Système de Recommandation de film

Sommaire

I. Introduction

A. Contexte et motivation

Les systèmes de recommandation sont devenus une composante essentielle des plateformes numériques modernes, jouant un rôle important dans la personnalisation de l'expérience utilisateur. Utilisés par des géants de la technologie comme Netflix, Amazon, et YouTube, ces systèmes aident à naviguer dans l'immense quantité de contenu disponible en proposant des recommandations personnalisées basées sur les préférences et les comportements passés des utilisateurs.

Le présent rapport se propose d'explorer la conception et l'implémentation d'un modèle de recommandation en utilisant les ensembles de données fournis par IMDb et MovieLens. Pour ce faire, il s'appuiera sur les meilleures pratiques et les ressources offertes par la plateforme Datascientest, ainsi que sur les informations recueillies sur internet concernant les retours d'expérience menés par les géants de la technologie dans le domaine des systèmes de recommandation.

B. Objectifs

Création d'un système de recommandation

- Analyser et préparer les ensembles de données IMDb et MovieLens pour la modélisation.
- Implémentation d'un système de recommandation pour générer des recommandations personnalisées basées sur les comportements et les préférences des utilisateurs.
- Consolidation d'un jeu de données en utilisant des techniques de web scraping pour enrichir les données disponibles et améliorer la précision du modèle, si nécessaire.

Rapports d'exploration, de data visualisation et de pre-processing des données

- Réalisation d'un rapport détaillé d'exploration des données, mettant en évidence les tendances, les corrélations et les caractéristiques significatives du jeu de données.
- Utilisation de techniques de data visualisation pour présenter de manière efficace les informations extraites des données.
- Mise en œuvre de processus de pre-processing des données pour nettoyer, transformer et préparer les données en vue de leur utilisation dans la modélisation.

Rapport de modélisation

- Développement et entraînement d'un modèle de recommandation basé sur le machine learning.
- Évaluation approfondie de la performance du modèle, en utilisant des métriques pertinentes et en comparant ses résultats avec des modèles de référence.

Rapport final et GitHub associé

- Compilation de tous les résultats, analyses et conclusions dans un rapport final détaillé.
- Publication du code source, des scripts et des documents associés sur GitHub pour une transparence et une reproductibilité maximales des résultats obtenus.

II. Revue de littérature

A. Qu'est ce qu'un système de recommandation ?

Un système de recommandation est un outil logiciel et une technique d'analyse de données qui offrent des suggestions de produits, de services ou de contenu à un utilisateur. Ces suggestions peuvent prendre la forme de recommandations de films, de musique, de livres, de produits à acheter, et même d'amis à ajouter sur les réseaux sociaux. Il existe plusieurs types de systèmes de recommandation :

- Filtrage Collaboratif (Collaborative Filtering) : Ces systèmes recommandent des éléments basés sur les préférences et les comportements similaires d'autres utilisateurs.
- Filtrage Basé sur le Contenu (Content-Based Filtering) : Ces systèmes recommandent des éléments similaires aux éléments que l'utilisateur a appréciés dans le passé, en se basant sur les caractéristiques du contenu.
- Systèmes Hybrides : Ces systèmes combinent plusieurs techniques de recommandation pour améliorer la précision des suggestions.

B. A quoi sert un système de recommandation ?

Les systèmes de recommandation servent plusieurs objectifs importants :

- Amélioration de l'expérience utilisateur : En proposant du contenu pertinent et personnalisé, les utilisateurs sont plus susceptibles de trouver ce qu'ils recherchent et de découvrir de nouveaux contenus qu'ils apprécieront.
- Augmentation de l'engagement et des revenus : En gardant les utilisateurs engagés et en augmentant la probabilité qu'ils consomment plus de contenu ou achètent plus de produits, les systèmes de recommandation peuvent avoir un impact direct sur les revenus des entreprises.
- Gestion et filtrage de l'information : Dans un monde où l'information est abondante, les systèmes de recommandation aident à filtrer le bruit et à mettre en avant les éléments les plus pertinents.

C. Pour qui ?

Les systèmes de recommandation sont utilisés dans de nombreux secteurs et par diverses plateformes :

- Streaming Vidéo et Musique : Netflix, YouTube, Spotify
- E-commerce : Amazon, eBay
- Réseaux Sociaux : Facebook, LinkedIn
- Bibliothèques Numériques : Goodreads, Kindle
- Applications de Rencontres : Tinder, OkCupid

D. Comment fonctionnent les systèmes de recommandation basés sur le ML ?

Les systèmes de recommandation basés sur le machine learning (ML) suivent un processus standard adaptable en fonction des besoins. Ils exploitent des algorithmes sophistiqués pour analyser d'importantes quantités de données et formuler des prédictions précises sur les préférences des utilisateurs. Pour cela, le processus standard comprend les étapes suivantes :

- Collecte de données : Cette phase implique l'extraction des données pertinentes, telles que les données des utilisateurs, du contenu et contextuelles, nécessaires au fonctionnement du système de recommandation.
- Prétraitement des données : Les données extraites sont ensuite soumises à un processus de nettoyage et de mise en forme afin de les préparer à l'entraînement du modèle. Cela inclut souvent la correction des erreurs, la suppression des valeurs manquantes et la normalisation des données.
- Entraînement du modèle : Les données prétraitées sont utilisées pour entraîner le modèle de recommandation, qui peut être basé sur des techniques telles que le filtrage collaboratif, le filtrage basé sur le contenu ou des approches hybrides combinant plusieurs méthodes.
- Évaluation et optimisation : Une fois le modèle entraîné, il est essentiel de l'évaluer pour vérifier sa performance. Cette évaluation permet d'identifier les éventuels défauts ou biais du modèle et de procéder à des ajustements pour optimiser ses performances en termes de précision et de pertinence des recommandations.

E. Quels sont les avantages et les limitations des systèmes de recommandation basés sur le ML ?

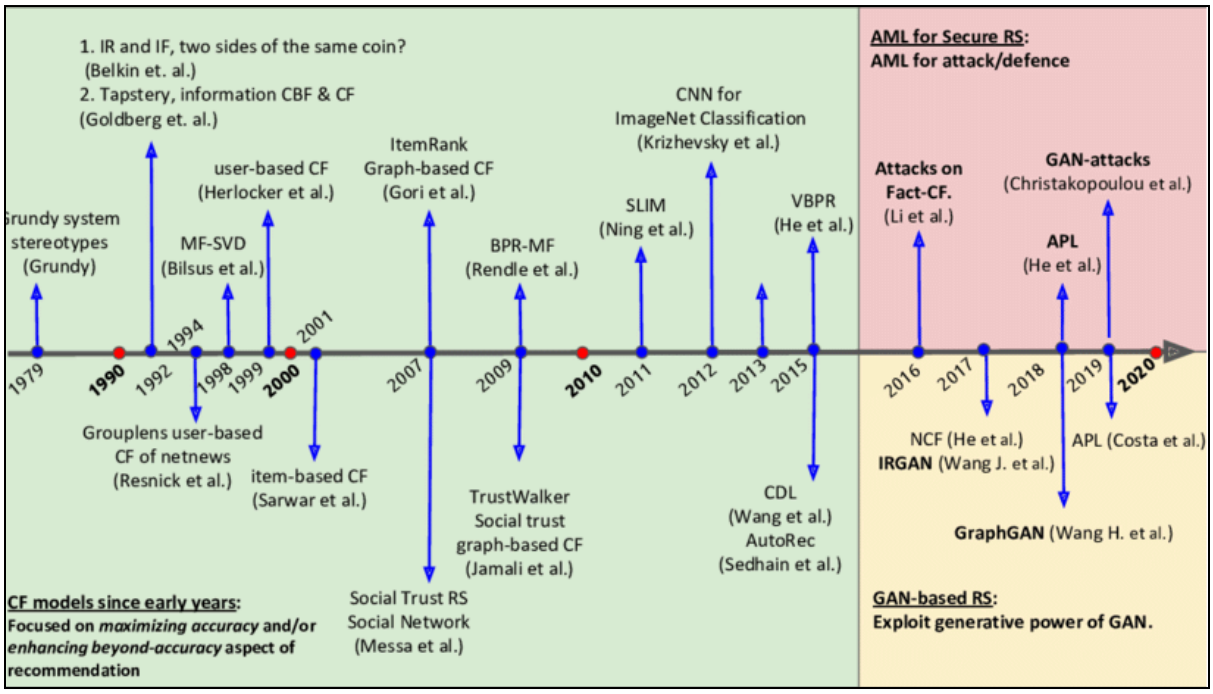
Les systèmes de recommandation basés sur le machine learning offrent de nombreux avantages. Tout d'abord, ils permettent une personnalisation des recommandations, ce qui signifie que chaque utilisateur reçoit des suggestions adaptées à ses préférences individuelles. En outre, ces systèmes aident les utilisateurs à découvrir de nouveaux contenus qu'ils pourraient ne pas avoir trouvés autrement, élargissant ainsi leurs horizons. En proposant des suggestions pertinentes, ils permettent également aux utilisateurs de gagner du temps en recherchant des contenus qui correspondent à leurs goûts. Enfin, en fournissant des

recommandations attrayantes, ces systèmes peuvent contribuer à augmenter l'engagement des utilisateurs sur une plateforme, favorisant ainsi la fidélité et la satisfaction des utilisateurs.

Malgré leurs avantages, les systèmes de recommandation basés sur le machine learning présentent également certaines limitations importantes. Tout d'abord, ils peuvent être affectés par les biais présents dans les données d'entraînement, ce qui peut conduire à des recommandations discriminatoires ou injustes. Par exemple, l'affaire de l'algorithme de recrutement d'Amazon illustre bien ces risques, mettant en lumière la possibilité de perpétuation de stéréotypes ou de discriminations involontaires. Dans ce cas, l'outil discriminait les femmes candidates, car le modèle avait été entraîné sur des CV soumis à l'entreprise au cours des dix années précédentes, période où les hommes étaient majoritairement embauchés pour des postes techniques. En conséquence, le système avait appris à associer certaines caractéristiques des CV masculins à une performance élevée.

De surcroît, le manque de transparence quant à la manière dont les systèmes de recommandation génèrent leurs suggestions peut susciter la méfiance des utilisateurs quant à leur fiabilité et à leur impartialité. Parfois, ces systèmes ont également tendance à recommander trop de contenu similaire, ce qui limite la diversité des choix offerts aux utilisateurs.

F. Quelle est l'histoire et l'évolution du système de recommandation ?



Le premier système de recommandation connu est Grundy, développé en 1979 par Elaine Alice Rich, une informaticienne américaine. Il utilisait des stéréotypes pour recommander des livres à des utilisateurs en se basant sur un modèle simple de filtrage collaboratif, où les recommandations étaient générées en analysant les achats et les évaluations des utilisateurs.

Le système identifiait les utilisateurs aux goûts similaires et recommandait aux utilisateurs des produits que les utilisateurs similaires avaient appréciés. Bien que Grundy ait été conçu pour les produits en général, son succès a démontré le potentiel des systèmes de recommandation pour fournir des suggestions personnalisées et pertinentes aux utilisateurs. Le système a inspiré le développement de nombreux autres systèmes de recommandation, et ses principes de base sont encore utilisés aujourd'hui dans les systèmes modernes.

Puis dans les années 1990, des modèles de recommandations plus complexes ont vu le jour, notamment le système de Tapestry développé par une équipe dirigée par David Goldberg. Le système se basait sur un filtrage collaboratif. Celui-ci effectuait une recherche en fonction du contenu et des réactions enregistrées des autres utilisateurs. Ou bien même le système de GroupLens créé par John Riedl et Paul Resnick, est l'un des premiers systèmes à utiliser le filtrage collaboratif automatisé pour recommander des articles de Usenet. Il utilisait les évaluations des utilisateurs pour prédire les évaluations d'autres utilisateurs pour les mêmes articles.

Ensuite dans les années 2000, les modèles basés sur les matrices ont monté en puissance, principalement grâce un à géant de la technologie, Amazon, qui a introduit un système de recommandation basé sur les items, où les produits similaires sont recommandés en fonction de ce que d'autres clients ont acheté, contribuant de manière significative à la popularité du commerce électronique. Dans les années 2006, un autre géant de la technologie, Netflix, a introduit les modèles de factorisation de matrice, qui ont marqué une avancée majeure en permettant de capturer les interactions complexes entre utilisateurs et items en décomposant la matrice utilisateur-item en produits de matrices de rang inférieur.

En 2010, les filtres collaboratifs neuronaux ont commencé à émerger, combinant les réseaux de neurones avec des techniques de filtrage collaboratif pour améliorer la précision des recommandations en capturant des relations non linéaires entre utilisateurs et items, et l'intégration de données provenant des réseaux sociaux a permis de raffiner les recommandations en tenant compte des relations sociales et des influences interpersonnelles.

G. Comment fonctionne l'algorithme de Netflix ?

L'algorithme de Netflix fonctionne en suivant plusieurs étapes clefs pour recommander des contenus personnalisés à chaque utilisateur. Pour ce faire, la plateforme collecte des données sur le comportement de visionnage de chaque utilisateur, y compris l'historique de visionnage et les éventuelles notes attribuées aux contenus. Ensuite, ces données sont utilisées pour modéliser les préférences de chaque utilisateur, tandis que l'historique de visionnage et les notes sont analysés pour comprendre les types de contenus que l'utilisateur apprécie.

Netflix collecte également une grande quantité de données sur chaque titre de contenu, y compris des métadonnées telles que le genre, le casting, le réalisateur, etc. Ces métadonnées sont ensuite analysées à l'aide de techniques d'analyse textuelle pour mieux comprendre le contenu et l'associer aux préférences des utilisateurs.

Une fois la collecte terminée, Netflix utilise deux types de techniques de recommandation, le filtrage collaboratif et le filtrage basé sur le contenu. Puis les recommandations sont intégrées à l'interface utilisateur de Netflix. Cela se traduit par la mise en avant des titres recommandés sur la page d'accueil et la création de listes personnalisées telles que "À regarder" ou "Séries similaires à celles que vous avez aimées".

Netflix mène également des tests A/B pour évaluer l'efficacité de différentes versions de son algorithme. En utilisant des données en temps réel, la plateforme optimise continuellement ses recommandations pour s'assurer qu'elles restent pertinentes et précises.

III. Données utilisées

A. IMDb datasets

Les ensembles de données IMDb comprennent plusieurs fichiers, chacun contenant des informations spécifiques sur les films, les émissions de télévision, les acteurs, les réalisateurs, et les évaluations des utilisateurs. Les principaux fichiers utilisés dans notre exploration sont :

- name.basics.tsv : Contient des informations sur les personnes (acteurs, réalisateurs, etc.) telles que le nom, la date de naissance, les professions connues, etc.
- title.akas.tsv : Contient des informations sur les titres alternatifs de chaque film ou émission.
- title.basics.tsv : Contient des informations de base sur les titres (films, émissions, épisodes) tels que le titre, l'année de sortie, le genre, etc.
- title.crew.tsv : Contient des informations sur les membres de l'équipe technique pour chaque titre, y compris les réalisateurs et les scénaristes.
- title.episode.tsv : Contient des informations sur les épisodes de séries télévisées, y compris les IDs des épisodes et les IDs des titres de la série auxquels ils appartiennent.
- title.principals.tsv : Contient des informations sur les principaux contributeurs (acteurs, réalisateurs, scénaristes) associés à chaque titre.
- title.ratings.tsv : Contient les notes moyennes et le nombre de votes pour chaque titre.

B. MovieLens 20M datasets

L'ensemble de données MovieLens 20M contient des informations sur 20 millions de notations de films données par 138 000 utilisateurs anonymes sur environ 27 000 films. Les principaux fichiers utilisés sont :

- ratings.csv : Contient les notations attribuées par les utilisateurs aux films, avec les colonnes userId, movieId, rating, et timestamp.
- movies.csv : Contient les informations sur les films, avec les colonnes movieId, title, et genres.
- tags.csv : Contient les tags attribués par les utilisateurs aux films, avec les colonnes userId, movieId, tag, et timestamp.

