

Recommandation

de Films dans la plateforme

MoviesForYou



Préparé par : AMIN ELFAQUIRI



PLAN

- 01 INTRODUCTION
- 02 COMPRÉHENSION DE BUSINESS
- 03 COMPRÉHENSION DES DONNÉES
- 04 TECHNOLOGIES UTILISÉES
- 05 DATA SOURCE 'API '
- 06 CRÉATION DU PRODUCER
- 07 CREATION DE CONSUMER
- 07 VISUALISATION AVEC KIBANA
- 08 CONSTRUCTION DU MODÈLE "ALS"
- 09 RECOMMANDATION L'API
- 10 CONCLUSION

INTRODUCTION

Le projet de recommandation de films de MoviesForYou est une initiative ambitieuse visant à intégrer les technologies Big Data et Intelligence Artificielle pour améliorer les suggestions de films à l'aide des données de MovieLens. Cette solution repose sur l'utilisation d'Apache Spark, Elasticsearch, Kibana, et Flask pour créer une expérience utilisateur personnalisée et dynamique.

Expression des besoins :

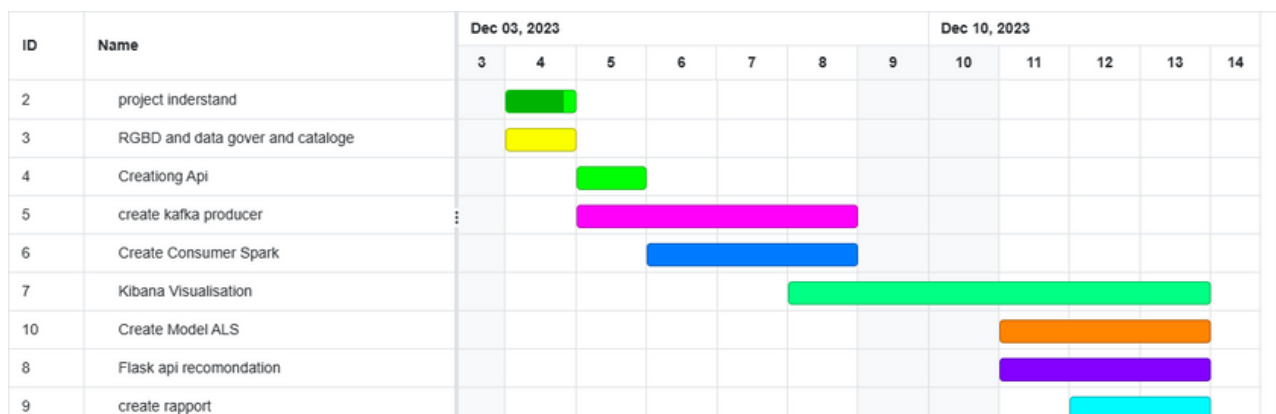
Objectif Global du Projet :

- Développer un système de recommandation de films en temps réel, intégrant des technologies Big Data et IA, pour offrir une expérience utilisateur personnalisée.

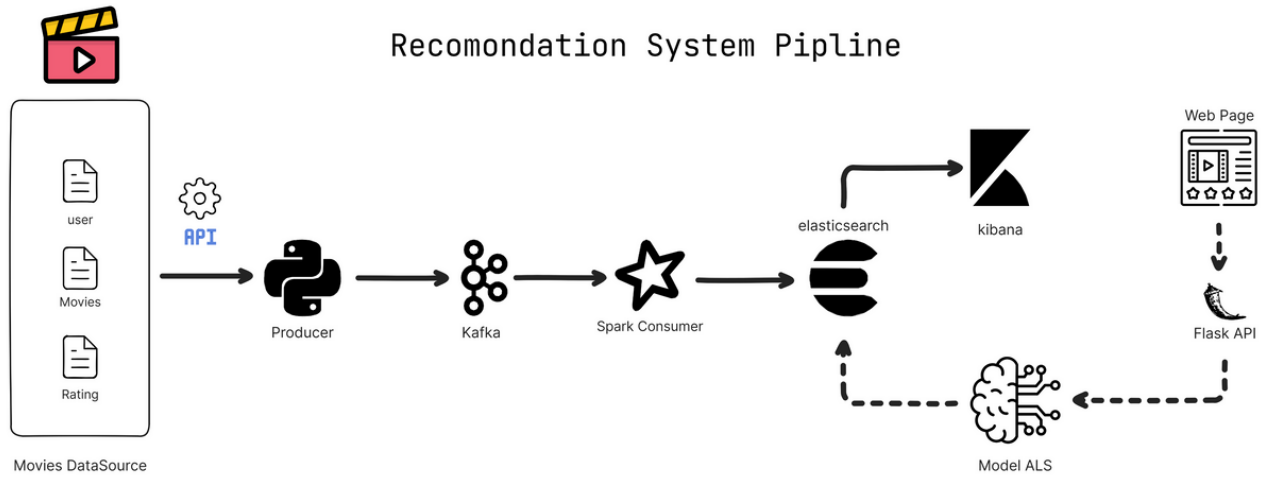
Fonctionnalités Principales :

