2.4.1 Introduction

Cette partie est consacrée à l'implémentation de notre système, nous allons faire une étude comparative entre les trois méthodes de filtrage : basé sur le contenu, collaboratif et hybride. Avant de déterminer laquelle est la meilleure,

2.4.2 Objectifs de la mise en œuvre

L'objectif principal de cette mise en œuvre est de concevoir et développer un système de recommandation performant en combinant les approches de filtrage collaboratif et de filtrage basé sur le contenu. En intégrant ces deux techniques complémentaires dans une approche hybride, le système vise à surmonter les limitations propres à chaque méthode prise isolément, notamment le problème du démarrage à froid, la rareté des données ou encore la personnalisation limitée.

Plus spécifiquement, les objectifs opérationnels de la mise en œuvre sont les suivants :

Collecte et préparation des données : Extraire, nettoyer et structurer les données nécessaires, incluant à la fois les caractéristiques des utilisateurs (profils, historique d'interaction) et les attributs des items (contenus pédagogiques, catégories, descripteurs).

Implémentation du filtrage basé sur le contenu : Mettre en place un modèle capable de recommander des éléments similaires aux préférences explicites et implicites des utilisateurs, en s'appuyant sur les descripteurs des contenus.

Implémentation du filtrage collaboratif : Construire un système exploitant les interactions et les évaluations des utilisateurs pour identifier des schémas de comportements et générer des recommandations à partir de la similarité entre utilisateurs ou entre items.

Conception de l'approche hybride : Intégrer les deux approches précédentes en définissant une stratégie de combinaison (par exemple, pondération des scores ou fusion des recommandations), afin d'en tirer parti de manière optimale.

Évaluation des performances du système : Mettre en œuvre des métriques d'évaluation standards (précision, rappel, F1-score, NDCG), ainsi que des indicateurs spécifiques au domaine e-Learning (taux de complétion, progression académique, réutilisation, satisfaction subjective) pour valider l'efficacité du système développé.

Cette démarche vise à offrir une expérience de recommandation plus personnalisée, plus pertinente et mieux adaptée à l'évolution des besoins des utilisateurs.

2.4.3 Environnement de développement

2.4.3.1 Google colab

(Google Collaboratory) est un environnement de développement en

Ligne gratuit proposé par Google, qui permet d'écrire et d'exécuter du code Python dans un notebook Jupiters, avec accès à des ressources matérielles comme les GPU et TPU.

Il est très utilisé dans les domaines de la data science, du machine Learning et de l'enseignement, car il ne nécessite aucune installation locale. [55]

2.4.3.2 Python

Python est un langage de programmation interprété, haut niveau et polyvalent, connu pour sa syntaxe claire et sa facilité d'apprentissage. Il est largement utilisé en développement web, science des données, automatisation, intelligence artificielle, et bien plus encore. [56]

Portabilité : Python fonctionne sur une grande variété de systèmes d'exploitation, aussi bien open source (comme les différentes variantes d'UNIX) que propriétaires (tels que MacOS, BeOS, NeXTStep, MS-DOS ou Windows).

Gratuité et liberté d'utilisation : Python est un langage libre et gratuit, y compris pour un usage commercial, sans aucune restriction. Syntaxe claire et expressive : Grâce à sa syntaxe simple et à ses structures de données évoluées (comme les listes et les dictionnaires), Python permet d'écrire des programmes à la fois concise et facilement lisibles.

Gestion automatique des ressources : Python gère la mémoire et les ressources

Système (comme les fichiers) automatiquement, via un système de comptage déréférences, similaire à un ramasse-miettes (gerbage collecter), mais avec un fonctionnement propre.

Orientation objet avancée : Python est un langage orienté objet, prenant en charge

L'héritage multiple, la surcharge des opérateurs, et une grande flexibilité dans la conception des objets.

Dynamique : Python peut exécuter dynamiquement du code généré sous forme de chaînes de caractères.

Orthogonal : Un petit nombre de concepts suffit à créer des structures complexes et puissantes.

Réflectif : Il permet à un objet de modifier ses propres attributs, méthodes, voire sa classe à l'exécution.