- links.csv : Contient pour chaque film, les identifiants correspondants dans les trois systèmes de bases de données (MovieLens, IMDb, et TMDb), avec les colonnes movieId, imdbId, tmdbId.
- genome-scores.csv : Contient l'identifiant du film associé à un tag et sa pertinence, avec les colonnes movieId, tagId et relevance.
- genome-tags.csv : Contient l'identifiant du tag et son contenu, avec les colonnes tagId et tag.

IV. Exploration des données

A. Introduction

En suivant le processus standard pour construire un modèle de ML : collecte des données, exploration des données, prétraitement des données, entraînement du modèle, et enfin évaluation et optimisation du modèle, nous avons commencé par la première étape en rassemblant les ensembles de données non commerciaux fournis par IMDb et l'ensemble de données MovieLens 20M, fournis par GroupLens, dans le dossier ***00_data_source*** de notre répertoire de projet ***dstrec***. Ensuite, nous sommes passés à l'exploration des deux ensembles de données. Cette étape est importante dans le processus de construction d'un modèle de recommandation. Elle permet de comprendre la structure et les caractéristiques des données, d'identifier les tendances, les anomalies, et les relations entre les variables, ainsi que de préparer les données pour l'analyse et le développement du modèle.

Et en parallèle, nous avons effectué la datavisualisation de l'exploration des données. Étant donné que c'est une part importante de la phase d'exploration des données. Elle apporte un éclairage complémentaire sur les données, notamment sur la visualisation des informations de manière unitaire ou croisée, sur la compréhension et l'appréhension des données.

Exploration des données

L'exploration des données a été réalisée au sein de plusieurs Jupyter Notebooks, chacun dédié à un fichier contenant des informations spécifiques sur les films :

IMDb datasets

Fichier d'évaluation	Fichier source
dstrec_evaluation_name_basics_data.ipynb	name.basics.tsv
dstrec_evaluation_title_akas_data.ipynb	title.akas.tsv
dstrec_evaluation_title_basics_data.ipynb	title.basics.tsv
dstrec_evaluation_title_crew_data.ipynb	title.crew.tsv
dstrec_evaluation_title_episode_data.ipynb	title.episode.tsv
dstrec_evaluation_title_principals_data.ipynb	title.principals.tsv
dstrec_evaluation_title_ratings_data.ipynb	title.ratings.tsv

MovieLens 20M datasets

Fichier d'évaluation	Fichier source
dstrec_evaluation_movies_data.ipynb	movies.csv
dstrec_evaluation_ratings_data.ipynb	ratings.cvs
dstrec_evaluation_tags_data.ipynb	tags.csv
dstrec_evaluation_links_data.ipynb	links.csv
dstrec_evaluation_genome_scores_data.ipynb	genome_score.csv
dstrec_evaluation_genome_tags_data.ipynb	genome_tags.csv

Visualisation des données

La visualisation des données a été réalisée au sein de plusieurs Jupyter Notebooks, chacun dédié à une phase spécifiques :

IMDb datasets

Graphique	Fichier de visualisation
Top 10 des professions les plus présentes	dstrec_visualisation_name_basics_data.ipynb
Proportion des valeurs nulles et renseignées	dstrec_visualisation_title_akas_data.ipynb
Top 20 des genres les plus présentes	dstrec_visualisation_title_basics_data.ipynb
Proportion des valeurs nulles et renseignées	dstrec_visualisation_title_crew_data.ipynb
Distribution de la note moyenne des films	dstrec_visualisation_title_rating_data.ipynb

MovieLens 20M datasets

Nom du graphique	Fichier de visualisation
Histogramme des 10 genres les plus courants	dsterc_visualisation_movielens.ipynb
Histogramme des 20 tags les plus courants	dsterc_visualisation_movielens.ipynb
Distribution des notes moyennes	dsterc_visualisation_movielens.ipynb

B. Exploration des données IMDb

Chargement des données

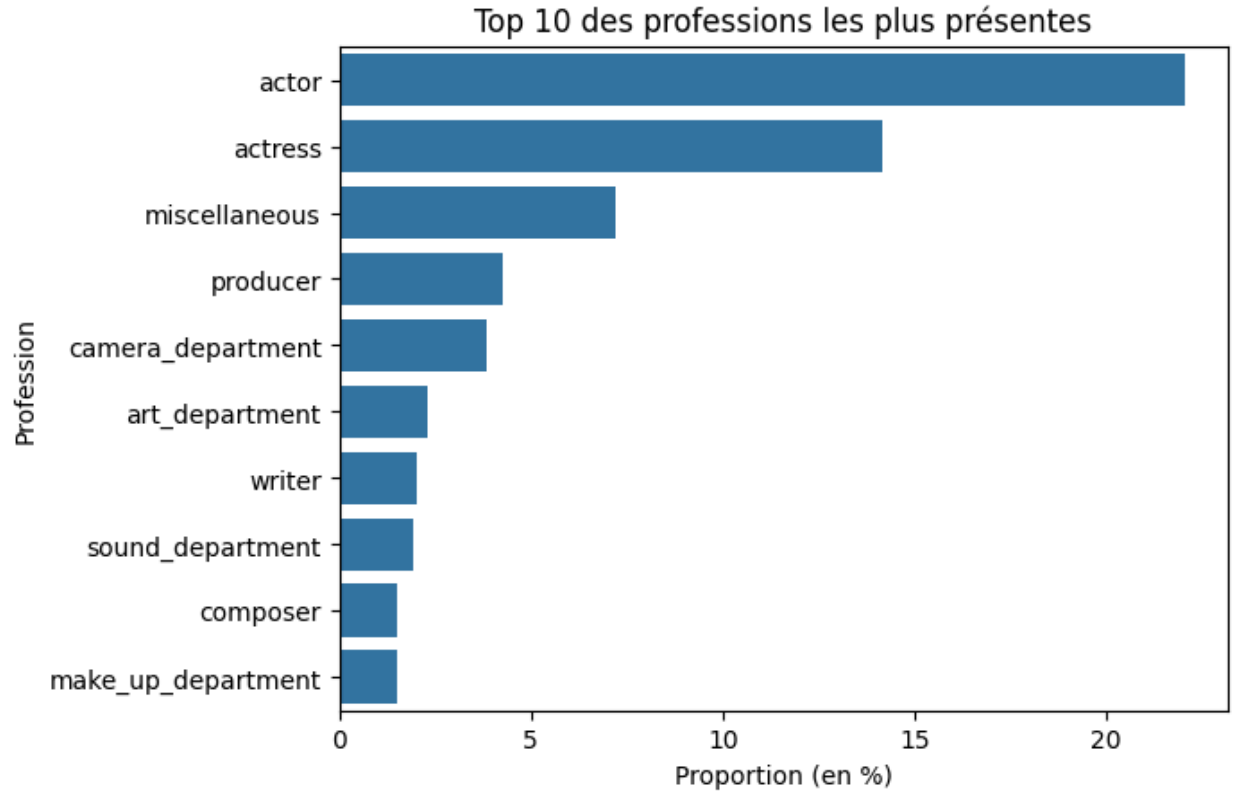
Lors du chargement des données de l'ensemble IMDb, nous nous sommes heurté à une volumineuse quantité de données, dépassant les 85 millions de lignes dans le jeu de données ***title.principals***. Après plusieurs essais, nous avons opté pour l'utilisation du paramètre ***low_memory*** et, si nécessaire, du paramètre ***nrows***, ce qui nous a permis d'analyser une partie représentative des données tout en garantissant une analyse efficace sans ralentir le serveur.

Analyse du dataset Name basics

Le jeu de données contenait 6 colonnes et 13 508 117 lignes. Voici un aperçu des première lignes du dataset :

nconst	primaryName	birthYear	deathYear	primaryProfession	knownForTitles
nm0000001	Fred Astaire	1899.0	1987.0	actor,miscellaneous,producer	tt0072308,tt0050419,tt0053137,tt0027125
nm0000002	Lauren Bacall	1924.0	2014.0	actress,soundtrack,archive_footage	tt0037382,tt0075213,tt0117057,tt0038355

Visualisation de la colonne *primaryProfession* :



Observation :

Nous avons constaté que le dataset ne contenait pas de valeurs dupliquées. Bien que ce jeu de données possédait de nombreuses valeurs manquantes, les colonnes ***nconst*** et ***primaryName***, qui nous intéressaient particulièrement, ont moins de 0.0005% de valeurs manquantes.

En outre, nous avons observé que les valeurs de la colonne ***knownForTitles*** n'étaient pas suffisamment explicites pour être exploitées dans les étapes suivantes de notre analyse.

Décision pour le prétraitement des données :

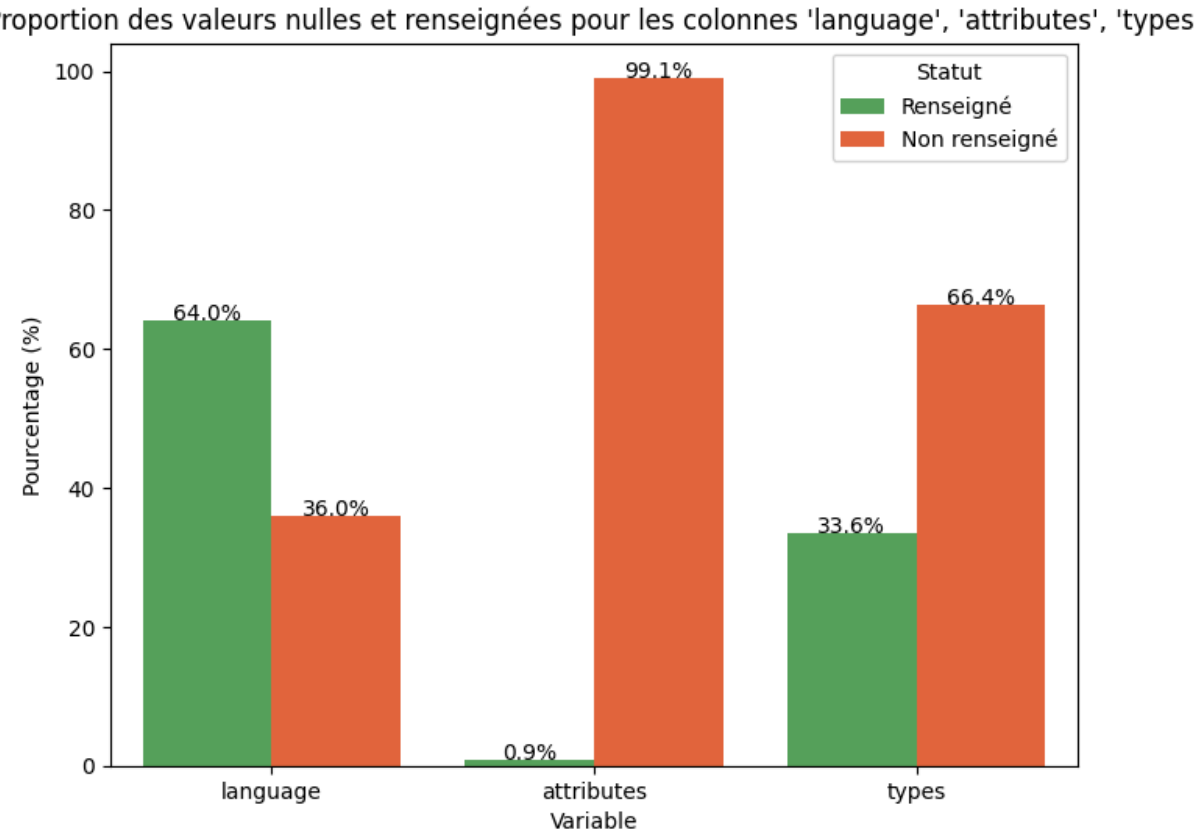
Par conséquent, pour la phase de pré-traitement des données, nous avons décidé de conserver uniquement les colonnes ***nconst*** et ***primaryName***.

Analyse du dataset Title akas

Le jeu de données contenait 8 colonnes et 48 459 150 lignes. Voici un aperçu des première lignes du dataset :

titleId	ordering	title	region	...	isOriginalTitle
tt0000001	1	Carmencita	NaN	...	1
tt0000001	2	Carmencita	DE	...	0

Visualisation des colonnes ayant un fort taux de valeurs manquantes :



Observation :

Nous avons observé que ce jeu de données n'apporte pas de plus-value pour le modèle de ML. Les colonnes censées fournir des informations supplémentaires sur les films contiennent énormément de valeurs manquantes. Par exemple, sur l'échantillon analysé :

- La colonne ***types*** a plus de 67% de valeurs manquantes.
- La colonne ***attributes*** a plus de 99% de valeurs manquantes.
- La colonne ***language*** a plus de 36% de valeurs manquantes.

Cette quantité importante de valeurs manquantes limite l'utilité de ces colonnes dans notre analyse.

Décision pour le prétraitement des données :

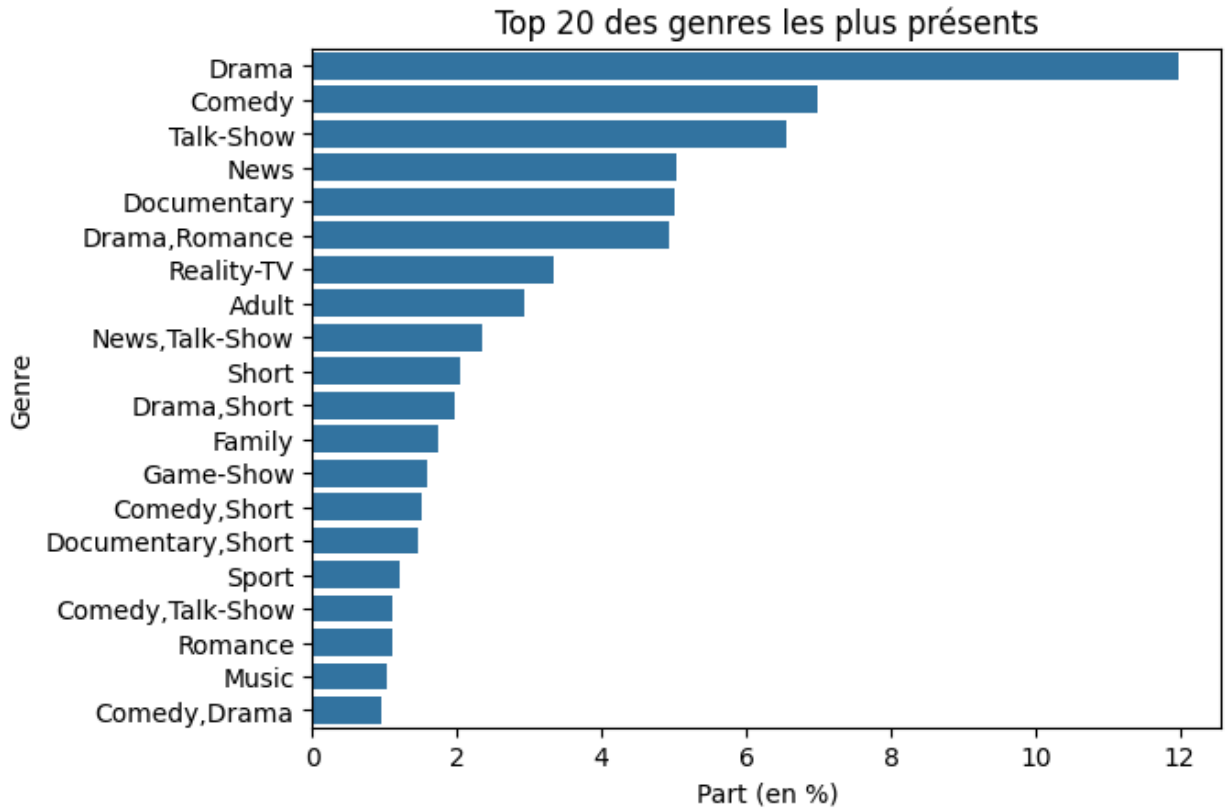
De surcroît, les titres alternatifs des films, bien qu'intéressants, ne sont pas essentiels pour l'objectif principal de notre modèle de recommandation. Ainsi, nous avons décidé de ne pas inclure ce dataset dans la phase de prétraitement des données.

