**Cahier des charges MLOps**

1. **Contexte et objectifs**

Au cours de ce projet, l’application développée devra permettre de réaliser un système de recommandation de films pertinent basé sur une étude de similarité entre les personnes regardant les mêmes films et les similarités existantes entre films.

Le commanditaire de l’application est IMDB qui voudrait améliorer son système de recommandation de films existant afin de le rendre plus ciblé selon chaque utilisateur. En effet, il souhaiterait prendre en compte également des informations sur les préférences de chaque utilisateur et un système de notes.

Les utilisateurs de cette application sont les personnes susceptibles d’être intéressées par le fait de regarder des films mais n’utilisant pas de plateformes de streaming tel Netflix ou Amazon Prime car proposant déjà un service de ce genre sur leur plateforme. Les utilisateurs ne sont donc peut-être pas forcément à l’aise avec l’outil informatique, cette application doit donc être assez intuitive et simple à utiliser.

L’administrateur de cette application sera le gérant du site IMDB.

Cette application sera intégrée en backend sur un site web.

Elle sera utilisée à partir d’une interface graphique/ API.

## **Modèle**

Les modèles étudiés sont spécifiques au système de recommandation. Ils sont basés sur des filtrages :

1. **Le premier type est basé sur le contenu ou Content-based filtering** : il repose sur une étude des similarités existantes au niveau des caractéristiques des films pour établir les recommandations. En pratique, chaque film est reconnu pour être plus ou moins apprécié par un utilisateur selon plusieurs caractéristiques tels que le genre, le réalisateur, les acteurs, etc. C’est ensuite sur la base de leurs préférences pour certains films que sont recommandés une liste de films présentant des caractéristiques proches.

**(+) :** Nécessite seulement d’avoir des informations sur les films.

**(-) :** Nécessite que les données renseignées de tous les films soient du même genre.

Nous utilisons un modèle Nearest Neighbors de la librairie scikit-learn qui permet de définir les plus proches voisins donc les films ayant les caractéristiques les plus proches au film entré en paramètre.

1. **Le deuxième type est basé sur le filtrage collaboratif** ou **Collaborative filtering** : il repose sur une étude des similarités existantes au niveau des caractéristiques des utilisateurs permises par une collecte de données individuelles (sur la base de leur comportement sur leur site, leur profil…). L’une des manières les plus utilisées est de prendre comme caractéristiques les notes mises par les utilisateurs pour chaque film et de prédire les notes sur des films non vus afin de recommander une liste de films à regarder. Le modèle utilisé est le SVD et est issu de la librairie surprise, une librairie Python conçue pour les systèmes de recommandations.Ce modèle se base sur le principe de factorisation de matrice et minimise les erreurs grâce à la descente de gradient stochastique.

L’hypothèse est qu’en pratique, des utilisateurs ayant des préférences similaires sur certains éléments en ont probablement sur d’autres. L’objectif est ainsi de constituer des groupes d’utilisateurs à préférences similaires. Pour deux utilisateurs donnés, si les deux apprécient beaucoup de films en commun, alors un film vu et apprécié par l’un sera fortement recommandé au second.

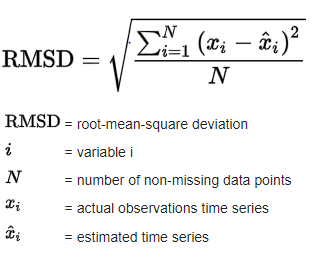
**(+) :** Les préférences de tout un chacun évolue dans le temps, il faut donc que le système puisse évoluer également dans le temps ce qui est rendu possible par ce système collaboratif.

**(-) :** Si un nouvel utilisateur utilise le système de recommandations ou si cet utilisateur présente des préférences/comportements différents à ceux précédemment étudiés, alors aucune similarité proche est possible, c’est ce que l’on appelle le “cold start”.

1. Les deux modèles utilisés ci-dessus présentant chacun leurs avantages et leurs limites, des systèmes de **recommandations hybrides** sont de plus en plus utilisés. Le principe est d’utiliser ces deux modèles simultanément afin d’en contourner certaines limites apportées par ces modèles pris séparément.