- Système de recommandation de films basé sur les données de MovieLens.
- Intégration des technologies Apache Spark, Elasticsearch, Kibana, et Flask.
- Visualisation des données à travers des tableaux de bord interactifs.

Planification :



Workflow :



COMPRÉHENSION DE BUSINESS

MoviesForYou, s'engage à maximiser la rétention des utilisateurs existants et à attirer de nouveaux utilisateurs sur leur plateforme de divertissement. L'équipe de développement explore les données dans le but de comprendre les comportements des utilisateurs et d'identifier des stratégies pour atteindre ces deux objectifs clés.

Analyse de Rétention :

L'équipe commence par analyser les données des utilisateurs qui sont restés actifs sur la plateforme pendant une période prolongée. Ils examinent les genres de films préférés, la fréquence de visionnage, et les périodes d'activité pour identifier des tendances.

Identification des Points de Départ :

Les développeurs étudient les données des nouveaux utilisateurs et identifient les points d'entrée les plus courants, tels que les recommandations initiales ou les fonctionnalités populaires. Ils cherchent à comprendre comment ces points d'entrée influent sur la rétention.

Analyse Comparative :

En comparant les données des utilisateurs fidèles avec celles des nouveaux utilisateurs, l'équipe cherche des différences significatives. Cela pourrait inclure des variations dans les genres préférés, la fréquence de la notation des films, ou les interactions avec d'autres fonctionnalités de la plateforme.

Segmentation Utilisateur :

En utilisant des critères tels que la fréquence d'utilisation et les préférences de genre, l'équipe segmente les utilisateurs en groupes distincts. Ces segments aideront à personnaliser les recommandations et les stratégies de rétention.

COMPRÉHENSION DES DONNÉES

Afin de mieux comprendre les données utilisées dans le projet de recommandation de films de MoviesForYou, nous examinons les différents fichiers fournissant des informations sur les utilisateurs, les films et les évaluations.

u.data - Jeu de Données Complet:

- Le fichier u.data contient l'ensemble complet de données avec 100 000 évaluations attribuées par 943 utilisateurs à 1682 films.
- Chaque utilisateur a évalué au moins 20 films.
- Les données comprennent les colonnes suivantes : user id | item id | rating | timestamp.
- Les horodatages sont en secondes Unix depuis le 1/1/1970 UTC.

u.info - Informations sur le Jeu de Données:

- Fournit le nombre total d'utilisateurs, d'articles (films) et d'évaluations dans le jeu de données.
- Offre un aperçu global des dimensions du jeu de données.

u.item - Informations sur les Films:

- Contient des détails sur les films sous forme d'une liste tabulée.
- Chaque ligne comprend les informations suivantes : movie id | movie title | release date | video release date | IMDb URL | ...
- Les 19 derniers champs indiquent les genres, avec 1 indiquant que le film appartient à ce genre, et 0 indiquant le contraire.
- Les identifiants de film correspondent à ceux du jeu de données u.data.

u.genre - Liste des Genres:

- Fournit une liste des genres possibles, utilisée pour interpréter les informations dans le fichier u.item.

u.user - Informations Démographiques sur les Utilisateurs:

- Donne des informations démographiques sur les utilisateurs, sous forme d'une liste tabulée.
- Comprend les colonnes suivantes : user id | age | gender | occupation | zip code.
- Les identifiants d'utilisateur correspondent à ceux du jeu de données u.data.

Pour plus d'informations sur la source de données, la gouvernance des données et le RGPD,
[cliquez ici](#)

TECHNOLOGIES UTILISÉES

Afin de réaliser le projet de recommandation de films de MoviesForYou avec succès, une configuration appropriée de l'environnement de développement et l'utilisation de technologies spécifiques sont nécessaires.

