RÉPUBLIQUE DU CAMEROUN

Paix-Travail-Patrie

Université de Yaoundé I

École Nationale Supérieure

Polytechnique de Yaoundé (ENSPY)

REPUBLIC OF CAMEROON

Peace-Work-Fatherland

The University of Yaounde I

National Advanced School of Engineering of Yaounde (NASEY)

ANI-IA 4048 : INTELLIGENCE ARTIFICIELLE ET APPLICATIONS :

RAPPORT DU DEVOIR SUR LE SYSTÈME DE RECOMMANDATION DE FILMS

Etudiants	Matricules
LOWE NYAT FRED NELSON	21P550
NGAKO DANGO AUDREY MARENE	21P561
NGUETSA TAMOKA RUSSELL YANSSEN	21P565

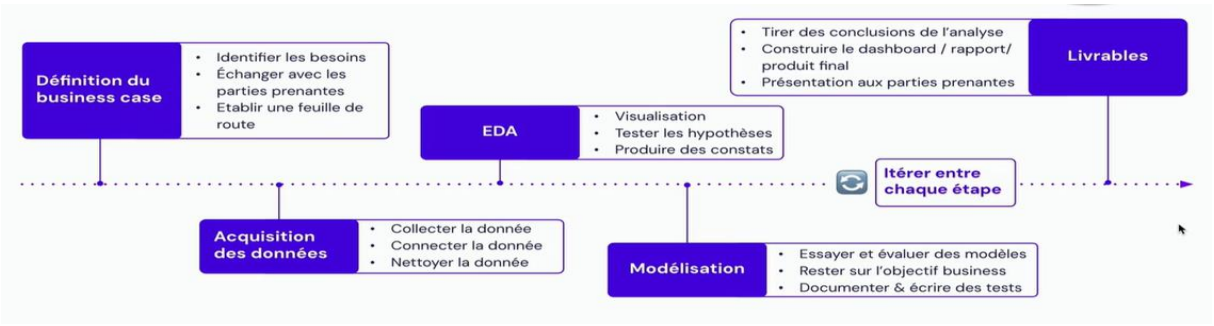
Table des matières

INTRODUCTION

Dans le cadre du cours d'intelligence artificielle et ses applications, il nous a été demandé de concevoir un système de recommandation de films fondé sur la similarité des préférences entre utilisateurs. Pour ce faire, nous avons exploité les données du fichier ml-latest.zip mis à disposition par la plateforme MovieLens.org, une référence dans le domaine des systèmes de recommandation.

Ce projet s'inscrit dans une démarche complète de Data Science, allant de la compréhension du besoin à l'évaluation des modèles. Avant toute modélisation, une étape préliminaire de cadrage a permis de clarifier les objectifs métier ainsi que la faisabilité technique du projet. Cela a impliqué :

- **Une analyse du besoin** : comprendre les attentes en matière de recommandation et les critères de succès attendus.
- **Une évaluation des données disponibles** : les données issues de MovieLens sont à la fois riches, pertinentes et de qualité, ne nécessitant ni collecte complémentaire, ni prétraitement complexe.
- **Une estimation des ressources nécessaires** : les modèles implémentés ne requérant pas de calculs lourds, une machine locale a suffi à l'exécution de l'ensemble du processus.



Notre approche repose sur un filtrage collaboratif basé utilisateur (User-User Filtering), en testant deux types de mesures de similarité : la similarité cosinus et la distance de Manhattan (Cityblock). Après avoir préparé et exploré les données, nous avons procédé à l'implémentation de ces modèles, puis à leur évaluation selon plusieurs métriques, telles que la précision à k (Precision@k) et l'erreur quadratique moyenne (RMSE).

Dans la suite de ce rapport, nous présenterons de manière structurée :

- Le prétraitement des données,
- Les analyses exploratoires réalisées,
- Les choix de modélisation ainsi que les résultats d'évaluation,
- Et enfin, les aspects techniques liés à l'implémentation.

Chaque section mettra en lumière les décisions prises, les outils utilisés, et les enseignements tirés tout au long de ce projet.

