DataScientest • com

Cahier des charges projet MLOps

1) Contexte et Objectifs

Contexte et objectifs du projets

OBJECTIFS

La société MovieLens possède un système de recommandation basé sur le web et une communauté virtuelle qui recommande des films à ses utilisateurs en fonction de leurs préférences. Actuellement leur site contient environ 11 millions de notes pour environ 8500 films et leur système de recommandation vieillissant nécessite une mise à jour.
Dans le cadre du développement du site web de la société MovieLens , nous avons été mandatés pour créer et mettre en place un nouveau système de recommandation de haut niveau afin d'améliorer les qualités de service du site et d'attirer de nouveaux clients .
Ce système de recommandation devra optimiser le fonctionnement sur plusieurs thèmes notamment :

- La recommandation de films personnalisés au profil de l'utilisateur tout en veillant à ne pas trop l'enfermer dans ses goûts (limiter l'effet bulle)
- Limiter au maximum l'effet coldstart pour les nouveaux films ajoutés à la base de données MovieLens
- Une refonte de l'interface pourra être envisagée
- les utilisateurs seront les utilisateurs du site MovieLens
- l'application restera intégrée au site web MovieLens

MODELE

Le système de recommandation utilisera deux modèle NearestNeighbors :

- Une approche centrée sur les utilisateurs, c'est-à-dire un modèle qui cherche des utilisateurs qui ont des comportements similaires avec l'utilisateur à qui l'on souhaite faire des recommandations. Les notes des utilisateurs similaires seront utilisées pour calculer une liste de recommandations pour cet utilisateur. La prédiction sera réalisée à partir d'analyse des matrices croisées de films et notations.

- Une approche centrée sur les films, dans ce cas le modèle appliquera la fonctionnalité : "les gens qui vu le film x ont aussi vu le film y", ainsi le modèle recommandera des films qui ont été vus par des utilisateurs qui ont vu le film x et les notes seront utilisées pour calculer les recommandations.

Métrique de performance : la nature du modèle ne permet pas d'utiliser une métrique classique, il nous sera donc nécessaire d'en construire une.
On pourra par exemple vérifier en feedback à t+ x jours après la recommandation que les utilisateurs aient regardé et noté un ou plusieurs films recommandés par les propositions passées. Puis à partir de ces informations construire une métrique permettant de vérifier la performance du système.

Robustesse : de la même façon que pour la métrique de performance nous n'avons pas de métrique de robustesse du modèle classique. Mais nous pourrons vérifier que les même recommandations se retrouvent plusieurs fois si on lance plusieurs prédictions avec le même profil

Temps d'entraînement : notre utilisation du modèle ne nécessite pas un entraînement très fréquent. On pourra par exemple demander un nouvel entraînement tous les 1000 feedback utilisateur (valeur à déterminer) ou tous les mois en fonction des nouveaux ajouts de film. Dans tous les cas nous ne serons que peu impacté par le temps d'entraînement, dont la durée n'aura pas à être challengé tant qu'elle reste en dessous des 24h

Accès aux modèle et données pour les différents utilisateurs
L'administrateur de l'application sera l'administrateur du site MovieLens qui aura accès aux données statistiques, l'accès à la partie "backend" sera réservée à nos services techniques en charge du SAV.
La recommandation de film sera être accessible et utilisable depuis le site MovieLens, par l'intermédiaire d'une interface graphique.
Cette interface devra être accessible à partir d'un bouton de type "Demander une recommandation de film" depuis le site internet. En sortie une liste de 1 à 5 films (à définir) devra être proposé à l'utilisateur, avec une image du film, son synopsis, année de sortie, acteurs et réalisateurs (résumé de type imdb dont la base de données fait partie des datas accessibles) et une proposition de mettre une note à chaque film recommandé.

Les films recommandés aux utilisateurs enregistrés pourront être stockés afin de lui proposer une notation ultérieure par feedback.
Il pourrait éventuellement être possible que l'utilisateur donne directement une notation au film proposé dans le cas où il l'aurait déjà vu.
Cela permettra d'une part d'alimenter la base de données du site, et d'autre part d'enrichir le profil de l'utilisateur afin d'être plus efficace dans les recommandations ultérieures.

Pour la recommandation sur profils les utilisateurs devront avoir un profil enregistré sur movieLens.