Analyse du dataset Title basics

Le jeu de données contenait 9 colonnes et 10 790 736 lignes. Voici un aperçu des première lignes du dataset :

tconst	titleType	primaryTitle	originalTitle	...	genres
tt0000001	short	Carmencita	Carmencita	...	Documentary,Short
tt0000002	short	Le clown et ses chiens	Le clown et ses chiens	...	Animation,Short

Visualisation de la colonne *genres* :



Observation :

Nous avons constaté que le jeu de données ne contenait pas de valeurs dupliquées, la colonne ***endYear*** indique l'année de fin pour les séries télévisées, la colonne ***runtimeMinutes*** possède 70% de valeurs manquantes, tandis que la colonne ***originalTitle*** a 99% de ses valeurs identiques à la colonne ***primaryTitle***.

Nous observons via le graphique que les valeurs ***DRAMA*** de la colonne ***genres*** sont les types de genre les plus présents dans le fichier. Et que les genres présents dans ce TOP 20 représentent en proportion la grande majorité des genres du fichier.

Décision pour le prétraitement des données :

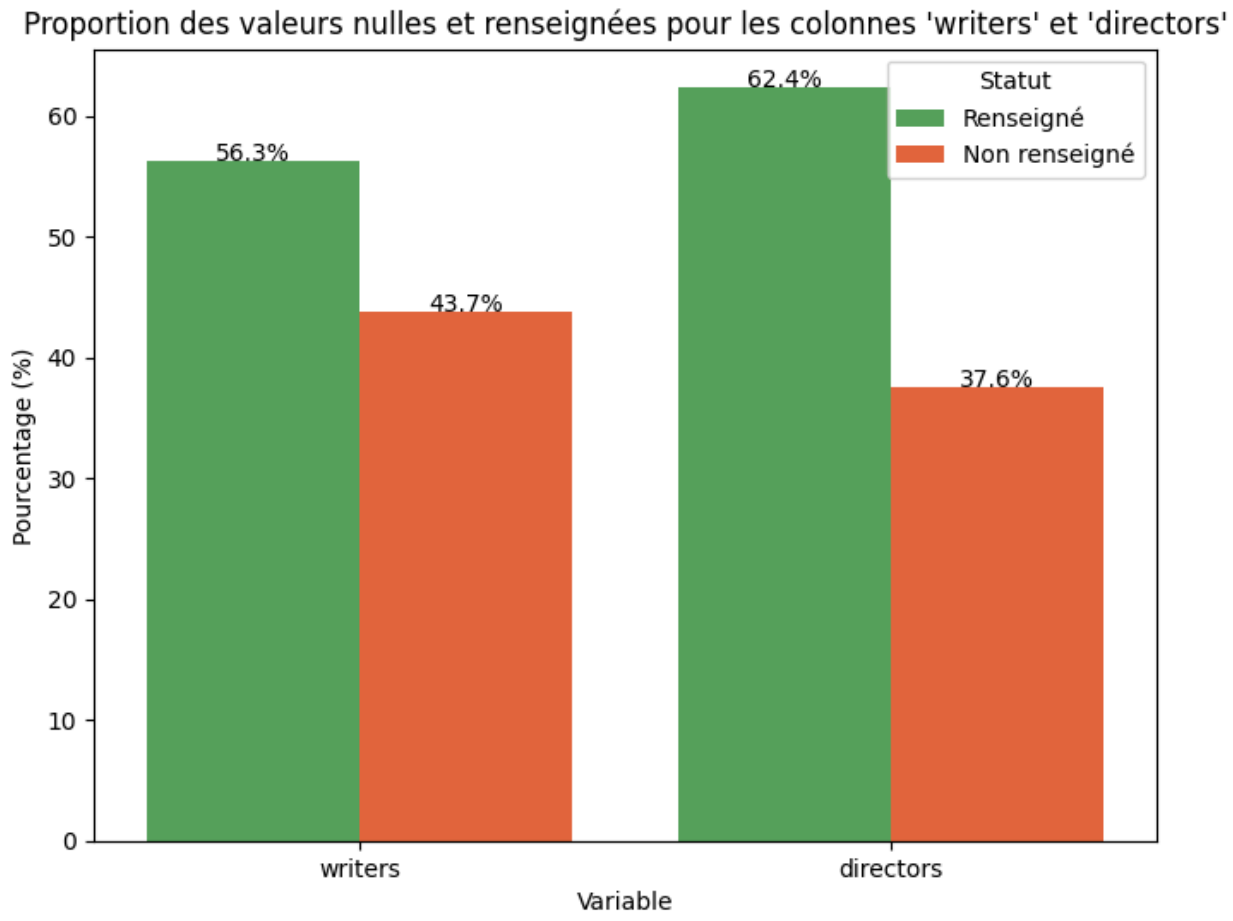
Nous allons extraire les colonnes ***originalTitle***, ***endYear***, ***runtimeMinutes*** et ***startYear*** du jeu de données, filtrer uniquement les lignes où la colonne ***titleType*** contient le mot "movie", convertir la colonne ***isAdult*** en un type de genre dans la colonne genres, après un changement de typage préalable, extraire les doublons de la colonne ***primaryTitle***.

Analyse du dataset Title crew

Le jeu de données contenait 3 colonnes et 10 143 819 lignes. Voici un aperçu des première lignes du dataset :

tconst	directors	writers
tt0000001	nm0005690	NaN
tt0000002	nm0721526	NaN

Visualisation des valeurs nulles pour les colonnes “writers” et “directors” :



Observation :

Nous avons relevé que ce jeu de données n'apporte potentiellement pas de plus-value pour le modèle de ML. Les colonnes censées fournir des informations supplémentaires sur les directeurs et les auteurs des films contiennent énormément de valeurs manquantes. Par exemple :

- La colonne ***directors*** a plus de 38% de valeurs manquantes.
- La colonne ***writers*** a plus de 44% de valeurs manquantes.

Cette quantité importante de valeurs manquantes limite l'utilité de ces colonnes dans notre analyse.

Décision pour le prétraitement des données :

En conséquence, nous avons décidé de ne pas inclure ce dataset dans la phase de prétraitement des données.

Analyse du dataset Title episode

Le jeu de données contenait 4 colonnes et 8 258 774 lignes. Voici un aperçu des première lignes du dataset :

tconst	parentTconst	seasonNumber	episodeNumber
tt0041951	tt0041038	1.0	9.0
tt0042816	tt0989125	1.0	17.0

Observation :

Nous avons constaté que le pourcentage des ***tconst*** des lignes de la colonne ***titleType*** contenant la valeur ***tvEpisode*** du dataset ***title.basics*** étaient présents à 99.97% dans ce jeu de données. En outre, cela nous a permis de confirmer que le jeu de données ***title.episode*** contenait exclusivement des informations sur les épisodes de séries de l'ensemble IMDb.

Décision pour le prétraitement des données :

Lors de la phase de prétraitement des données, nous extrairons directement les données liées aux épisodes des séries à partir de ***title.basics***, sans avoir besoin du jeu de données ***title.episode***.

Analyse du dataset Title principals

Le jeu de données contenait 6 colonnes et 85 883 465 lignes. Voici un aperçu des première lignes du dataset :

tconst	ordering	nconst	category	job	characters
tt0000001	1	nm1588970	self	NaN	["Self"]
tt0000001	2	nm0005690	director	NaN	NaN

Observation :

Nous avons observé que le jeu de données ne contenait pas de valeurs dupliquées, les colonnes ***job*** et ***characters*** possèdent plus de 50% de valeurs manquantes, tandis que la colonne ***ordering*** n'apporte pas de plus-value pour la suite. Néanmoins, la colonne ***category*** pourrait être remodelée, pour inclure les noms des participants aux films dans les prédictions du modèle de machine learning, en particulier les acteurs et les producteurs.

Décision pour le prétraitement des données :

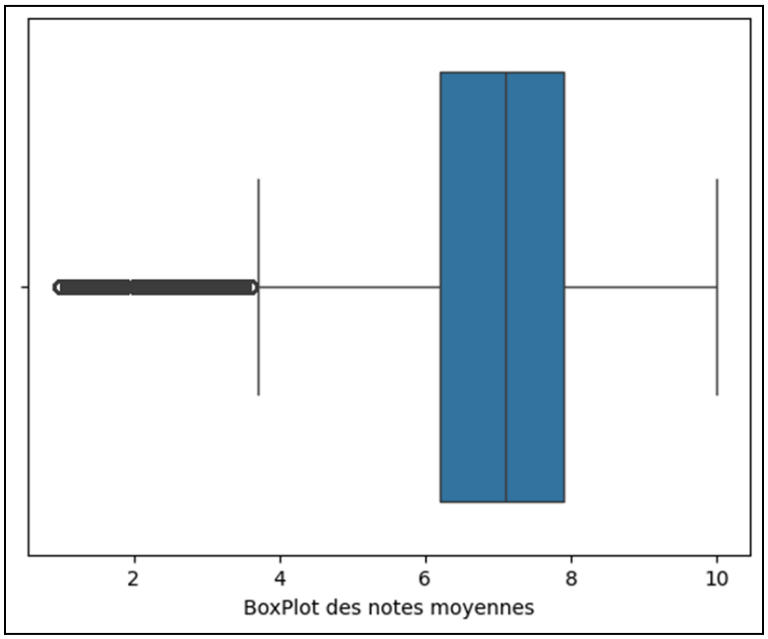
Lors de la phase de prétraitement des données, nous extrairons les colonnes ***job***, ***characters*** et ***ordering***, puis effectuerons un remodelage (pivot) de la colonne category pour mieux structurer les données. Ensuite, nous extrairons les nouvelles colonnes inutiles ou qui contiennent beaucoup de valeurs manquantes telles que : ***archive_footage, archive_sound, casting_director, cinematographer, composer, production_designer, self***.

Analyse du dataset Title rating

Le jeu de données contenait 3 colonnes et 1 439 968 lignes. Voici un aperçu des première lignes du dataset :

tconst	averageRating	numVotes
tt0000001	5.7	2058
tt0000002	5.7	276

Visualisation de la distribution de la note moyenne des films :



Observation :

Nous avons relevé que le jeu de données ne contenait pas de valeurs dupliquées et ne possédait pas de valeurs manquantes, et qu'il nous permet de connaître les notes attribuées aux films par les utilisateurs.

En outre, la visualisation a permis de constater que les notes allaient de 0 à 10 avec une moyenne autour de 7, et que 50% des notes étaient entre 6 et 8. En conséquence, il n'y a pas à priori de valeurs aberrantes détectables et un taux faible de notes inférieures à 4.

Décision pour le prétraitement des données :

Lors de la phase de prétraitement des données, aucune action spécifique n'est nécessaire pour ce dataset jusqu'à la phase de fusion entre IMDb et MovieLens.

Récapitulatif de l'exploration des données IMDb

Nom dataset	Nombre de col	Nombre de ligne	Colonne à extraire	Colonne à garder	Commentaire
Name basics	6	13 508 117	birthYear, deathYear, primaryProfession, knownForTitles	nconst, primaryName	A retravailler
Title akas	8	48 459 150	/	/	A exclure du ML
Title basics	9	10 790 736	originalTitle, isAdult, startYear, endYear, runtimeMinutes	tconst, titleType, primaryTitle, genres	A retravailler titleType : uniquement: "movie"
Title crew	3	10 143 819	/	/	A exclure du ML
Title episode	4	8 258 774	/	/	A exclure du ML
Title principals	6	85 883 465	job, characters, ordering	tconst, nconst, category	A retravailler
Title rating	3	1 439 968	/	tconst, averageRating, numVotes	

C. Exploration des données MovieLens 20M

Chargement des données

Les datasets présents dans les données Movielens 20M n'ont pas posé de problème particulier lors de leur chargement, les données étaient lisibles et ne comportaient pas d'erreurs particulières.

Analyse du dataset Movies

Le jeu de données a 3 colonnes et 27 278 lignes. Voici un aperçu des première lignes du dataset :

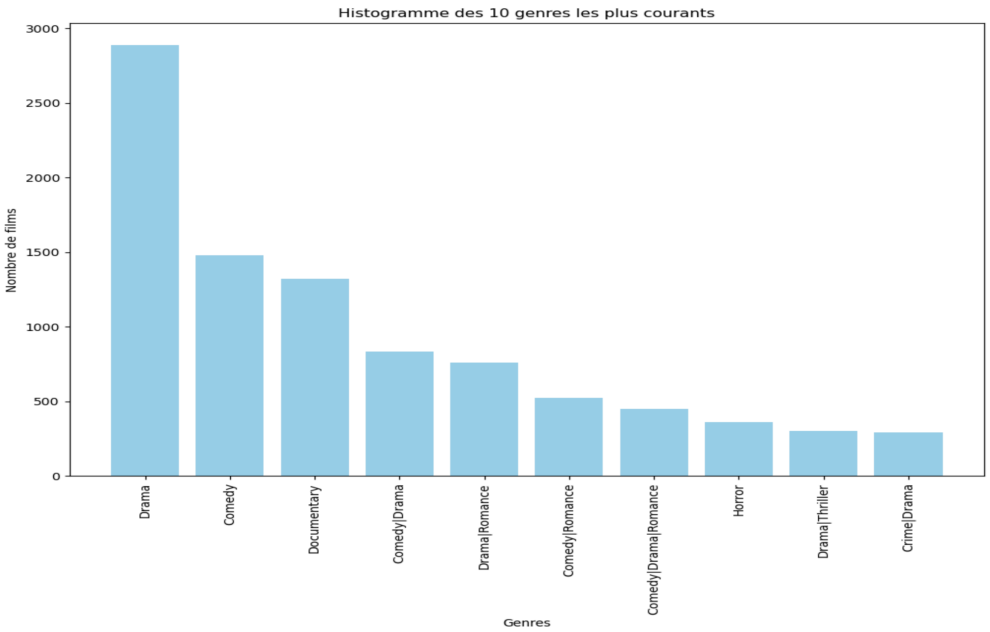
movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	Father of the Bride Part II (1995)	Comedy

Observation :

Le dataset ne contient aucune valeur manquante ni aucune valeur dupliquée.

Décision pour le prétraitement des données :

Ce jeu de données contient des éléments essentiels pour notre système de données, nous le gardons donc pour le prétraitement des données. La colonne ***movieId*** nous permettra d'ailleurs de le relier aux autres datasets de movielens.



Analyse du dataset Tags

Le jeu de données a 4 colonnes et 465 564 lignes. Voici un aperçu des première lignes du dataset :

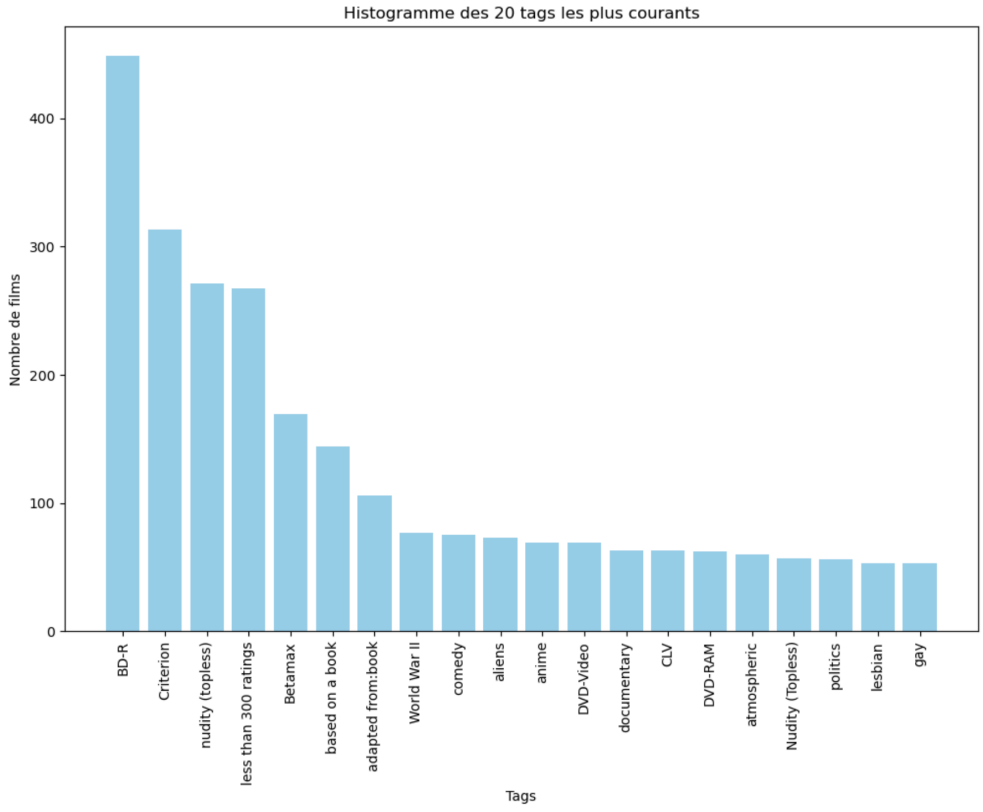
userId	movieId	tag	timestamp
18	4141	Mark Waters	1240597180
65	208	dark hero	1368150078
65	353	dark hero	1368150079
65	521	noir thriller	1368149983
65	592	dark hero	1368150078

Observation :

Les colonnes ***userId***, ***movieId*** et ***timestamp*** ne contiennent aucune valeur nulle. La colonne ***tag*** contient 16 valeurs non renseignées, le taux de remplissage de cette variable est de 99,99%.

