
MTI820 - Rapport de travail de session

**Prototype d'un système de
recommandation de films**

Réalisé par :
DESCHILDRE Paul,
DJAMEN DJOFANG Vanick,
MEGUENANI Mohamed El Amine,
OLLIVIER Emma,
PINEDA MESA Kenia,
VIERA Baptiste

Hiver 2023

**Génie de Technologies de l'information
Ecole de technologie supérieur**

Table des matières

1	Introduction	3
2	Revue de littérature	4
2.1	Les types de systèmes de recommandations	4
2.2	Les défis de performance	4
2.3	Les défis éthiques et de société	4
2.4	L'interface utilisateur	5
2.5	Gestion des données et de leur étiquetage	5
3	Présentation des données	6
3.1	Source des données	6
3.2	Extraction	7
3.2.1	Sélection des informations concernant les films	7
3.2.2	Extraction des informations concernant les acteurs et le réalisateur	7
3.2.3	Extraction des informations concernant les genres	8
3.2.4	Extraction des périodes phares de l'année pour recommander un film à partir des mots clefs de celui-ci	8
3.2.5	Extraction de l'information sur le succès du film	9
3.2.6	Pistes d'amélioration pour la table des notes	9
3.2.7	Technologies utilisées	9
3.3	Modélisation	9
4	Exploitation de données	11
4.1	Analyse et nettoyage de données	11
4.1.1	Analyse et détection des anomalies	11
4.1.2	Nettoyage de données	12
4.2	Création du tableau de bord	12
4.2.1	Étape de développement	13
4.2.2	Publication et partage	15
5	Réalisation du prototype	16
5.1	Déploiement de bout en bout	16
5.1.1	Option 1 : Microsoft Azure	16
5.1.2	Option 2 : Snowflake et AWS	17
5.2	Interface utilisateur	22
5.2.1	Page d'inscription des utilisateurs	23
5.2.2	Page des recommandations	24
5.2.3	Page de recherche de films	26
6	Contributions	27

TABLE DES MATIÈRES

7 Conclusion	28
8 Bibliographie	29

1 Introduction

Les systèmes de recommandation se sont, depuis l'explosion de l'utilisation d'Internet, fait une place dans la vie des utilisateurs. Ils sont utilisés dans de nombreux domaines comme pour la recherche Internet, la vente en ligne et aussi pour la proposition de contenu. En effet, les systèmes de recommandation sont énormément exploités par les plateformes de streaming (diffusion) en ligne, aussi bien de musiques, de vidéos ou encore de films et de séries. Ces systèmes ont une place de grande importance pour ces plateformes, c'est effectivement grâce à ceux-ci qu'elles vont pouvoir garder leur utilisateurs en s'assurant de leur proposer du contenu qui les intéressent et ainsi se frayer une place à travers la concurrence. De par leur importance, la mise en place de systèmes de recommandation fait face à de nombreux défis de performance et d'éthique. Il s'agit de proposer du contenu personnalisé pour l'utilisateur qu'il aimera sans l'ennuyer, sans l'enfermer dans ses propres goûts ou lui proposer du contenu inapproprié.

Dans le cadre de ce projet, nous avons choisi d'implémenter un prototype de système de recommandation de film. Ce prototype doit permettre à un utilisateur à partir d'une série d'informations sur lui qu'il fournira d'avoir des suggestions pertinentes pour lui.

Ce rapport est alors un aperçu des différentes étapes de la réalisation de ce prototype. Tout d'abord, nous présenterons une revue de la littérature avec laquelle nous nous sommes documentés pour ce projet. Puis, nous exposerons les données que nous avons utilisées et la manière dont on les a traitées et exploitées. Enfin nous parlerons de la réalisation du prototype en tant que tel en présentant son architecture et son implémentation.

2 Revue de littérature

2.1 Les types de systèmes de recommandations

L'implémentation des systèmes de recommandation est un sujet qui intéresse aussi bien la recherche que l'industrie, qui cherche à faire des recommandations toujours plus pertinentes et personnalisées pour ses utilisateurs. Pour ce faire, les méthodes basées sur l'intelligence artificielle et plus particulièrement sur l'apprentissage machine sont utilisées. Elles sont notamment présentées par Roy et Dutta dans leur étude [1]. Ils expliquent alors que les méthodes de recommandation peuvent être de plusieurs types. Il y a notamment le filtrage collaboratif qui se base sur les interactions qu'ont les autres utilisateurs et l'utilisateur cible avec le contenu, que ce soit par le biais de notes ou des vues, pour donner à l'utilisateur une proposition pertinente. Par exemple : si un utilisateur A pour lequel on souhaite faire une recommandation a aimé un certain nombre de mêmes films qu'un autre utilisateur B, il sera estimé que ces deux utilisateurs ont des goûts similaires et on pourra proposer à notre utilisateur A d'autres films qu'a aimé l'utilisateur B. Il y a aussi le filtrage basé sur le contenu qui, lui, va suggérer des items en se basant sur ses propriétés. Par exemple, si un utilisateur met beaucoup de notes positives à des thrillers, le système aura davantage tendance à suggérer des thrillers à l'utilisateur.

2.2 Les défis de performance

Toutefois ces méthodes font encore face à de nombreux défis, présentés notamment par Schedl *et al.* dans leur étude [2], comme celui du départ à froid (cold start problem). Lorsqu'un nouvel utilisateur s'inscrit sur une plateforme, il n'a à l'origine interagit avec aucun contenu, il n'a pu encore rien regarder ou écouter ni rien noter. Il faut alors trouver une manière de contourner le problème pour ces utilisateurs afin d'éviter qu'ils arrivent sur la plateforme, ne trouvent rien à leur goût puis partent aussitôt.

2.3 Les défis éthiques et de société

Il y a également des défis éthiques et de société liés aux systèmes de recommandations. C'est notamment ce qu'expliquent Milano *et al.* dans leur étude [3]. Effectivement, dans le cas de la méthode du filtrage collaboratif c'est l'ensemble des utilisateurs qui influence les recommandations. Ainsi, il serait tout à fait possible de recommander à un utilisateur du contenu heurtant sa sensibilité, ce qui lui donnerait alors une expérience d'utilisation de la plateforme très désagréable. En effet, lors de l'implémentation d'un système de recommandation, il faut, de manière générale, veiller à ne pas proposer de contenu inadapté pour l'utilisateur, comme par exemple un film pour adulte à un mineur. Il y a également, avec le filtrage collaboratif, un risque de toujours prédire les mêmes films. En effet, il semblerait logique que les blockbusters américains dans le cas des films et les "tubes" dans le

cas des musiques soient très populaires sur la plateforme et soient alors proposés plus facilement que les autres films ou chansons. Ainsi, ils seront davantage regardés ou écoutés et donc encore plus recommandés. On tombe alors dans une boucle sans fin qui s'alimente et le système perd en diversité.