2) Base de données

La base de données est disponible à partir de ce lien :
https://grouplens.org/datasets/movielens/20m/

Les données disponibles sont répartis dans 6 fichiers :

- “**ratings**” contenant pour chaque utilisateur l’ID du film noté avec sa note et un timestamp
- “**tags**” contenant les tags attribué par les utilisateurs pour chacun des films qu’ils ont notés
- “**moviesID**” contenant permettant de relier chaque ID de film au titre de film, avec son année de sortie intégré dans la colonne title construite de la façon suivante : “nom du film” (“année de sortie”)
- “**genome-scores**” contenant pour chaque ID de film l’ID de tag qui lui a été attribué par un algorithme de machine learning avec la pertinence de ce tag, avec une mesure numérique de la pertinence entre 0 et 1 (1 étant le maximum).
- “**genome-tags**” contenant pour chaque ID de tag de “génome-score” sa correspondance.
- “**links**” donnant la correspondance entre les ID de film utilisé sur movieLens et les ID imbd et tmdb.

Notre modèle utilisera en priorité les données donnant les nom de film, les id d’utilisateurs, les notations de film par utilisateur, les tags génome et tags utilisateurs.
On pourra éventuellement utiliser en plus la pertinence des tags, le genre de chaque film et leur année de sortie suivant les améliorations désirées.
L’information de timestamp pourra éventuellement être utilisée pour proposer aux utilisateurs des films de même genre que ceux visionnés récemment - et proposer en priorité des suites de film dans des cas de film type saga.

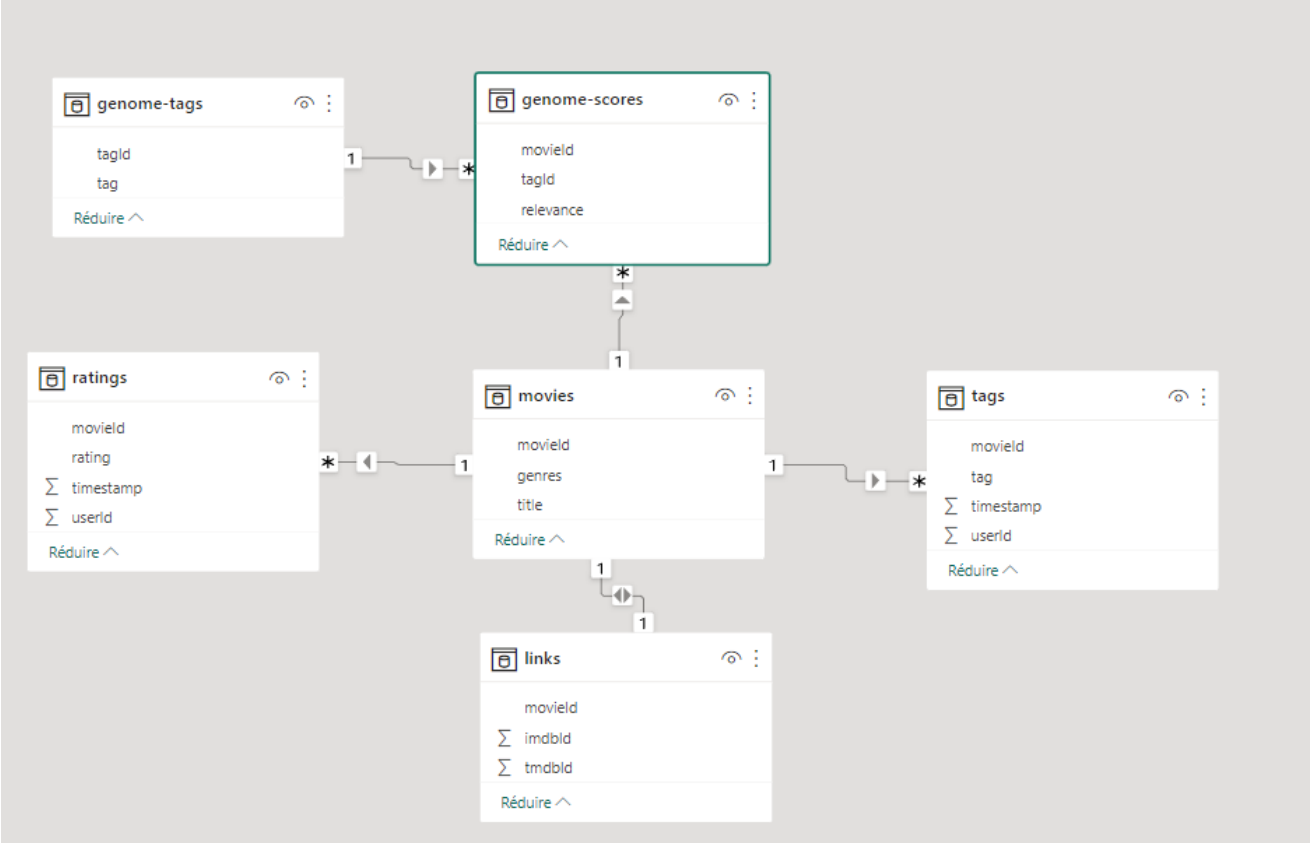
Les données disponibles sur IMBD et TMDB ne seront pas utilisées dans un premier temps.