1. Prétraitement des données

Dans l'ensemble des fichiers .csv du fichier zippé, nous avons utilisé 2 ***rattings.csv et movies.csv.*** Afin d'avoir accès :

- Aux utilisateurs ayant déjà noté un film,
- A tous les films,
- A toute les notes sur les films.

Aussi, contrairement aux données du fichier movies.csv qui a été entièrement chargé, pour des raisons de performance de nos machines de travail, nous nous sommes limités aux **1000001** premières lignes du fichier rattings.csv sur un total de **33000000** (donc un ratio **3,03%** des notes totales) lignes afin de garantir :

- L'accès à la plupart des notes d'un utilisateur (d'où la non utilisation d'une méthode de sélection aléatoire),
- L'accès à tous les films,
- Le bon fonctionnement de la RAM.

Nous avons donc obtenu un total de **9561 utilisateurs ayant noté des films et 25825 films notés sur 86000 films au total sans valeurs manquantes**.

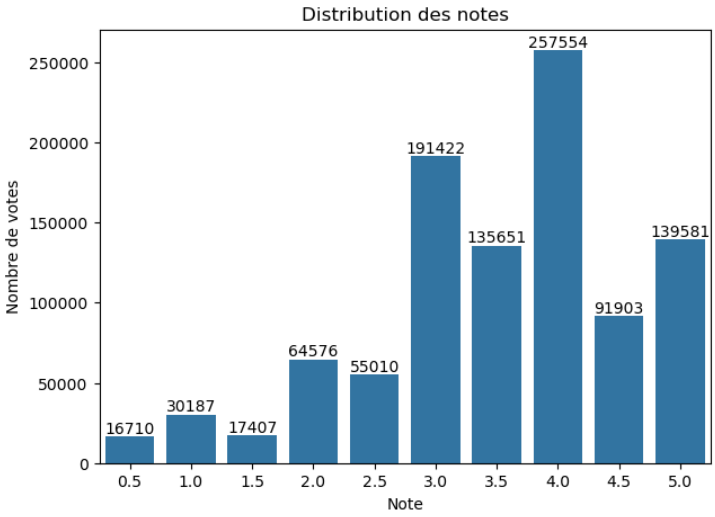
Par la suite, nous avons rencontré un problème d'identifiant après la fusion des 2 dataframe. C'est-à-dire, dans le dataframe, nous pouvions avoir des users 1, 2, 5, 7 sans avoir les users 3, 4 et 6. Ce qui créait problème lors de la création de la matrice user-item ? Pour ça, on a créé un tableau indice_user et un tableau indice_item qui contiennent les anciens id et les nouvelles id par exemple (1,2,5,6) => (1,2,3,4) puis on a ajouté deux colonnes sur le tableau principale qui contient ces nouveaux IDs. Ce traitement est très couteux, ça nous a pris presque 12 minutes.

A présent, passons aux analyses.