Dans une autre étude menée par Seaver [4], il est question de ne pas enfermer l'utilisateur dans ses propres goûts. Effectivement, selon cet article, un bon système de recommandations est un système qui fait découvrir du contenu nouveau à ces utilisateurs. Un utilisateur interagissant plusieurs fois avec un contenu similaire risque alors de n'avoir plus que ce type de contenu de proposer et si au bout d'un certain temps cela ne l'intéresse plus il finira fatalement, lassé, par quitter la plateforme.

2.4 L'interface utilisateur

En plus d'un système de recommandation performant, les plateformes qui proposent du contenu doivent également avoir une interface utilisateur efficace. Gomez-Urbe et Hunt présentent dans leur étude [5] l'interface utilisateur de la plateforme de streaming Netflix. D'après leur article, un utilisateur classique perd de l'intérêt s'il cherche un film plus de 90 secondes. C'est pourquoi la plateforme affiche les prédictions en ligne en indiquant à l'utilisateur pourquoi ce film ou cette série lui est proposé. Cela lui permet de choisir plus rapidement, aucun temps ne lui est laissé pour se désintéresser. Sur l'interface utilisateur, il y a également un équilibre entre les recommandations générales, c'est-à-dire globalement les tendances ainsi que les recommandations ciblées pour l'utilisateur en question.

2.5 Gestion des données et de leur étiquetage

Les modèles de recommandation, comme tout modèle d'apprentissage machine, ont besoin d'un grand nombre de données pour apprendre convenablement. Dans leur étude, Roh, Heo et Euijong Whang [6] font état de l'importance des données initiales pour l'apprentissage. En effet, celles-ci sont aussi importantes que le modèle en soit. Ils expliquent dans leur article comment tirer un maximum des données initiales pour un apprentissage le plus pertinent possible. Un des problèmes courants est souvent le manque de données et plus particulièrement le manque de données étiquetées. Pour cette problématique, des techniques d'apprentissage semi-supervisé et non-supervisé sont expliquées. Dans le cas des systèmes de recommandation, il s'agit principalement des notes attribuées aux éléments. Il peut alors arriver que, pour certains éléments, il y ait très peu de notes ou de visionnages. Il faut alors générer des étiquettes en croyant celles existantes. C'est ce sur ce principe que sont basées beaucoup de méthodes d'apprentissage semi-supervisé.

3 Présentation des données

3.1 Source des données

Afin de réaliser le prototype de notre système de recommandation, il a fallu se procurer des données concernant les films ainsi qu'une série de notes associées à ceux-ci. Pour ce faire, nous avons choisi d'utiliser une base de données ouverte provenant de *MovieLens* ainsi que de *The Movie Database* (TMDb) provenant la plateforme *Kaggle* [7]. Grâce à cette base de données, il est alors possible d'avoir des informations de base sur les films sortis après 2017 telles que leur titre, leur date de sortie, leur pays d'origine, leur résumé, les genres auxquels ils sont attribués ou encore leur affiche. En plus de cela, la base de données donne des informations concernant les acteurs qui ont joué dans chaque film ainsi que l'équipe de tournage. Il y a également une série de mots clefs associés à chaque film. En outre, cette base de données propose pour chaque film une série de notes allant de 0 à 5 données par des utilisateurs, ici anonymisés. Les données sont fournis au format *.csv* de la manière suivante :

- Fichier contenant les informations sur les films
movies_metadata : id, adult, belongs_to_collection (json), budget, genres (json), home_page, original_language, original_title, overview, popularity, poster_path, production_company (json), production_countries (json), release_date, revenue, runtime, spoken_languages, status, tagline, title, video, vote_average, vote_count
- Fichier contenant les informations sur les crédits du film
credits : id, cast (json), crew (json)
- Fichier contenant les informations sur les mots clefs associés aux films
keywords : id, keywords (json)
- Fichier contenant les informations sur les votes
rating : id, userId, movieId, rating, timestamp

De plus, en complément de ces données, nous avons trouvé pertinent d'utiliser une autre base de données provenant de la plateforme *Data.world* [8] recensant les films ayant eu du succès entre 1975 et 2015. Celle ci contient un fichier *.csv* qui est le suivant :

- blockbuster-top_ten_movies_per_year_DFE : audience_freshness, poster_url, rt_audience_score, 2015_inflation, adjusted, genres, genre_1, genre_2, genre_3, imdb_rating, length, rank_in_year, rating, release_date, studio, title, world_wide_gross, year

Bien sûr, toutes ces données ne sont pas vraiment exploitables comme telles et il s'agit alors de réaliser des opérations de nettoyage.

3.2 Extraction

3.2.1 Sélection des informations concernant les films

Notre première base de données (*MovieLens* et *The Movie Database*) fournit beaucoup d'informations sur chaque film. Néanmoins, pour notre prototype de système de recommandation nous n'avons pas besoin de toute cette information. Nous avons alors effectué un tri parmi les colonnes que nous avons jugées intéressantes pour l'implémentation de notre prototype.. Nous avons alors choisi de conserver l'identifiant du film, son titre, son affiche, la moyenne de ses votes ainsi que son nombre de votes. Nous avons également fait le choix de garder le pays d'origine du film afin de pouvoir proposer à l'utilisateur des films venant de chez lui ainsi que l'information concernant le fait qu'un film est un film pour adulte ou non. Cela permet de ne pas proposer à un utilisateur mineur des films qui ne lui sont pas destinés.