Cette base de données sera amenée à évoluer à partir des nouvelles notations d’utilisateur et de l’ajout de nouveaux films.
Les utilisateurs pourront attribuer des notes et tags aux films au fur à mesure de leur utilisation de la plateforme.
Le système pourra éventuellement les inciter à attribuer de nouvelles notes et tags

aux films qui leur auront été proposés en affichant sur la page d'accueil du site une partie "évaluation des films recommandés". Cette partie contiendra la liste des films recommandés précédemment à l'utilisateur afin de lui permettre d'accéder plus facilement à leur notation.
La notation des films précédemment recommandés nous sera utile pour évaluer la robustesse et la performance des recommandations du modèle, il sera donc nécessaire que l'utilisateur y ait un accès facile et incitatif.

L'ajout de nouveau film sera étudié avec attention afin de limiter le "ColdStart", on pourra par exemple inciter l'utilisateur à regarder ces nouveautés en les proposant lors de sa connexion au service de recommandation.

Architecture de la base de données :



3) API

Les différents accès à la base de données et aux métadonnées du site web Movielens se feront par l'intermédiaire plusieurs API (application programming interface) reparti en plusieurs groupes à des degrés de sécurité différentes selon les utilisateurs.

Pour la consultation et l'échange des données nous utiliserons le protocole https que le site utilise déjà.

Les API à utiliser seront les suivantes :

- Une route de base pour vérifier le fonctionnement de l'API
- Une route par modèle pour avoir une prédiction (modèles random, user et movie), pour lesquelles l'authentification sera nécessaire
- Une route pour accéder à l'historique des prédiction d'un utilisateur pour laquelle l'authentification sera nécessaire
- une route pour donner une nouvelles notation pour laquelle l'authentification sera nécessaire
- Une route permettant d'avoir une liste des films les mieux notés pour un genre prédeterminé, pour laquelle l'authentification sera nécessaire
- Une route pour la création d'un nouvel utilisateur et pour laquelle on demandera une notation minimum de 3 films lors de la création

4) Testing & Monitoring

- **Tester le bon fonctionnement des modèles :**

Le modèles random n'aura pas de test unitaire.
Les modèles user et movie auront tous les deux 2 tests similaires pour vérifier leur stabilité ainsi que la diversité des proposition :

- un premier test de stabilité pendant lequel nous lanceront 100 prédictions avec une sortie de 10 films. (à partir d'un utilisateur ou d'un film à identifier). Le critère de validité sera qu'au moins un film soit présent sur au moins 80% des prédictions.
- une second test ou l'on comparera les prédictions avec 10 films en sortie pour 3 films ou 2 utilisateurs. Et on vérifiera qu'au moins un film est en écart sur l'ensemble des prédictions pour s'assurer que les modèles ne réaliser pas toujours la même prédiction.

- **Tester le bon fonctionnement des différents endpoints de l'API**

Des test unitaires seront effectués sur les endpoint de l'API, et notamment son fonctionnement et son temps de réponse seront validés.
Les tests API seront lancés à chaque push github, lorsqu'une mise à jour sera réalisée.

- **Tester le bon fonctionnement du process d'ingestion de nouvelles données**

La base de données est testé à chaque fois qu'elle est misé à jour, par lancement de test unitaire sur github action.
On vérifiera notamment qu'elle n'est pas vide, qu'elle comporte le bon nombre de colonne et que les format des variable de chaque colonne est respecté.

