

DATASCIENTEST



PROJET :

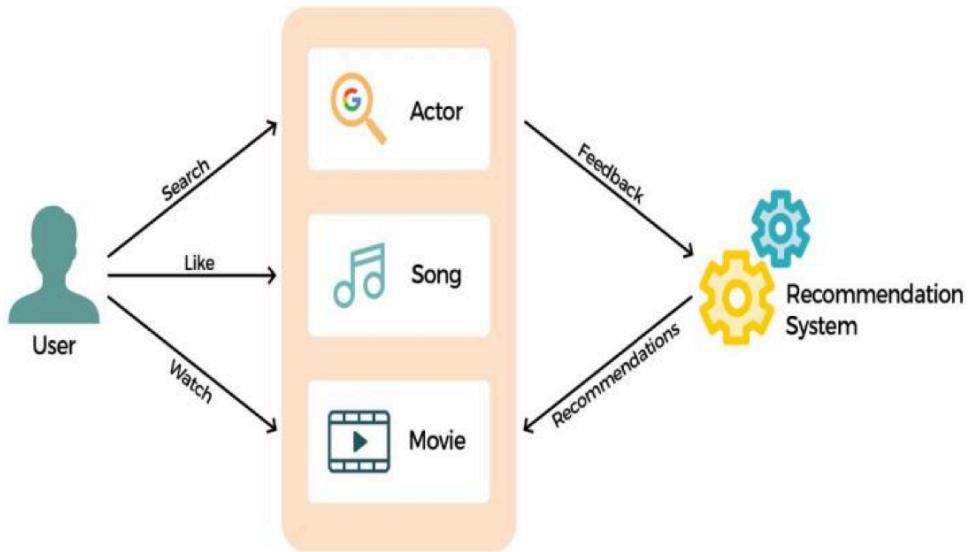
Système de Recommandation de Films

Exploration, data visualisation et pre-processing des données

CONTEXTE

Un système de recommandation est une application permettant de prédire le comportement d'un utilisateur à partir de données issues d'un historique de préférences.

C'est une forme de filtrage de l'information.



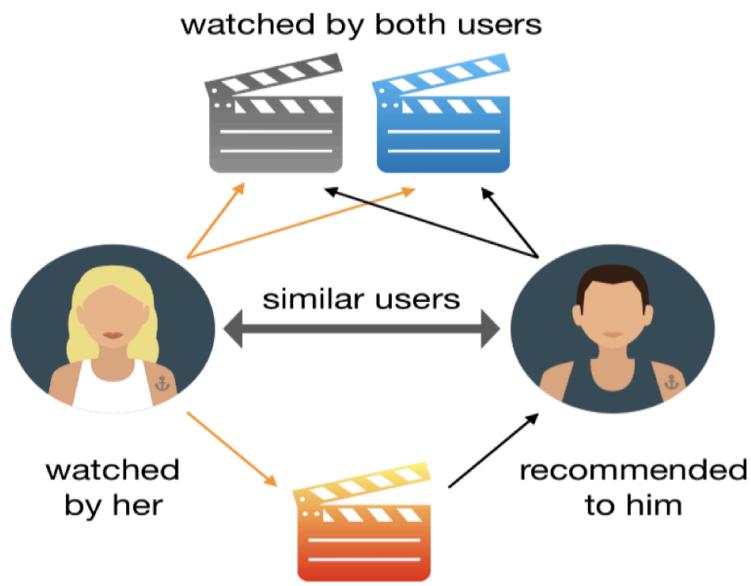
Les systèmes de recommandations sont utilisés dans différents domaines : dans les bibliothèques, sur les sites commerciaux, dans le tourisme, la vidéo à la demande, etc.

Les applications telles que Amazon (recommandation de produits), Netflix (recommandation de séries et films), Spotify (recommandation de musique), ou encore Youtube (recommandation de vidéos) sont des systèmes de recommandations connus.

Dans le cas de ce projet, nous allons concevoir un système de recommandation utilisé pour prédire la « note » ou la « préférence » qu'un utilisateur donnerait à un film. Pour ce faire nous allons mettre en place un algorithme capable de d'analyser les préférences et les besoins des autres utilisateurs afin de prédire et recommander des films à un utilisateur.

Les 3 techniques principales de recommandation sont le filtrage collaboratif, le filtrage basé sur le contenu et enfin le filtrage hybride.

Filtrage collaboratif

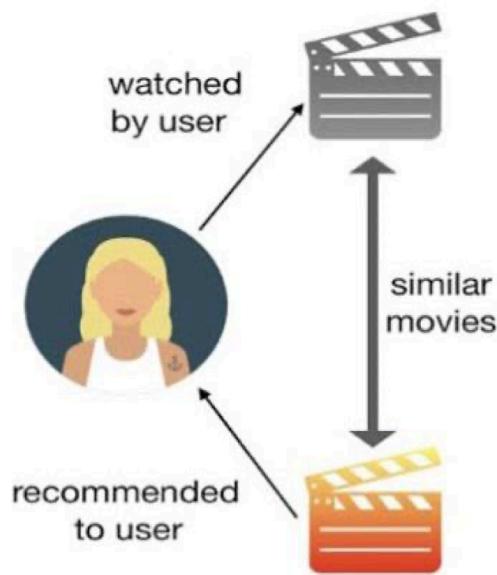


Le filtrage collaboratif consiste à associer des utilisateurs en fonction de leurs préférences, et à leur proposer les produits des utilisateurs similaires. Ainsi, pour chaque utilisateur, les données sont récoltées afin d'établir son profil. Celles-ci sont soit recueillies à leur insu, soit par l'intermédiaire de formulaires remplis sur internet. Le profil d'un utilisateur est alors constitué en croisant les informations de profils similaires. Le système de filtrage collaboratif a été popularisé par Amazon avec la fonctionnalité "les gens qui ont acheté x ont aussi acheté y".

Parmi les filtrages collaboratifs, on distingue tout d'abord les filtrages de type actif. Ces systèmes utilisent les goûts explicites de ses utilisateurs pour réaliser des prévisions. Ces données peuvent être récoltées, par exemple, par un système de note. Cette méthode présente l'avantage de permettre de reconstruire l'historique d'un individu, mais elle présente le défaut de permettre un biais de déclaration.

On distingue également les filtrages de type passif. Dans ce cas, le système effectue une analyse du comportement de l'utilisateur en « arrière-plan » pour en déduire ses goûts et préférences sans que celui-ci ait besoin de donner consciemment ses opinions. C'est une méthode utilisée entre autres par Facebook et Amazon. Cette méthode permet qu'aucune information ne soit demandée aux utilisateurs, et donc qu'il n'y ait pas de biais de déclaration. En revanche, les données récupérées sont plus difficilement attribuables et contiennent des biais d'attribution. Un exemple typique est la multi-utilisation d'un compte par plusieurs utilisateurs.

Filtrage basé sur le contenu



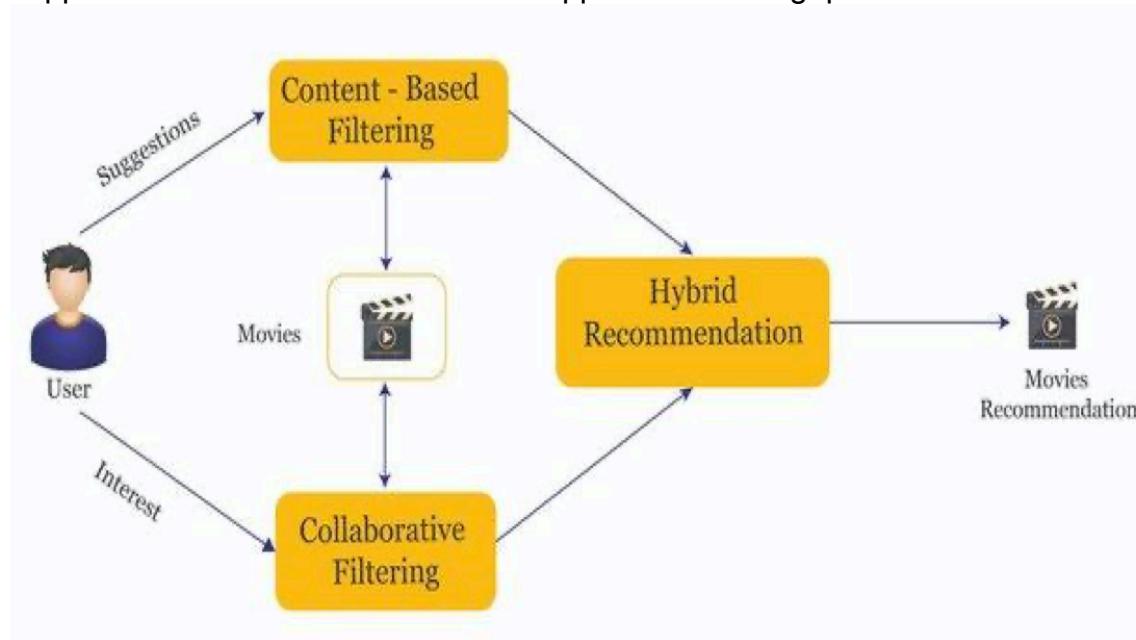
Le filtrage basé sur le contenu s'intéresse plutôt aux caractéristiques des produits proposés. L'idée ici est de sélectionner du contenu en se basant sur les caractéristiques du produit, et en les comparant aux caractéristiques qui intéressent l'utilisateur.