2. Analyses

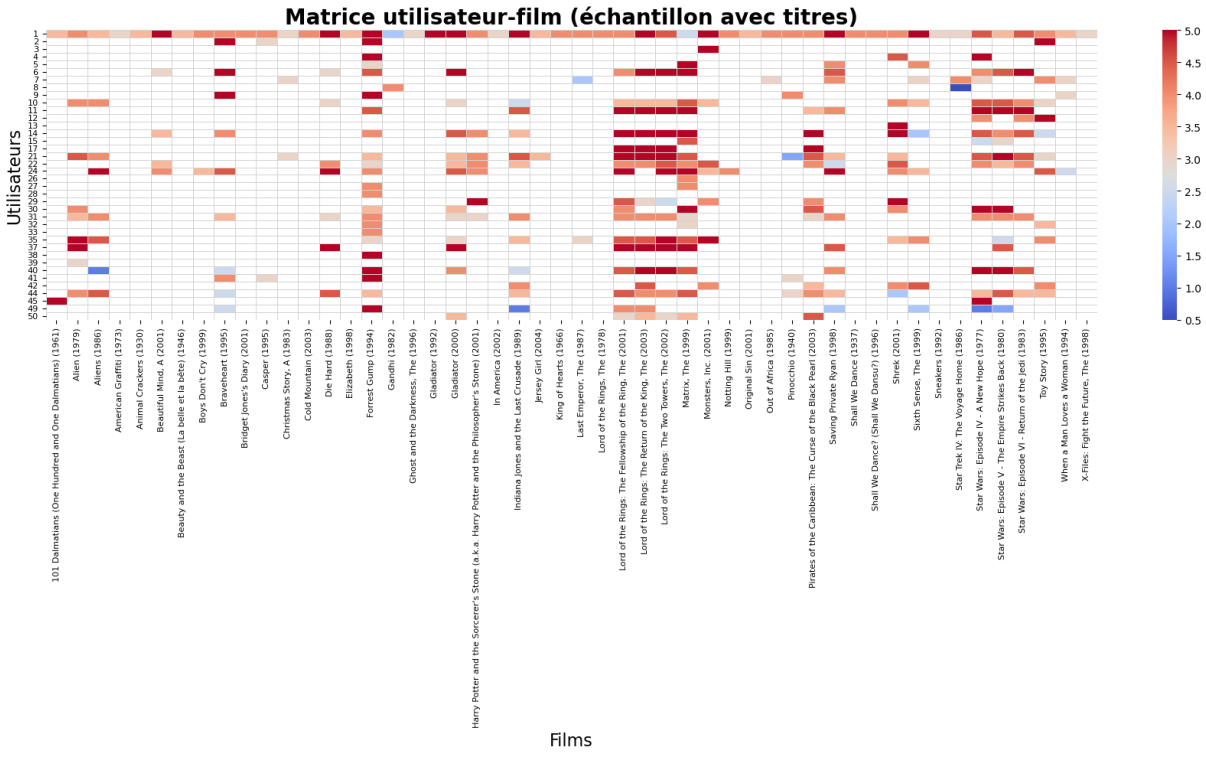
Nous avions eu à faire 5 analyses principalement :

- La distribution des notes :



On se rend donc compte qu'il y a très peu de vote faible mais beaucoup de vote fort **(principalement 4.0)**

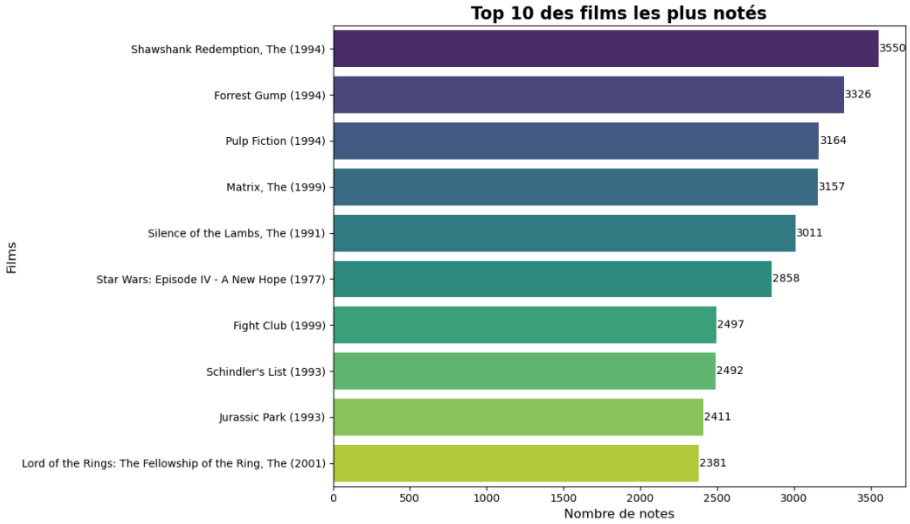
- La matrice user-film :