- **Quand faut-il ré-entraîner le modèle ? (périodiquement, lorsque les performances sont trop faibles)**

2 types d'upgrade seront à réalisé :
    - soit tous les 1000 feedback réalisé par les utilisateur
    - soit tous les mois

La performance n'est pas un critère de réentrainement étant donnée la nature du modèle.

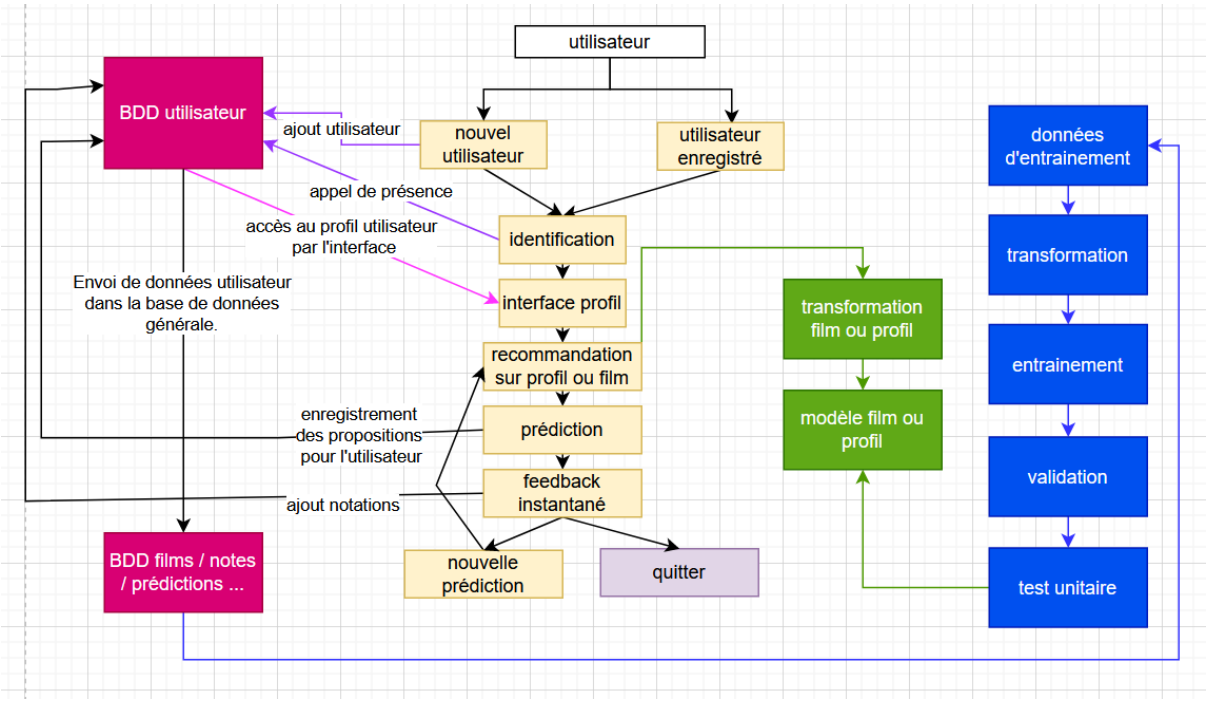
- **Sur quelles données faut-il ré-entraîner le modèle ? (sur l'intégralité du jeu de données, sur un échantillon des données les plus récentes…)**