On note certaines limites aux systèmes de recommandation :

- Un enfermement uniquement vers les produits appréciés de l'utilisateur, sans possibilité d'ouverture d'esprit vers quelque chose de nouveau,
- un moyen de capter l'attention et d'orienter les achats des usagers, de manière inconsciente puisque l'usager n'a aucune conscience du fonctionnement de l'algorithme,
- la pertinence de leurs propositions, étant donné que les algorithmes ne représentent les biens que par une liste d'attributs, sans prise en compte de leur dimension symbolique ni de l'expérience de l'usager et du contexte dans lequel il se trouve (l'usager n'est pas figé dans le temps : ses goûts et son environnement changent au cours du temps),
- une utilisation non affichée et non explicitée par les systèmes de recommandations des navigations et des traces laissées à leur insu par les internautes, ce qui pose la question de la transparence et de l'éthique.
- Enfin, les algorithmes de recommandation peuvent conduire à des résultats biaisés vers certains contenus

Filtrage Hybride

Un système de recommandation hybride utilise des composants de différents types d'approches de recommandation ou s'appuie sur leur logique.



Il existe certains problèmes liés au système de recommandation comme :

A. Problème du « départ à froid » (Cold-starter problem)

Lorsqu'un utilisateur s'inscrit pour la première fois, il n'a regardé aucun film. Ainsi, le système de recommandation ne dispose pas de film sur lequel il puisse donner des résultats. C'est ce qu'on appelle un problème de démarrage à froid.

Le filtrage basé sur le contenu est un moyen de traiter ce problème de démarrage à froid.

B. Problème de rareté des données

Ce problème se produit lorsque l'utilisateur a évalué très peu d'éléments sur la base desquels il est difficile pour le système de recommandation d'obtenir des résultats précis.

C. Problème de flexibilité

Dans ce cas, l'encodage se fait de manière linéaire sur les éléments. Le système fonctionne efficacement uniquement lorsque l'ensemble de données est de taille limitée.

OBJECTIFS

Les objectifs de ce projet sont les suivants :

- Réaliser un système de recommandation de films par application de méthode de filtrage collaboratif (collaborative filtering) et basé sur le contenu (content-based filtering),
- Consolider un jeu de données par web scrapping,
- Appliquer des algorithmes de Deep Learning.

Personne dans le groupe ne dispose d'expertise relative à la problématique adressée.

Pour mener à bien ce projet, nous ne sommes pas rentrés en contact avec des experts métiers, mais nous avons utilisé la documentation en ligne afin de nous orienter, en particulier sur les phases de pre-processing et feature engineering. Cela nous a permis de comprendre la façon dont il fallait cadrer le projet, et d'appréhender quelle était la forme de données permettant une application efficace des algorithmes de Machine et Deep Learning. Une bibliographie est donnée à la fin de ce rapport.

CADRE

Les jeux des données utilisés sont issus des bases :

<https://grouplens.org/datasets/movielens/20m/>

<https://www.imdb.com/interfaces/>

Ces données sont disponibles en libre téléchargement sur internet.

Les données récoltées contiennent 26744 films et plus de 20 millions de notations.

PERTINENCE

Au regard de nos objectifs, nous considérons que le genre de films est une variable très importante, ainsi que le nombre de notations pour un film. La variable cible est la note (sur 5) attribuée à un film.

PRE-PROCESSING ET FEATURE ENGINEERING

Après avoir importé les données, il a été nécessaire de les nettoyer et de les traiter afin d'obtenir un dataset prêt pour le début de la modélisation.

Les étapes sont listées ci-après :

- Import des 2 datasets : movies et ratings
- Modification de la colonne 'genres' de movies pour avoir des listes de genres
- Ajout d'une colonne par genre et attribution des valeurs 1 (présence du genre) ou 0 (absence du genre)

- Jointure de ‘movies’ et ‘ratings’ dans une colonne ‘data’
- Suppression des 534 valeurs manquantes (pas d’intérêt pour les films non notés)
- Suppression des colonnes ‘timestamp’ et ‘genres’

La première étape dans l’exploration des données consiste à étudier la colonne «genres ». En effet cela est très important pour la suite du projet, en particulier dans le cas d’un filtrage basé sur le contenu.

Il y'a 19 genres différents et 246 films qui n'ont pas de genre attribué.

	genre	count
0	Drama	13344
1	Comedy	8374
2	Thriller	4178
3	Romance	4127
4	Action	3520
5	Crime	2939
6	Horror	2611
7	Documentary	2471
8	Adventure	2329
9	Sci-Fi	1743
10	Mystery	1514
11	Fantasy	1412
12	War	1194
13	Children	1139
14	Musical	1036
15	Animation	1027
16	Western	676
17	Film-Noir	330
18	(no genres listed)	246
19	IMAX	196

L’étape suivante était d’ajouter à la base de données une colonne dédiée à chaque genre.

On montre ci-après les 5 premières lignes du dataset. On remarque déjà que les 5 premières lignes correspondent au même film : « Toy Story (1995) ». En effet, il y a autant de lignes que de notations.

	movielid	title	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	...
0	1	Toy Story (1995)	0	1	1	1	1	0	0	0	...
1	1	Toy Story (1995)	0	1	1	1	1	0	0	0	...
2	1	Toy Story (1995)	0	1	1	1	1	0	0	0	...
3	1	Toy Story (1995)	0	1	1	1	1	0	0	0	...
4	1	Toy Story (1995)	0	1	1	1	1	0	0	0	...
...	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western	no_genres	userId	rating	...
...	0	0	0	0	0	0	0	0	3.0	4.0	...
...	0	0	0	0	0	0	0	0	6.0	5.0	...
...	0	0	0	0	0	0	0	0	8.0	4.0	...
...	0	0	0	0	0	0	0	0	10.0	4.0	...
...	0	0	0	0	0	0	0	0	11.0	4.5	...

Pour information, les dimensions du dataset sont actuellement de (20000263, 24). Il peut être envisageable de réduire le nombre de lignes en définissant un seuil permettant d'éliminer les films les moins bien notés.

Afin d'agrémenter encore plus notre base de données, notamment concernant les aspects de filtrage basé sur le contenu, nous vons effectué un Webscrapping afin de récupérer les 100 meilleurs acteurs, 100 meilleures actrices et 100 meilleurs réalisateurs et réalisatrices suivant un classement effectué sur le site imdb.