Décision pour le prétraitement des données :

Lors de la phase de prétraitement nous procéderons à la suppression de la colonne ***timestamp*** qui ne nous sera pas utile pour la suite. Nous tenterons également de réduire le nombre de ligne en ne gardant qu'une ligne par film avec le tag le plus utilisé sans tenir compte des userId.



Analyse du dataset Rating

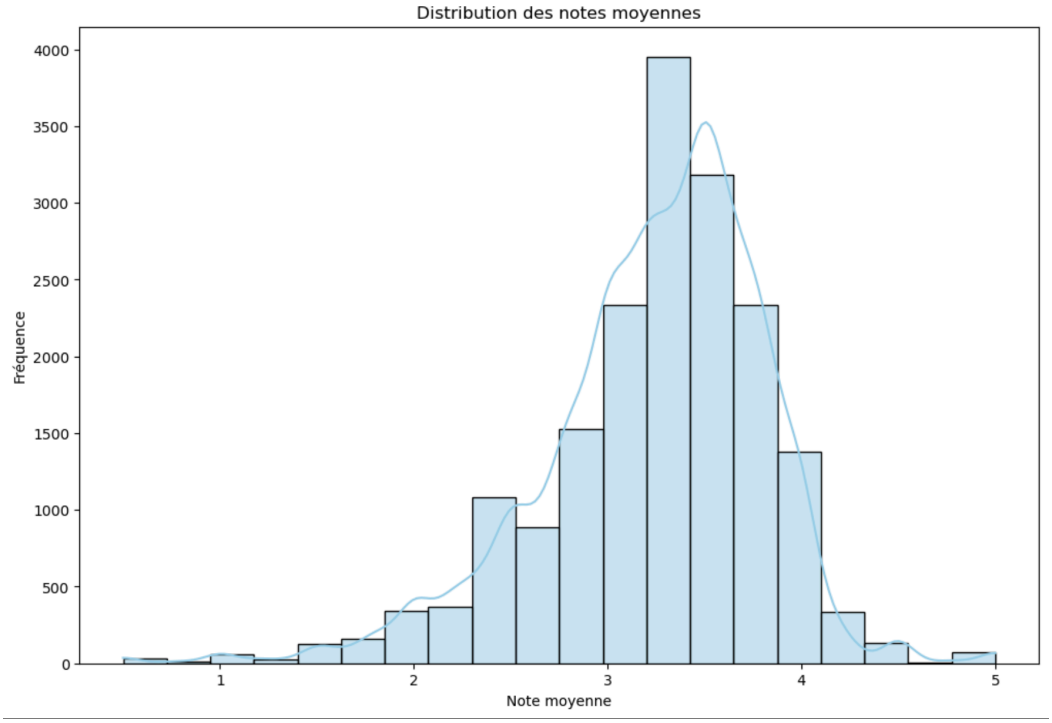
Le jeu de données a 4 colonnes et 20 000 263 lignes. lignes. Voici un aperçu des première lignes du dataset :

userId	movieId	rating	timestamp
1	2	3.5	1112486027
1	29	3.5	1112484676
1	32	3.5	1112484819
1	47	3.5	1112484727
1	50	3.5	1112484580

Le dataset ne contient aucune valeur manquante ni aucune valeur dupliquée.

Décision pour le prétraitement des données :

Lors du prétraitement comme précédemment nous supprimons la colonne ***timestamp***. Nous ferons également une moyenne des notes obtenues par film pour réduire également le nombre de lignes et améliorer la lisibilité de notre dataset final.



Analyse du dataset Links

Le jeu de données a 3 colonnes et 27 278 lignes. Voici un aperçu des première lignes du dataset :

movieId	imdbId	tmdbId
1	114709	862.0
2	113497	8844.0
3	113228	15602.0
4	114885	31357.0
5	113041	118662

Nous avons constaté que la colonne **tmdbId** comporte 252 valeurs manquantes soit 0,92%.

Décision pour le prétraitement des données :

Lors du prétraitement nous supprimerons la colonne **tmdbId** qui ne nous servir pas, tandis que la colonne ***movieId*** pourra être reliée à nos autres datasets de movielens et la colonne ***imdbId*** sera à transformer afin d'avoir les identifiants complets et pouvoir relier notre dataset movielens avec le dataset imdb.

Analyse du dataset Genome-scores

Le jeu de données a 3 colonnes et 11 709 768 lignes. Voici un aperçu des première lignes du dataset :

movieId	tagId	relevance
1	1	0.02500
1	2	0.02500
1	3	0.05775
1	4	0.09675
1	5	0.14675

Le dataset ne contient aucune valeur manquante ni aucune valeur dupliquée.

Décision pour le prétraitement des données :

Lors du prétraitement nous n'utiliserons pas ce dataset au vu de son contenu si voulons conserver une ligne par film nous ne garderons pas tous les tags de chaque film.

Analyse du dataset Genome-tags

Le jeu de données contient 2 colonnes et 1 128 lignes. Voici un aperçu des première lignes du dataset :

tagId	tag
1	007
2	007 (series)
3	18th century
4	1920s
5	1930s

Le dataset ne contient aucune valeur manquante ni aucune valeur dupliquée.

Décision pour le prétraitement des données :

Lors du prétraitement nous n'utiliserons pas ce dataset non plus puisque nous utiliserons les tags du dataset tags comme indiqué précédemment.

Récapitulatif de l'exploration des données MovieLens 20M

Nom dataset	Nombre de col	Nombre de ligne	Colonne à extraire/supprimer/modifier	Colonne à garder	Commentaire
Movies	3	27 278	/	movieId, title, genres	Base de prétraitement
Tags	4	465 564	timestamp, userId, tag	movieId	A retravailler
Rating	4	20 000 263	timestamp, userId, rating	movieId	A retravailler
Links	3	27 278	tmdbId, imdId	movieId	A retravailler
Genome-scores	3	11 709 768	/	/	A exclure du ML
Genome-tags	2	1 128	/	/	A exclure du ML

V. Prétraitement des données

A. Introduction

Ensuite, nous avons poursuivi avec la phase de prétraitement des données. Cette étape consiste à préparer les données et à garantir la pertinence des données en vue de construire le modèle de machine learning. Le prétraitement comprend diverses opérations telles que le nettoyage des données, la modification du typage des colonnes, la transformation de colonnes, etc.

Le prétraitement des données a été réalisée au sein de plusieurs Jupyter Notebooks, chacun dédié à une phase spécifiques :

IMDb datasets

Fichier d'évaluation	Fichiers source
000_dstrec_preparation_title_basics.ipynb	title.basics.tsv | title.ratings.tsv
001_dstrec_preparation_name_basics.ipynb	name.basics.tsv
002_dstrec_preparation_title_principals.ipynb	title.principals.tsv title_basics_tconst.csv
003_dstrec_transformation_title_principals.ipynb	title.principals.csv | name.basics.csv
004_dstrec_transformation_imdb.ipynb	title.principals.csv | title.basics.csv

MovieLens 20M datasets

Fichier d'évaluation	Fichiers source
005_dstrec_transformation_movielens.ipynb	movies.csv | tags.csv | ratings.csv | links.csv
006_dstrec_transformation_movielens_with_directors.ipynb	movielens.csv

Dstrec datasets

Fichier d'évaluation	Fichiers source
007_dstrec_transformation_des_donnees.ipynb	imdb.csv | movielens.csv

B. Prétraitement des données IMDb

Chargement des données

Lors du prétraitement des données IMDb, nous avons rencontré la même problématique à celle de la phase d'exploration en raison de la quantité massive de données à traiter. Cependant, contrairement à la phase précédente, nous ne pouvions pas utiliser le paramètre ***nrows*** car nous devions traiter l'ensemble des données disponibles.

Considération technique :

Nous avons envisagé d'utiliser une base de données pour gérer les opérations, mais nous avons rapidement constaté que dès le premier fichier cela prenait énormément de temps. Nous avons également envisagé d'utiliser la librairie Dask, qui est capable de gérer des datasets plus grands que la mémoire disponible et d'exécuter des tâches en parallèle. Cependant, les opérations de réduction de données prévues allaient diminuer considérablement le volume des données. De plus, nous avions constamment besoin de visualiser les résultats, ce qui est plus simple et pratique avec Pandas. Pandas est également mieux optimisée pour des jeux de données qui peuvent être entièrement chargés en RAM.

Solution adoptée :

Nous avons finalement opté pour l'utilisation de Pandas avec des boucles utilisant le paramètre ***chunksize***. Cette approche permet de charger progressivement de grandes quantités de données sans dépasser la mémoire disponible. Néanmoins, entre chaque notebook, il nous a été essentiel de réactualiser le kernel pour libérer la mémoire et pouvoir continuer les opérations dans les notebooks suivants, afin d'éviter les problèmes de mémoire saturée et pour garantir un flux de travail fluide.

Préparation du dataset Title basics

Chargement des données :

Pendant le chargement des données du jeu de données ***title.basics***, nous avons utilisé la fonction ***clean_and_filter_data***. Celle-ci a supprimé les colonnes à extraire identifiées lors de la phase d'exploration des données, elle a transformé le typage de la colonne ***isAdult***, afin qu'elle puisse par la suite l'insérer en un type de genre dans la colonne ***genres***. Puis la fonctionne a supprimé la colonne ***isAdult***. Ensuite, elle a filtré les lignes pour ne conserver que celles où la colonne ***titleType*** avait la valeur ***movie***. Et enfin elle a fusionné le dataset ***title.basics*** avec celui de ***title.ratings*** afin d'inclure les informations de notation des films.

Suppression des valeurs manquantes et dupliquées :

Nous avons ensuite procédé à la suppression des valeurs manquantes pour garantir la qualité des données et avons géré les doublons en utilisant le nombre de votes (***numVotes***) comme critère de filtrage, conservant uniquement la ligne avec le plus grand nombre de votes en cas de doublon.

Vérification et sauvegarde des données :

Après ces étapes de nettoyage, nous avons effectué des analyses supplémentaires pour nous assurer que toutes les transformations et suppressions avaient été correctement appliquées et que le dataset était prêt pour la transformation des données. Puis, nous les avons sauvegardées dans un fichier CSV.

En outre, nous avons créé un fichier CSV supplémentaire contenant uniquement la colonne ***tconst*** avec les mêmes lignes que le dataset principal. Cette approche permet d'optimiser les performances et de réduire l'utilisation de la RAM pour les prochaines étapes de préparation des données des autres datasets.

Préparation du dataset Name basics

Chargement des données :

Lors du chargement des données du jeu de données ***name.basics***, nous avons décidé d'exclure les colonnes identifiées comme à extraire lors de la phase d'exploration des données : ***birthYear, deathYear, primaryProfession, knownForTitles***.

Affichage des premières lignes du nouveau dataset :

nconst	primaryName
nm0000001	Fred Astaire
nm0000002	Lauren Bacall

Sauvegarde des données :

Après la suppression des colonnes non désirées pour le reste des opérations, nous avons sauvegardé le nouveau jeu de données dans un fichier CSV.

Ces étapes nous ont permis de simplifier le dataset ***name.basics*** en ne conservant que les informations essentielles (***nconst*** et ***primaryName***), facilitant ainsi les opérations futures et optimisant l'efficacité du traitement des données.

Préparation du dataset Title principals

Chargement des données :

Lors du chargement des données du jeu de données ***title.principals***, nous avons chargé uniquement les lignes correspondantes aux lignes du dataset ***title_basics_tconst*** contenant la colonne ***tconst***.

Sauvegarde des données :

Après avoir filtré les données, nous avons sauvegardé ce nouveau jeu de données dans un fichier CSV. Cette sauvegarde permet d'assurer une utilisation cohérente et efficace de ces données filtrées pour les étapes ultérieures du processus.

Transformation du dataset Title principals

Chargement des données :

Après le chargement des données du jeu de données ***title.principals***, nous l'avons mergé avec le dataset ***name.basics*** via la colonne ***nconst***. Au cours de cette fusion, nous avons supprimé les colonnes ***ordering***, ***job*** et ***characters***, jugées inutiles pour la suite de l'analyse. Ensuite, nous avons également supprimé la colonne ***nconst*** qui n'était plus nécessaire après la fusion.

Affichage des premières lignes du nouveau dataset :

tconst	category	primaryName
tt0000009	actress	Blanche Bayliss
tt0000009	direc	Alexander Black

Remodelage (pivot) de la colonne category :

Nous avons remodelé la colonne ***category*** en utilisant les valeurs de la colonne ***primaryName***. Cette opération a permis de transformer les catégories en nouvelles colonnes, avec les noms des participants correspondants. Après cette transformation, nous avons analysé et vérifié les nouvelles colonnes créées à partir des valeurs de la colonne ***category***. Nous avons extrait les colonnes qui contenaient soit une grande quantité de valeurs manquantes, soit celles qui n'apportaient pas de plus-value.

Sauvegarde des données :

Puis, nous avons sauvegardé le nouveau jeu de données transformé dans un fichier CSV.

Transformation du dataset IMDb

Chargement des données :

Dans cette dernière étape des transformations du jeu de données d'IMDb, nous avons chargé et fusionné le dataset ***title.principals*** avec ***title.basics***, rassemblant ainsi toutes les données souhaitées pour l'union avec le jeu de données ***MovieLens***.

Affichage des premières lignes du nouveau dataset :

tconst	primaryTitle	genres	...	actor
tt0111161	The Shawshank Redemption	Drama	...	Tim Robbins
tt0468569	The Dark Knight	Action,Crime,Drama	...	Christian Bale

Sauvegarde des données :

Puis, nous avons sauvegardé le nouveau jeu de données transformé dans un fichier CSV.

C. Prétraitement des données MovieLens

Choix des datasets utilisés

Comme nous l'avons indiqué précédemment nous faisons le choix de ne pas tenir compte des datasets genome-scores.csv et genome-tags.csv qui ne nous semblent pas pertinents pour la construction de notre dataset final et qui ne pourront être reliés aux données Imdb. Nous travaillerons donc sur les datasets ***movies.csv, links.csv*** et ***ratings.csv***.

Préparation du dataset movies

Dans un premier temps nous nous occupons de la colonne ***title.*** Les valeurs de cette dernière sont composées comme suit : Titre du film (année), par exemple : Toy Story (1995).
Par souci de simplicité et pour que les titres correspondent entre le jeu de données Movielens et Imdb nous décidons de supprimer l'année dans de nos titres.

Dans un second temps, nous conservons le reste des données sans apporter de modification : ***movieId*** et ***genres.***

Préparation du dataset links

Nous commençons dans ce dataset par supprimer la colonne ***tmdbId*** qui d'une part contient des valeurs manquantes et qui d'autre part ne nous servira pas car nous n'avons pas de données liées à tmdb.

Concernant la colonne ***imdbId*** nous nous rendons compte qu'il y a un problème à résoudre. En effet les identifiants imdb sont normalement au format "*tt0000000*" (par exemple : tt0114709), or dans notre jeu de données les deux "t" et les "0" ont été supprimé pour ne garder que les chiffres composant la fin de l'identifiant (exemple : 114709).

Afin que nos données comportent les véritables identifiants et que ceux-ci concordent avec le jeu de données Imdb nous transformons les valeurs de la colonne ***imdbId*** pour qu'elles respectent le format des identifiants imdb.