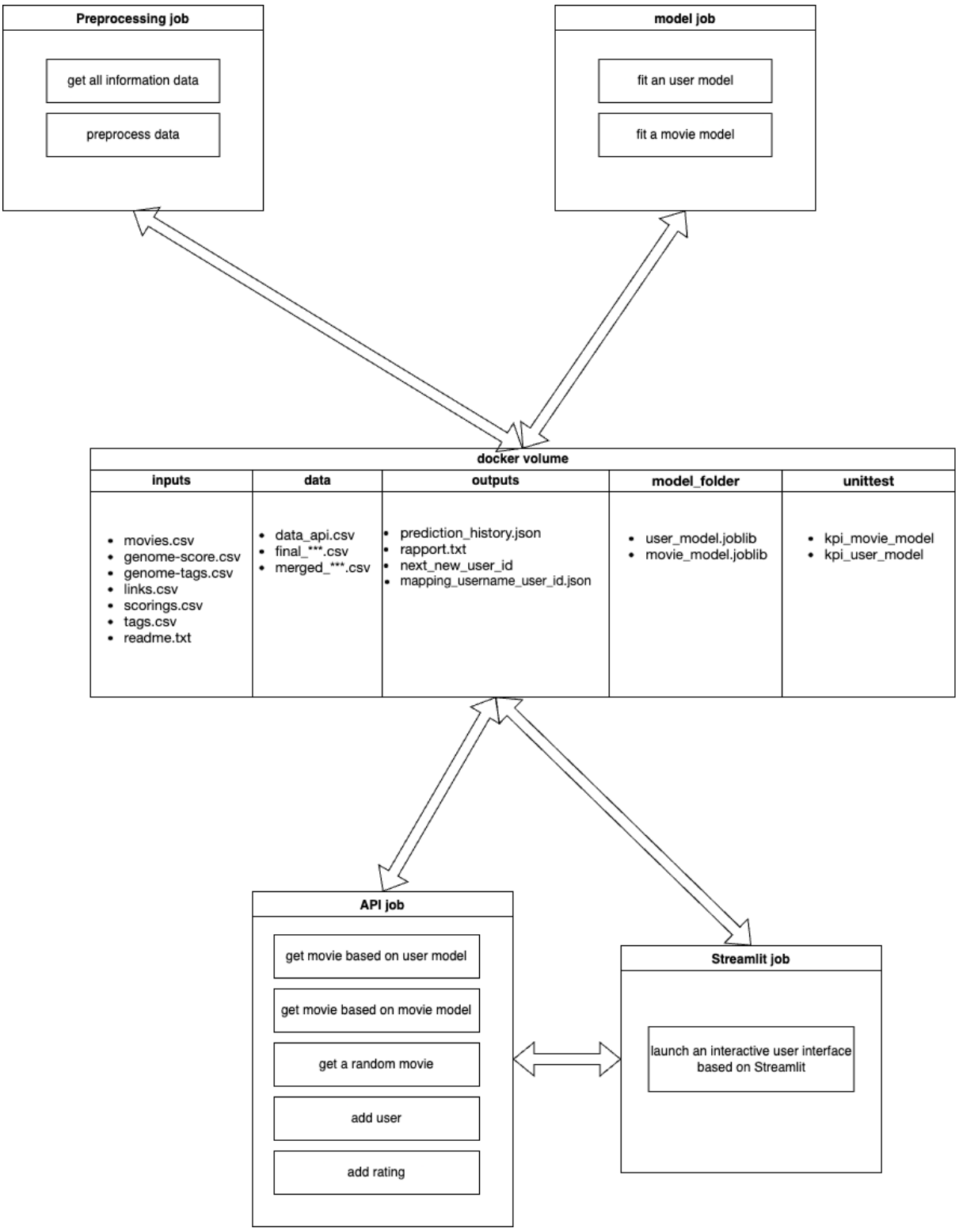
sur l'intégralité du jeu de données

- **Que faire lorsque le modèle n'atteint pas le seuil de performance requis ? (envoyer un mail d'alerte aux personnes concernées, bloquer l'application)**
  pas de test de la performance en direct.

Il n'y pas de criticité à utiliser un modèle peu performant ; d'autant que la performance sera difficilement évaluer en temps réel. Si un problème est détecté, un mail d'alerte automatique sera envoyé à l'administrateur du site afin qu'il puisse décider de la procédure à suivre.

5) Schéma d'implémentation




Preprocessing job
get all information data
preprocess data
model job
fit an user model
fit a movie model
docker volume
inputs
data
outputs
model_folder
unittest
movies.csv
genome-score.csv
genome-tags.csv
links.csv
scorings.csv
tags.csv
readme.txt
data_api.csv
final_***.csv
merged_***.csv
prediction_history.json
rapport.txt
next_new_user_id
mapping_username_user_id.json
user_model.joblib
movie_model.joblib
kpi_movie_model
kpi_user_model
API job
get movie based on user model
get movie based on movie model
get a random movie
add user
add rating
Streamlit job
launch an interactive user interface based on Streamlit