(-) : deux approches simultanées + ressources en calcul pour traiter données provenant de deux méthodes

Au final, le système de recommandation créé se basera sur le contexte afin de choisir l’une des trois approches présentées ci-dessus.

Le besoin du client est d’avoir une réponse instantanée après avoir saisi sa demande. Or sur cette base de données et ses 19 millions de lignes, le temps d’entraînement est relativement long. Il faut donc charger le modèle entraîné et prédire directement la sortie du modèle. Cela nécessite donc également une vigilance sur le modèle entraîné et les prédictions faites sur de nouvelles données éventuellement : si les données d’entraînement s’écartent trop des nouvelles données insérées, alors le modèle perdra en performance : c’est le problème du data drift.

L’évaluation du modèle SVD se fera grâce à l’étude de la RMSE (Root-Mean-Squared-Error) qui vient calculer la différence entre les notes connues pour une liste de films vus et les prédictions faites sur ces mêmes films. La RMSE ne donne qu’une simple valeur, afin de juger le reflet de sa valeur sur les données, il a été décidé de la retrancher par rapport à la valeur moyenne de la base de données. Nous avons obtenu un score entre 20 et 25% et une RMSE aux alentours de 0.8 sur les premiers entraînements ce qui est plutôt correct comme qualité de prédictions.

L’évaluation du modèle Nearest Neighbors se fera grâce à la distance euclidienne entre le film entré en paramètre et le film recommandé. Le seuil à ne pas dépasser étant de 3.0 sur la moyenne des 20 recommandations.

L’une des contraintes liées à ce projet était que la prédiction faite par le modèle soit rapide : l’utilisateur de la future application souhaite faire une recherche qualitative mais également rapide de films basée sur ses préférences, les similarités entre films ou bien les films les plus tendances. Pour cela, il a été décidé d’entraîner le modèle en amont et de sauvegarder ce modèle dans un fichier que l’on aura plus qu’à charger lors de l’utilisation de l’API. La prédiction prend seulement quelques secondes pour retourner une liste de 20 films se basant sur un utilisateur et un film.

1. **Base de données**

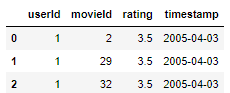
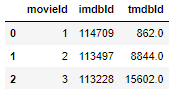
Les données qui ont été utilisées sont issues de deux sources web différentes :

* [MovieLens](https://grouplens.org/datasets/movielens/20m/) contient 20 millions de notes sur 27000 films différents. Plus de 138 000 personnes choisies aléatoirement ont alimenté cette base de données et noté au moins 20 films chacun. Les données ont été récoltées entre 1995 et 2015. Cette base de données contient 6 fichiers différents : links.csv / movies.csv / ratings.csv
* [IMDB](https://developer.imdb.com/non-commercial-datasets/) contient des caractéristiques de films (année de sortie, genre, réalisateur(s), écrivain(s),durée, note moyenne, etc). Cette base de données contient 7 fichiers différents : title.ratings / title.episode / title.crew / name.basics / title.akas / title.basics / title.principals. Cette source d’information est mise à jour quotidiennement et téléchargée toutes les semaines afin de mettre à jour la base de données des modèles utilisés.

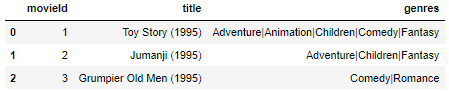
Une fois ces deux bases de données nettoyées et liées, nous avons un dataframe de plus de 5 millions de données donc très conséquents.

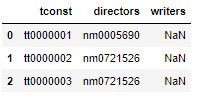
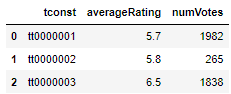
Dans le cadre de ce projet et l’hypothèse d’une éventuelle continuité dans le projet, il faut réfléchir à l’actualisation possible de données et l‘acquisition de nouvelles données. Par rapport aux informations récoltées des films, il s’agirait de mettre à jour la base de données IMDB régulièrement. Pour ce faire, nous utiliserons Airflow afin qu’un dag soit exécuté chaque semaine et vienne mettre à jour les données IMDB. En ce qui concerne les notes et les utilisateurs, il faudrait permettre l’acquisition de nouvelles notes et de nouveaux utilisateurs dans l’API avec la création d’un utilisateur avec authentification et la possibilité par la suite d’ajouter des films vus ainsi qu’une note pour chacun d’entre eux, les films étant normalement dans la base de données remises à jour quotidiennement par IMDB.