Apache Spark :

- **Utilité:** Apache Spark est utilisé pour le traitement des données massives. Il facilite le chargement, le nettoyage, et l'ingénierie des caractéristiques des données volumineuses de MovieLens.

Elasticsearch:

- **Utilité:** Elasticsearch est utilisé comme base de données pour stocker les données de films et d'utilisateurs. Il permet une recherche rapide et efficace pour les recommandations en temps réel.

Kibana:

- **Utilité:** Kibana est utilisé pour la visualisation des données stockées dans Elasticsearch. Il permet de créer des tableaux de bord interactifs pour interpréter les insights obtenus.

Flask:

- **Utilité:** Flask est utilisé pour développer une API qui interagit avec le modèle de recommandation. Il reçoit les titres de films des utilisateurs et retourne des recommandations personnalisées.

Modèle de Recommandation - ALS (Alternating Least Squares):

- **Utilité:** Le modèle ALS est utilisé pour les recommandations de films en se basant sur les évaluations passées des utilisateurs.

DATA SOURCE 'API '

Dans cette phase du projet, nous avons créé une API Flask afin de fournir un accès facile et structuré aux données de recommandation de films. Le processus de création de cette API implique deux étapes majeures : la transformation des données brutes en format JSON et l'utilisation de Flask pour créer des points d'accès spécifiques.

Étape 1 : Transformation des Données en JSON

Pour faciliter la manipulation et la transmission des données, nous avons converti les fichiers texte contenant les informations sur les films, les utilisateurs et les notations en un format JSON. Le script Python utilise la bibliothèque Pandas pour charger les données, les fusionner en fonction des identifiants pertinents, et ensuite convertir le DataFrame résultant en un fichier JSON. Ce fichier JSON, nommé 'movies.json', contient toutes les informations nécessaires pour alimenter notre API.

script dans le repo : [change_to_json.py](#)

Étape 2 : Création de l'API Flask

Une fois les données transformées, nous avons développé une API Flask pour fournir un point d'accès centralisé aux informations sur les films.

Voici quelques points clés de l'API :

- Trois routes distinctes sont mises en place :
 - a. **/api/movies/<int:page>** : Cette route prend en charge la pagination des films, permettant aux utilisateurs de récupérer une page spécifique de résultats. Chaque page affiche jusqu'à 10 films.
 - b. **/api/user/<int:user_id>** : Cette route permet aux utilisateurs de récupérer les films notés par un utilisateur spécifique, identifié par son ID.
 - c. **/api/movie/<int:movie_id>** : Permet aux utilisateurs de récupérer des détails spécifiques sur un film en fournissant son ID

script dans le repo : [MovieRec_Api.py](#)

Format de données dans l'api :

DATA SOURCE 'API '

```
{
  "movieId": 428,
  "movie_title": "Harold and Maude (1971)",
  "release_date": "01-Jan-1971",
  "IMDb_URL": "http://us.imdb.com/M/title-exact?Harold%20and%20Maude%20(1971)",
  "unknown": 0,
  "Action": 0,
  "Adventure": 0,
  "Animation": 0,
  "Children's": 0,
  "Comedy": 1,
  "Crime": 0,
  "Documentary": 0,
  "Drama": 0,
  "Fantasy": 0,
  "Film-Noir": 0,
  "Horror": 0,
  "Musical": 0,
  "Mystery": 0,
  "Romance": 0,
  "Sci-Fi": 0,
  "Thriller": 0,
  "War": 0,
  "Western": 0,
  "userId": 184,
  "rating": 4,
  "timestamp": 889909551,
  "age": 37,
  "gender": "M",
  "occupation": "librarian",
  "zip": "76013"
}
```

CRÉATION DU PRODUCER

Dans cette étape du projet, nous avons développé le Producer, un composant essentiel assurant la transmission régulière des données movies vers Kafka. Le rôle de ce Producer est de récupérer les informations actualisées sur les films depuis l'API Flask préalablement mise en place, puis de les publier dans le topic Kafka nommé "MovieRecommender".

Fonctionnement du Producer :

1- Récupération des Données depuis l'API Flask :

- Le Producer utilise la fonction `get_data(page)` pour extraire les données à partir de l'API Flask. À chaque itération de la boucle principale, une nouvelle page de données est récupérée, garantissant une mise à jour régulière des informations.