Introspectif : Python fournit des outils d'analyse du programme (comme les debugger ou profiler), souvent écrits en Python lui-même. [57]

2.4.4 Technologies et outils utilisés

Librairies

Surprise : Surprise est une bibliothèque Python spécialisée dans la construction et l'évaluation de systèmes de recommandation, notamment les systèmes de filtrage collaboratif. Elle propose plusieurs algorithmes prêts à l'emploi tels que SVD (Singulier Value Decomposition), KNN (K-Nearest Neighbors) et d'autres méthodes basées sur la factorisation de matrices

numpy : NumPy est la bibliothèque fondamentale pour le calcul scientifique en Python ,C'est une bibliothèque Python qui fournit un objet tableau multidimensionnel, divers objets dérivés (comme les tableaux masqués et les matrices), ainsi qu'un ensemble de routines pour effectuer rapidement des opérations sur les tableaux, notamment des opérations mathématiques, logiques, de manipulation de forme, de tri, de sélection, d'entrée/sortie, des transformations de Fourier discrètes, de l'algèbre linéaire de base, des opérations statistiques élémentaires, des simulations aléatoires et bien plus encore.

Pandas : Pandas est une bibliothèque Python open source, largement utilisée pour les tâches de science des données, d'analyse de données et d'apprentissage automatique. Elle est construite au-dessus d'une autre bibliothèque appelée Numpy, qui offre un support pour les tableaux multidimensionnels. En tant que l'une des bibliothèques les plus populaires pour la manipulation de données, Pandas fonctionne très bien avec de nombreux autres modules de science des données dans l'écosystème Python.

Collections : Le module collections étend les types de données intégrés de Python en fournissant des structures de données supplémentaires très performantes. Il est largement utilisé en data science, en algorithmes et dans le traitement efficace des données.

operator : Le module operator fournit des fonctions équivalentes aux opérateurs standards de Python (addition, multiplication, comparaison, etc.). Il permet de simplifier l'écriture de certaines expressions, notamment lorsqu'on utilise des fonctions d'ordre supérieur, Ce

module est utile pour rendre le code plus lisible et performant dans certaines manipulations de données.

heapq : Le module heapq fournit une implémentation du tas binaire (heap) en Python. Un tas permet de maintenir efficacement une structure de type file de priorité, où l'élément le plus petit (ou le plus grand) est toujours accessible rapidement. heapq est utile pour des algorithmes gourmands comme les files de priorité, les tris partiels (k plus petits éléments), ou encore dans les algorithmes de graphes (ex. : Dijkstra).

math : Le module math fournit un grand nombre de fonctions mathématiques de base : trigonométrie, logarithmes, puissances, fonctions hyperboliques, arrondis, factoriels, etc. Contrairement à numpy qui travaille sur des tableaux, math opère sur des scalaires (nombres simples). C'est un module léger et rapide, souvent utilisé pour les calculs numériques classiques.

itertools : Le module itertools propose des fonctions permettant de manipuler efficacement les itérateurs : permutations, combinaisons, regroupements, répétitions infinies, chaînes d'itérations, etc. Très utile en algorithmique, il permet d'écrire des boucles complexes de façon concise et optimisée, en minimisant l'utilisation de mémoire grâce à sa nature paresseuse (lazyevaluation).

sklearn : Un package essentiel pour le langage de programmation Python, fréquemment utilisé dans les projets d'apprentissage automatique, s'appelle **Scikit-learn**.Scikit-learn se concentre principalement sur les outils de machine Learning, notamment les algorithmes statistiques, mathématiques et polyvalents qui constituent la base de nombreuses technologies d'apprentissage automatique. Gratuit et open source, Scikit-learn joue un rôle crucial dans le développement de nombreux types d'algorithmes pour le machine Learning et les technologies associées. [58]

2.4.5 Mise en œuvre

2.4.5.1 Dataset

-Item_id : Le numéro d'identification du cours

-User_id: Le numéro d'identification de l'étudiant

-Watch_percentage : Le pourcentage d'achèvement des cours

-Rating : Les notes attribuées à un cours par les étudiants

-Name : Le nom du cours

-Description : Description du cours

-Created_at : Le temps de création du cours

-Theme : Le genre auquel appartient un cours.