3.2.2 Extraction des informations concernant les acteurs et le réalisateur

Afin de réaliser notre prototype nous avons décidé de garder du fichier "*credits.csv*" uniquement les acteurs et le réalisateur du film. En effet, il sera pratique de cette manière de recommander à l'utilisateur des films avec les mêmes acteurs ou réaliser par la même personne qu'un autre film qu'il a aimé. Dans le fichier "*credits.csv*" les informations concernant les acteurs et l'équipe de tournage sont stockées au format json.

```

1 [{ 'cast_id': 14,
2   'character': 'Woody (voice)',
3   'credit_id': '52fe4284c3a36847f8024f95',
4   'gender': 2,
5   'id': 31,
6   'name': 'Tom Hanks',
7   'order': 0,
8   'profile_path': '/pQFoyx7rp09CJTAb932F2g8Nlho.jpg'}, [{ 'cast_id': 15,
9   'character': 'Buzz Lightyear (voice)',
10  'credit_id': '52fe4284c3a36847f8024f99',
11  'gender': 2, 'id': 12898,
12  'name': 'Tim Allen',
13  'order': 1, ...}], ... ]

```

Listing 1 – Données sur les acteurs d'un film

Pour notre modélisation finale, nous avons choisi de garder le nom des trois premiers acteurs principaux pour chaque film, c'est une information qui nous est donnée par l'entrée "*order*".

```

1 [{ 'credit_id': '52fe4284c3a36847f8024f49',
2   'department': 'Directing',

```


3 PRÉSENTATION DES DONNÉES

```

3 'gender': 2,
4 'id': 7879,
5 'job': 'Director',
6 'name': 'John Lasseter',
7 'profile_path': '/7EdqiNbr4FRjIhKHYPdFfEEEEFG.jpg'},
8 {...} ... ]

```

Listing 2 – Données sur l'équipe de tournage d'un film

Pour ne sélectionner que les réalisateurs parmi tous les membres d'une équipe de tournage, il suffisait de sélectionner uniquement la personne dont l'entrée *"job"* a comme valeur *"Director"*.

3.2.3 Extraction des informations concernant les genres

De la même manière que pour les acteurs et les réalisateurs, il a fallu extraire les genres du film à partir d'un format *.json*. Pour chaque film, il y avait plusieurs genres proposés et dans notre approche, nous avons décidé de ne garder que les trois premiers genres qui sont les principaux.

3.2.4 Extraction des périodes phares de l'année pour recommander un film à partir des mots clefs de celui-ci

Il est possible de savoir, dans l'année, quand il est le plus pertinent de recommander un film à partir du fichier contenant les mots clefs (mis eux aussi en format *.json*). En effet, certains mots clefs font référence à une période de l'année tel que *Noël* ou encore *Halloween*. Nous avons, en fonction de différentes périodes de l'année, identifié un certain nombre de mots clefs dont il faudra vérifier la présence pour chaque film.

- Halloween : "halloween", "ghost", "witch", "fear"
- Noel : "christmas"
- Saint Valentin : "love", "lover", "romance"
- Vacances d'été : "summer", "summer vacation"
- 11 Novembre : "world war i"
- 8 mai : "world war ii"
- action de grâce : "thanksgiving"

Ainsi, pour chaque film sera associé l'ensemble des périodes ci-dessus. Celles-ci prendront la valeur 1 si un de ses mots clefs fait partie des mots clefs du film et 0 sinon.

3 PRÉSENTATION DES DONNÉES

3.2.5 Extraction de l'information sur le succès du film

Dans le but de savoir si le film a eu du succès ou non, nous avons réalisé une jointure externe entre notre première base de données venant de *MovieLens* et de *The Movie Database* sur le titre du film ainsi que sur son année de sortie. Dans le cas où un film existe dans les deux bases de données, il est indiqué comme étant un film ayant connu le succès.

3.2.6 Pistes d'amélioration pour la table des notes

Dans le fichier des notes, nous retrouvons une colonne avec les identifiants des utilisateurs, une colonne avec les identifiants des films et une colonne avec les notes. Le souci est que certains films sont très peu notés par rapport à d'autres, ce qui a pour impact que le modèle d'apprentissage machine ne parvient pas à générer des prédictions pour ces films. Afin d'améliorer le modèle et régler ce problème, nous aurions pu utiliser des techniques d'apprentissage non supervisé pour générer plus de notes.

3.2.7 Technologies utilisées

Dans le but d'effectuer toutes ces opérations de nettoyage, la bibliothèque *Pandas* a été utilisée. *Pandas* est une bibliothèque implémentée pour le langage *Python* qui permet de manipuler et d'analyser des données, provenant notamment comme dans notre cas de fichiers *.csv*, en mettant celle-ci dans un format de type "*Data-frame*".

3.3 Modélisation

Nous avons modélisé nos différentes tables grâce à un schéma de type "*snowflake*". La table de fait de ce schéma est la table des notes attribuées par les utilisateurs aux films. Cette table est alors aux dimensions "*Film*" et "*Utilisateur*". Ces dimensions sont à leur tour liées à d'autres dimensions. Le film a une dimension pour ses genres, ses acteurs, son réalisateurs, les périodes de l'année pour lesquelles il va être davantage recommandé, sa popularité et sa date de sortie. L'utilisateur a quant à lui une dimension pour sa date d'anniversaire et une dimension enregistrant ses genres favoris.