Préparation du dataset ratings

Pour commencer nous décidons de supprimer la colonne ***timestamp*** qui ne nous sera pas utile dans notre jeu de données et pour notre modèle.

Comme indiqué précédemment nous souhaitons avoir une ligne par film, afin de réaliser cela nous ne pouvons donc pas garder toutes les notes pour chaque utilisateur. Nous décidons donc de créer une colonne ***average_ratings*** comportant la note moyenne pour chaque film. Une fois cette colonne créée nous la rajoutons à notre dataframe ratings et nous arrondissons la note moyenne à une décimale.

Puis nous supprimons la colonne ***rating*** originale de notre dataframe.

La note étant désormais la même pour toutes les lignes d'un même film, nous supprimons la colonne ***userId*** qui nous ne sera plus utile puisque nous avons supprimer la colonne contenant la note attribuée par le user.

Afin d'être sûre de n'avoir qu'une ligne par film nous supprimons les doublons dans le dataframe ratings.

Préparation du dataset tags

Comme dans notre dataset ratings, nous supprimons la colonne ***timestamp.***

La colonne ***tag*** contient des valeurs manquantes, nous décidons simplement de supprimer les lignes contenant ces valeurs manquantes.

Dans la logique de nos précédents prétraitements nous souhaitons une ligne par film, afin d'aboutir à cela nous choisissons de rechercher le tag qui ressort le plus pour chaque film et de conserver ce dernier comme valeur dans une nouvelle colonne intitulée ***most_common_tag***. Puis nous supprimons les colonnes ***tag*** et ***userId*** du dataframe original avant de supprimer les doublons pour conserver une ligne par film.

Création du dataset movielens

Après avoir effectué les différentes phases de préparation ci-dessus, nous allons désormais passer à la préparation de notre dataset pour les données ***movielens***.

Nous procédons à la fusion de nos quatre dataframes nettoyés via la colonne ***movieId*** communes à ces différents datasets. Nous déterminons d'ailleurs ***movieId*** comme index de notre dataset movielens.

Le jeu de données comporte 659 doublons dans la colonne title, nous décidons donc de supprimer ces doublons.

Voici un aperçu des cinq premières lignes de notre dataset ainsi créé :

movieId	title	genres	imdbId	average_rating
1	Toy Story	Adventure|Animation|Children| Comedy|Fantasy	tt0114709	3.9
2	Jumanji	Adventure|Children|Fantasy	tt0113497	3.2
3	Grumpier Old Men	Comedy|Romance	tt0113228	3.2
4	Waiting to Exhale	Comedy|Drama|Romance	tt0114885	2.9
5	Father of the Bride Part II	Comedy	tt0113041	3.1

Le dataset movielens final contient 5 colonnes et 18352 lignes, il ne comporte aucune donnée manquante. Nous sauvegardons le tout et l'enregistrons dans le fichier movielens.csv.

Essai de webscraping : Ajout des réalisateurs (directors)

Par la suite, nous avons également tenté de faire du webscraping sur le site imdb.com afin d'ajouter à movielens une colonne contenant les réalisateurs (directors) pour chaque film. En effet, nous considérons que le réalisateur d'un film peut être utile dans les critères de recommandations.

Nous remettons ci-dessous la fonction créée à cet effet :

```
header = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/91.0.4472.124 Safari/537.36'
}

def fetch_director(imdb_id):
    url = f"https://www.imdb.com/title/{imdb_id}/"
    try:
        response = requests.get(url, headers=header)
        if response.status_code != 200:
            return 'N/A'
        soup = BeautifulSoup(response.content, 'html.parser')
        director_tag = soup.find('a', href=lambda x: x and 'tt_ov_dr' in x)
        director_name = director_tag.text.strip() if director_tag else 'N/A'
        return director_name
    except requests.exceptions.RequestException as e:
        print(f"Error fetching data for IMDb ID {imdb_id}: {str(e)}")
        return 'N/A'

with ThreadPoolExecutor(max_workers=10) as executor:
    directors = list(executor.map(fetch_director, movielens_final['imdbId']))

movielens_final['Director'] = directors
```

Nous enregistrons ces résultats dans un nouveau fichier nommé movielens_with_directors.csv.

D. Prétraitement final des données

Au sein de ce prétraitement final notre objectif sera de nettoyer les deux datasets précédemment créés pour ensuite les fusionner et transformer nos données.

Ces étapes nous permettront ainsi d'obtenir notre jeu de données final, jeu qui sera utilisé pour notre modélisation.

Préparation des données de IMDb

Dans un premier temps nous affichons les données manquantes du dataset ***imdb*** pour avoir une vision des éléments qui devront être revus pendant ce prétraitement :

Nom de la colonne	Nombre de valeur(s) manquante(s)
tconst	0
titleType	0
primaryTitle	0
genres	59 214
averageRating	297 910
numVotes	297 910
actor	159 729
actress	194 922
composer	301 291
director	46 632
producer	204 624

Notre jeu de données actuelles comporte à ce stade 572 547 lignes et 11 colonnes.

Suppression des lignes dont la valeur averageRating est manquante :

Au regard de la taille de notre jeu de données, nous faisons le choix de supprimer les lignes dont la note moyenne est absente car cette note va être l'une des valeurs les plus importantes pour notre modèle de recommandation de films.

Suite à ce retrait, notre jeu de données présente désormais 274 637 lignes et 11 colonnes.

Renommage des colonnes :

Afin de prédire la suite de notre transformation des données nous renommons dès maintenant les colonnes du dataset imdb afin que les noms des colonnes correspondent au dataset movielens et soient compréhensibles de tous. Ainsi ***tconst*** devient ***imdbId*** et ***primaryTitle*** devient ***title***.

Suppression des colonnes titleType et composer :

Pour notre objectif de création d'un modèle de recommandation nous estimons que les colonnes ***titleType*** et ***composer*** ne nous seront pas utiles, nous décidons donc de supprimer directement ces colonnes.

Affichage des premières lignes du nouveau dataset :

imdbId	title	genres	...	producer
tt0111161	The Shawshank Redemption	Drama	...	Niki Marvin
tt0468569	The Dark Knight	Action,Crime,Drama	...	Christopher Nolan

Préparation des données de MovieLens

Tout comme pour ***imdb*** nous commençons par afficher les données manquantes de ce dataset afin d'avoir une vision des éventuels problèmes à résoudre :

Nom de la colonne	Nombre de valeur(s) manquante(s)
movieId	0
title	0
genres	0
imdbId	0
average_rating	0
most_common_tag	0

Notre jeu de données comporte à ce stade 18 352 lignes et 5 colonnes sans aucune valeur manquante.

Suppression des colonnes most_common_tag et movieId :

La colonne ***movieId*** ne nous servira plus car elle n'est pas commune avec les données ***imdb***, celle-ci nous a permis uniquement de regrouper nos données movielens à l'étape précédente.

Quant à la colonne ***most_common_tag*** nous faisons le choix de ne pas la garder car nous estimons que celle-ci ne nous servira pas dans le cadre de la modélisation.

Renommage des colonnes :

Comme pour les données ***imdb*** nous voulons que les colonnes soient claires et communes, nous procédons donc à la modification de la colonne ***average_rating*** en ***averageRating***.

Reformatage de la colonne averageRating :

Dans le jeu de données ***movielens*** nos notes moyennes sont sur 5 alors que dans le jeu de données ***imdb*** ces dernières sont sur 10. Dans un souci de clarté et de cohérence nous multiplions par deux les notes moyennes de ***movielens*** afin qu'elles soient également sur 10.

Affichage des premières lignes du nouveau dataset :

title	genres	imdbId	averageRating
Toy Story	Adventure|Animation|Children|Comedy|Fantasy	tt0114709	7.8
Jumanji	Adventure|Children|Fantasy	tt0113497	6.4

Fusion des deux jeux de données

Avant de procéder à la fusion finale de nos deux datasets nous avons effectué des tests afin de voir si des lignes (correspondant à un film) sont communes au deux jeux de données. Ceci nous a indiqué que seuls 2 343 lignes présentes dans ***movielens*** ne sont pas communes avec ***imdb***. Nous traiterons plus bas le sort de ces lignes.

Fusion des deux jeux de données et gestion des colonnes communes :

Nous procédons à la fusion des jeux ***imdb*** et ***movielens*** via la colonne ***imdbId***. Concernant les colonnes communes, sachant que la note ***imdb*** est reliée à un nombre de vote pour ne pas fausser les données nous garderons la note ***imdb***. La données ***IMDB*** étant la plus fiable en termes de recherche de film, nous décidons de prioriser la donnée ***imdb*** pour les films se trouvant dans les deux datasets pour les colonnes communes. Nous supprimons les colonnes que nous avions créées lors de la fusion.

Nous décidons également de supprimer les 2 343 lignes qui ne se trouvaient que dans le dataset ***movielens*** car ces dernières ne comportent pas de valeurs pour les autres colonnes de notre dataset et risqueraient donc de complexifier notre travail.

Vérification des doublons :

Nous constatons que nous n'avons pas de doublons après la fusion et les traitements effectués sur celle-ci.

Gestion des valeurs manquantes :

Une fois la fusion effectuée nous nous intéressons aux valeurs manquantes de notre jeu de données :

Nom de la colonne	Nombre de valeur(s) manquante(s)
imdbId	0
numVotes	0
actor	37 037
actress	49 541
director	4 323
producer	66 793
title	0
genres	8 526
averageRating	0

Au vu du grand nombre de lignes dans notre jeu de données, nous décidons tout simplement de supprimer les lignes contenant des valeurs manquantes afin d'avoir un dataset propre et sans valeur manquante afin que le modèle puisse fonctionner au mieux. Nous supprimons ainsi les lignes dont les valeurs sont manquantes dans les colonnes ***genres, director, actor, actress*** et ***producer***.

Notre jeu de données contient désormais 164 108 lignes et 9 colonnes.

Fusion des colonnes actor et actress :

Pour des raisons de simplicité et de lisibilité nous préférons regrouper les colonnes ***actor*** et ***actress*** sous la forme ***actor/actress*** afin d'avoir dans la même colonne l'acteur et l'actrice principale du film. On crée ainsi la colonne ***actor_actress*** et supprimons les colonnes de base.

Réorganisation du jeu de données :

Afin d'avoir un ordre de colonne cohérent et avant de procéder à la transformation de la colonne ***genres***, nous ré-organisons l'ordre de nos colonnes dans le dataset comme suit : ***'imdbId'***, ***'title'***, ***'genres'***, ***'averageRating'***, ***'numVotes'***, ***'director'***, ***'actor_actress'***, ***'producer'***.

Transformation de la colonne genres en booléens :

Afin de faciliter le travail de notre futur modèle de recommandation, nous transformons la colonne genres en booléens. Pour ce faire nous cherchons toutes les valeurs prises par la colonne genre en prenant en compte le fait que plusieurs genres peuvent être attribués à un même film en utilisant notamment les fonctions split pour séparer les genres et get_dummies pour créer les colonnes booléennes. Pour finir cette étape nous supprimons les colonnes qui nous ont permis de faire cette transformation.

Création du fichier final :

Nous enregistrons notre dataset final sous le nom dstrec.csv.

Affichage des premières lignes du dataset final :

imdbId	title	averageRating	numVotes	...	forAdult
tt0000009	Miss Jerry	5.4	212.0	...	0
tt0000574	The Story of the Kelly Gang	6.0	900.0	...	0

VI. Exploitation des données

A. Introduction

Dans cette phase d'exploitation des données, notre objectif principal était de tester et d'évaluer plusieurs modèles et types de systèmes de recommandation pour identifier celui qui correspondait le mieux à nos besoins. Cette étape nous a permis de comparer les performances des différents modèles, et de comprendre leurs mécanismes internes à travers des techniques d'interprétabilité.

L'exploitation des données a été réalisée au sein de plusieurs Jupyter Notebooks, chacun dédié à un modèle spécifique :

Modèle évaluée avec les librairies suivantes	**Fichiers d'évaluation**
Annoy (hybride)	000_dstrec_annoy_h_modelisation.ipynb
Autoencodeur (hybride)	001_dstrec_autoencodeur_h_modelisation.ipynb
TfidfVectorizer avec TruncatedSVD (hybride)	002_dstrec_tfidf_svd_h_modelisation.ipynb
TfidfVectorizer (collaboratif)	003_dstrec_collaboratif_modelisation.ipynb

B. Exploitation des données Autoencodeur

Test du modèle

Dans le cadre de notre projet sur les systèmes de recommandation, nous avons tenté d'élaborer un modèle hybride combinant les techniques de filtrage basées sur le contenu et le filtrage collaboratif.

Ce modèle utilise le deep learning et notamment autoencodeur. Notre objectif était de créer un système capable de fournir des recommandations à la fois précises et personnalisées, en exploitant la puissance du deep learning.

Nous avons cherché à évaluer la performance de ce modèle, en comparant ses résultats avec d'autres méthodes plus traditionnelles, afin de déterminer dans quelle mesure le deep learning pouvait améliorer l'efficacité des recommandations.

Chargement et Préparation des données :

Une fois les données chargées, nous avons entrepris plusieurs étapes de préparation pour optimiser leur utilisation dans notre modèle de recommandation.

Tout d'abord, nous avons effectué les mêmes étapes de prétraitement que celles utilisées pour le modèle Annoy, afin de garantir une cohérence entre les deux modèles pour les comparer entre eux. Cela implique la suppression de la colonne ***imdbId***, l'ajout de la colonne

weightedAverageRating, la conversion de toutes les colonnes pertinentes en entiers, et la sélection de la colonne ***title*** comme cible.

Réduction de la Dimensionnalité avec TruncatedSVD

Nous avons opté pour le même nombre de composantes pour la décomposition par ***TruncatedSVD*** que le modèle Annoy, c'est à dire 100 composantes.

Construction et entraînement de l'Autoencodeur

Après avoir déterminé le nombre optimal de composantes pour ***TruncatedSVD***, nous avons procédé à la construction du modèle basé sur des ***autoencodeurs***. Cette architecture comprenait des couches d'encodeurs et de décodeurs, permettant de réduire la dimensionnalité des données tout en capturant les principales caractéristiques pour la recommandation.

Afin de prévenir tout surapprentissage et d'améliorer l'efficacité de l'entraînement, nous avons intégré la technique ***d'EarlyStopping***. Cette méthode surveille la performance du modèle sur les données de validation et interrompt l'entraînement si les performances cessent de s'améliorer après un certain nombre d'époques. Ainsi, cela permet de conserver le meilleur modèle obtenu avant que le processus d'entraînement ne commence à stagner, garantissant une utilisation optimale des ressources de calcul et une performance maximale du modèle.

Evaluation du modèle

Après l'entraînement et l'application du modèle basé sur les ***autoencodeurs***, nous avons procédé à son évaluation pour vérifier sa performance sur les données de test, notamment sur le MSE (Mean Squared Error) et le Tess Loss. Et les résultats obtenus nous ont révélé que le modèle est performant, avec une faible perte sur les données de test et un MSE relativement bas.

Tests du Modèle :

Après avoir construit et optimisé notre modèle, nous avons procédé à une série de tests pour évaluer la qualité des recommandations générées. Malheureusement, les résultats obtenus n'ont pas répondu à nos attentes. Les recommandations étaient souvent incohérentes, et chaque nouvel entraînement du modèle produisait des suggestions de films de manière très aléatoire.

En outre, en comparaison avec le modèle Annoy, ce modèle s'est révélé plus lourd à charger et moins performant en termes de vitesse d'exécution et de stabilité des résultats.

Conclusion

En conséquence, bien que le deep learning présente des avantages théoriques, nous avons décidé de privilégier l'utilisation du modèle Annoy, qui s'est avéré plus rapide, plus léger, et capable de fournir des recommandations plus cohérentes et pertinentes.

C. Exploitation des données Tfidf avec SVD

Test du modèle

Dans le cadre de notre projet sur les systèmes de recommandation, nous avons tenté d'élaborer un modèle hybride combinant les techniques de filtrage basées sur le contenu et le filtrage collaboratif.

Ce modèle utilise notamment ***TfidfVectorizer*** et ***TruncatedSVD*** pour analyser et réduire la dimensionnalité des données relatives aux films. Notre objectif était de créer un système capable de fournir des recommandations précises et personnalisées aux utilisateurs.