2- Sérialisation des Données :

- Les données extraites sont sérialisées au format JSON à l'aide de la fonction `serializer(message)`. Cette étape est cruciale pour formater correctement les données en vue de leur traitement par Kafka.

3- Initialisation du Producer Kafka :

- Le Producer Kafka est instancié avec les paramètres nécessaires, tels que les adresses des serveurs de démarrage Kafka et la fonction de sérialisation spécifiée.

4- Boucle Principale - Envoi Continu :

- Structuré dans une boucle infinie, le script assure un flux régulier de données vers Kafka. Cette approche permet au système de recommandation de toujours disposer des informations les plus récentes.

5- Itération sur les Pages de Données :

- À chaque itération, le Producer extrait les données de la page actuelle de l'API Flask. Cette méthodologie permet de couvrir progressivement l'ensemble des données disponibles de manière itérative.

6- Itération sur Chaque Film de la Page :

- Pour chaque page, le Producer itère sur les films extraits. Chaque film est ensuite envoyé individuellement au topic Kafka "MovieRecommender".

7- Gestion des Délais :

- Une pause de 3 secondes entre chaque envoi (`time.sleep(3)`) est intégrée pour éviter une surcharge rapide du topic Kafka. Cette stratégie assure une distribution régulière des données et limite la charge sur le système.

CREATION DE PRODUCER

8- Maintien d'un Flux Continu :

- La boucle principale est conçue pour s'exécuter en continu, garantissant ainsi que le Producer reste actif, collectant et envoyant constamment des données à Kafka.

9- Intégration du Topic dans le Producer :

- Le Producer est configuré pour diriger les données vers le topic spécifié, "MovieRecommender". Cette configuration garantit que les messages sont dirigés vers le bon canal de communication au sein du système Kafka.

script dans le repo : [Producer.py](#)

CREATION DE CONSUMER

Intégration de Spark Streaming et Elasticsearch

Dans cette phase cruciale du projet, nous avons développé un script de consommation (Consumer) basé sur Apache Spark Streaming. Ce script se connecte à un topic Kafka spécifique, nommé "MovieRecommender", récupère en temps réel les données émises par le Producer, puis effectue des transformations avant d'insérer les résultats dans Elasticsearch.

Connexion à Elasticsearch et Définition des Mappings :

- Le script débute par la connexion à Elasticsearch en utilisant les informations d'identification, et procède à la définition des mappings pour trois indices distincts : "movies-user_rating", "movies-info", et "user-info". Ces mappings définissent la structure des données à stocker dans chaque index.

```
mapping_rating = {
  "mappings": {
    "properties": {
      "userId": {"type": "integer"},
      "age": {"type": "integer"},
      "gender": {"type": "keyword"},
      "release_date": {"type": "date"},
      "timestamp": {"type": "date"},
      "rating": {"type": "integer"},
      "movieId": {"type": "integer"},
      "movie_title": {"type": "keyword"},
      "genres": {"type": "keyword"},
    }
  }
}

mapping_movie = {
  "mappings": {
    "properties": {
      "movieId": {"type": "integer"},
      "movie_title": {"type": "keyword"},
      "release_date": {"type": "date"},
      "genres": {"type": "keyword"}
    }
  }
}

mapping_user = {
  "mappings": {
    "properties": {
      "userId": {"type": "integer"},
      "age": {"type": "integer"},
      "gender": {"type": "keyword"}
    }
  }
}
```

CREATION DE CONSUMER

Intégration de Spark Streaming et Elasticsearch

1- Initialisation de la Session Spark :

- Une session Spark est instanciée, avec la configuration nécessaire incluant les dépendances requises pour Kafka et Elasticsearch.

2- Lecture des Données depuis Kafka :

- Les données en streaming sont lues depuis le topic Kafka "MovieRecommender". La lecture est effectuée sous forme de flux continu.