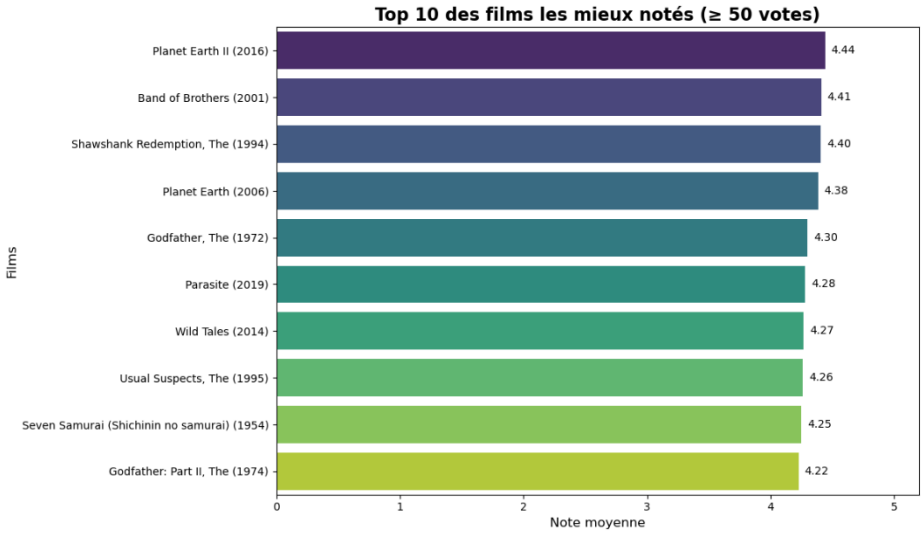
De cet échantillon de la matrice user-item, on constate qu'il y a beaucoup plus de notes positives que négatives. Les espaces vides montrent justement que la plupart des 30 premiers utilisateurs n'ont pas tous notés les 30 premiers films. Par contre si on utilisait toutes les lignes de notre dataframe, on aurait certes toutes les cellules colorées mais une matrice pratiquement illisible de par sa grandeur.

- Le taux de sparsité : ou parcimonie en français, est une caractéristique qui décrit une faible concentration ou densité au sein d'un ensemble ou d'une structure. Ceci dit, une faible sparsité caractérise un ensemble très touffu et serré. Dans notre cas, la sparsité est d'une valeur de **99,59%** ce qui montre que notre ensemble de donnée est très peu dense. Ceci peut aussi s'expliquer par le fait qu'il y a très peu de chance que des utilisateurs aient notés tous les films.

- Top 10 des films les plus notés :



- Top 10 des films les mieux notés :



Par la suite nous avons divisé l'ensemble de donnée en données de test (20%) et d'entrainement (80%) tout en maintenant la sparsité au maximum comme suit :

```
--- Sparsité de l'ensemble d'entraînement (ratings train) ---
Nombre d'utilisateurs uniques : 8912
Nombre de films uniques : 23814
Nombre total de notes : 795486
Nombre de combinaisons User-Movie possibles : 212230368
Taux de sparsité (train) : 0.9963 (99.63 %)
-------------------------------------------------
--- Sparsité de l'ensemble de test (ratings test) ---
Nombre d'utilisateurs uniques : 8912
Nombre de films uniques : 14036
Nombre total de notes : 203063
Nombre de combinaisons User-Movie possibles : 125088832
Taux de sparsité (test) : 0.9984 (99.84 %)
-------------------------------------------------
```

3. Modélisation et Evaluation

On va commencer par créer les modèles Memory based.

- Les modèles User-Based : "Les utilisateurs qui sont similaires à vous, ont aimé aussi ..."
- Les modèles Item-Based : "Les utilisateurs qui ont aimé ça, ont aimé aussi ..."

Pour expliquer plus :

- Le modèle User-Based : va prendre un utilisateur, trouve les utilisateurs les plus similaires à lui en se basant sur la note, Puis recommande les items aimé par ces utilisateurs (ça prend un user et retourne des items)
- Le modèle Item-Based : prend un item, cherche les utilisateurs qui ont aimé cet item, trouve les items aimé par ces utilisateurs (ça prend un item et retournes une liste des items)

Pour le faire, ont utilisé 2 métriques **le cosine similaire et cityblock** (distance de Manhattan). Pour le faire, on a commencé par créer les matrices user-item train et test. Ce sont les deux matrices qui vont croiser les notes de utilisateurs et des items. Puis, on a créé nos 4 modèles Memory Based à la fin, on a créé une fonction pour faire les prédictions selon le modèle.

Remarque : Lors de notre travail, nous avons eu à commenté les instructions chargées de créer les modèles Item_Based car il étai moins performant que les autres et n'entraient pas dans le cadre de ce travail qui consiste à se baser sur les préférences des utilisateurs.