Nous avons dans un premier temps fait deux tests de modèles qui n'étaient pas concluants. Le premier ne donnait pas de résultats assez bons et le second ne fonctionnait pas en raison d'un problème de mémoire. Nous avons tout de même laissé en commentaire, à la fin de notre fichier, les lignes de codes créées pour ces tests.

Installation et importation des librairies :

Nous avons préparé notre environnement de développement en installant les librairies nécessaires, en particulier pour le traitement des données et l'apprentissage automatique.

Chargement et Préparation des données :

Après avoir chargé nos données qui ont été nettoyées et préparées au préalable, nous avons également supprimé la colonne ***imdbId*** afin qu'elle n'influe pas sur notre modèle. Ce nettoyage était essentiel pour garantir l'intégrité des données utilisées dans les recommandations.

Configuration du Modèle TF-IDF et SVD :

Nous avons configuré ***TfidfVectorizer*** pour réduire les caractéristiques des descriptions textuelles à 5000 termes maximaux, afin de capturer l'essence du contenu sans surcharger le modèle de données inutiles. Ensuite, ***TruncatedSVD*** a été utilisé pour réduire le nombre de dimensions à 100 composantes principales, optimisant ainsi l'efficacité de la transformation des données.

Création des Poids des Variables :

Les poids ont été attribués à chaque variable pour différencier leur impact dans le calcul des recommandations. Les genres comme ***Action***, ***Adventure***, etc., ont reçu un poids de 3, reflétant leur importance dans la préférence des utilisateurs. D'autres attributs comme ***averageRating*** et ***director*** ont reçu des poids de 2, tandis que les autres colonnes ont reçu des poids de 1, pour moduler leur influence sur le système de recommandation. Ceci nous permet de déterminer selon nous les caractéristiques qui nous semblent les plus importantes à prendre en compte pour effectuer notre recommandation. Le titre quant à lui a reçu un poids de 0 puisqu'il ne doit pas intervenir dans la recommandation mais uniquement en tant que cible de sortie de notre modèle.

Préparation des Features :

Nous avons traité les colonnes du DataFrame en fonction de leur type. Pour les colonnes de type objet, nous avons appliqué TfidfVectorizer suivi de TruncatedSVD pour les transformer en un espace latent réduit. Pour les colonnes numériques, nous avons appliqué une pondération directe en multipliant les valeurs par leurs poids respectifs. Ces matrices transformées ont ensuite été concaténées en une seule matrice sparse.

Création du Modèle Hybride de Recommandation :

Nous avons développé une fonction ***get_recommendations*** pour calculer la similarité cosine entre les films et générer une liste de films recommandés basés sur un titre de film donné. Cette fonction récupère le vecteur de caractéristiques du film demandé et calcule sa similarité avec tous les autres films dans la base de données, renvoyant les 10 recommandations les plus proches.

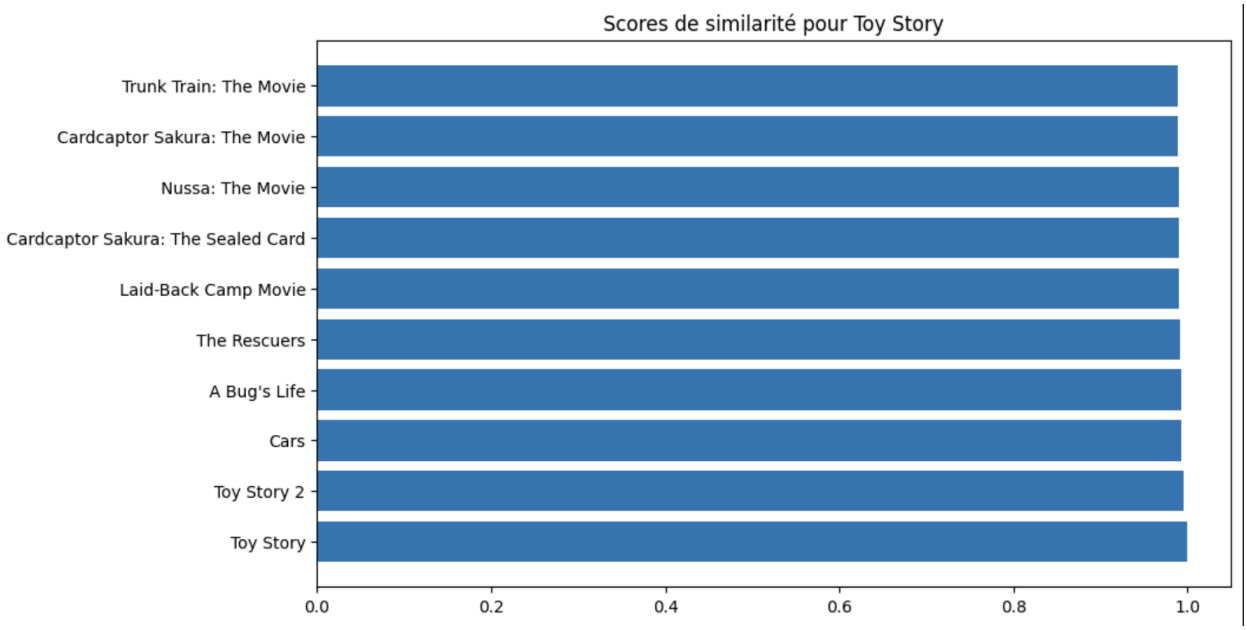
Tests du Modèle :

Des tests ont été réalisés pour démontrer la fonctionnalité de notre système de recommandation. Par exemple, des recommandations pour des films comme "Miss Jerry", "Toy Story" et "Inception" ont été générées pour illustrer la capacité du modèle à identifier des films pertinents basés sur différents genres et styles.

Interprétabilité du modèle

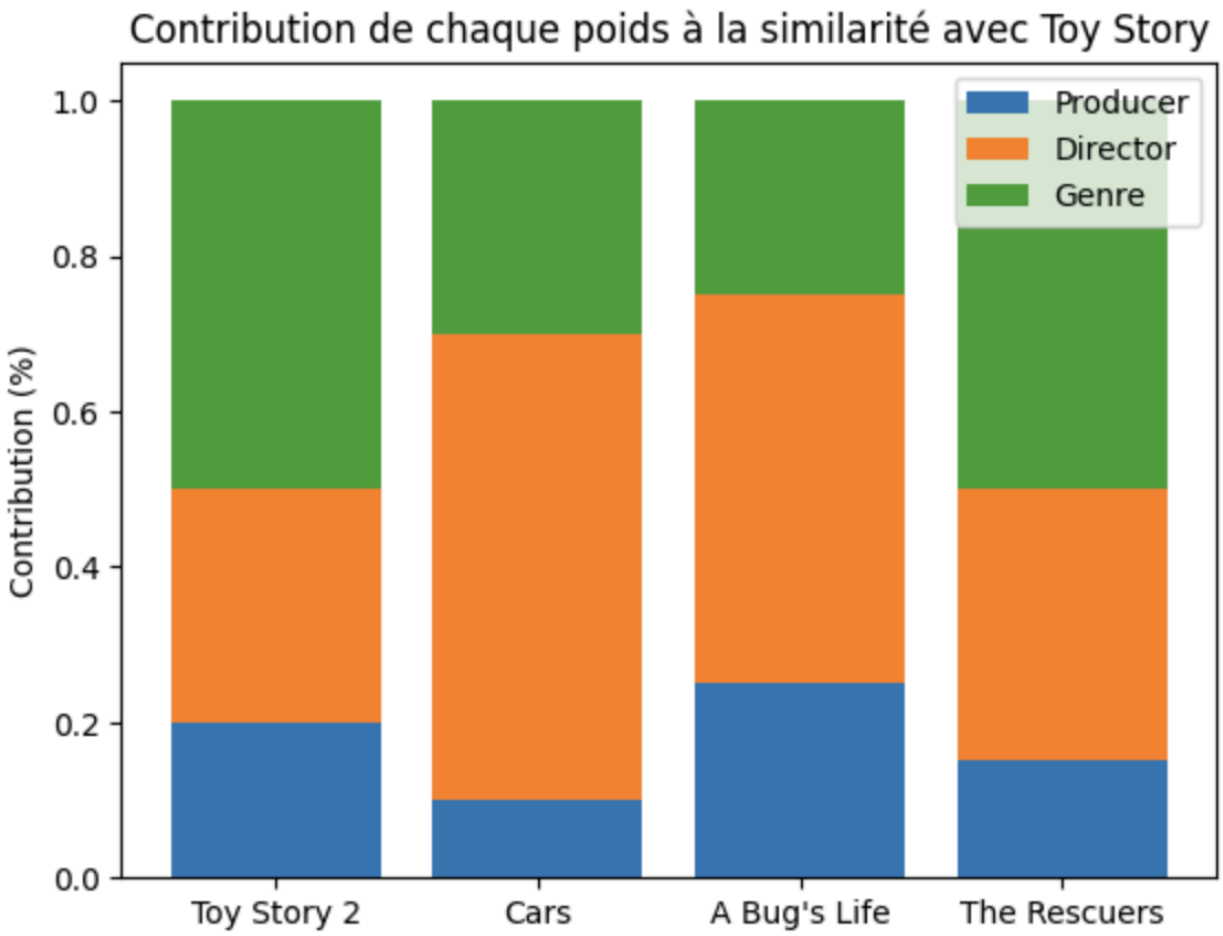
Nous nous sommes penchés sur l'interprétabilité du modèle pour nous assurer que les recommandations faites sont non seulement précises mais aussi compréhensibles et justifiables.

Nous avons commencé par explorer l'interprétabilité relative à la similarité des films. Pour cela, nous avons calculé la similarité cosinus entre les vecteurs caractéristiques des films. L'objectif était de comprendre pourquoi certains films sont recommandés lorsqu'on cherche des suggestions similaires à un film donné, en l'occurrence Toy Story. En sélectionnant un sous-ensemble de films potentiellement proches, nous avons pu observer que les scores de similarité étaient élevés pour des films comme Toy Story 2, Cars et A Bug's Life. Ces résultats sont conformes à nos attentes, car ces films partagent des éléments communs avec Toy Story, tels que le genre, le réalisateur, ou des thèmes narratifs similaires. Cette analyse confirme que le modèle propose des recommandations logiques basées sur des similarités tangibles.



Ensuite, nous nous sommes penchés sur l'interprétabilité relative aux poids attribués aux différentes caractéristiques des films, comme le réalisateur, le genre ou le producteur. Nous avons mené des tests en ajustant dynamiquement les poids de ces caractéristiques pour observer les modifications dans les recommandations. Par exemple, nous avons considérablement augmenté le poids du réalisateur pour voir comment cela influencerait les suggestions. Les résultats ont montré que cette modification faisait ressortir des films réalisés par le même réalisateur ou partageant des similarités stylistiques ou thématiques avec Toy Story. Cela démontre que le modèle est sensible aux variations de poids des caractéristiques, ce qui permet de mieux comprendre quels éléments influencent le plus les recommandations.

Pour rendre ces analyses encore plus transparentes, nous avons réalisé une visualisation des effets des différents poids sur les recommandations. À l'aide de barres empilées, nous avons illustré comment chaque caractéristique contribue aux recommandations pour des films comme Toy Story 2, Cars, A Bug's Life, et The Rescuers. Le graphique a clairement montré que, par exemple, le genre jouait un rôle prépondérant dans la recommandation de Toy Story 2 et The Rescuers, tandis que pour Cars et A Bug's Life, c'était davantage le réalisateur qui influençait les résultats. Cette analyse graphique nous a permis de confirmer que le modèle fonctionne de manière logique, en s'appuyant sur des caractéristiques significatives, ce qui renforce la cohérence et la pertinence des recommandations proposées.



En somme, l'interprétabilité de ce modèle de recommandations nous a permis de valider que les suggestions faites sont bien justifiées par les données et les caractéristiques des films. Cette approche assure que les utilisateurs peuvent avoir confiance dans les recommandations du système, car elles reposent sur des bases solides et compréhensibles.

Conclusion

Le modèle développé a réussi à offrir des recommandations personnalisées, et les ajustements des poids ainsi que la fonction de recommandation personnalisée ont joué un rôle clé dans l'optimisation des résultats. Les tests non concluants, quant à eux, ont mis en lumière des domaines nécessitant des ajustements, et nous envisageons de continuer à affiner et à améliorer notre système en testant notamment d’autres modèles.

D. Exploitation des données Tfidf, modèle collaboratif

Test du modèle

Dans le cadre de notre projet sur les systèmes de recommandation, nous avons comme objectif de créer un système capable de fournir des recommandations personnalisées aux utilisateurs.

Nous avons testé plusieurs modélisations basées sur le jeu de données disponible. Certaines modélisations n'ont pas donné de résultats concluants. L'un des modèles testés est un modèle basé sur le filtrage collaboratif.

Ce modèle utilise notamment la fonction TfidfVectorizer. Cette fonction convertit le texte en vecteurs de mots.

Installation et Importation des Librairies :

Nous avons préparé notre environnement de développement en installant les librairies nécessaires, en particulier pour le traitement des données et l'apprentissage automatique.

Chargement et Préparation des données :

Nous avons utilisé les données préalablement nettoyées et préparées dans le chapitre décrit précédemment.

Préparation des Features :

Nous avons traité les colonnes du DataFrame en fonction de leur type à l'exception de la colonne ***imdbId, title, averageRating et numVotes*** .

Pour les colonnes **actor_actress**, **producer** et ***director***, nous avons effectué un traitement des données afin que chaque nom soit unique. Nous avons également retraité les différents genres.

Création du Modèle :

Nous avons développé une fonction ***get_recommendations*** pour calculer la similarité COSINE entre les films et générer une liste de films recommandés basés sur le titre de film donné. Cette fonction récupère le vecteur de caractéristiques du film demandé et calcule sa similarité avec tous les autres films dans la base de données, renvoyant les 10 recommandations les plus proches.

Tests du Modèle :

Des tests ont été réalisés pour démontrer la fonctionnalité de notre système de recommandation. Nous avons pris comme exemple le film "***Star Trek***".

Interprétabilité du modèle

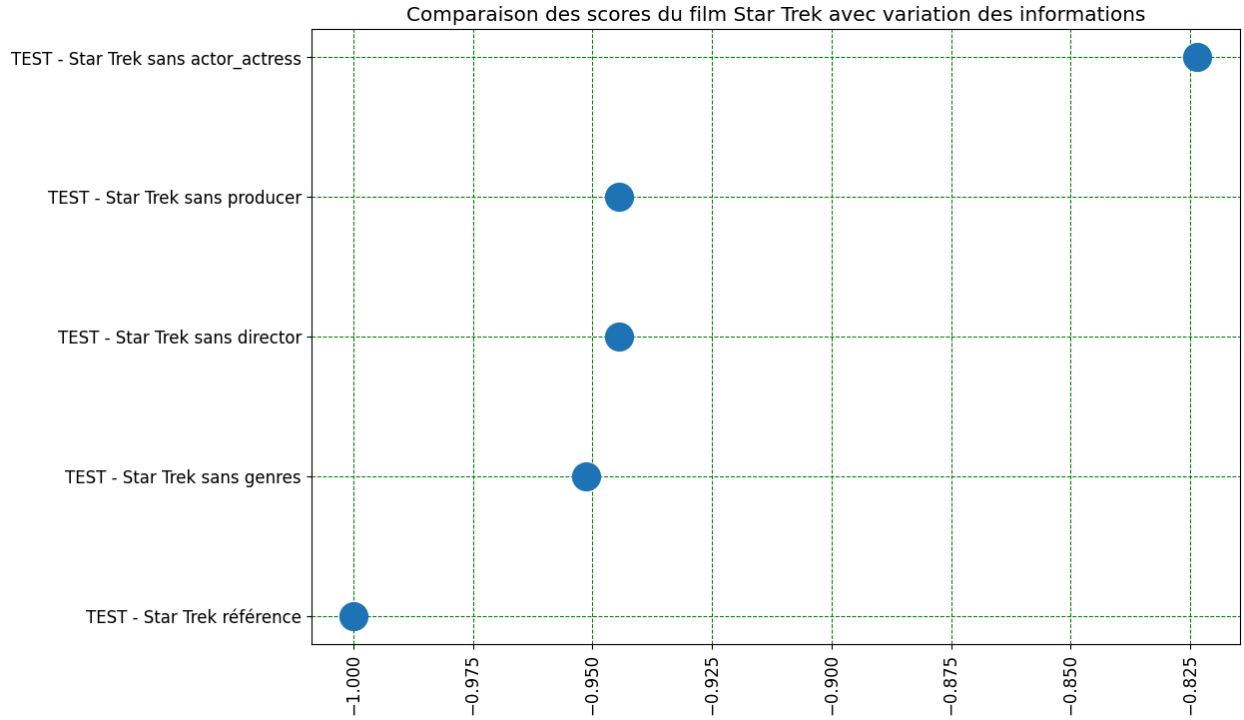
Nous nous sommes penchés sur l'interprétabilité du modèle pour nous assurer que les recommandations faites sont non seulement précises mais aussi compréhensibles et justifiables.