Un schéma d’architecture des données est présenté ci-dessous :

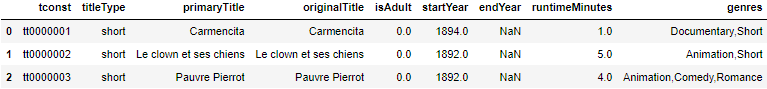


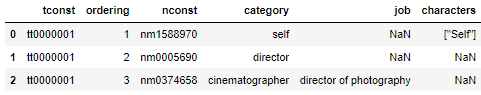
MovieLens





IMDB





La base de données MovieLens peut être fusionnable grâce à la colonne *movieId* tandis que la base de données IMDB l’est grâce à la colonne *tconst* représentant l’identifiant du film IMDB. Ces deux bases de données sont fusionnables grâce à cette même colonne *tconst* et la colonne *imdbld.*

1. **API**

Plusieurs routes sont disponibles sur cet API. Les premières concernent les utilisateurs de la base de données :

* /new\_user → un user est ajouté avec génération d’userId et password
* /delete\_user → un user peut être supprimé de la base de données
* /update\_user → possibilité de modifier les informations du user

Cette base de données utilisateur est conservée dans le fichier database.db et est issue de SQlite3. Si le fichier database.db est inexistant, il peut être créé en lançant le fichier init\_db.py.

D’autres routes concernent les notes mises par ces utilisateurs :

* /new\_rating → possibilité d’ajouter une note à un user existant selon un film
* /delete\_rating → une note peut être supprimé de la base de données
* /update\_rating → il est possible de modifier une note s' il y a eu une erreur lors de sa création ou si lors d’un second visionnage, la première note ne correspond plus à celle que veut laisser le user.

Cette base de données est également disponible dans le fichier database.db.

Enfin la dernière route concerne le système de recommandation en lui même :

* /recommendation\_system → génère une liste de films recommandés en fonction d’un user id et d’un film en utilisant l’un des modèles détaillé dans la partie 2.

1. **Testing & Monitoring**

Les tests unitaires mis en oeuvre visent à tester le bon fonctionnement du/de la :

* **Processus de construction de la base de données :** le bon téléchargement des bases de données initiales, la phase de préprocessing pour une bonne fusion en une unique base de données contenant en particulier les variables clés et sous le bon format
* **Phase modélisation :** le preprocessing spécifique à chaque modèle, le respect du seuil de performance, la bonne attribution du modèle en fonction

des informations disponibles. En effet, dans le cadre des systèmes de recommandation le modèle retenu sera de type : basé sur le contenu (content based filtering) si des informations sont disponibles sur le contenu du films et pas sur les utilisateurs et basé sur un filtrage collaboratif *(collaborative filtering)* dans le cas inverse. Enfin dans le cas d’informations conjointement disponibles, il devra être fait appel à une modélisation hybride

* **Phase API :** le bon fonctionnement des différents endpoints et le mauvais fonctionnement quand les circonstances requises ne sont pas d’actualité, la bonne mise en forme des inputs collectés à travers les activités du user (ex: mail, mot de passe)

Les performances du modèle sont évaluées sur un jeu de test.

Les jeux tests et train pourront être actualisés pour tenir compte des nouvelles informations récoltées à travers l’API. Plus précisément, le jeu test sera systématiquement actualisé pour tester la qualité des prédictions sur de nouvelles données. Le jeu train, et donc le processus d’entraînement, sera lui réactualisé lorsque les performances du modèle auront été significativement altérées avec un RMSE >=1.

De plus, le réentraînement du modèle se fera de manière hebdomadaire, en parallèle de la mise à jour des données IMDB avec Airflow, avec une réactualisation des deux jeux de données, pour permettre cette fois, une optimisation régulière du modèle basée sur la collecte de nouvelles informations en continue.

Enfin, en cas de non atteinte du seuil de performance, une alerte sera envoyée aux développeurs afin de les inviter au réentraînement du modèle.

1. **Schéma d’implantation**