Pour ce qui est de l'évaluation, nous avons utilisé 2 méthodes : **La precision@k et le RMSE (Root Mean Square Error).**

La précision@k mesure la proportion de recommandations parmi les K premières qui sont réellement pertinentes pour l'utilisateur :

$$Precision@K = (Nbre\ de\ recommendation\ pertinentes\ dans\ les\ K\ premières)/K$$

Par exemple si pour un utilisateur, on recommande 5 films et que 3 de ces films sont réellement pertinents (notés ≥ 4), alors : $Precision@K = 3/5 = 0.6$

Cependant, cette métrique d'évaluation n'était pas assez pertinente pour notre tâche car **elle dépend de fortement de K** (le nombre de recommandation voulue) et **ne tient pas compte des notes précises ou du score de confiance.** D'où ces résultats pour k = 5

```
Precision@5 (Cosine) : 0.0000
Precision@5 (Cityblock) : 0.0000
```

RMSE ou alors Erreur Quadratique Moyenne évaluer **l'exactitude des prédictions de notes** par le modèle :

$$RMSE = \sqrt{\left(\frac{1}{N}\right)\sum_{i=1}^{N}(\hat{r}_i - r_i)^2}$$

Avec :

- $\hat{r}_i$ = note prédite
- r_i = note réelle
- N = nombre total de prédictions

Son avantage est qu'il mesure directement l'erreur de prédiction et est utile pour l'affichage des notes prédites (ce qui entre très bien dans notre tâche). Ce qui suit donc présente les résultats du RMSE pour les 2 modèles :

```
RMSE (Cosine) : 0.9645
RMSE (Cityblock) : 0.9747
```

On remarque donc que selon le RMSE, le modèle basé sur la similarité cosinus fait moins d'erreur que celui basé sur la distance de Manhattan. C'est donc ce modèle dont on servira par la suite pour faire nos tests et principalement notre implémentation.

- Résumé des deux modèles de filtrage collaboratif (User-User) :

Modèle	Type de distance	Fonction utilisée	Principe
recommend_movies_user_user_sparse	Cosine Similarity	`cosine_similarity`	Compare l'orientation des vecteurs de notes entre utilisateurs (plutôt que leur valeur absolue). Deux utilisateurs sont similaires s'ils aiment/détestent les mêmes films, peu importe les notes exactes.
recommend_movies_user_user_cityblock	Cityblock (Manhattan) Distance	`pairwise_distances(metric='cityblock')`	Compare la distance absolue entre les notes des utilisateurs. Plus la somme des différences est faible, plus les utilisateurs sont similaires. Transformée ensuite en similarité par `1 / (1 + distance)`.

- Comparaison approfondie : Cosine Similarity vs Cityblock Distance :

Critère	Cosine Similarity	Cityblock (Manhattan) Distance
Définition	Mesure l'angle (l'orientation) entre deux vecteurs	Mesure la somme des différences absolues entre deux vecteurs
Formule mathématique	$\cos(\theta) = A \cdot B / (\|A\| \times \|B\|)$	$D = \Sigma\|a_i - b_i\|$
Échelle prise en compte	Non : normalise les vecteurs	Oui : prend en compte la valeur absolue des différences
Sensibilité aux notes	Faible : seuls les motifs sont importants (ex. tendances de vote)	Forte : deux notes proches sont considérées comme plus similaires
Principe	Deux utilisateurs sont similaires s'ils ont voté dans la même direction	Deux utilisateurs sont similaires s'ils ont donné des notes très proches
Valeur de sortie	Entre -1 et 1 (proximité vectorielle)	Distance réelle (plus petite = plus proche)
Type de mesure	Similarité directe	Distance qu'on convertit souvent en similarité par : 1 / (1 + distance)
Comportement typique	Regroupe les utilisateurs avec les mêmes préférences relatives	Regroupe les utilisateurs qui notent très pareil (mêmes niveaux)
Cas d'usage adapté	Recommandation basée sur goûts similaires	Recommandation basée sur proximité absolue des notations
Avantage principal	Ignore les biais de notation individuels (ex : note sévère ou indulgente)	Capte mieux les cas où les utilisateurs sont très rigoureux ou similaires numériquement
Inconvénient principal	Peut ignorer des écarts de valeur importants (ex : 2 vs 5)	Sensible aux biais de notation (note tout à 3 ou tout à 5)
Popularité dans les reco	Très utilisée (standard pour filtrage collaboratif User-User / Item-Item)	Moins fréquente, mais utile dans des modèles spécifiques