	item_id	language	name	nb_views	description	created_at	Difficulty	Job	Software	Theme	duration	type
0	510	en	What is OneDrive for Business?	1114.0	OneDrive for Businessis an online libraryto st...	2016	Beginner	[]	['OneDrive']	['Discover']	42.0	tutorial
1	511	en	Add, restore, delete documents on OneDrive	547.0	To create a new document in OneDriveclick New,...	2016	Beginner	[]	['OneDrive']	['Share', 'Produce', 'Organize']	122.0	tutorial
2	512	en	Share documents	607.0	By default, all content is private, so you don...	2016	Beginner	[]	['OneDrive']	['Share', 'Collaborate']	176.0	tutorial
3	513	en	Recycle bin	278.0	When you delete a file or afolder, OneDrive gi...	2016	Beginner	['Accounting', 'Financial', 'Human resources',...	['OneDrive']	['Produce', 'Organize']	46.0	tutorial
4	514	en	Sync OneDrive for Business to your computer (W...	312.0	The one drive synchronization application is a...	2016	Beginner	[]	['OneDrive']	['Produce', 'Organize']	163.0	tutorial

Figure 2.4.5.1: En-tête du dataset *items*

	user_id	item_id	watch_percentage	created_at	rating
0	224557	510	100	2018-09-28 16:18:29	10
1	224557	615	100	2018-09-28 16:22:22	10
2	224557	7680	100	2018-09-28 16:23:34	10
3	224293	510	100	2018-09-28 17:20:30	10
4	224293	515	100	2018-09-28 17:40:02	10

Figure 2.4.5.2: En-tête du dataset *explicit-rating*

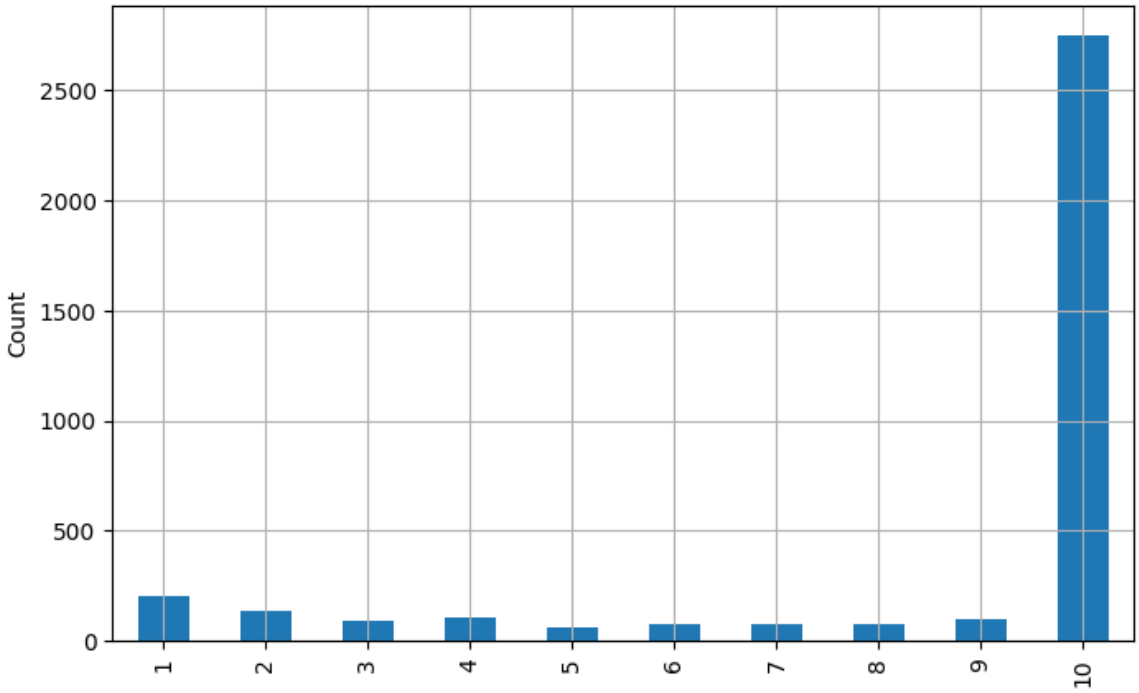

Figure 2.4.5.3 : Répartition des notes

Cette figure montre comment les notes ont été réparties dans le jeu de données par les étudiants. Comme on peut le voir, la majorité d'entre eux ont attribué la note de 10 à leurs cours, et la deuxième note la plus fréquente est 1. En dehors de cela, il n'y a pas de grande différence entre les autres notes attribuées.