3- Transformation des Données :

Sérialisation et Structuration :

- Les données lues depuis Kafka sont initialement sérialisées en format JSON, puis structurées à l'aide d'un schéma prédéfini. Le schéma est spécifié pour chaque champ, déterminant le type de données attendu.

```
schema = StructType([
    StructField("Action", IntegerType()),
    StructField("Adventure", IntegerType()),
    StructField("Animation", IntegerType()),
    StructField("Children's", IntegerType()),
    StructField("Comedy", IntegerType()),
    StructField("Crime", IntegerType()),
    StructField("Documentary", IntegerType()),
    StructField("Drama", IntegerType()),
    StructField("Fantasy", IntegerType()),
    StructField("Film-Noir", IntegerType()),
    StructField("Horror", IntegerType()),
    StructField("Musical", IntegerType()),
    StructField("Mystery", IntegerType()),
    StructField("Romance", IntegerType()),
    StructField("Sci-Fi", IntegerType()),
    StructField("Thriller", IntegerType()),
    StructField("War", IntegerType()),
    StructField("Western", IntegerType()),
    StructField("unknown", IntegerType()),
    StructField("IMDb_URL", StringType()),
    StructField("age", IntegerType()),
    StructField("gender", StringType()),
    StructField("movieId", IntegerType()),
    StructField("movie_title", StringType()),
    StructField("rating", IntegerType()),
    StructField("release_date", StringType()),
    StructField("timestamp", IntegerType()),
    StructField("userId", IntegerType()),
])
```

4- Filtrage des Données Nulles :

- Certaines colonnes peuvent contenir des valeurs nulles après la sérialisation. Nous effectuons un filtrage pour exclure les lignes contenant au moins une valeur nulle dans l'une des colonnes spécifiées.

CREATION DE CONSUMER

Intégration de Spark Streaming et Elasticsearch

5- Création de la Colonne "genre" :

- Nous créons une colonne "genre" en utilisant les colonnes spécifiques aux genres des films. Si la valeur de la colonne correspondante est égale à 1, le nom du genre est ajouté à la liste. Les colonnes de genre sont ensuite supprimées.

6- Ajustement des Types de Données :

- Pour garantir la conformité avec les types d'indexation Elasticsearch, nous ajustons les types de données de certaines colonnes. Par exemple, les colonnes "release_date" et "timestamp" sont converties respectivement en types "date" et "timestamp".

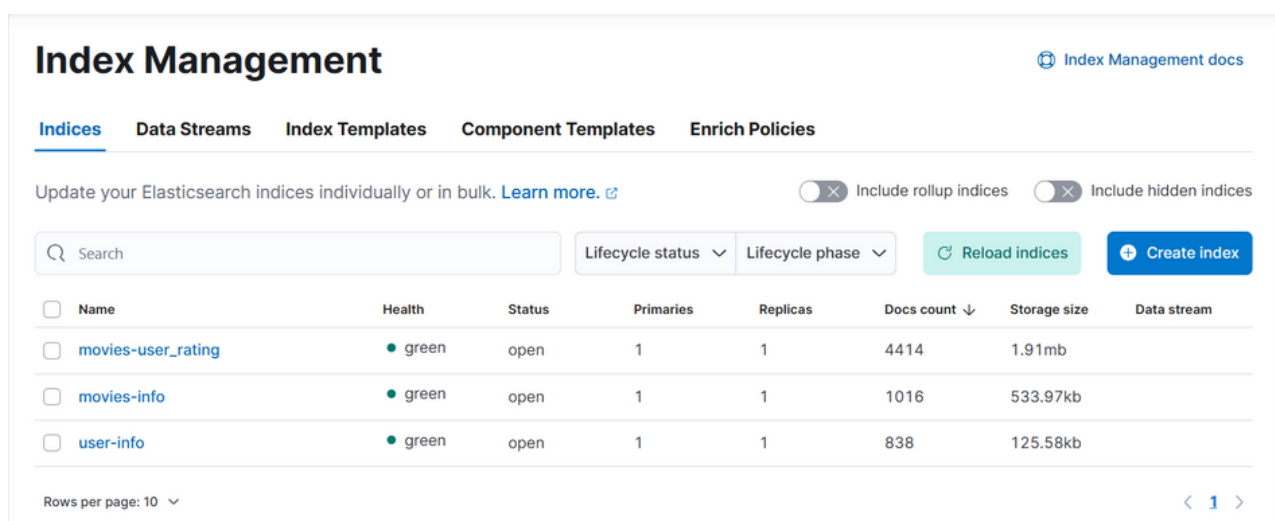
7- Indexation des Données dans Elasticsearch :

- Trois DataFrames distincts sont créés, correspondant à chaque index Elasticsearch ("movies-user_rating", "movies-info", "user-info"). Ces DataFrames sont ensuite insérés en parallèle dans leurs index respectifs dans Elasticsearch.