- Cas d'usage recommandé pour chaque modèle :

Cas d'usage	Cosine Similarity	Cityblock (Manhattan)
L'utilisateur note peu, mais les tendances générales sont importantes	Très adapté	Moins bon
On veut une comparaison précise des notes attribuées	Moins précis	Plus adapté
Les utilisateurs ont des styles de notation très différents (certains notent sévèrement, d'autres généreusement)	Insensible à l'échelle des notes	Sensible aux écarts
Données très sparse (peu de films en commun entre utilisateurs)	Robuste	Moins fiable
Performance observée (dans ton cas)	Meilleure précision et RMSE	Moins bon résultat

4. Implémentation

Pour l'implémentation sur interface web simple, nous avons eu recours à des bibliothèques telles que :

- Flask : pour la communication entre le modèle et l'interface.
- Joblib, pickle, panda, numpy et scipy pour recharger les paramètres sauvegardés pour la recommandation.

La page principale de l'interface propose à l'utilisateur d'entré son id et le nombre de recommandation qu'il souhaite. Puis après un laps de temps, il reçoit ses recommandations dans un tableau constitué :

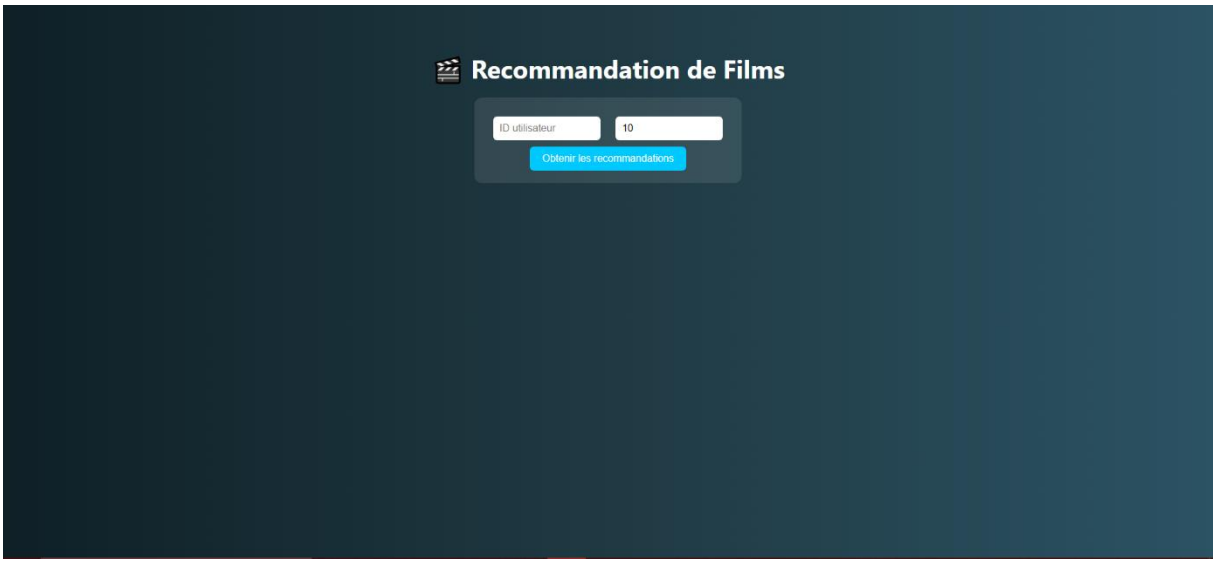
- Une colonne pour les noms des films recommandés avec pour la plupart des liens hypertexte qui mène à une page web dédié au visionnage de ce film (notamment le site themoviedb.org à partir du fichier ***links.csv***),
- Une colonne qui présente les notes susceptibles d'être attribuée aux films par l'utilisateur,
- Une colonne présentant le genre de films recommandés.