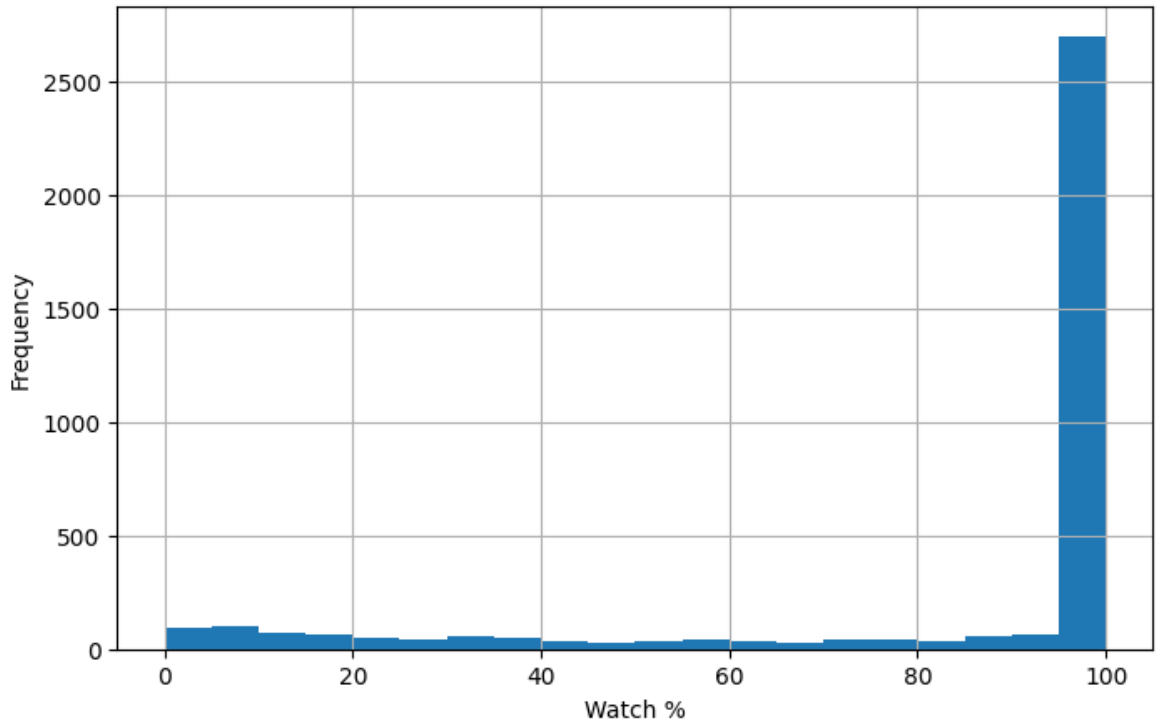

Figure 2.4.5.4: Pourcentage de visionnage

Comme nous pouvons le constater, la plupart des étudiants ont terminé leurs cours à 100 %, mais certains n'ont pas réussi à les terminer ou ne les ont presque pas commencés. Cela reflète le comportement des étudiants sur la plateforme d'apprentissage en ligne.

2.4.6 Prétraitement

Dans cette phase, nous avons effectué un prétraitement de notre jeu de données, en commençant par la lecture de celui-ci, l'extraction et l'utilisation des colonnes nécessaires, jusqu'au nettoyage des valeurs inutiles.