Nous avons donc voulu tester les features sur lesquelles s'appuient les recommandations.

Nous avons déterminé qu'il existe 4 catégories de features :

- producer
- director
- actor_actress
- genres (regroupement de l'ensemble des genres attribués au film)

Pour tester les différents éléments intervenant dans le calcul du score, nous avons fait varier les éléments pris en compte dans le calcul du score pour le film "Star Trek" : nous avons enlevé successivement le ***producer***, le ***director***, les ***actor_actress*** et les ***genres***. A chaque fois, nous avons calculé le score suite au changement de donnée.



Le modèle semble bien équilibré et utilise efficacement l'ensemble des éléments. Chacun d'entre eux intervient et contribue à l'élaboration du score de la recommandation. Toutefois, celui qui a un impact important sur le calcul du score est l'élément ***actor_actress***.

Nous avons ensuite effectué un second test, afin de calculer des scores avec certains éléments tout en comparant les variations. Pour cela, nous avons réalisé les étapes suivantes :

- Enregistrement des recommandations originales avec l'ensemble des informations
- Calcule des scores en supprimant la variable "genres"

Ci-joint les résultats :

	movie_indices_x	recommanded_title	score_x	movie_indices_y	score_y
0	97378	Star Trek Into Darkness	1.000000	97378	1.000000
1	123732	Star Wars: Episode VII - The Force Awakens	0.689292	123732	0.657455
2	125577	Star Trek Beyond	0.635425	125577	0.592822
3	106415	Super 8	0.626866	106415	0.612545
4	124099	Star Wars: Episode IX - The Rise of Skywalker	0.610929	124099	0.625374
5	124293	The Cloverfield Paradox	0.424393	124293	0.380634
6	143146	Mission: Impossible - Fallout	0.405170	143146	0.397311
7	141037	Overlord	0.399627	141037	0.355677
8	122189	Mission: Impossible - Rogue Nation	0.397849	122189	0.389570
9	83363	Cloverfield	0.395643	83363	0.367521
10	90227	Mission: Impossible - Ghost Protocol	0.387064	90227	0.378226

Nous constatons que retirer les genres modifie les scores et entraîne un impact sur les recommandations. Nous pouvons en conclure qu'avec les tests d''interprétabilité effectués que le modèle de recommandations s'appuie sur les informations et les caractéristiques des films pour faire ses recommandations. Les recommandations produites sont donc fiables et peuvent être expliquées.

Conclusion

Le modèle utilise l'ensemble des éléments dans son calcul de score de similarité. Le modèle développé a été testé sur différents films et il offre de bonnes recommandations.

E. Exploitation des données Annoy

Test du modèle

Dans le cadre de notre projet sur les systèmes de recommandation, nous avons tenté d'élaborer un modèle hybride combinant les techniques de filtrage basées sur le contenu et le filtrage collaboratif.

Ce modèle utilise notamment ***Annoy*** (Approximate Nearest Neighbors Oh Yeah) une libraire de python utilisée par Spotify pour leur recommandation musicale. Notre objectif était de créer un système capable de fournir des recommandations précises et personnalisées aux utilisateurs.

Chargement et Préparation des données :

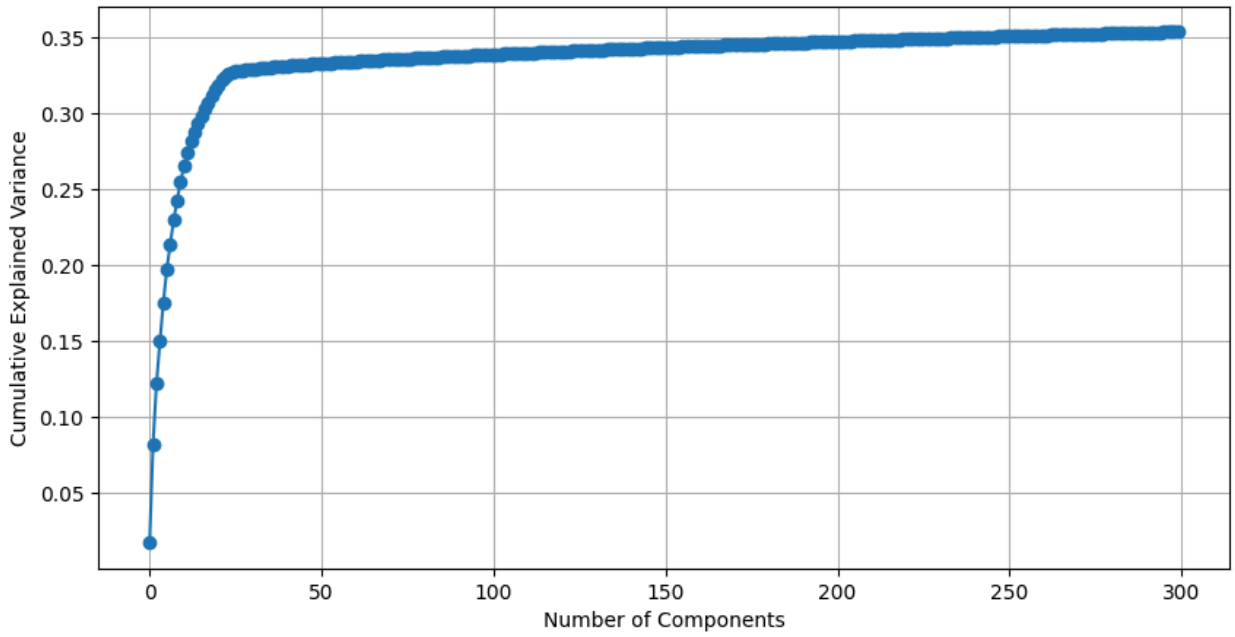
Une fois les données chargées, nous avons entrepris plusieurs étapes de préparation pour optimiser leur utilisation dans notre modèle de recommandation.

Tout d'abord, nous avons retiré du dataframe la colonne ***imdbId***, qui ne nous était plus utile pour la suite des opérations. Ensuite, nous avons ajouté une nouvelle colonne, ***weightedAverageRating***, afin de calculer la note moyenne pondérée (WAR) pour chaque film. Cette étape a permis d'ajouter une pondération aux notes des films, prenant en compte à la fois la moyenne des notes et le nombre de votes, ce qui permet d'obtenir une estimation plus fiable et équilibrée de la qualité d'un film. Par ailleurs, étant donné que l'algorithme Annoy ne supporte que des données de type entier, nous avons converti toutes les colonnes pertinentes en entiers.

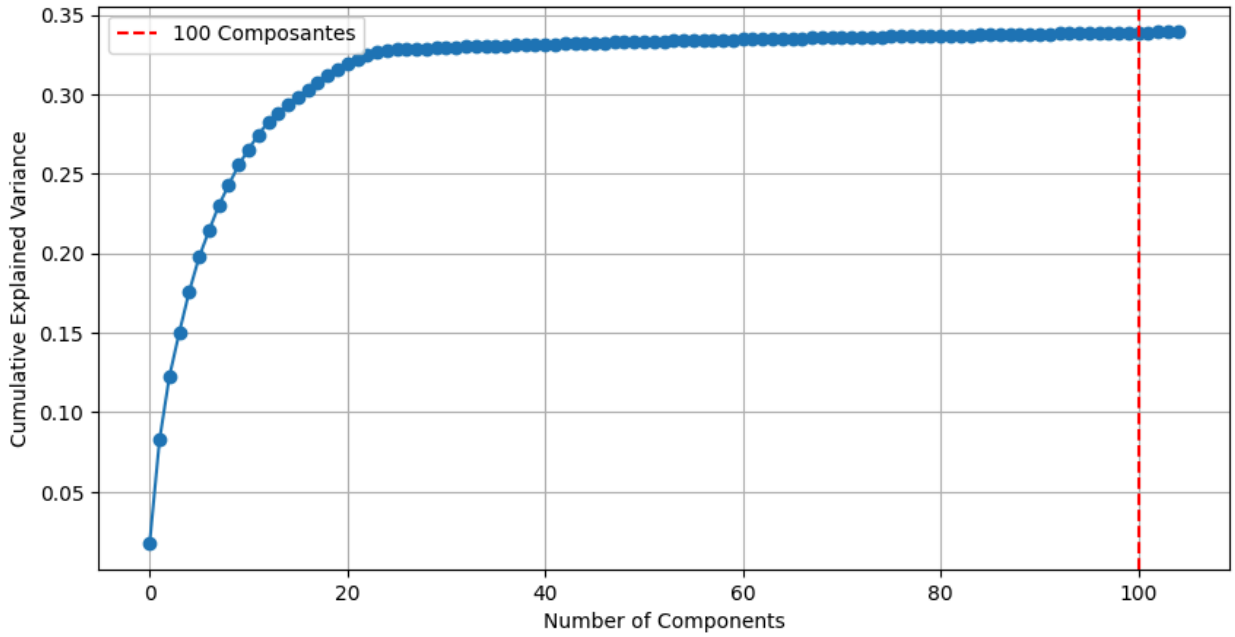
Enfin, nous avons sélectionné la colonne ***title*** comme cible, que nous avons stockée à côté du jeu de données de features de contenu.

Réduction de la Dimensionnalité avec TruncatedSVD

Après la normalisation des données, nous avons cherché à déterminer le nombre optimal de composantes pour la décomposition par ***TruncatedSVD***. Pour ce faire, nous avons créé un graphique en utilisant 300 composantes, afin d'analyser visuellement le point à partir duquel l'ajout de nouvelles composantes n'apporte plus de gain significatif en termes de variance expliquée.



L'analyse graphique a révélé que 100 composantes représentaient un choix optimal, car la variance expliquée cumulée atteignait 33,88% à ce stade, et commençait à stagner au-delà. En comparaison, avec 300 composantes, la variance expliquée n'était que légèrement supérieure, atteignant 35,40%. Cette stagnation après 100 composantes indique qu'au-delà de ce point, l'ajout de nouvelles composantes n'apporte qu'un gain marginal, justifiant ainsi notre choix de retenir 100 composantes pour le modèle.



Test des différentes Métriques et nombres d'arbres

Après avoir déterminé le nombre optimal de composantes pour ***TruncatedSVD***, nous avons procédé à des tests pour évaluer différentes combinaisons de métriques et de nombres d'arbres afin de déterminer les paramètres les plus appropriés pour notre modèle.

Nous avons expérimenté plusieurs métriques et quantités d'arbres pour identifier la configuration la plus performante sur ce jeu de données. Après de nombreux essais, il s'est avéré que l'utilisation de la métrique Euclidean combinée avec 10 arbres offrait les meilleurs résultats.

Tests du Modèle :

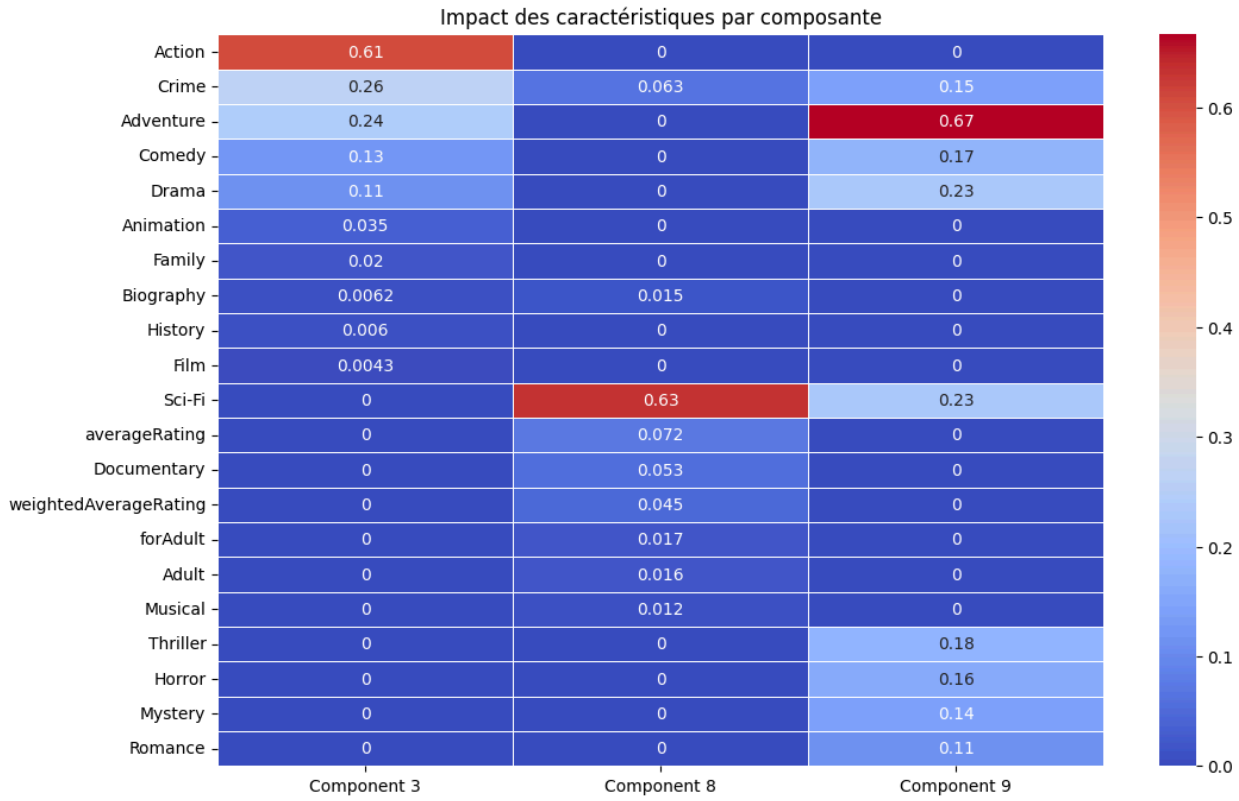
Après avoir construit et optimisé le modèle, nous l'avons soumis à des tests en l'appliquant à plusieurs films pour évaluer la qualité des recommandations générées. Les résultats obtenus étaient non seulement pertinents, mais également cohérents avec les attentes, ce qui a confirmé l'efficacité du modèle.

En plus de fournir des recommandations précises, le modèle s'est avéré être rapide à charger et à exécuter, ce qui le rend non seulement performant en termes de précision, mais aussi pratique à utiliser dans un environnement de production.