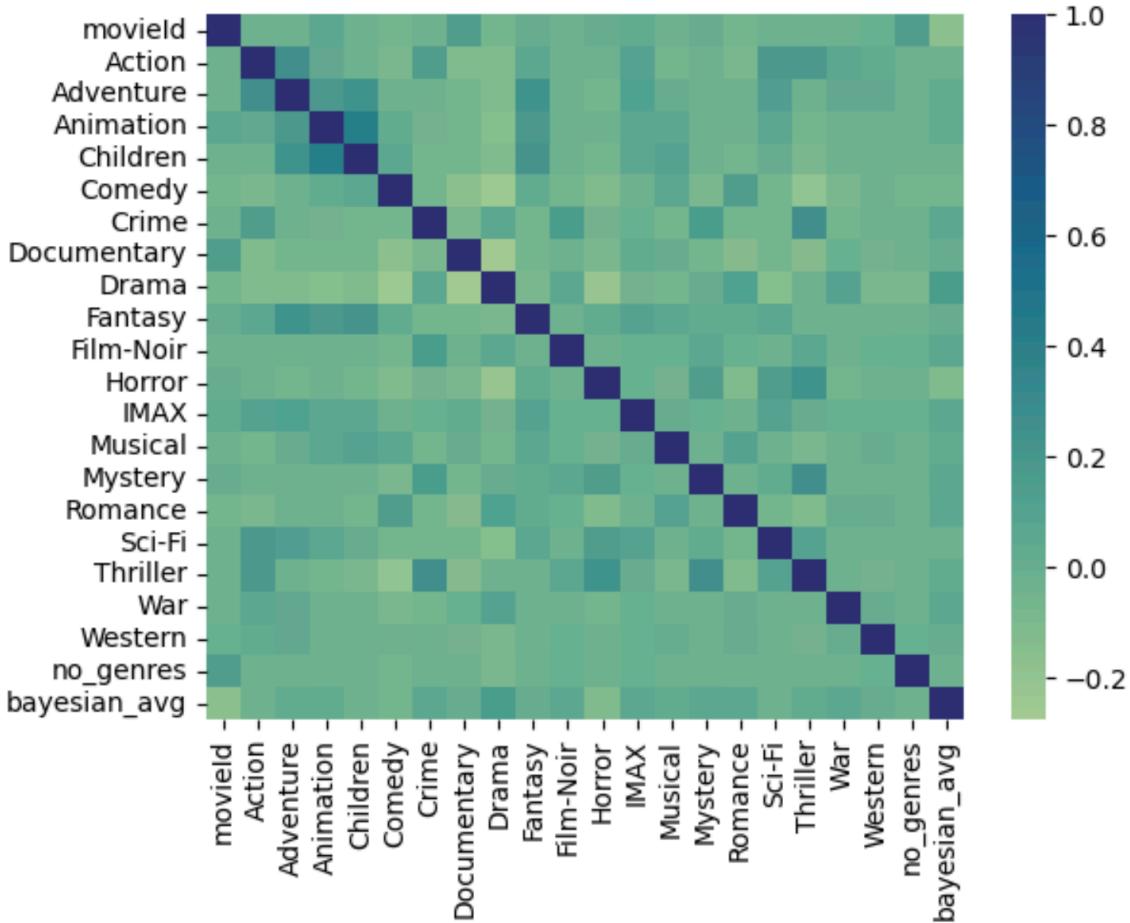
Les liens sont donnés dans la bibliographie en fin de rapport. Il arrive en effet souvent que des utilisateurs soient « fan » d'un acteur, d'une actrice, ou encore d'un certain réalisateur, et soient amenés à aimer le jeu d'un acteur ou le style d'un réalisateur indépendamment du genre du film concerné. Cela apporte donc une réelle amélioration à la qualité du jeu de données.

```
from bs4 import BeautifulSoup as bs
import requests

url_actors = 'https://www.imdb.com/list/ls050274118/'
url_actresses = 'https://www.imdb.com/list/ls000055475/'
url_directors = 'https://www.imdb.com/list/ls053823383/'
```

VISUALISATIONS ET STATISTIQUES

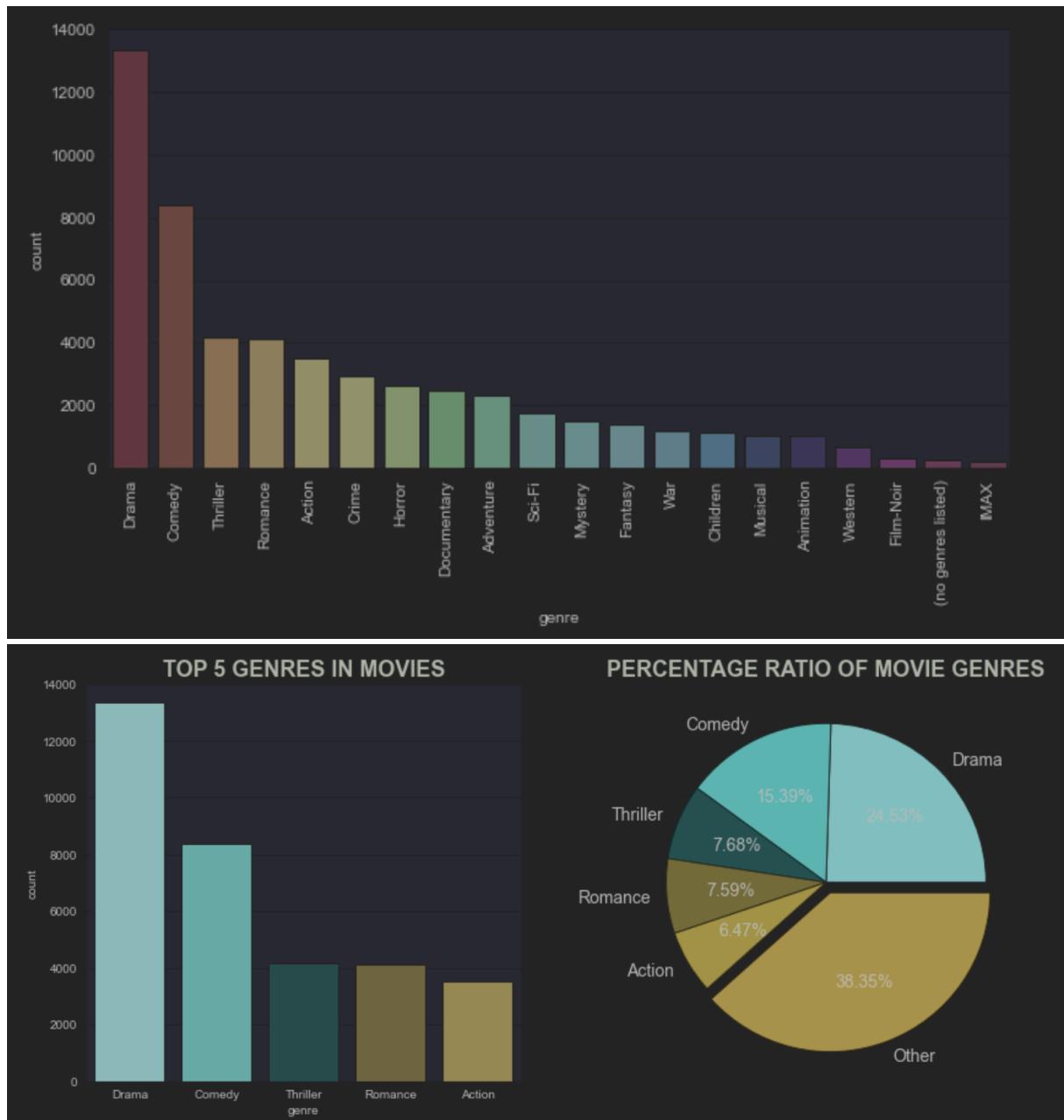
Afin d'établir des relations éventuelles entre les différentes variables, on trace la matrice de corrélation du dataframme 'movies'. Les seules corrélations présentes se trouvent entre les variables 'Animation' et 'Children', ce qui semble cohérent. On peut également noter quelques corrélations entre 'Crime', 'Thriller', et 'Mystery' voire 'Horror', mais elles sont limitées. On remarque qu'il n'y aucune corrélation entre la variable cible et les variables explicatives.



On peut également analyser les genres pour voir ceux qui sont les plus représentés.

On remarque que dans toute la banque de films, le genre 'Drama' est le plus représenté, suivi des 'Comedy'. Ces deux genres représentent à eux seuls 40%.

On peut en déduire une tendance générale sur les films qui sortent à suivre un attrait des utilisateurs pour ces 2 genres.



Dans la base de données « rating » il y'a **27262** films dont **26744** films notés c'est à dire environ 2% (518 films) n'ont pas de note.