Nous avons utilisé cette fonction pour supprimer les lignes contenant des valeurs manquantes, car elles pourraient nuire au bon fonctionnement de nos algorithmes, et afin de maintenir un jeu de données propre.

```python
def clean_dataset(dataset) :

    for idx in dataset.index :
        for column in dataset :
            if dataset.loc[idx, column] == None or str(dataset.loc[idx, column]).lower() == "nan" :
                dataset = dataset.drop(index = idx)
                break
    return dataset
```

Figure 2.4.6: Jeu de données nettoyé

2.4.7 L'implémentations

L'implémentation du système de recommandation hybride a été réalisée en Python à partir de trois modules principaux développés manuellement : le filtrage basé sur le contenu, le filtrage collaboratif et l'algorithme hybride.

- *Filtrage basé sur le contenu*

Le filtrage basé sur le contenu repose sur la construction d'une *matrice de similarité entre les cours en* fonction de plusieurs caractéristiques :

- *Genres des cours* : la similarité est calculée par le produit scalaire normalisé (cosine similarité) entre les vecteurs de genres.
- *Année de publication :* une fonction exponentielle est utilisée pour modéliser la décroissance de similarité en fonction de la différence d'années.
- *Descriptions textuelles :* la similarité est mesurée en comptant les mots-clés communs présents dans les descriptions des cours.

La prédiction d'une note pour un utilisateur donné est ensuite effectuée en identifiant les *k* cours les plus similaires aux cours déjà notés, puis en calculant une moyenne pondérée des notes de ces voisins.

```
Function __init__(k, sim_options):
        Call parent constructor AlgoBase
        Set self.k

    Function fit(trainset, load=False):
        Call parent fit with trainset
        Get genres from dataset
        Get years from dataset
        Get descriptions from dataset

        If load is True:
            Import similarity matrix from file
            Return self

        Initialize similarity matrix with zeros

        For each item A in trainset:
            For each item B after A in trainset:
                Get genre similarity between A and B
                If genre similarity is 0:
                    Set similarity to 0
                Else:
                    Compute year similarity
                    Compute description similarity
                    Combine all similarities into one score
                Save similarity score in matrix (both directions)
        Return self

    Function computeGenreSimilarity(course1, course2, genres):
        Compute cosine similarity between genre vectors
        Return result

    Function computeYearSimilarity(course1, course2, years):
        Compute exponential decay based on year difference
        Return result

    Function computedescriptionsimilarity(course1, course2, descriptions):
        Compare common terms in descriptions
```

Figure 2.4.7.1 : code pour filtrage par contenu

```
Function computeYearSimilarity(course1, course2, years):

Function computedescriptionsimilarity(course1, course2, descriptions):
    Compare common terms in descriptions
    Compute similarity percentage
    Return result

Function exportSimilarities():
    Save similarity matrix to file

Function importSimilarities():
    Load similarity matrix from file
    Return matrix

Function predict(u, i):
    If student u or course i not in trainset:
        Raise PredictionImpossible

    Initialize neighbor list
    For each item rated by student u:
        Get similarity between i and rated item
        Add (similarity, rating) to neighbors

    Select top-k neighbors based on similarity
    Compute weighted average of ratings
    Return predicted rating

Function EstamedRatingsForstudent(u):
    Initialize empty list prid
    For each item in trainset:
        Predict rating for student u and item
        Append rating to prid
    Return prid
```

Figure 2.4.7.2 : Suite du code de filtrage par contenu

- *Filtrage collaboratif*

Le filtrage collaboratif est implémenté via une factorisation de matrices personnalisée. Deux matrices de facteurs latents sont initialisées : l'une pour les utilisateurs (étudiants), l'autre pour les items (cours). Le modèle est entraîné par descente de gradient stochastique sur plusieurs itérations (epochs), en ajustant progressivement les facteurs pour minimiser l'erreur de prédiction.

