axes d'amélioration peuvent être envisagés. D'abord, l'augmentation de la longueur des séquences pourrait être poursuivie, en testant des séquences encore plus longues ou en ajustant les paramètres du modèle pour optimiser les performances. De plus, l'intégration d'autres types de données ou de nouvelles caractéristiques, comme des informations supplémentaires sur les comportements des utilisateurs ou des interactions entre produits, pourrait enrichir les modèles et améliorer leur capacité de prédiction.

En somme, bien que des résultats intéressants aient été obtenus, il reste encore plusieurs pistes à explorer pour affiner les modèles et accroître leur performance dans des contextes réels. Les travaux futurs devraient également porter sur l'évaluation de l'impact des différentes configurations de données et de modèles sur la capacité de prédiction, afin de mieux comprendre les facteurs qui influencent les résultats obtenus.

# 8 Annexe

## 8.1 Annexe 1 : Tableau descriptif du jeu de données 'products data'

Le tableau ci-dessous présente un résumé des principales caractéristiques du dataframe utilisé dans cette analyse. Le dataframe contient un total de 82 966 lignes et 49 colonnes.

| Résumé | Valeurs |
|---|---|
| Nombre de lignes | 82 966 |
| Nombre de colonnes | 49 |

### Types de Données

Les types de données des colonnes sont répartis comme suit :

| Type de Donnée | Nombre de Colonnes |
|---|---|
| booléen | 38 |
| chaîne de caractères | 1 |
| catégoriel | 10 |

### Variables Catégorielles

Les variables catégorielles dans le dataframe sont les suivantes, avec le nombre de valeurs uniques pour chaque colonne :

| Variable | Nombre de Valeurs Uniques |
|---|---|
| product_id | 82 815 |
| department_key | 36 |
| class_key | 1 003 |
| subclass_key | 3 452 |
| sector | 6 |
| brand_key | 4 852 |
| shelf_level1 | 39 |
| shelf_level2 | 285 |
| shelf_level3 | 1 176 |
| shelf_level4 | 1 022 |

### Variables Booléennes

Les variables booléennes et leur taux de valeurs 'True' sont les suivantes :

| Variable | Valeur True | Taux |
|---|---|---|
| bio | 6 726 | 0.081 |
| sugar_free | 243 | 0.0029 |
| aspartame_free | 18 | 0.00022 |
| gluten_free | 1 427 | 0.017 |
| halal | 819 | 0.0099 |
| casher | 256 | 0.0031 |
| eco_friendly | 62 | 0.00075 |
| local_french | 7 287 | 0.088 |
| artificial_coloring_free | 892 | 0.011 |
| taste_enhancer_free | 964 | 0.012 |
| naturality | 0 | 0 |
| antibiotic_free | 278 | 0.0034 |
| reduced_sugar | 1 177 | 0.014 |
| vegetarian | 6 820 | 0.082 |

| Variable | Valeur True | Taux |
|---|---|---|
| pesticide_free | 5 215 | 0.063 |
| grain_free | 54 | 0.00065 |
| no_added_sugar | 632 | 0.0076 |
| salt_reduced | 0 | 0 |
| nitrite_free | 1 556 | 0.019 |
| fed_without_ogm | 705 | 0.0085 |
| no_added_salt | 0 | 0 |
| no_artificial_flavours | 41 | 0.00049 |
| porc | 4 088 | 0.049 |
| vegan | 246 | 0.003 |
| frozen | 1 999 | 0.024 |
| fat_free | 419 | 0.0051 |
| reduced_fats | 49 | 0.00059 |
| fresh | 1 178 | 0.014 |
| alcool | 3 568 | 0.043 |
| lactose_free | 0 | 0 |
| phenylalanine_free | 415 | 0.005 |
| palm_oil_free | 6 910 | 0.083 |
| ecoscore | 82 966 | 1 |
| produits_du_monde | 1 547 | 0.019 |
| regional_product | 4 557 | 0.055 |
| national_brand | 59 119 | 0.71 |
| first_price_brand | 4 230 | 0.051 |
| carrefour_brand | 19 554 | 0.24 |

#### Variables de Type Chaîne de Caractères

Pour les variables de type chaîne de caractères, les statistiques suivantes ont été observées :

| Variable | NA | NA % | Plus Court | Plus Long |
|---|---|---|---|---|
| product_description | 0 | 0 | ABC | VP 75 CL |

## 8.2 Annexe 2 : Code pour création de la séquence

Listing 1: Fonction pour créer des séquences de longueur fixe

```
# Fonction pour creer des sequences de longueur fixe (10) et leurs cibles
def create_sequences(data, sequence_length=10):
    sequences = []
    targets = []
    for seq in data:
        for i in range(len(seq) - sequence_length):
            sequences.append(seq[i:i+sequence_length])
            targets.append(seq[i+sequence_length])
    return np.array(sequences), np.array(targets)
```

# 9 Bibliographie

Christopher Olah, *Understanding LSTMs*, https://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015. Consulté le 7 janvier 2025.

Wikipedia, *Long short-term memory*, https://fr.wikipedia.org/wiki/R%C3%A9seau_de_neurones_r%C3%A9currents#Long_short-term_memory, consulté le 3 janvier 2025.

OpenAI, *ChatGPT (version 4)*, https://chat.openai.com

Anthropic, *Claude (version 1)*, https://claude.ai.