3 PRÉSENTATION DES DONNÉES

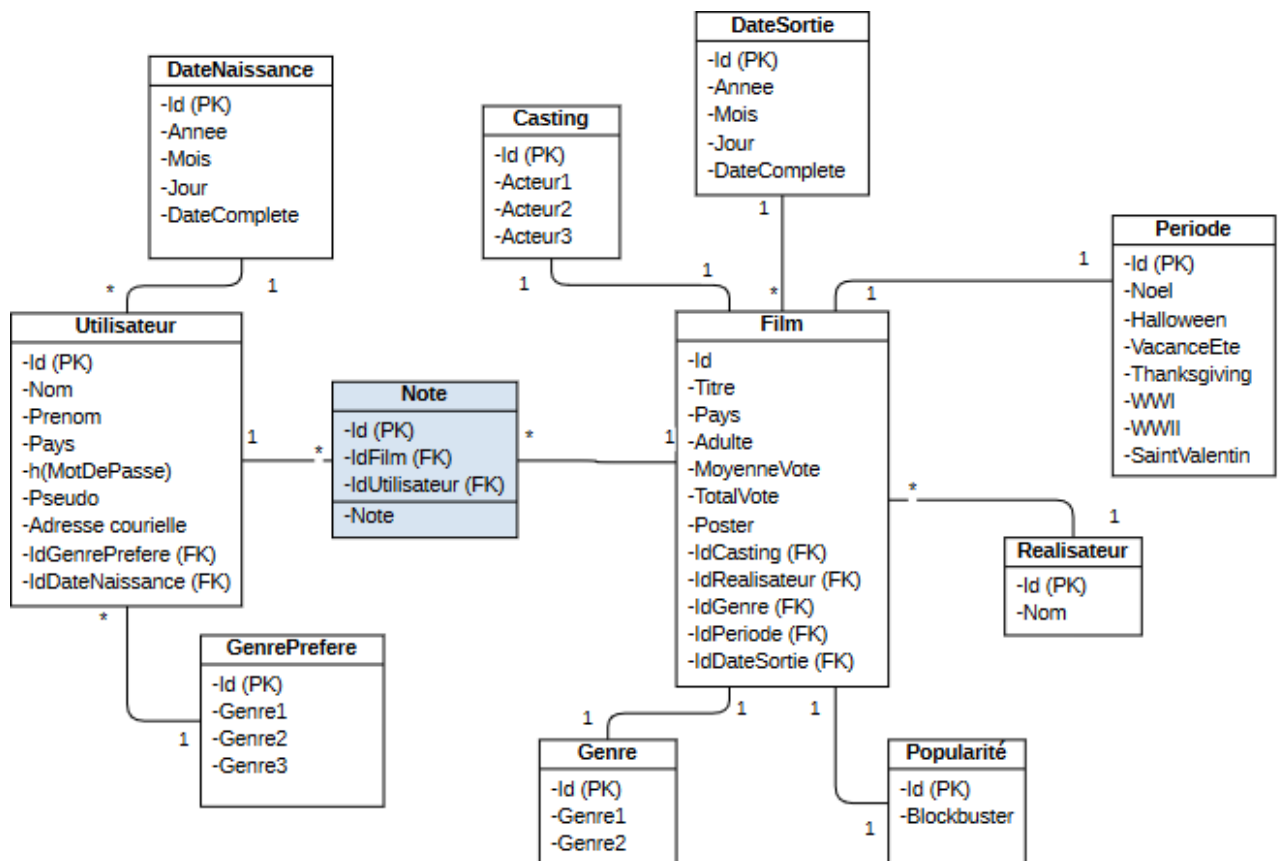


Figure 1 – UML de l'architecture de données

4 Exploitation de données

L'exploitation de données est devenue une pratique courante dans de nombreuses organisations à travers le monde. Elle consiste à recueillir, à stocker et à analyser des données pour en extraire des informations utiles.

Dans le cadre du développement de notre plateforme de diffusion, l'exploitation de données peut nous aider à mieux comprendre les préférences des utilisateurs et à améliorer la recommandation de contenu, et ainsi nous aider à identifier les tendances et améliorer l'efficacité opérationnelle.

4.1 Analyse et nettoyage de données

Avant d'exploiter les données, il est essentiel de comprendre la qualité de ces données. Dans un premier temps, il sera important qu'on analyse et on nettoie nos données pour détecter les données aberrantes, manquantes, les valeurs nulles et les doublons. Ensuite, il faudra corriger les types de variables pour assurer l'uniformité et la cohérence des données. En effet, la présence de données aberrantes ou de doublons dans notre ensemble de données peut affecter la précision des analyses et conduire à des interprétations erronées. Aussi, les données manquantes ou les valeurs nulles peuvent réduire la taille de l'échantillon et fausser les résultats.

Pour cela, nous utiliserons l'environnement *Jupyter Notebook*. Nous utiliserons également la bibliothèque *Pandas*, qui est une bibliothèque open-source de traitement de données pour *Python*. *Pandas* est utilisée pour la manipulation et l'analyse des données, telles que les fichiers *.csv*, en utilisant des *DataFrames*.

4.1.1 Analyse et détection des anomalies

Au début, nous chargerons les fichiers *.csv* dans des *DataFrame* en utilisant la fonction `'read_csv'`, afin de pouvoir explorer les données et détecter les anomalies. Pour cela, nous utiliserons les fonctions de *Pandas* telles que :

- `"describe()"` pour obtenir des statistiques descriptives sur les données.
- `"info()"` pour afficher des informations sur les données.
- `"isnull()"` pour vérifier les valeurs nulles.
- `"duplicated()"` pour détecter les doublons.
- `"dtype"` pour identifier les types de données de chaque colonne.
- `"count()"` pour compter le nombre d'éléments non nuls dans chaque colonne.
- `"groupby()"` pour regrouper les données selon les valeurs d'une ou plusieurs colonnes.
- `"unique()"` pour retourner les valeurs uniques d'une colonne.

- "sort_values()" permet de trier les données selon une ou plusieurs colonnes.

Ces manipulations d'exploration nous aident à mieux comprendre la structure de nos données.

4.1.2 Nettoyage de données

Une fois les anomalies détectées, il faut nettoyer les données en supprimant les doublons, en remplissant les données manquantes avec des valeurs appropriées et en supprimant les lignes contenant des valeurs nulles ou aberrantes. Aussi, nous allons changer et ajuster les types de nos données pour éviter toute confusion lors de l'analyse. Voici des exemples de code utilisé lors de cette opération de nettoyage :

- `film.drop_duplicates()` : cette fonction supprime toutes les lignes en double dans la table `film`.
- `film_cl.drop_duplicates(subset=['Id'])` : cette fonction supprime toutes les lignes en double en se basant sur la colonne 'Id'.
- `film_cl.drop_duplicates(subset=['Title', 'Pays'], keep='last')` : cette fonction supprime toutes les lignes en double en se basant sur les colonnes 'Title' et 'Pays' et en gardant la dernière occurrence.
- `film_cl.drop_duplicates(subset=['Title'], keep='last')` : cette fonction supprime toutes les lignes en double en se basant sur la colonne 'Title' et en gardant la dernière occurrence.
- `film_cl.dropna(subset=['Id'])` : cette fonction supprime toutes les lignes où la colonne 'Id' contient des valeurs manquantes (NaN).
- `film_cl.dropna(subset=['Title'])` : cette fonction supprime toutes les lignes où la colonne 'Title' contient des valeurs manquantes (NaN).
- `popularite.drop_duplicates(subset=['Id'])` : cette fonction supprime toutes les lignes en double en se basant sur la colonne 'Id'.
- `film_cl['IdAuteur'].astype('int64')` : cette fonction convertit le type de données de la colonne 'IdAuteur' en int.

4.2 Création du tableau de bord

Une fois que nos données sont bien nettoyées, il est temps de les explorer en profondeur pour trouver des informations pertinentes et mieux comprendre leur contenu.

Pour cela, nous allons utiliser des techniques d'analyse et de visualisation de données en développant un tableau de bord interactif avec des graphiques visuels

4 EXPLOITATION DE DONNÉES

en utilisant l'outil *Power BI*, car ce dernier permet une connexion facile à nos données, offre une grande variété de graphiques et de visualisations pour la représentation des données et est facile à utiliser.