Chaque prédiction est ensuite obtenue en calculant le produit scalaire entre les vecteurs de facteurs utilisateur et item correspondants.

```
Class Collaborative inherits from AlgoBase:

    Function __init__(learning_rate, n_epochs, n_factors, reg):
        Set self.learning_rate
        Set self.n_epochs
        Set self.n_factors
        Set self.reg

    Function fit(trainset):
        Call parent fit with trainset
        Initialize self.students_f with random values
        Initialize self.courses_f with random values
        For each epoch in number of epochs:
            For each (studentID, courseID, rating) in trainset:
                Compute error between rating and dot product of student and course factors
                Update student factor
                Update course factor

    Function predict(u, i):
        If student or course unknown in trainset:
            Raise PredictionImpossible
        Compute estimated rating as dot product of student and course factors
        Return estimated rating

    Function EstamedRatingsForstudent(u):
        Initialize empty list prid
        For each item in trainset:
            Compute predicted rating for student u and item
            Append rating to prid
        Return prid
```

Figure 2.4.7.3 : code de filtrage collaboratif

- *Approche hybride*

L'approche hybride combine de manière équilibrée les prédictions des deux modèles précédents. Elle est encapsulée dans une classe Hybride qui contient une liste de recommander (le modèle collaboratif et le modèle basé sur le contenu).

Lors de la prédiction, le système exécute chaque algorithme séparément sur l'utilisateur et l'item cibles, récupère les différentes estimations produites, puis calcule leur moyenne pour générer une prédiction finale. Cette combinaison simple mais efficace permet d'exploiter simultanément les avantages de la personnalisation comportementale du collaboratif et de la pertinence des attributs du contenu spécifiques.

```
Function __init__(recommendersList):
    Set self.recommenders

Function fit(trainset):
    For each algorithm in self.recommenders:
        Call fit on algorithm with trainset

Function predict(uid, iid, r_ui=None, clip=True, verbose=False):
    Initialize empty list predictions
    For each algorithm in self.recommenders:
        Append prediction result to predictions
    Initialize mean to 0
    For each (uid, iid, r_ui, est, details) in predictions:
        Add est to mean
    Compute average of estimates
    Construct final prediction list with average estimate
    Return prediction

Function test(testset, verbose=False):
    For each (uid, iid, r_ui) in testset:
        Predict and collect results
    Return list of predictions

Function EstamedRatingsForstudent(u):
    Initialize empty list prid
    For each item in trainset of first recommender:
        Predict rating for student u and item
        Append rating to prid
    Return prid
```

Figure 2.4.7.4 : code de l'approche hybride

2.4.8 Évaluation expérimentale

Comme nous pouvons le voir, voici les résultats de nos algorithmes en fonction des métriques présentées ci-dessous.

Approches	*Précision*	*Rappel*	*F1-Score*	*NDCG*
Filtrage à BC	*98.47%*	*99.78%*	*99.12%*	*96.23%*
Filtrage à Collaboratif	*99.23%*	*99.91%*	*99.56%*	*96.31%*
Hybride	*99.57%*	*99.93%*	*99.74%*	*98.15%*

Tableaux 2.4.8.1 : Résultats comparatifs des approches de recommandation selon les métriques d'évaluation

Approches	*Taux de complétion*	*Progression académique*	*Réutilisation*	*Pertinence subjective*
Filtrage à BC	*61.37%*	*75.38%*	*99.76%*	*40.22%*
Filtrage à Collaboratif	*60.28%*	*77.43%*	*99.96%*	*42.11%*
Hybride	*65.23%*	*78.62%*	*99.97%*	*45.26%*

Tableaux 2.4.8.2 : Évaluation qualitative des approches de recommandation selon l'engagement et la pertinence perçue