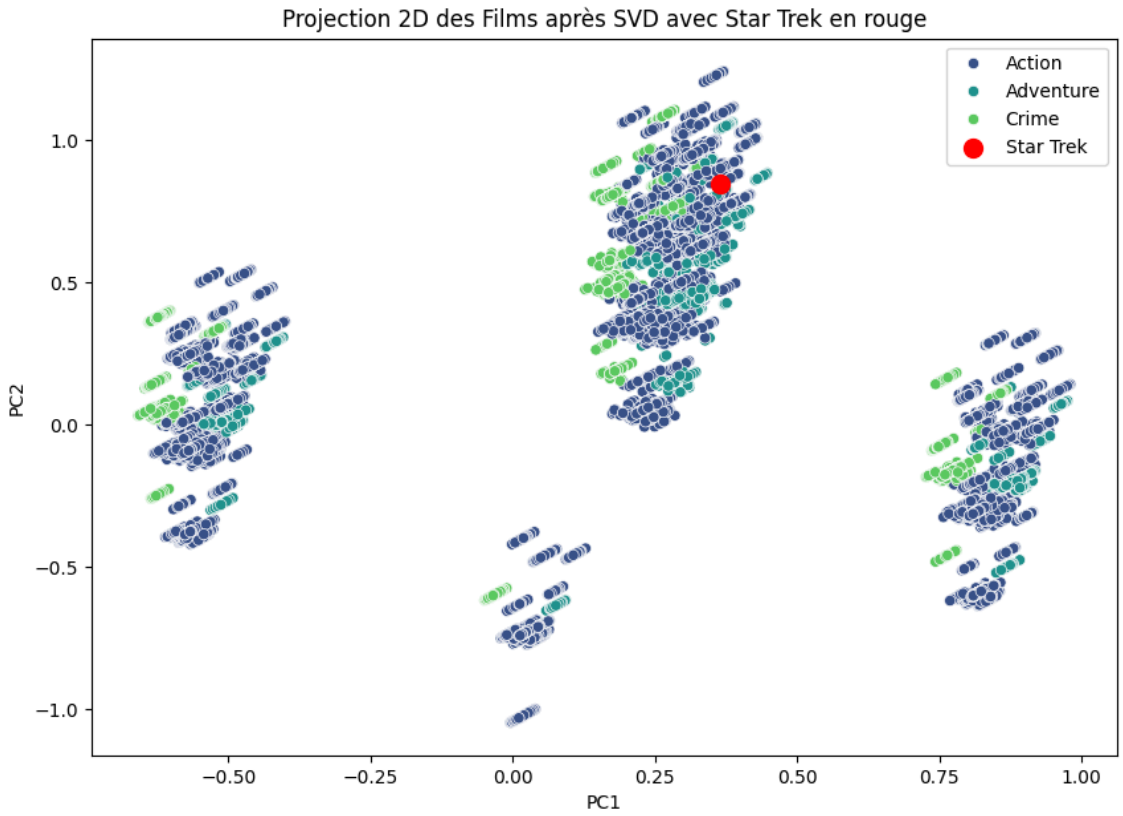
Interprétabilité du modèle

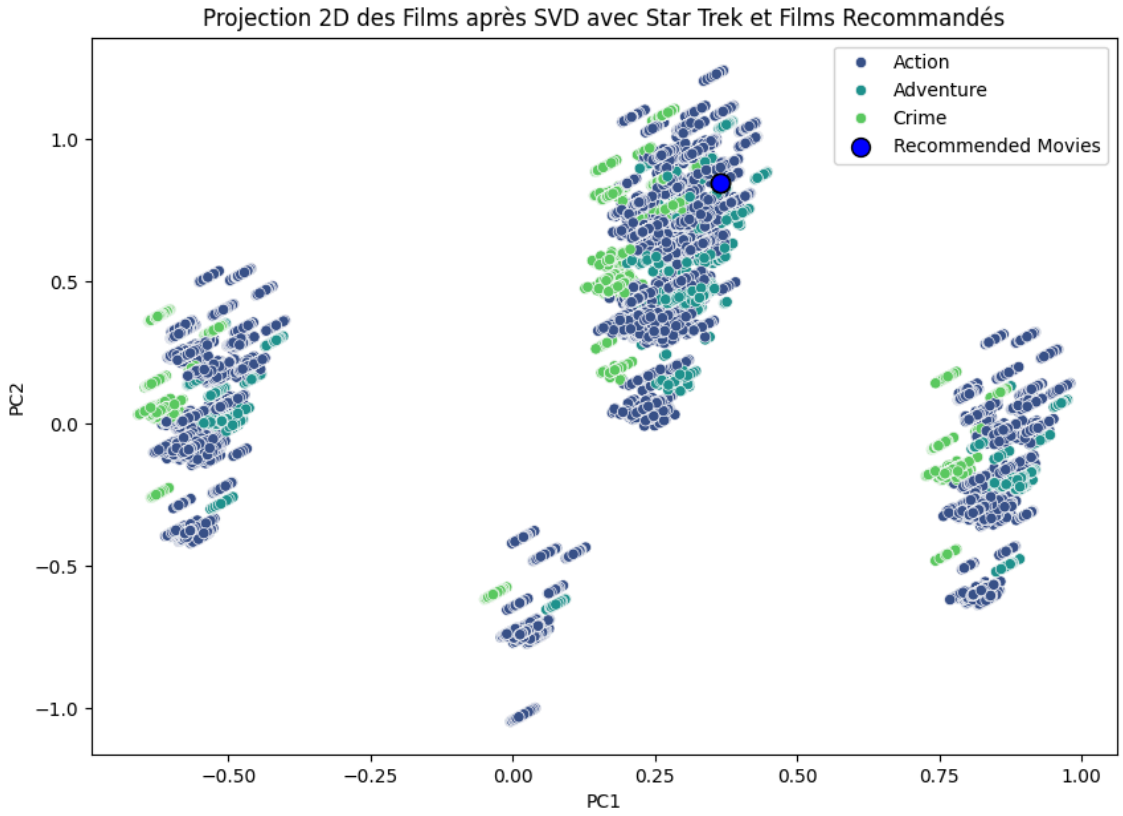
Ensuite, nous avons procédé à des analyses d'interprétation du modèle en utilisant **SHAP** (SHapley Additive exPlanations), en prenant le film ***Star Trek*** comme exemple. **SHAP** nous a permis de comprendre comment le modèle prenait ses décisions en identifiant les colonnes les plus influentes pour chaque recommandation.

L'analyse a révélé que le modèle organise les colonnes en composants principaux, et que certains de ces composants ont un impact significatif sur les recommandations en fonction des caractéristiques spécifiques du film cible. Par exemple, pour ***Star Trek***, certains composants étaient plus déterminants dans le processus de recommandation, influençant directement les films proposés en relation avec celui-ci.

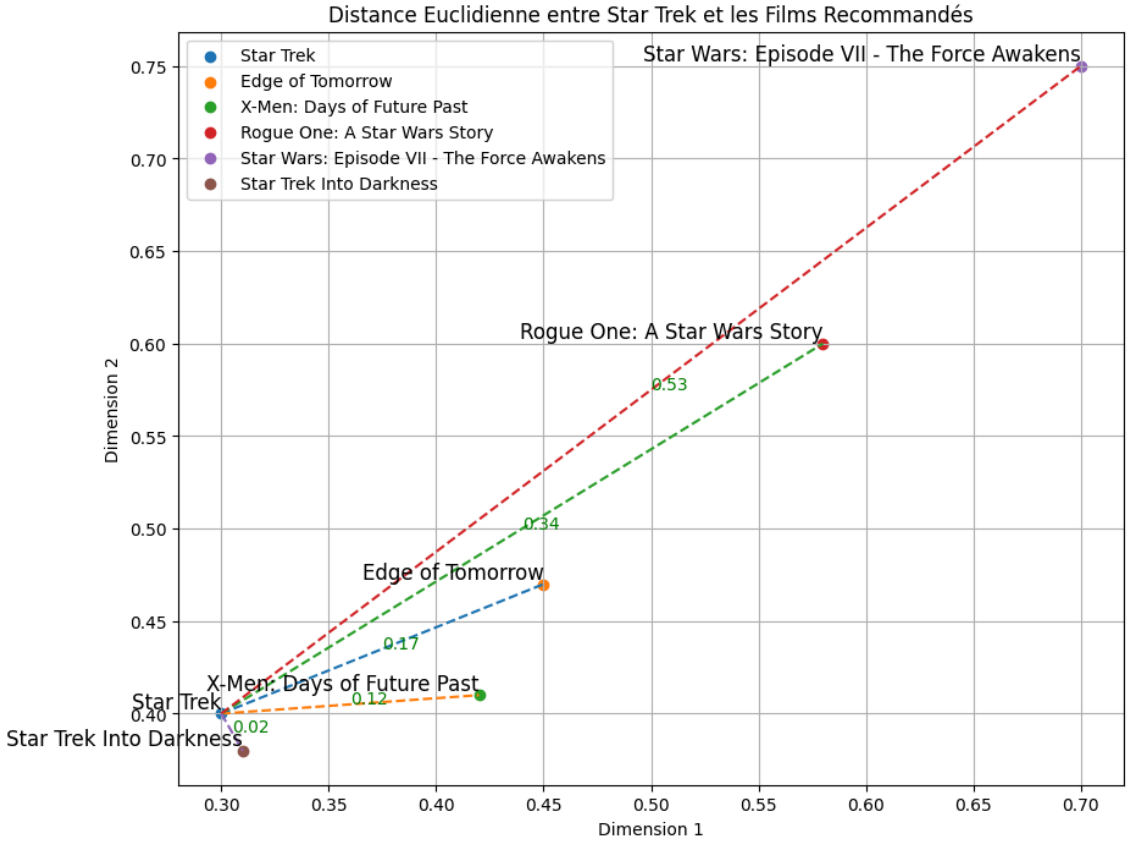


Par la suite, nous avons visualisé en 2D l'espace entre les films afin de mieux comprendre comment le modèle choisit les films après avoir sélectionné les composants les plus appropriés pour les recommandations. Cette visualisation a révélé que le modèle classe les films en fonction de la proximité de leurs attributs dans le dataframe. Par exemple, les films associés aux genres Action, Adventure, et Crime sont regroupés ensemble. Pour confirmer nos analyses et conclusions sur le fonctionnement du modèle, nous avons également visualisé cette projection en prenant ***Toy Story*** comme film de référence.


Projection 2D des Films après SVD avec Star Trek en rouge
Action
Adventure
Crime
Star Trek
PC1
PC2
1.0
0.5
0.0
−0.5
−1.0
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


Projection 2D des Films après SVD avec Star Trek et Films Recommandés
Action
Adventure
Crime
Recommended Movies
PC1
PC2
1.0
0.5
0.0
−0.5
−1.0
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

Enfin, nous avons zoomé sur les 5 films recommandés par le modèle pour ***Star Trek***, ce qui a confirmé que le modèle privilégie les films ayant la plus faible distance par rapport au film cible.



Cette approche nous a permis de valider la cohérence des recommandations générées par le modèle, tout en offrant une meilleure visibilité sur le fonctionnement interne du modèle et sur la manière dont les différentes caractéristiques des films contribuent aux recommandations finales.

Conclusion

En conclusion, le modèle basé sur la librairie Annoy s'est révélé être le plus performant parmi ceux que nous avons testés. Il a démontré une grande rapidité lors du chargement et de l'exécution, tout en utilisant moins de mémoire que les autres modèles. En outre, il a fourni les résultats les plus pertinents, offrant des recommandations de films cohérentes et fiables. Grâce à son efficacité et à sa précision, ce modèle s'est avéré être la meilleure option pour notre système de recommandation.

VII. Conclusion

Résultat obtenus

Notre système de recommandation de films, développé en utilisant les ensembles de données IMDb et MovieLens, a démontré une capacité prometteuse à personnaliser les suggestions de films basées sur les préférences des utilisateurs et les similarités avec des films qui ont été aimés par les utilisateurs eux-mêmes. Grâce à des méthodes avancées de traitement de données et de machine learning ainsi que des tests de différents modèles, le système a réussi à offrir des recommandations pertinentes, ce qui constitue une avancée significative vers une expérience utilisateur enrichie et personnalisée.

Nous avons mis à profit les connaissances acquises durant les cours et même au-delà, car nous nous sommes également documentés sur d'autres méthodes existantes.

Et le résultat obtenu a été un succès. Le modèle choisi, Annoy, a particulièrement mis en évidence son efficacité et sa robustesse lors des tests, tout en se distinguant par sa rapidité d'exécution et ses faibles exigences en termes de ressources matérielles par rapport aux autres modèles testés.

En outre, grâce à l'efficacité de l'équipe, nous avons non seulement pu finaliser les livrables du projet bien avant la date prévue pour la soutenance, mais également mener à bien des tâches supplémentaires, telle que l'analyse de l'interprétabilité du modèle.

Difficultés rencontrées

Les deux principales difficultés rencontrées ont été liées au matériel et à la coordination au sein de l'équipe. À maintes et maintes reprises, l'infrastructure matérielle s'est révélée insuffisante, en particulier en ce qui concerne la puissance de calcul, qui ne parvenait pas à supporter la charge des traitements requis. En conséquence, nous avons dû explorer des alternatives pour contourner ces limitations. C'est d'ailleurs cette contrainte qui nous a conduits à privilégier le modèle Annoy, lequel a su fonctionner de manière fluide malgré les ressources limitées du serveur.

En ce qui concerne la coordination au sein de l'équipe, le fait que nous venions de divers horizons professionnels a représenté à la fois une force et une faiblesse. Cela a été une force dans la mesure où nos perspectives variées ont enrichi les solutions apportées aux livrables et au traitement des données. Cependant, cela a également constitué une faiblesse, car nous avons dû apprendre à collaborer efficacement et à adopter un langage commun. Néanmoins, malgré ces défis susdits, nous avons réussi à finaliser les livrables bien avant la date de la soutenance.

Analyse rétrospective

L'un des enseignements que nous pourrions faire est l'importance de bien connaître les données que l'on exploite. La phase d'exploration des données est essentielle et détermine la réussite du projet. La phase de prétraitement des données est également primordiale car de ce jeu de données final dépendra la qualité des résultats obtenus suite aux modélisations.

En outre, le choix minutieux des caractéristiques du modèle de machine learning s'est avéré essentiel pour obtenir des résultats pertinents et en adéquation avec nos objectifs.

Perspectives d'amélioration

Une des principaux axes d'amélioration serait d'optimiser le workflow Git en mettant en place une structure plus rigoureuse. Cela inclurait la création de branches dédiées aux nouvelles fonctionnalités "***features***", d'une branche de développement "***dev***", et d'autres branches, afin de rendre le travail plus flexible, mieux structuré et organisé. La branche principale "***main***" serait ainsi réservée exclusivement aux mises en production, garantissant une meilleure gestion du cycle de développement.

Un autre axe d'amélioration serait d'élargir les critères du modèle d'apprentissage, en intégrant davantage de variables relatives aux préférences et choix des utilisateurs. Cela permettrait de mieux refléter les comportements dans un contexte d'entreprise réel. Pour l'avenir, nous pouvons envisager d'explorer des modèles de deep learning plus sophistiqués pour améliorer la précision des recommandations. L'intégration de feedbacks utilisateurs en temps réel pourrait également aider à affiner les systèmes de recommandation de manière dynamique.

Montée en compétence

Ce projet a été une opportunité précieuse pour l'équipe de monter en compétence sur les outils de data science et de machine learning au travers d'un cas concret. Il nous a notamment permis de se confronter au traitement de vastes sources de données et de comprendre l'importance de l'exploration et du prétraitement des données afin d'envisager des modèles pertinents et performants.

À travers ce projet, nous avons développé des compétences essentielles liées au métier de Data Scientist, notamment en matière d'exploitation des données, de réflexion critique, ainsi que dans le choix du type de modèle de machine learning et de ses hyperparamètres. Nous avons également découvert des technologies de pointe utilisées dans ce domaine, telles que Streamlit.

Par ailleurs, sur le plan humain, ce fut une belle et enrichissante rencontre entre des personnes de différents profils et de différents horizons, qui ne se connaissaient pas au départ mais qui ont réussi à collaborer efficacement pour arriver à un résultat abouti.

Nous avons acquis les bases de la recherche et de la pensée logique, des compétences indispensables pour obtenir des résultats pertinents et alignés avec les besoins exprimés.

Ce projet nous a permis d'appréhender le travail en équipe autant dans ses avantages que dans ses inconvénients, une belle introduction au travail d'un datascientist.

VIII. Références

- Wikipedia. "Recommender System." Disponible à: Wikipedia - Recommender System.
- Onespire. "History of Recommender Systems." Disponible à: Onespire - History of Recommender Systems.
- ACM Computing Surveys. "A Survey of Recommender Systems." Disponible à: /mnt/data/ACM_CSUR___Survey_AdL___CameraReady.pdf.
- https://grouplens.org/datasets/movielens/20m/
- https://www.imdb.com/interfaces/
- https://help.netflix.com/fr/node/100639
- https://www.lesmondesnumeriques.net/wp-content/uploads/2019/02/Comprendre-lalgorithme-de-recommandation-de-Netflix-1.pdf
- https://github.com/spotify/annoy