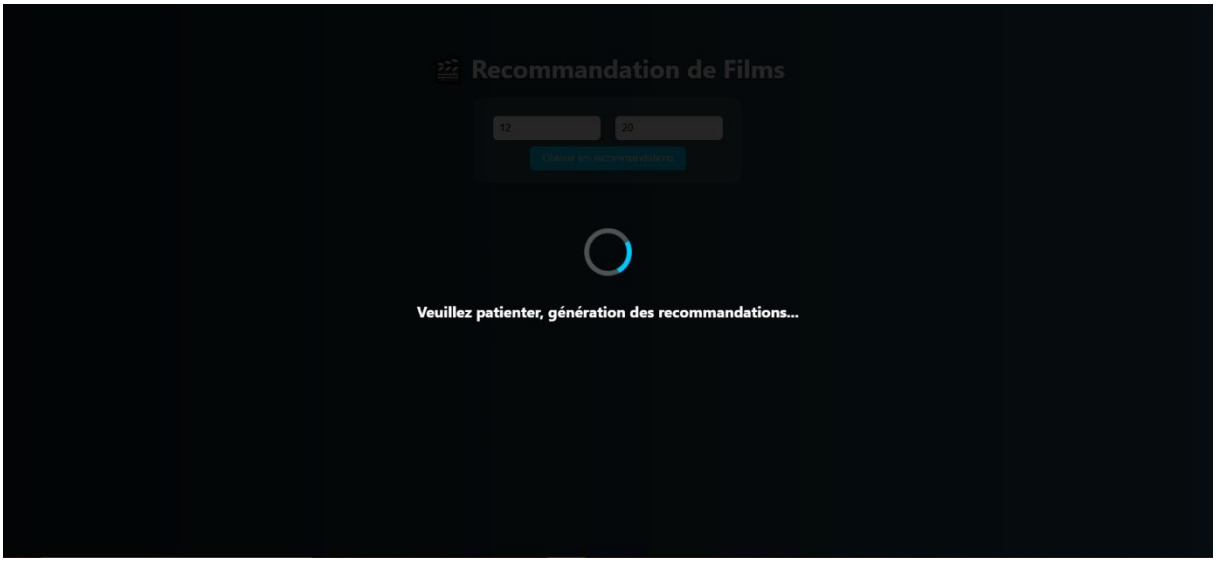
Il est à noté que le temps d'affichage des recommandations **de moins 60 secondes quel qu'en soit le nombre de recommandation voulue.**

Ci-dessous sont présenté quelques images de l'interface de notre implémentation.

Conclusion

En somme, après traitement, analyse et étude des données, nous avons abouti à un modèle de recommandation basée sur la similarité cosinus entre les utilisateurs afin de recommander les meilleurs films susceptibles d'être apprécié par le sujet de la recommandation avec une note probable qu'il pourrait appliquer à ce film. Puis nous avons implémenté cela sur une légère interface web qui prend en entré l'id de l'utilisateur et le nombre de films qu'il voudrait avoir comme recommandation ; il obtient donc en sorti les films recommandés, la note probable qu'il pourrait appliquer au film, ainsi que le genre du film.


Recommandation de Films
ID utilisateur
10
Obtenir les recommandations


Recommandation de Films
12
20
Obtenir les recommandations
Veuillez patienter, génération des recommandations...


Recommandation de Films
ID utilisateur
10
Obtenir les recommandations
Titre du film
Note prédite
Genres
In My Dreams (2014)
5.00
Drama|Fantasy|Romance
Ab-normal Beauty (2004)
5.00
Horror
Goth (2008)
5.00
Drama|Horror|Mystery|Thriller
Looking for Alibrandi (2000)
5.00
Drama
Kipchoge: The Last Milestone (2021)
5.00
Documentary
Beyond the Gates (2016)
5.00
Adventure|Horror
American Mary (2012)
5.00
Horror|Thriller
Friend Zone (2019)
5.00
Romance
Heart of Me, The (2002)
5.00
Drama|Romance
Dum Laga Ke Haisha (2015)
5.00
Children|Comedy|Romance
My Tomorrow, Your Yesterday (2016)
5.00
Drama|Romance
Day-O (1992)
5.00
Children|Comedy|Fantasy