8- Lancement du Streaming et Attente d'Insertion :

- Le script utilise Spark Streaming pour traiter les données en continu. Trois threads distincts sont créés pour insérer les données dans les indices Elasticsearch correspondants. Le processus fonctionne de manière asynchrone pour améliorer l'efficacité.

script dans le repo : [Consumer.py](#)

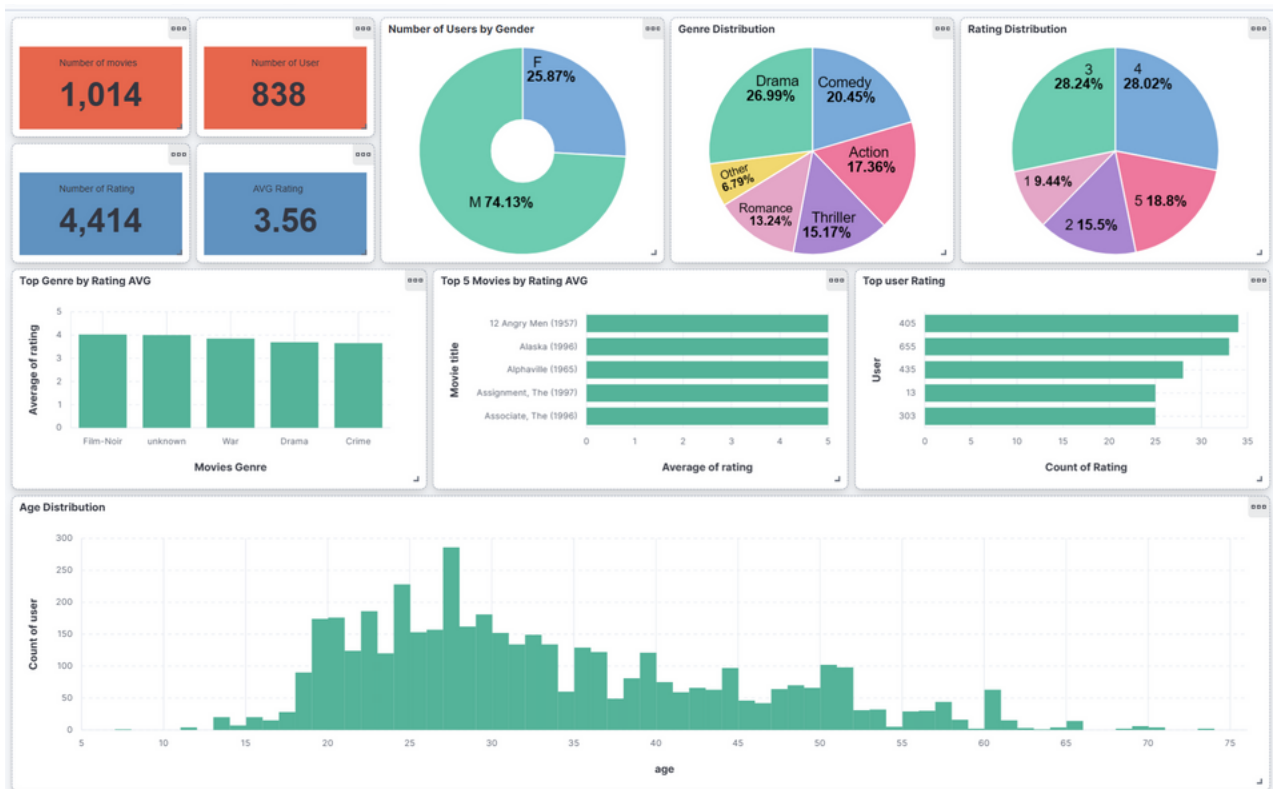


The screenshot shows the 'Index Management' page in Elasticsearch. It features a table with columns for Name, Health, Status, Primaries, Replicas, Docs count, Storage size, and Data stream. Three indices are listed: 'movies-user_rating', 'movies-info', and 'user-info', all with a 'green' health status and 'open' status. The table also includes a search bar, filters for lifecycle status and phase, and buttons for 'Reload indices' and 'Create index'.