Le tableau de bord que nous allons développer fournira des informations importantes sur nos données telles que les films les plus populaires, les genres de films les plus appréciés, les films les mieux notés, les pays les plus représentés par les films, etc. Il permet également à l'utilisateur de filtrer les données en fonction de différents critères tels que le genre, l'année de sortie, le pays d'origine etc. Cela nous aidera à détecter les tendances et les relations entre nos données afin d'obtenir des informations clefs et d'être capable de prendre de meilleures décisions.

4.2.1 Étape de développement

Afin de développer notre tableau de bord, nous allons commencer par connecter *Power BI* à nos fichiers *.csv* pour charger les données. Une fois que nos tables de données sont chargées à partir des différents fichiers, nous pouvons procéder à la transformation des données (dans notre cas, nous avons simplement modifié le format des dates et remplacé les points par des virgules dans les champs de type *float*). N'oublions pas que cette étape est importante car elle permet d'obtenir des données propres et prêtes à être analysées.

Après avoir chargé et transformé nos tables, nous allons maintenant développer notre modèle de données et créer les relations entre les différentes tables en nous basant sur le schéma UML développé dans la partie précédente. Cela permet de créer un rapport plus riche et plus précis en combinant les données de différentes tables.

Vous trouverez ci-dessous notre modèle de relations dans *Power BI*.

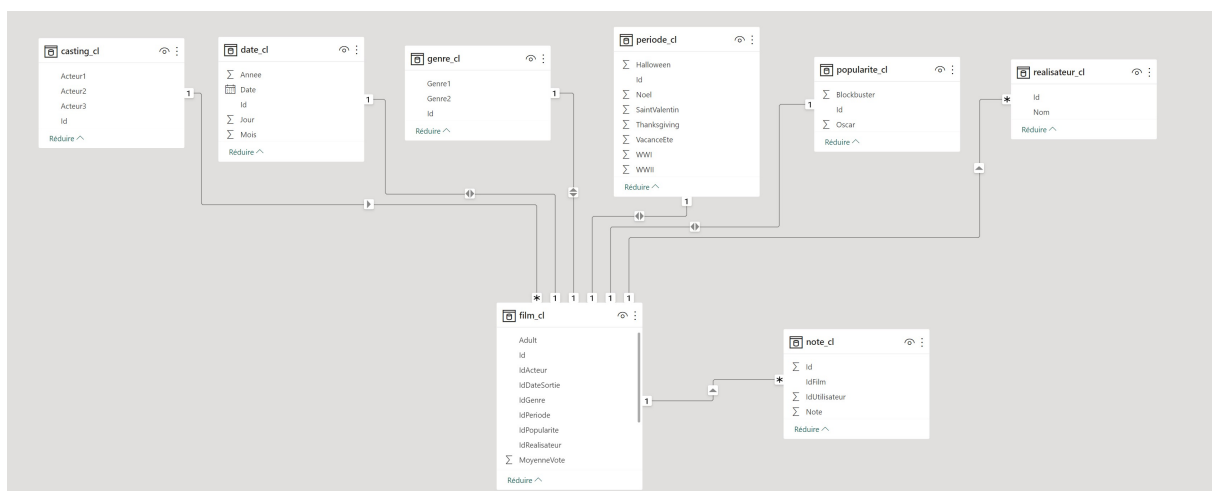


Figure 2 – Modèle de relation entre les tables

Après avoir implémenté notre modèle, nous passons à la création de visualisations. Nous avons choisi de développer six types de visualisations différents pour

4 EXPLOITATION DE DONNÉES

présenter les données de manière variée et compréhensible.

Tout d'abord, nous avons créé une carte des données en utilisant deux tables : la table des films et la table des périodes. Cette carte affiche le nombre de films, le total des votes, la moyenne des votes et le nombre total de films pour chaque période (Halloween, Noël, Saint-Valentin, etc.). Cela nous donne une vue d'ensemble des tendances et des différences des données.

Ensuite, nous avons développé deux graphiques à barres pour des comparaisons entre différentes catégories. Le premier est un classement des films ayant obtenu le maximum de notes, où nous avons appliqué un filtre pour limiter la liste à 20 films. Le deuxième est un classement des films les mieux notés, pour lequel nous avons également appliqué le même filtre limitant à 20.

Nous avons également créé un graphique en anneau en utilisant la table des films pour afficher le nombre de films par pays et le pourcentage de la distribution de films pour chaque pays. Pour voir rapidement quels pays sont les plus représentés dans notre ensemble de données.

Pour représenter la répartition des films à travers les années, nous avons créé un graphique en courbe en utilisant les deux tables (*films* et *date*).

Enfin, nous avons créé un histogramme groupé en utilisant la table des genres et la table des films pour comprendre les genres les plus populaires dans notre ensemble de données. Cette visualisation montre la distribution des films par genre, et permet de voir clairement quels genres ont le plus grand nombre de films.

Vous trouverez ci-dessous la visualisation de notre tableau de bord.