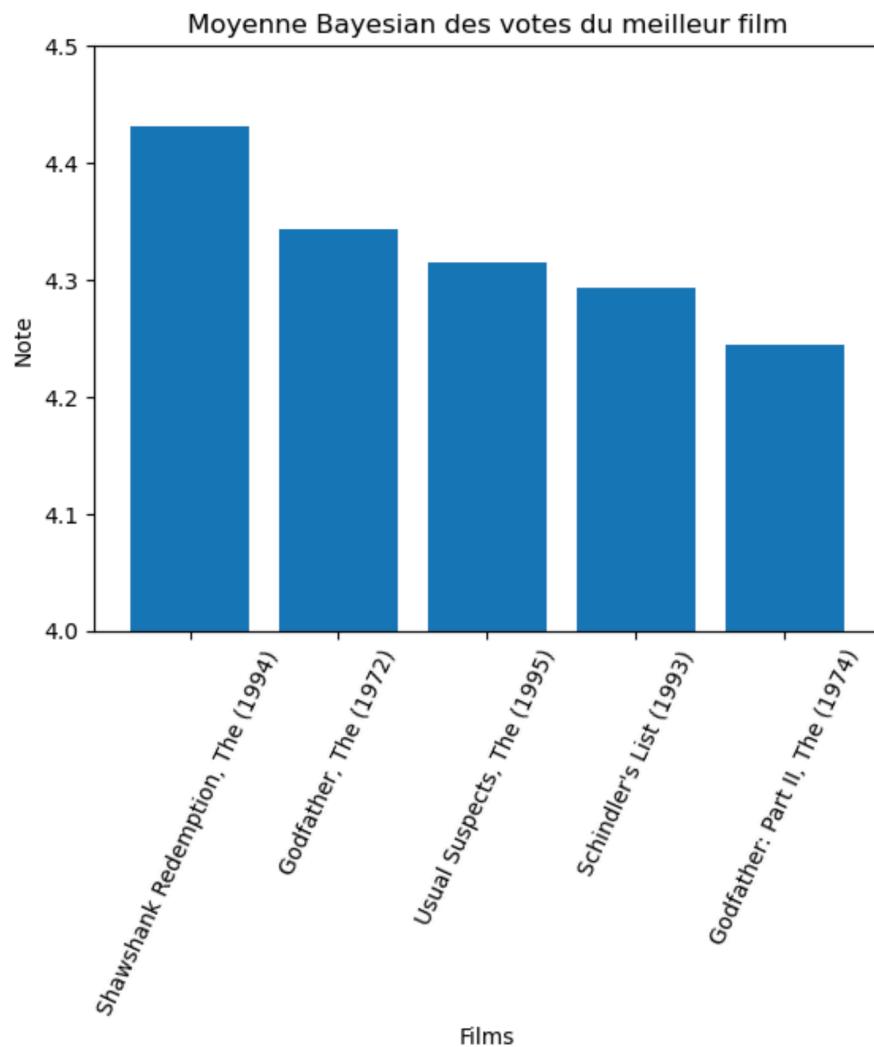
Après avoir effectué la jointure des datasets ‘movies’ et ‘ratings’, on regroupe le tout par film, en fonction de la note moyenne des utilisateurs.

Pour ce faire, on utilise plutôt la moyenne Bayesian permettant de prendre en compte le nombre de notations et ainsi la popularité des films à noter.

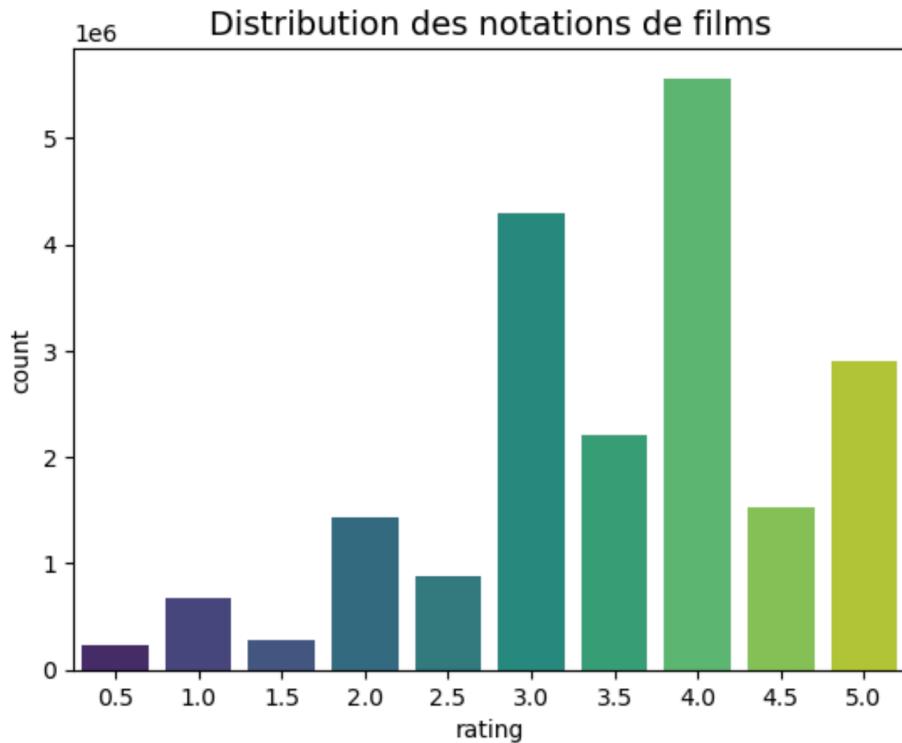
Il ne serait pas raisonnable d'avoir dans le classement un film ayant une unique notation de 5, et arrivant donc en tête des notations. Il est nécessaire de pondérer.

On obtient ainsi les 5 films les mieux notés, avec prise en compte du nombre de votes.

	movied	count	mean		title	bayesian_avg
315	318	63366	4.446990	Shawshank Redemption, The (1994)	4.431666	
843	858	41355	4.364732	Godfather, The (1972)	4.342857	
49	50	47006	4.334372	Usual Suspects, The (1995)	4.315561	
523	527	50054	4.310175	Schindler's List (1993)	4.292849	
1195	1221	27398	4.275641	Godfather: Part II, The (1974)	4.245286	

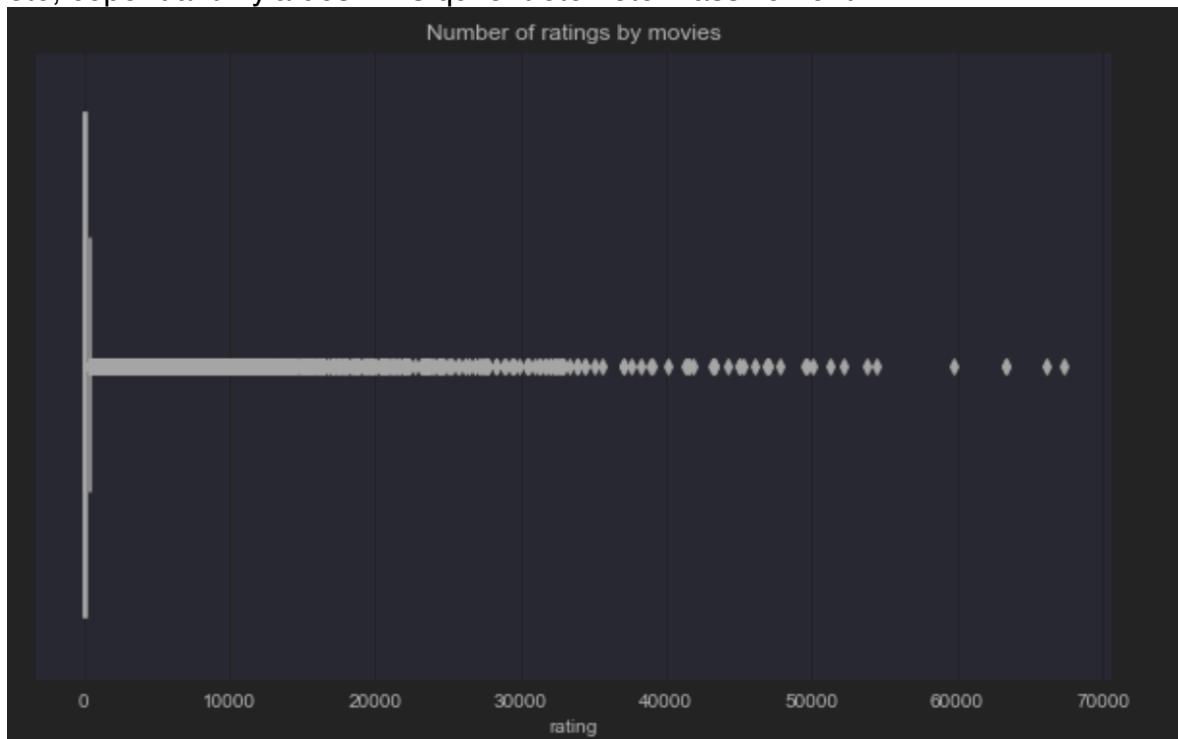


On regarde à présent la distribution des données, en particulier des notations (rating).



On remarque une tendance à noter suivant des chiffres ronds. De plus, la note de 4 est la plus donnée, suivie de la note de 3, et enfin de 5.

La figure suivante nous permet de déduire que la majorité des films ont été très peu noté, cependant il y'a des films qui ont été noté massivement.



Afin de ne pas fausser les prédictions, nous avons décidé de garder les films qui ont été suffisamment notés.

Il est donc indispensable d'éliminer les films très peu noté car cela ne reflète pas un avis collectif mais plutôt plusieurs avis individuels.

Modélisation

Classification du problème

Notre problème s'apparente d'une part à un problème de clustering, et d'autre part à un problème de régression. En effet, il s'agit de définir une classe de films pour un critère donné en entrée de l'application. Par exemple, pour un film aimé, on propose 10 autres films similaires. Ou encore, pour un utilisateur qui indique les genres de films qu'il aime, on lui propose un certain nombre de films. On peut donc considérer cela comme une façon de créer une classe à partir d'un film.