<input type="checkbox"/>	Name	Health	Status	Primaries	Replicas	Docs count ↓	Storage size	Data stream
<input type="checkbox"/>	movies-user_rating	● green	open	1	1	4414	1.91mb	
<input type="checkbox"/>	movies-info	● green	open	1	1	1016	533.97kb	
<input type="checkbox"/>	user-info	● green	open	1	1	838	125.58kb	

VISUALISATION AVEC KIBANA

Dans cette étape cruciale du projet, nous avons effectué une visualisation approfondie de l'ensemble de nos données à l'aide de Kibana. L'objectif principal était d'obtenir une compréhension claire et approfondie des tendances



CONSTRUCTION DU MODÈLE "ALS"

nous nous sommes concentrés sur la construction du modèle de recommandation, utilisant la méthode ALS (Alternating Least Squares) pour générer des suggestions de films pertinentes. Cette section détaille les étapes spécifiques de la construction du modèle.

Sélection du Modèle :

Le choix du modèle est une étape stratégique. Nous avons opté pour le modèle ALS en raison de sa capacité à traiter les données implicites, telles que les évaluations de films. L'ALS offre une solution collaborative robuste, permettant une personnalisation efficace des recommandations pour chaque utilisateur.

Entraînement du Modèle :

Nous avons procédé à l'entraînement du modèle en utilisant le jeu de données préparé au cours de la phase de prétraitement des données avec Apache Spark. Ce processus a permis au modèle de découvrir les relations subtiles entre les utilisateurs et les films, ajustant ses paramètres pour optimiser la précision des recommandations.

Évaluation du Modèle :

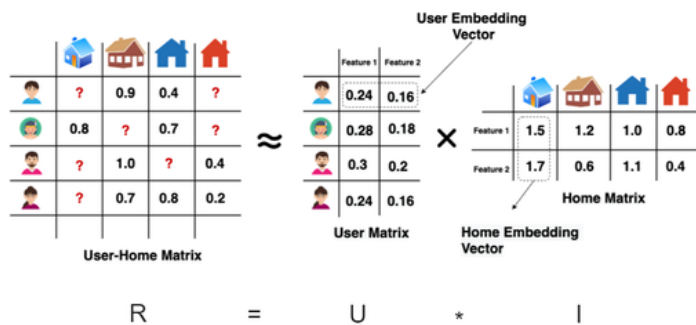
L'évaluation du modèle est essentielle pour mesurer sa performance. Nous avons utilisé des métriques telles que la RMSE (Root Mean Squared Error) pour évaluer la précision du modèle par rapport aux évaluations réelles. Cette évaluation itérative a contribué à ajuster les hyperparamètres pour améliorer les performances du modèle.

Intégration des Données dans Elasticsearch :

Une fois le modèle entraîné et évalué avec succès, nous avons procédé à l'intégration des résultats dans Elasticsearch. Cela a permis de fournir un accès rapide et efficace aux recommandations de films à l'aide d'indices dédiés, facilitant ainsi une expérience utilisateur fluide.

CONSTRUCTION DU MODÈLE "ALS"

Matrix Factorization



ALS

1 - Initialization:

Randomly initialize user matrix U and item matrix I.

2 - Alternating Optimization:

User Update: Fix I and solve for U using least squares optimization.

Item Update: Fix U and solve for I using least squares optimization.

3 - Repeat these steps for a certain number of iterations or until convergence.

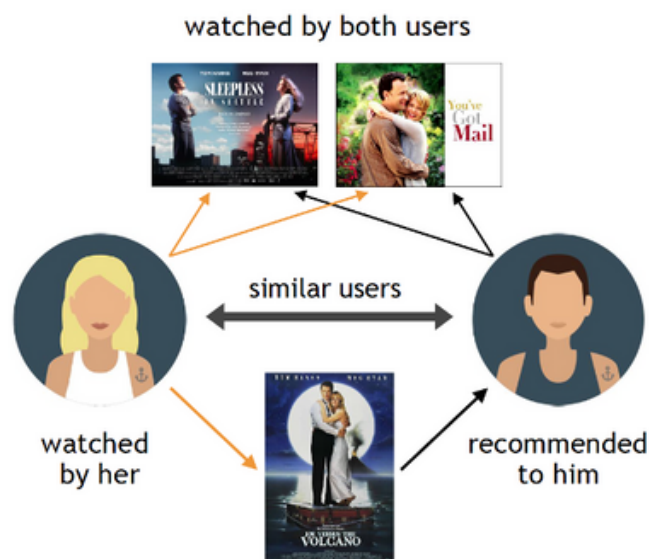
4 - Prediction:

Obtain the predicted ratings matrix by multiplying the updated U and I matrices: $R_{\text{predicted}} = U * I$.