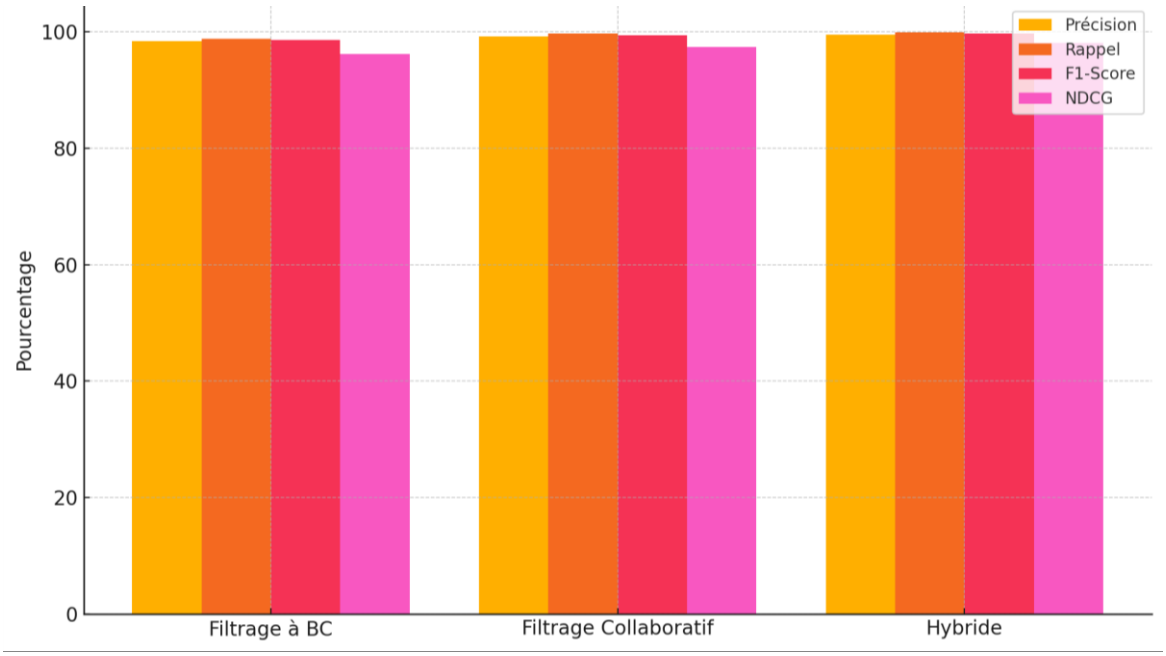

Figure 2.4.8.1 : Métriques classiques

Dans ce graph, on compare les performances des trois approches selon les métriques standards de recommandation : Précision, Rappel, F1-Score et NDCG
On observe que l'approche hybride domine systématiquement les deux autres sur l'ensemble des indicateurs. Elle atteint une précision de 99.57% et un rappel de 99.93%, traduisant une capacité remarquable à proposer des recommandations à la fois pertinentes et complètes. Le F1-Score élevé (99.74%) confirme cet équilibre entre précision et rappel. Le NDCG à 98.15% indique également que les éléments les plus pertinents apparaissent très tôt dans la liste de recommandations, optimisant ainsi l'expérience utilisateur.