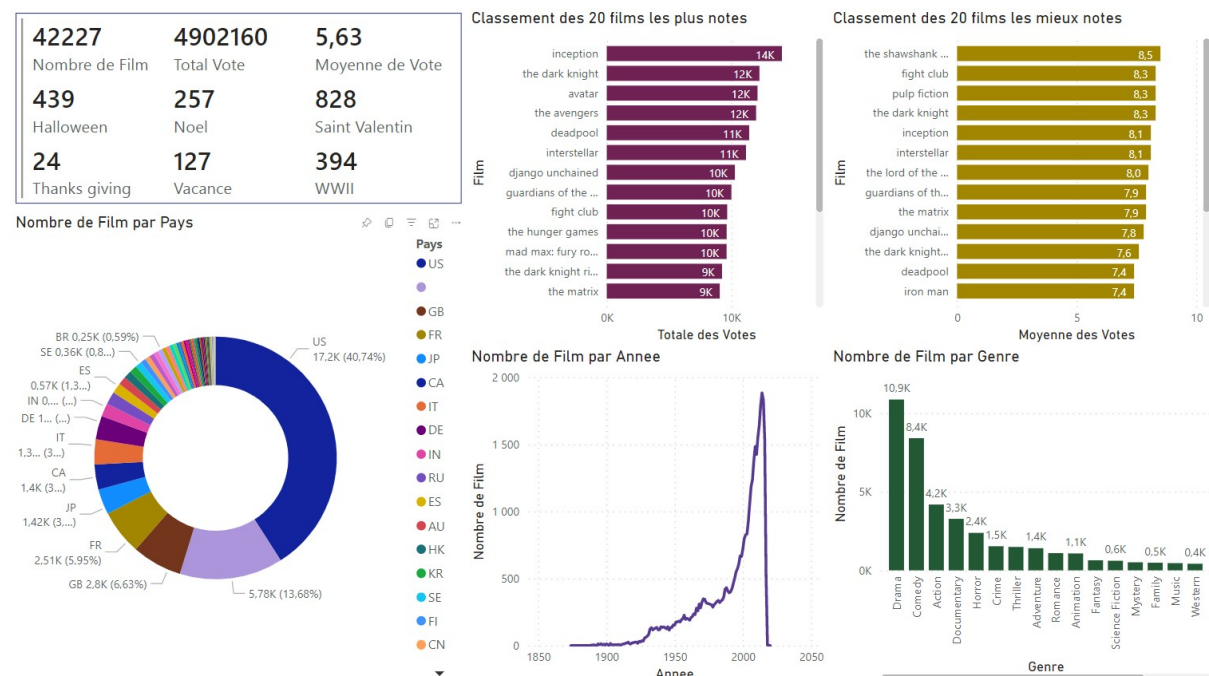
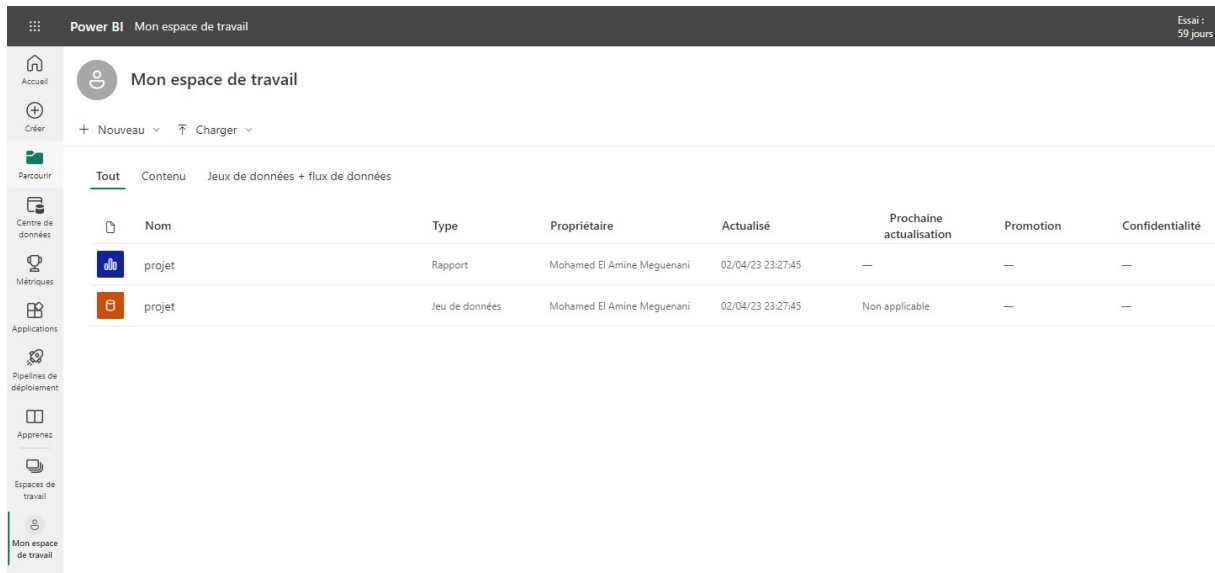


Figure 3 – Visualisation de notre tableau de bord

4 EXPLOITATION DE DONNÉES

4.2.2 Publication et partage

La publication du tableau de bord est l'étape finale du processus et permet de partager facilement les informations et les visualisations avec d'autres utilisateurs sur le service infonuagique de *Power BI*. Une fois publié, le tableau de bord peut être partagé avec d'autres utilisateurs. Il peut également être intégré à d'autres applications et sites Web pour une visualisation facile des données.





Power BI Mon espace de travail		Essai : 59 jours					
Mon espace de travail							
+ Nouveau ▾ ⌵ Charger ▾ ⌵							
Parcourir							
Tout Contenu Jeux de données + flux de données							
	Nom	Type	Propriétaire	Actualisé	Prochaine actualisation	Promotion	Confidentialité
	projet	Rapport	Mohamed El Amine Meguenani	02/04/23 23:27:45	—	—	—
	projet	Jeu de données	Mohamed El Amine Meguenani	02/04/23 23:27:45	Non applicable	—	—

Figure 4 – Exemple de publication du tableau de bord

5 Réalisation du prototype

5.1 Déploiement de bout en bout

5.1.1 Option 1 : Microsoft Azure

Nous avons dans un premier temps décidé d'utiliser les divers outils de Microsoft Azure pour faire un déploiement de bout en bout de notre modèle de recommandations. Nous nous sommes basés sur un tutoriel de Microsoft [10]. Cependant, ce tutoriel, datant de 2019, a été développé avec la version Spark 2.3 et non avec la version 3.0 ce qui a engendré un nombre considérable de problèmes de compatibilité. Presque chaque ligne du tutoriel était non fonctionnelle. Nous avons dû faire de multiples recherches pour réussir la migration non seulement vers la version de Spark 3.0 mais aussi vers les nouvelles versions des outils spécifiques à Microsoft Azure qui sont les suivants :

- Blob Storage : service de stockage d'objets cloud qui permet de stocker et de récupérer de grands volumes
- Databricks pour Azure : plateforme de traitement de données unifiée permettant de mettre en place des systèmes de bout en bout.
- Cosmos DB : service de base de données multi-modèle distribué qui offre une haute disponibilité, une évolutivité horizontale et une latence faible pour les applications web et mobiles.
- Kubernetes : système de gestion de conteneurs qui permet de déployer, de gérer et de mettre à l'échelle des applications conteneurisées sur des clusters de serveurs.
- Web App : service de développement web qui permet de créer et de déployer des applications sur le web à grande échelle en utilisant une infrastructure cloud.