5 - Recommendation:

Use the predicted ratings to make recommendations for users based on unrated items with the highest predicted r

Collaborative Filtering



RECOMMANDATION L'API

L'élaboration de l'API Flask représente une étape clé dans le déploiement du système de recommandation. Cette interface permet aux utilisateurs d'interagir avec le modèle de recommandation en fournissant des entrées personnalisées et en récupérant des suggestions de films adaptées à leurs préférences.

Conception de l'API

La conception de l'API a été pensée de manière à offrir une expérience utilisateur fluide. Nous avons défini les points d'entrée pour accepter les requêtes utilisateur, en se concentrant principalement sur la réception du titre d'un film. Cette conception modulaire permet une extensibilité future pour intégrer de nouvelles fonctionnalités.

Traitement des Requêtes Utilisateur

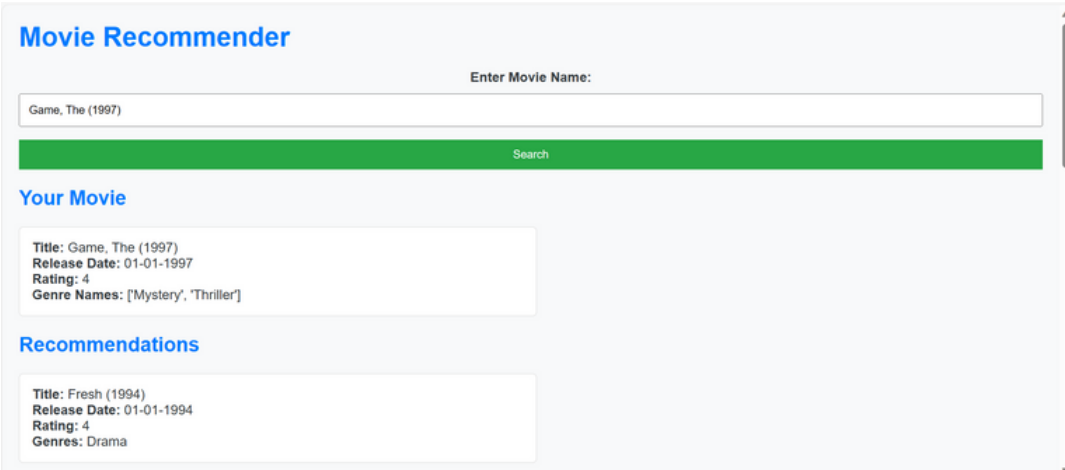
L'API reçoit les requêtes des utilisateurs et extrait les informations nécessaires, telles que le titre du film spécifié. Grâce à ces données, l'API recherche les utilisateurs qui ont interagi avec ce film particulier.

Interaction avec le Modèle ALS

Une fois les utilisateurs identifiés, l'API utilise ces informations pour solliciter le modèle ALS. Ce dernier génère alors des recommandations basées sur les préférences et les historiques de visionnage des utilisateurs spécifiés.

Retour des Recommandations aux Utilisateurs

Les recommandations obtenues du modèle sont retournées à l'utilisateur sous forme de réponse JSON. Cette structure permet une intégration aisée avec des applications front-end ou d'autres systèmes, garantissant une expérience utilisateur cohérente.



The screenshot displays a web application titled "Movie Recommender". It features a search bar with the placeholder text "Enter Movie Name:" and a green "Search" button. Below the search bar, the interface is divided into two main sections: "Your Movie" and "Recommendations". The "Your Movie" section displays details for the movie "Game, The (1997)", including its release date (01-01-1997), a rating of 4, and genres "Mystery" and "Thriller". The "Recommendations" section displays details for the movie "Fresh (1994)", including its release date (01-01-1994), a rating of 4, and the genre "Drama".

Section	Movie Title	Release Date	Rating	Genres
Your Movie	Game, The (1997)	01-01-1997	4	['Mystery', 'Thriller']
Recommendations	Fresh (1994)	01-01-1994	4	Drama

CONCLUSION

Le projet de recommandation de films de Jay-Z Entertainment représente une initiative ambitieuse visant à améliorer l'expérience utilisateur sur leur plateforme de divertissement. En suivant la méthodologie CRISP-DM, l'équipe a parcouru plusieurs phases cruciales pour développer un système de recommandation efficace, intégrant le Big Data, l'IA, et des technologies telles que Apache Spark, Elasticsearch, Kibana, Flask, et Kafka.