Il est également possible de voir ce problème comme un problème de régression. En fonction des notations données aux films par l'utilisateur, le modèle prédit la note que l'utilisateur donnera à un film qui n'a pas encore été noté.

Le projet fait partie de la famille des systèmes de recommandation.

La métrique de performance principale utilisée pour comparer nos modèles est la RMSE. En effet, elle permet de mesurer l'écart entre les prédictions et les valeurs réelles, ce qui permet de procéder à des comparaisons entre algorithmes.

Nous avons également utilisé, en cours d'étude, d'autres métriques telles que la MSE, la MAE et la FCP.

Choix du modèle et optimisation

Le gestion du volume des données et les temps de calculs nous ont conduit à retenir 1500 films et 1500 utilisateurs pour tester nos modèles et effectuer des comparaisons entre modèles. On reste ainsi dans les tailles de données classiques qu'on trouve dans la littérature.

Dans une première partie, nous avons fait le choix d'utiliser un modèle de **content-based filtering** basé sur les genres des films ainsi que sur les acteurs, actrices, réalisateurs, et réalisatrices principaux. Chaque film est alors représenté par un vecteur des différents paramètres. La méthode utilisée ensuite est celle de la matrice de similarité cosinus qui permet de cibler les vecteurs qui ont l'angle le plus faible avec un film donné et ainsi d'établir la liste des films les plus semblables.

Exemple des 10 meilleurs films pour le film « Godfather, The (1972) :

```
Dimensions of our genres cosine similarity matrix: (1500, 1500)
Recommendations for Godfather, The (1972):

   4          Godfather: Part II, The (1974)
  68          On the Waterfront (1954)
 208          Dog Day Afternoon (1975)
 376          Serpico (1973)
 463          Donnie Brasco (1997)
 269          Once Were Warriors (1994)
 470          Hate (Haine, La) (1995)
 669          Scent of a Woman (1992)
 710          Maria Full of Grace (Maria, Llena eres de grac...
 758          Boyz N the Hood (1991)

Name: title, dtype: object
```

Exemple pour Django Unchained (2012)

```
Dimensions of our genres cosine similarity matrix: (1500, 1500)
Recommendations for Django Unchained (2012):

707           Tombstone (1993)
349           Gladiator (1992)
592           Ray (2004)
1212          Rush (2013)
1232          Jet Li's Fearless (Huo Yuan Jia) (2006)
1330          Basketball Diaries, The (1995)
1441          Silverado (1985)
146           Inglourious Basterds (2009)
302           Kill Bill: Vol. 2 (2004)
80            Treasure of the Sierra Madre, The (1948)
Name: title, dtype: object
```

Initialement, il n'était pris en compte que les genres des films. Par la suite, nous avons introduit comme paramètres des films les 100 acteurs, 100 actrices, et 100 réalisateurs / réalisatrices principaux. En effet, il arrive fréquemment que les utilisateurs soient attirés par un film d'un acteur, d'une actrice, ou d'un réalisateur qu'ils apprécient.

Dans une deuxième partie, nous proposons un modèle de **filtrage collaboratif**.

Dans un premier temps, nous avons implémenté simplement une méthode kNN afin de trouver les films les plus proches d'un film donné en terme de notations des différents utilisateurs. Cette méthode est proche de celle utilisant la matrice de similarité cosinus pour le filtrage basé sur le contenu. Mais la différence réside dans le fait que les vecteurs ne sont pas constitués du contenu du film, mais plutôt des notations des différents utilisateurs.

On donne ci-dessous les résultats pour le film « Godfather, The (1972) :

```
Because you watched Godfather, The (1972):
Godfather: Part II, The (1974)
Goodfellas (1990)
Chinatown (1974)
French Connection, The (1971)
Raging Bull (1980)
Taxi Driver (1976)
Apocalypse Now (1979)
Serpico (1973)
Casino (1995)
```

Et pour le film Django Unchained (2012) :

```
Because you watched Django Unchained (2012):
Wolf of Wall Street, The (2013)
Looper (2012)
Skyfall (2012)
Dark Knight Rises, The (2012)
Silver Linings Playbook (2012)
Edge of Tomorrow (2014)
Captain Phillips (2013)
Prisoners (2013)
Argo (2012)
```

Les résultats sont différents que ceux obtenus avec la matrice de similarité cosinus, ce qui est normal, les deux modélisations ne s'appuie pas sur le même type de données. On voit néanmoins des similarités entre les deux résultats de modélisation, pour le premier film.

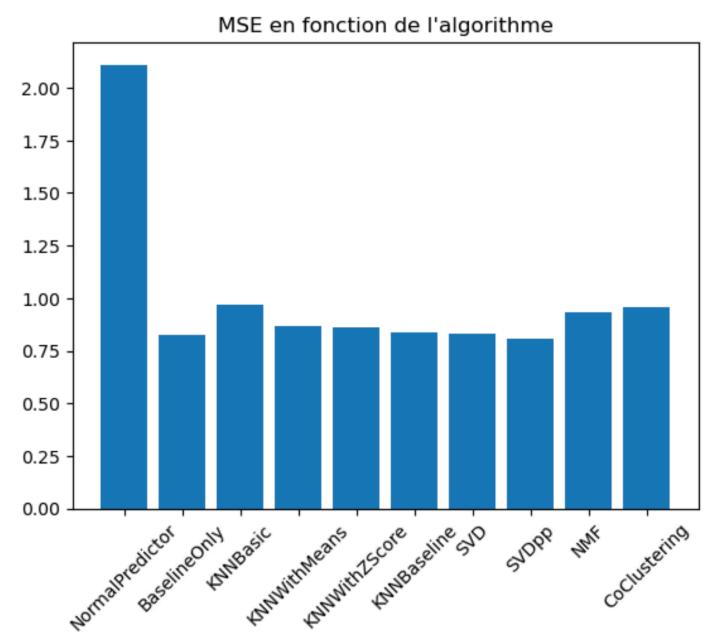
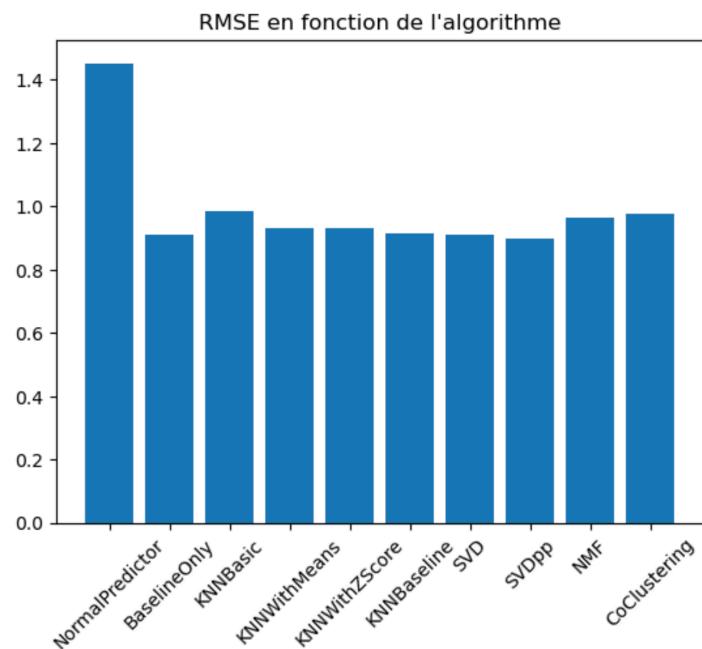
Afin de mesurer la performances de modèles de machine Learning et de Deep learning, nous avons choisi d'utiliser la bibliothèque surprise d'une part, et un modèle de Deep Learning simple d'autre part. Dans un premier temps, nous avons choisi de tester l'ensemble des modèles proposés dans la bibliothèque surprise, à savoir :