L'objectif de ce déploiement est d'utiliser un modèle de recommandation collaboratif qui s'appelle ALS sur notre jeu de données contenant les notes de films que nous stockerons dans Blob Storage afin de pouvoir prédire les 10 meilleures recommandations pour un utilisateur donné. Ces recommandations seront alors stockées dans Cosmos DB pour permettre un accès rapide aux données. Ce modèle sera déployé sur Kubernetes et sera utilisé en tant que Service Web App. En d'autres termes et pour simplifier, notre modèle sera sauvegardé et pourra être utilisé sous la forme d'API. Ces différentes étapes ont été réalisées à partir de Databricks pour Azure. Du fait des performances par défaut très faibles, nous avons dû augmenter les capacités de la plateforme en termes de CPU virtuels pour pouvoir exécuter dans un temps raisonnable l'entraînement du modèle de machine learning de recommandation que nous sauvegardons. Concernant le déploiement du modèle sur

Azure Kubernetes (AKS), nous avons au préalable utilisé par l'intermédiaire de Databricks l'outil Azure Machine Learning Service qui permet de créer une image d'un script qui récupère le TOP 10 des prédictions contenues dans Cosmos DB correspondant à un utilisateur donné. Ensuite, après avoir créé un environnement pour le modèle contenant les différentes dépendances et un cluster AKS, nous avons pu déployer notre modèle. A partir d'un script python, nous pouvons désormais faire appel à notre modèle en tant qu'API afin de récupérer le TOP 10 des prédictions pour un utilisateur donné contenu dans Cosmos DB.

Cependant après plusieurs jours de travail sur cette implémentation, nous avons consulté notre facture sur Azure qui s'est élevée à environ 200\$. Bien que nous n'ayons pas été assez précautionneux sur ce point, cela a été une réelle mauvaise surprise et nous a malheureusement contraint d'abandonner cette solution que nous avions mise en place... Au vu du nombre d'heures consacrées, cela a été une réelle déception, mais nous avons tout de même pu acquérir de nouvelles compétences en termes de déploiement de bout en bout sur Azure d'un modèle d'apprentissage machine.

5.1.2 Option 2 : Snowflake et AWS

Etant donné que la solution de *Microsoft Azure* n'était pas envisageable d'un point de vue financier, nous avons décidé de nous tourner vers de nouvelles technologies. A la suite de plusieurs recherches, et en nous basant sur un répertoire GitHub [11], nous avons décidé d'utiliser le service infonuagique d'*Amazon AWS* et la plateforme de traitement de données cloud-native *Snowflake*. Le choix d'*Amazon AWS* s'est justifié principalement par le fait que nous avons à notre disposition de nombreux outils gratuits ou à faible coût pour entraîner mais aussi déployer notre modèle de recommandation sur le nuage. Nous utilisons principalement *AWS SageMaker* pour y parvenir. En ce qui concerne l'algorithme de recommandation utilisé, nous avons utilisé l'algorithme *SVD* disponible dans la bibliothèque *Python Surprise*. La sélection de *Snowflake* se justifie également par un aspect financier (400\$ de crédits gratuits) mais également pour sa caractéristique cloud-native qui permet de stocker, traiter et analyser un grand volume de données de manière efficace et évolutive. *Snowflake* se différencie de ses concurrents par ses performances. En effet, cette plateforme permet de traiter de grandes quantités de données très rapidement grâce à son architecture de traitement de requêtes massivement parallèle. *Snowflake* permettra ainsi non seulement de contenir l'ensemble de nos données sous forme de table *SQL* relationnelles, mais aussi nous permettra de faire appel à notre modèle de recommandation stocké dans le nuage à partir de requêtes *SQL* basées sur des fonctions externes que nous définirons ultérieurement.

Les étapes qui nous avons réalisé sont les suivantes :

- Etape 1 : Charger l'ensemble des données relatives aux films dans Snowflake ;
- Etape 2 : Créer une image *docker* de l'entraînement et d'inférence personnalisée pour *SageMaker* ;

5 RÉALISATION DU PROTOTYPE

- Etape 3 : Créer une application sans serveur pour connecter *Snowflake* et *SageMaker* en déployant des fonctions *AWS Lambda* et *API Gateway*;
- Etape 4 : Entraînement, déploiement mais aussi inférence du modèle à l'aide des fonctions externes de *Snowflake*.

Etape 1 : Charger l'ensemble des données relatives aux films dans Snowflake

Nous allons dans un premier temps, charger l'ensemble de nos dix fichiers *.csv* dans Snowflake. Pour importer correctement les fichiers *.csv*, il est nécessaire de renseigner dans les requêtes *SQL* le format des données en spécifiant principalement que notre séparateur est la virgule (par défaut), que nous n'importons pas l'entête (nom des colonnes) des fichiers *.csv* ou encore que nous acceptons les champs encadrés par des doubles guillemets, entre autres.

Voici un exemple de création d'une table *SQL* qui contient les données importées à partir du fichier *film.csv*

```

1 create or replace TABLE MOVIELENS.PUBLIC.FILM (
2   ID NUMBER(38,0) NOT NULL,
3   TITLE VARCHAR(255),
4   PAYS VARCHAR(50),
5   ADULT VARCHAR(50),
6   MOYENNEVOTE FLOAT,
7   TOTALVOTE FLOAT,
8   POSTER_LEFT VARCHAR(255),
9   IDACTEUR NUMBER(38,0),
10  IDREALISATEUR NUMBER(38,0),
11  IDGENRE NUMBER(38,0),
12  IDPOPULARITE NUMBER(38,0),
13  IDPERIODE NUMBER(38,0),
14  IDDATESORTIE NUMBER(38,0),
15  POSTER_RIGHT VARCHAR(255),
16  primary key (ID),
17  foreign key (IDACTEUR) references MOVIELENS.PUBLIC.CASTING(ID),
18  foreign key (IDREALISATEUR) references MOVIELENS.PUBLIC.REALISATEUR(ID),
19  foreign key (IDGENRE) references MOVIELENS.PUBLIC.GENRE(ID),
20  foreign key (IDPOPULARITE) references MOVIELENS.PUBLIC.POPULARITE(ID),
21  foreign key (IDPERIODE) references MOVIELENS.PUBLIC.PERIODE(ID),
22  foreign key (IDDATESORTIE) references MOVIELENS.PUBLIC.DATE_SORTIE(ID)
23 );
24
25 COPY INTO FILM FROM @my_stage/film_cl.csv.gz FILE_FORMAT = my_file_format;

```

Listing 3 – Création de la table des films

Une bonne pratique lorsque nous développons des modèles de machine learning est d'utiliser des petits jeux de données ainsi que de procéder à des tests localement afin de gagner en termes de ressources qu'il s'agisse du temps ou d'argent si notre

modèle rencontre un problème. De plus, plus les problèmes sont identifiés tôt dans la " pipeline ", plus simple est de trouver une solution. C'est pour cette raison, que nous avons créé une table qui contient seulement 1000 lignes extraites de la table Note qui contient un total de 20 millions de notes.