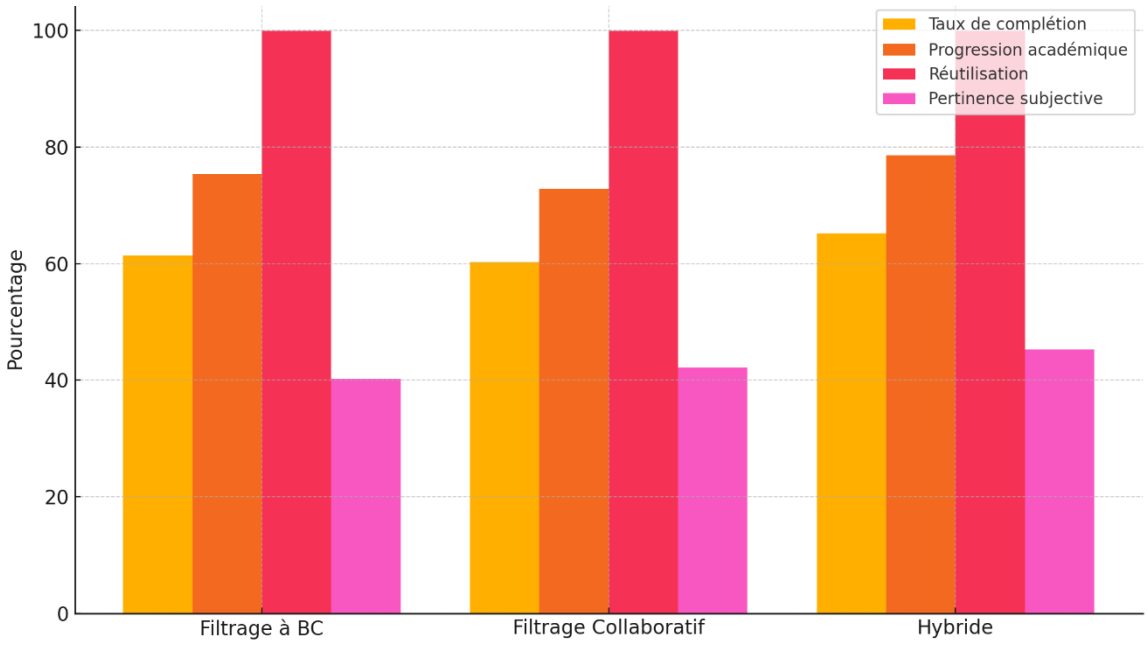

Figure 2.4.8.2 : Métriques spécifiques

Le second graphique met en évidence les performances des trois approches selon des critères adaptés au domaine éducatif. Là encore, l'approche hybride se distingue par des valeurs supérieures sur tous les indicateurs. Le taux de complétion des cours recommandés atteint 65.23%, montrant un engagement plus fort des apprenants. La progression académique atteint 78.62%, démontrant un réel impact pédagogique. Le taux de réutilisation, proche de 100% pour toutes les approches, reste légèrement plus élevé pour l'hybride (99.97%), indiquant une forte fidélisation. Enfin, la pertinence subjective, bien que globalement plus faible, reste la meilleure pour l'hybride (45.26%), traduisant une satisfaction perçue supérieure.

2.4.9 Discussion

L'analyse des résultats obtenus à partir des différentes approches de recommandation permet de dégager plusieurs observations importantes sur l'efficacité comparative des méthodes testées.

Tout d'abord, en ce qui concerne les métriques classiques de recommandation (Précision, Rappel, F1-Score et NDCG), l'approche hybride surpasse systématiquement les approches individuelles. La précision du modèle hybride atteint 99.57%, contre 98.47% pour le filtrage à base de contenu et 99.23% pour le filtrage collaboratif. De même, le rappel et le

F1-Score suivent cette tendance, avec des valeurs maximales respectivement de 99.93% et 99.74% pour l'approche hybride. Le NDCG, qui tient compte de la position des items pertinents dans la liste, est également meilleur pour l'approche hybride (98.15%), ce qui indique une meilleure qualité de classement des recommandations.

En parallèle, les métriques spécifiques au domaine e-Learning confirment également la supériorité de l'approche hybride. Le taux de complétion des cours est plus élevé pour l'hybride (65.23%) par rapport au filtrage à base de contenu (61.37%) et au filtrage collaboratif (60.28%). La progression académique, qui mesure l'amélioration des compétences après consommation des contenus recommandés, est elle aussi la plus importante dans le cas de l'approche hybride (78.62%). De plus, les taux de réutilisation du système sont globalement élevés pour toutes les approches, avec toutefois un léger avantage à l'hybride (99.97%), ce qui démontre une satisfaction globale et une fidélisation accrue des utilisateurs. Enfin, la pertinence subjective, qui reflète la satisfaction perçue des utilisateurs, atteint 45.26% pour l'hybride, contre 40.22% et 42.11% pour les deux autres méthodes.

Ces résultats confirment l'intérêt de l'approche hybride qui parvient à combiner les avantages des deux méthodes de base : la capacité du filtrage collaboratif à capturer les similarités comportementales entre utilisateurs, et la capacité du filtrage à base de contenu à exploiter les caractéristiques propres aux ressources pédagogiques. Cette combinaison permet d'améliorer significativement la qualité des recommandations aussi bien en termes d'adéquation objective des contenus qu'en termes d'impact pédagogique mesurable.

2.4.10 Conclusion

Au vu des résultats obtenus, il apparaît clairement que l'approche hybride offre les performances les plus équilibrées et les plus satisfaisantes, aussi bien sur les métriques classiques que sur les indicateurs spécifiques au domaine de l'e-Learning. Sa capacité à combiner les forces du filtrage collaboratif et du filtrage à base de contenu lui permet de générer des recommandations à la fois pertinentes, diversifiées et pédagogiquement efficaces. Ces résultats confirment l'intérêt d'adopter des approches hybrides pour améliorer la qualité des systèmes de recommandation dans les environnements d'apprentissage en lign