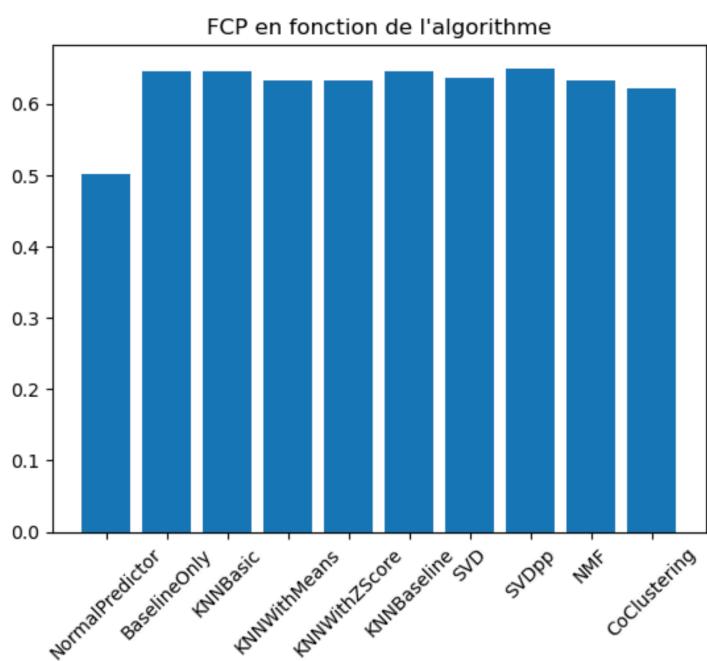
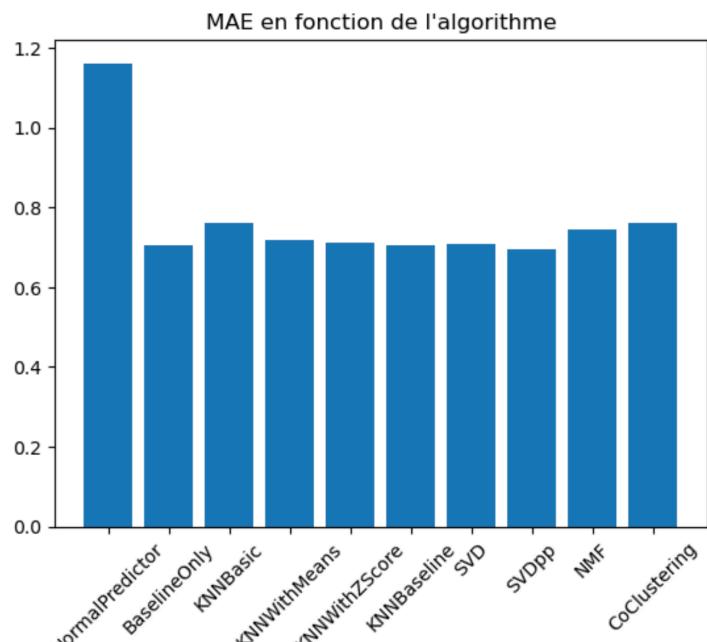
- NormalPredictor
- BaselineOnly
- KNNBasic
- KNNWithMeans
- KNNWithZScore
- KNNBaseline
- SVD
- SVDpp
-

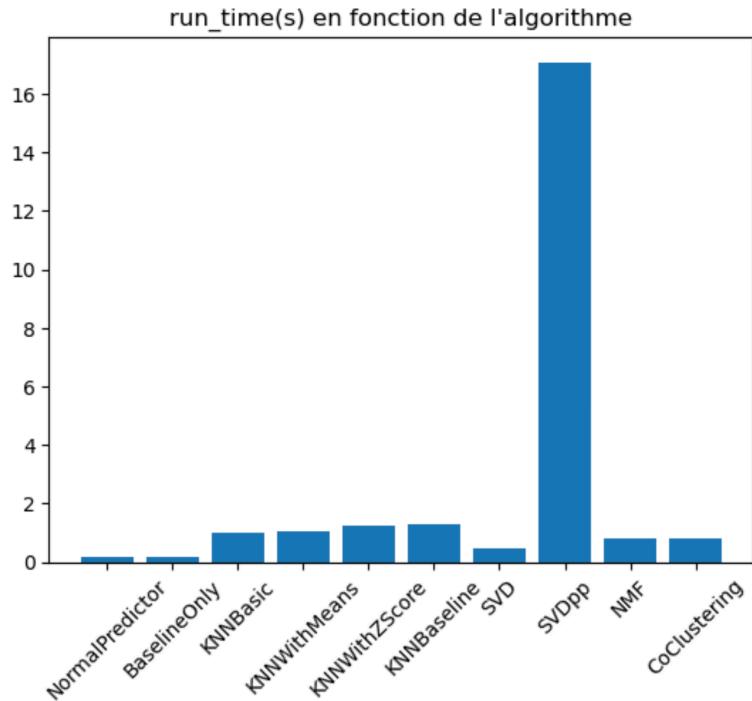
Les résultats suivants sont les premiers obtenus avec la bibliothèque surprise.

metrics	NormalPredictor	BaselineOnly	KNNBasic	KNNWithMeans	KNNWithZScore	KNNBaseline	SVD	SVDpp	NMF	CoClustering
RMSE	1.456287	0.909029	0.984053	0.931154	0.929272	0.914656	0.911782	0.895985	0.963067	0.97706
MSE	2.120772	0.826334	0.96836	0.867048	0.863546	0.836596	0.831346	0.802789	0.927498	0.954646
MAE	1.163619	0.706397	0.760345	0.717458	0.712126	0.704861	0.70757	0.693803	0.743694	0.76206
FCP	0.491307	0.644769	0.645364	0.632677	0.631684	0.645152	0.632384	0.649466	0.63651	0.61765
run_time(s)	0.16096	0.195397	0.990459	1.070926	1.236557	1.298838	0.477265	17.087445	0.788586	0.785985

Cela nous donne une première tendance sur les résultats. Les algorithmes SVDpp et SVD semblent être les plus performants.



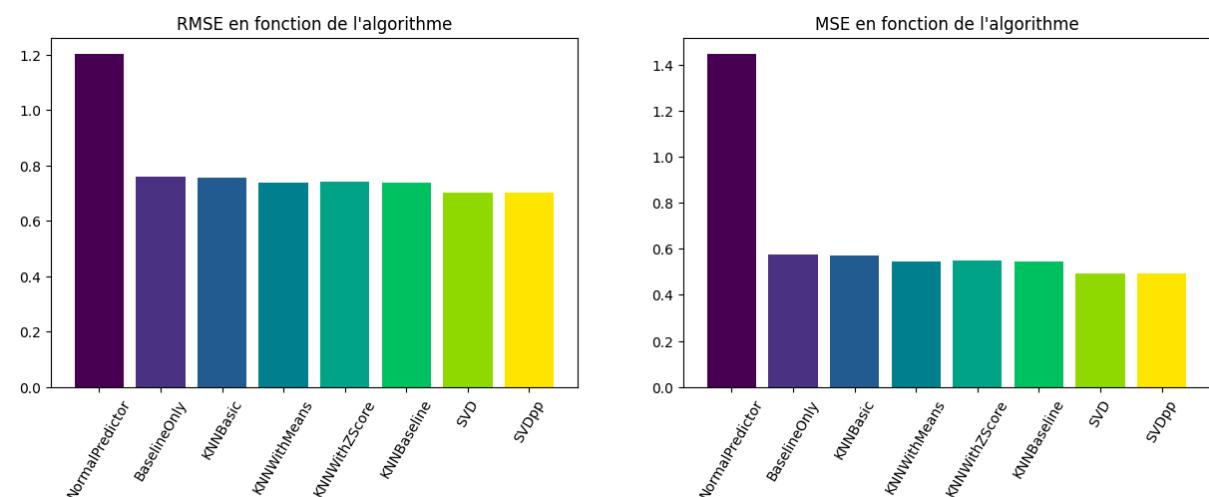




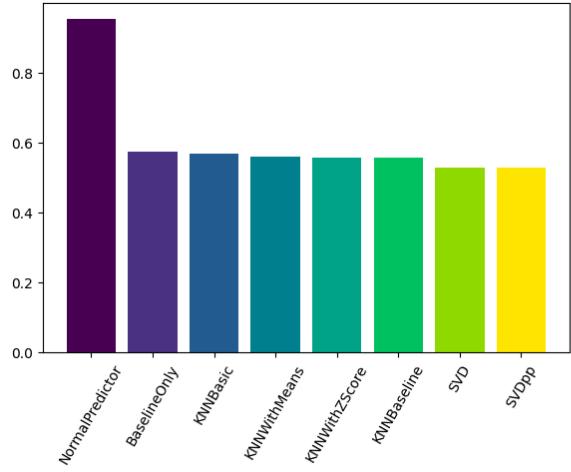
L'algorithme « Normal Predictor » donne des résultats bien inférieur à tous les autres. Il est intéressant de noter que l'algorithme SVDpp, bien que donnant les meilleurs résultats, a un temps de fonctionnement plus de 10 fois supérieur aux autres algorithmes. Ici, au vu de la quantité de données, le temps d'exécution de l'algorithme est une métrique à prendre en compte. Les résultats retenus avec SVDpp sont très proches de ceux obtenus avec SVD, qui permet un temps d'exécution bien moindre.

Après amélioration du jeu de données, notamment par la restriction du nombre de films et du nombres de votes (1500 meilleurs films, et 1500 utilisateurs qui votent le plus), nous obtenons de bien meilleurs résultats.

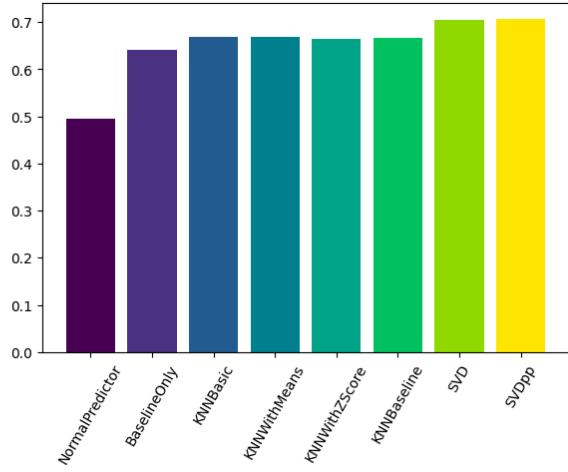
Metrics	NormalPredictor	BaselineOnly	KNNBasic	KNNWithMeans	KNNWithZScore	KNNBaseline	SVD	SVDpp
RMSE	1.201617	0.758862	0.756709	0.738908	0.740822	0.738216	0.701510	0.700808
MSE	1.443883	0.575871	0.572608	0.545986	0.548817	0.544963	0.492116	0.491132
MAE	0.951996	0.574506	0.566304	0.558429	0.557298	0.557580	0.528177	0.528267
FCP	0.495581	0.640275	0.668453	0.667718	0.665238	0.667363	0.705166	0.705882
Run_time (s)	0.618975	0.774094	32.503812	33.421974	33.377476	34.694283	4.940695	216.989252



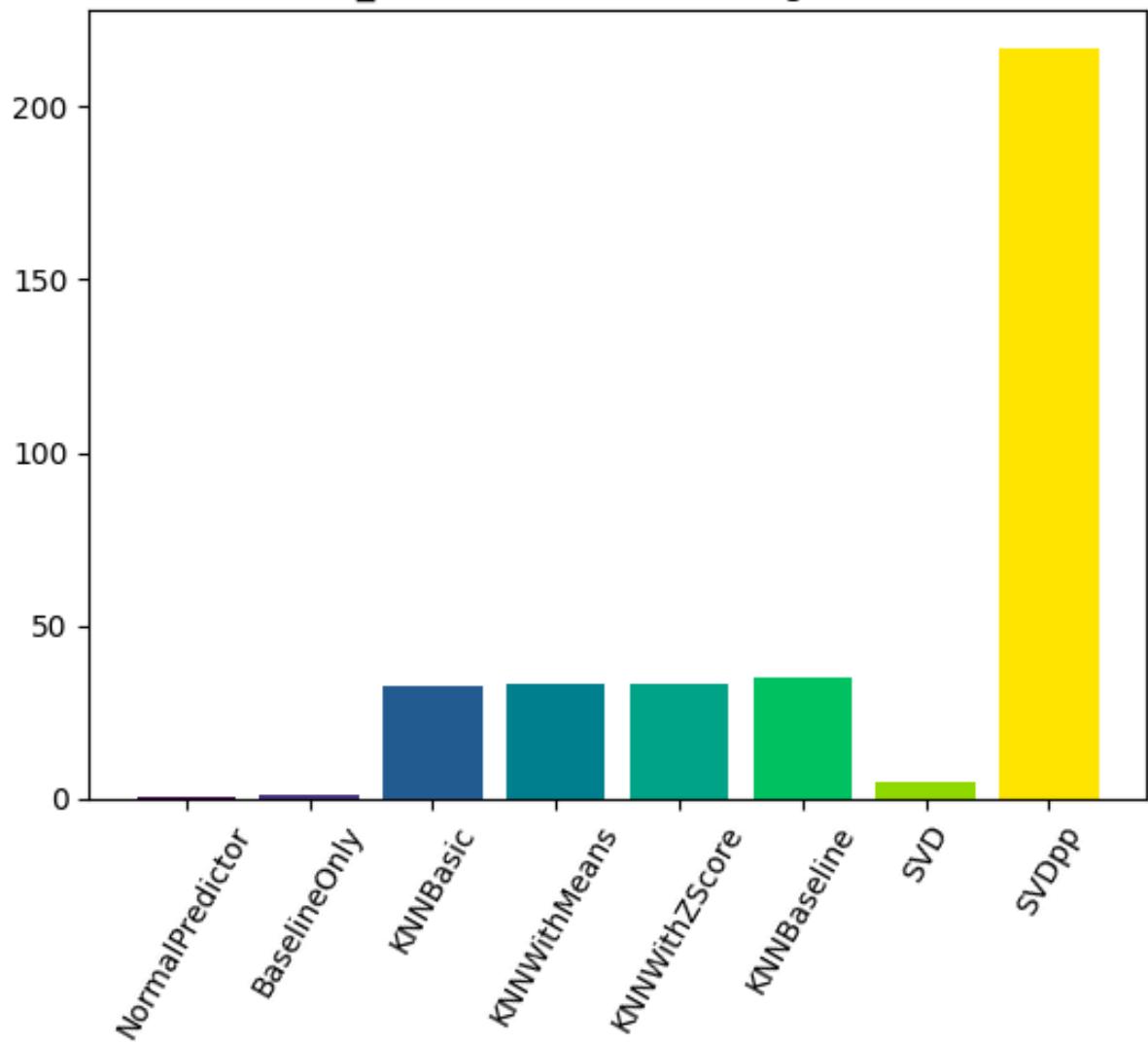
MAE en fonction de l'algorithme



FCP en fonction de l'algorithme



Run_time en fonction de l'algorithme



L'algorithme NormalPredictor donne des résultats moins bons que les autres modèles. L'algorithme SVDpp donne des bons résultats mais son temps de calcul est de beaucoup supérieur à celui des autres algorithmes. Finalement, l'algorithme SVD donne non seulement les meilleurs résultats, mais présente aussi un des temps de calcul les plus court. C'est donc l'algorithme qu'on retient pour notre modélisation.

On cherche par la suite à optimiser les hyperparamètres avec un GridSearchCV en prenant comme métrique le RMSE, et avec un nombre d'échantillons de 5.

```
# Recherche des meilleures hyperparamètres
param_grid = {
    'n_factors': [10, 100, 500],
    'n_epochs': [5, 20, 50],
    'lr_all': [0.001, 0.005, 0.02],
    'reg_all': [0.005, 0.02, 0.1]}

gs_model = GridSearchCV(
    algo_class = SVD,
    param_grid = param_grid,
    n_jobs = -1,
    joblib_verbose = 5)

gs_model.fit(data)

# Entrainement du modèle avec les paramètres qui minimisent la RMSE
best_SVD = gs_model.best_estimator['rmse']
best_SVD.fit(trainset)
```

On obtient une RMSE moyenne de 0.7, soit la valeur correspondant au résultat par défaut. Il n'y a pas d'amélioration significative du résultat en faisant varier les hyperparamètres.

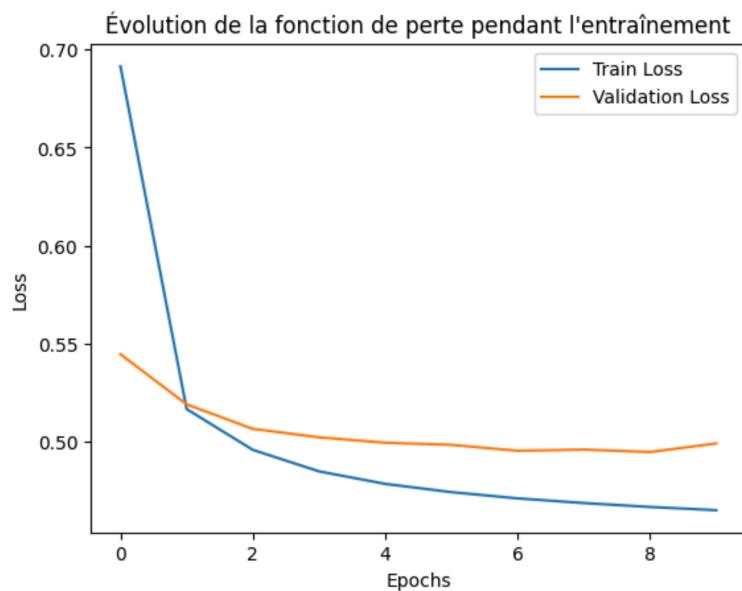
Nous avons également fait tourner un algorithme de Deep Learning simple constitué de deux entrées correspondants aux utilisateurs et aux films. Chaque entrée est constituée d'un input, d'une couche d'embedding et d'une couche de flatten.

Layer (type)	Output Shape	Param #	Connected to
<hr/>			
input_3 (InputLayer)	[(None, 1)]	0	[]
input_4 (InputLayer)	[(None, 1)]	0	[]
embedding_2 (Embedding)	(None, 1, 10)	15000	['input_3[0][0]']
embedding_3 (Embedding)	(None, 1, 10)	15000	['input_4[0][0]']
flatten_2 (Flatten)	(None, 10)	0	['embedding_2[0][0]']
flatten_3 (Flatten)	(None, 10)	0	['embedding_3[0][0]']
dot_1 (Dot)	(None, 1)	0	['flatten_2[0][0]', 'flatten_3[0][0]']
dense_2 (Dense)	(None, 64)	128	['dot_1[0][0]']
dense_3 (Dense)	(None, 1)	65	['dense_2[0][0]']
<hr/>			
Total params: 30193 (117.94 KB)			
Trainable params: 30193 (117.94 KB)			
Non-trainable params: 0 (0.00 Byte)			

L'ensemble est ensuite suivi d'une couche dense avec une fonction d'activation de type 'relu'. Enfin, la couche de sortie (output) est une couche dense de dimension 1.

Le modèle est ensuite compilé avec pour fonction de perte la MSE, pour optimiser la fonction 'adam', et la métrique suivie est la MSE.

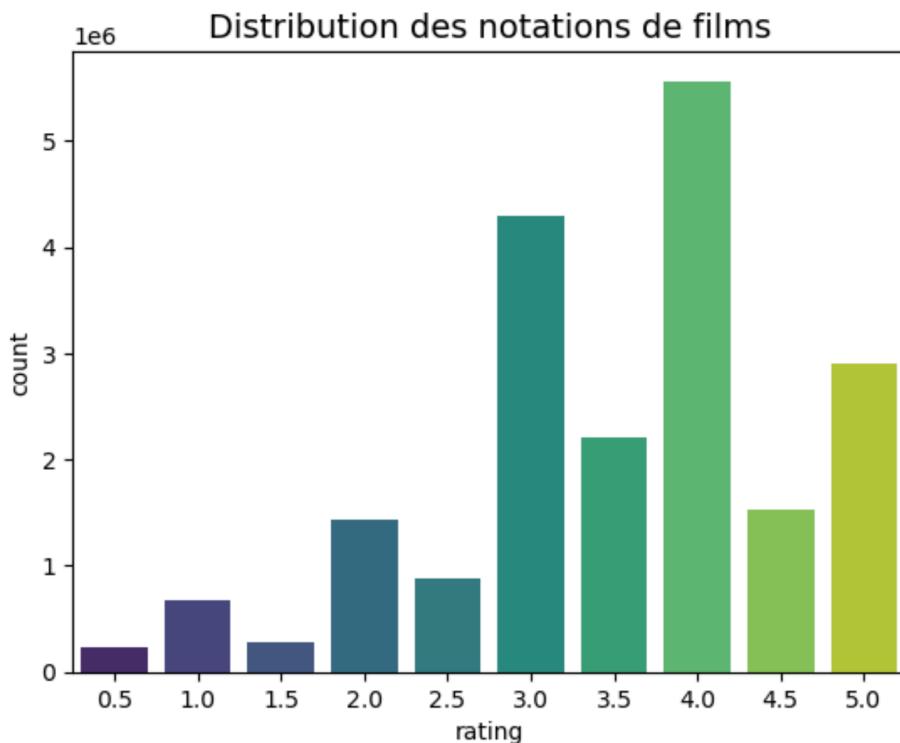
Les résultats obtenus sur 10 époques sont de 0.49 sur l'ensemble de validation, ce qui est équivalent aux résultats obtenus par l'algorithme SVD.



Interprétation des résultats

Tout d'abord, on sent aisément qu'il y a une variabilité des notations dues à l'utilisateur lui-même. Celui-ci peut être enclin à noter de façon inconsciente par comparaison rapprochée avec le dernier film qu'il a vu par exemple. Ou on peut imaginer qu'un utilisateur note mieux un film de Noël à la période de Noël, quand il est plus disposé à visionner ce genre de films. L'utilisateur, dans sa notation ne peut pas être considéré comme 100% fiable et c'est une source d'erreur. Une voie d'amélioration serait de prendre en considération la période de la notation et de pondérer la note en conséquence.

La partie préprocessing nous montre également que l'utilisateur a tendance à noter suivant des chiffres ronds (voir figure ci-dessous), ce qui est difficile à expliquer mathématiquement. On est plutôt face à un comportement purement humain. Le modèle ne prend pas compte cela. Une voie d'amélioration pourrait alors être de trouver un moyen de rapprocher les notes des chiffres ronds de façon systématique pour reproduire cette tendance chez les individus. On pourrait également considérer un ensemble de données constitués que de chiffre ronds, comme il en existe. Les utilisateurs sont alors uniquement invités à donner une note entre 1 et 5.



Nous n'avons pas utilisé de techniques d'intéterprétabilité.

Au cours du projet, la modification apportée au jeu de données consistant à garder seulement les utilisateurs ayant le plus noté et les films les plus et mieux notés (moyenne bayésienne) a permis d'améliorer significativement les résultats. On peut interpréter cela de plusieurs façons. Les utilisateurs ayant visionné et noté un grand nombre de films ont une évaluation plus fiable et plus constante, ce qui est plus

facilement modélisante. Inversement, les utilisateurs n'ayant pas visionné (et noté) beaucoup de films ont une évaluation plus sujette à fluctuation.

Conclusions tirées

Difficultés rencontrées lors du projet

Le principal verrou scientifique a été la façon de poser le problème. En effet, il n'était pas possible de séparer les jeux de données simplement en un ensemble d'entraînement et un ensemble de test, cela amenant à des ensembles de données de dimensions différentes. Il a donc fallut s'adapter et comprendre la façon avec laquelle il fallait procéder dans le cas particulier des systèmes de recommandation.

Dans un premier temps, nous avons utilisé les librairies habituelles pour implémenter des algorithmes de Machine Learning tels que KNN ou la factorisation matricielle.

L'utilisation anticipée de la librairie surprise nous aurait apporté un gain de temps, car elle permet à partir d'un preprocessing minimal d'arriver rapidement aux résultats. Elle est cependant plutôt adaptée aux systèmes de recommandation de type filtrage collaboratif.

L'implémentation des réseaux de neurones a aussi pris du temps, les cours ayant été abordés qu'en fin de formation. Mais nous avons néanmoins pu faire tourner plusieurs modèles avec des architectures différentes. Nous avons retenus un modèle simple qui donne des résultats équivalents à ceux de l'algorithme SVD.

La gestion du volume de données et les temps de calcul associés ont également été un point de réflexion. L'ensemble initial comportait plus de 20 millions de notations sur un ensemble de plus de 26 000 films. Pour des besoins de calcul et pour augmenter la qualité du jeu de données, nous avons fait le choix de restreindre les jeux de données à 1500 films et 1500 utilisateurs.

Bilan

Les travaux ont été menés de la façon suivante :

Kemayou : Pre-processing.

Kéran : Implémentation du code, rédaction des rapports, machine learning.

Khaled : présentation Streamlit, mise en forme des rapports.

Skander : implémentation du code, recherche sur les algorithmes de Deep Learning.

Suite du projet

Afin d'améliorer la performance des modèles, il pourrait être intéressant d'apporter plus d'informations sur les utilisateurs (collaboratif filtering) et sur les films (content-based filtering). Cela pourrait passer par des données sur l'âge, le milieu social-professionnel, le lieu de résidence etc. Pour ce qui est du contenu, il peut-être pertinent de prendre en compte l'année de sortie des films, et leur leur origine géographique par exemple.

Il serait également intéressant de poursuivre plus en profondeur l'utilisation du Deep Learning pour ce type de système de recommandation. Dans le cadre de nos travaux, le Deep Learning ne permet pas d'avoir des résultats plus intéressants que ce que donne la factorisation matricielle par l'algorithme SVD. Un changement de type d'architecture de modèle pourrait permettre d'explorer d'autres voies.

Enfin, réaliser un système Hybride permettrait de lier les performances obtenues en content-bases filtering et en collaborative filtering. L'enjeu sera alors de parvenir à pondérer au mieux les deux approches afin d'obtenir le système le plus performant possible.

Bibliographie

https://fr.wikipedia.org/wiki/Système_de_recommandation

<https://developers.google.com/machine-learning/recommendation>

<https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada>

<https://www.jillcates.com/pydata-workshop/html/tutorial.html>

https://fr.wikipedia.org/wiki/Filtrage_collaboratif

<https://www.imdb.com/list/ls050274118/>

<https://www.imdb.com/list/ls000055475/>

<https://www.imdb.com/list/ls053823383/>