Etape 2 : Créer une image docker de l'entraînement et d'inférence pour SageMaker

Le service Amazon SageMaker nous permet d'utiliser nos propres algorithmes de machine learning en utilisant des images Docker. Une image Docker peut être définie comme étant une application qui contient l'ensemble des bibliothèques, des dépendances et des fichiers exécutables permettant de la faire fonctionner de manière isolée, c'est-à-dire indépendamment de l'environnement système hôte. Les images Docker permettent également de faciliter la collaboration entre les développeurs en évitant les différents problèmes de compatibilité. Dans notre cas, nous allons empaqueter dans l'image Docker le code python pour entraîner notre modèle de machine learning mais également le code python pour faire l'inférence du modèle avec la commande suivante :

```
1 docker build -t snf-recommender-lab:latest .
```

En nous basant sur un code disponible sur un répertoire github, nous avons utilisé une fonction d'entraînement python qui sera exécuté par SageMaker et qui comporte les étapes suivantes :

- Récupération des identifiants et connexion avec notre compte Snowflake
- Chargement dans un dataframe python du jeu de données d'entraînement présent dans Snowflake à partir d'une requête SQL
- Entraînement du modèle de recommandation par filtrage collaboratif SVD en utilisant la cross-validation afin d'éviter le surapprentissage du modèle.
- Inférence sur le jeu de données afin de prédire le TOP 10 des recommandations pour chaque utilisateur
- Sauvegarde de l'artefact du modèle dans AWS S3 qui est un service de stockage objet permettant de déployer grâce à SageMaker le modèle afin de faire des prédictions.

Ensuite, une fois qu'une image Docker est créée, elle peut être utilisée pour créer des conteneurs Docker, qui sont des instances exécutables de l'image permettant un déploiement efficace et compatible avec presque tout type de systèmes. Pour se faire, nous avons utilisé Amazon Elastic Container Registry (Amazon ECR) qui est un service de registre de conteneurs entièrement géré par AWS. Cela nous

permettra de stocker, de gérer et de déployer notre image Docker de notre modèle de machine learning de recommandation. En d'autres termes, créer un nouveau rôle d'accès avec des permissions spécifiques dans AWS permet à SageMaker d'utiliser l'image pour l'entraînement et le déploiement de notre modèle afin de faire des prédictions.

Etape 3 : Créer une application sans serveur pour connecter Snowflake et SageMaker en déploiement des fonctions AWS Lambda et API Gateway.

Afin d'intégrer Snowflake avec SageMaker, nous avons mis en place l'AWS API Gateway ainsi que des fonctions AWS Lambda. A noter que l'API Gateway qui contient trois méthodes servira de déclencheur au trois fonctions AWS Lambda (entraînement, déploiement, appel). Les fonctions AWS Lambda permettent d'exécuter des scripts sur nos données sans avoir besoin de serveur. Pour y parvenir, nous avons utilisé le Serverless Framework afin de créer et de déployer les différentes fonctions par programmation et non par le biais de l'interface graphique. Afin d'autoriser les différents accès, nous avons créé un nouveau rôle attaché à des permissions spécifiques dans AWS afin de permettre à Snowflake d'assumer mais aussi d'invoquer les API Gateway. A noter que l'API Gateway repose sur l'architecture REST API qui dans notre cas utilise la méthode HTTP de type POST qui permet au trois méthodes l'API d'envoyer des données. Ensuite, nous avons créé un document de configuration renseignant les différents rôles créés ainsi que le chemin d'accès de l'image de notre modèle de machine learning contenu dans AWS ECR. Enfin, comme dit précédemment, grâce au Serverless Framework, nous avons pu déployer l'API Gateway ainsi que les fonctions AWS Lambda. La prochaine étape est de connecter l'API Gateway à Snowflake. Pour se faire nous avons créé une requête SQL (voir ci-dessus) permettant de se connecter au point de terminaison de l'API Gateway afin d'accéder aux ressources.

```
1 create or replace api integration snf_recommender_api_integration
2 api_provider = aws_api_gateway
3 api_aws_role_arn = '<role>'
4 enabled = true
5 api_allowed_prefixes = ('<https://api_point_terminaison_url>');
```

Ensuite, nous avons créé dans Snowflake les trois fonctions externes suivantes :

- Fonction "train_and_get_recommendations" pour l'entraînement et les prédictions
- Fonction "deploy_model" pour le déploiement du modèle
- Fonction "invoke_model" pour l'appel du modèle

Chaque fonction externe prend la forme de la requête suivante en spécifiant l'API et le point de terminaison de la méthode ("train" ou "deploy" ou "invoke") contenu dans l'API Gateway.

5 RÉALISATION DU PROTOTYPE

```

1 create or replace external function train_and_get_recommendations
2 (table_entree varchar, table_sortie varchar)
3   returns variant
4   api_integration = snf_recommender_api_integration
5   as '<entraînement_point_termination_url>';

```

Etape 4 : Entraînement, déploiement mais aussi inférence du modèle à l'aide des fonctions externes de Snowflake

Une fois les fonctions externes créées, nous avons créé une table "ratings_train_data" qui contient les 10 000 premières lignes de notre table "Note". Nous avons procédé comme tel du fait de nos ressources financières limitées. En effet, la table "Note" contenant 20 millions de lignes, il aurait été très onéreux de faire l'entraînement sur AWS SageMaker. A noter qu'un nombre de 10 000 lignes reste bien suffisant pour obtenir des prédictions relativement précises. Nous avons également créé une table qui contient les résultats des prédictions (TOP10) des utilisateurs. Pour déclencher l'entraînement de notre modèle de machine learning de recommandation, nous utilisons la requête SQL suivante :

```

1 Select train_and_get_recommendations(    donnees_entrainement    ,
      recommandation_predite    );

```

Cette requête fait appel à la fonction externe que nous avons préalablement créée et prend en paramètre la table du jeu de donnée d'entraînement ainsi que la table qui contiendra le résultat de la requête c'est-à-dire les prédictions. Cette requête va alors déclencher un " Training Job " dans AWS SageMaker qui se chargera de créer un modèle basé sur l'algorithme SVD et de prédire le TOP 10 des recommandations pour les utilisateurs. Nous avons également implémenté des prédictions en temps réel. En d'autres termes, l'utilisateur sera en mesure de connaître une note estimée d'un film qu'il n'a pas encore noté afin de lui permettre de savoir si le film pourrait éventuellement lui plaire. Pour se faire, nous avons créé un point de terminaison dans AWS SageMaker nous permettant d'accéder à l'artefact de notre modèle entraîné qui est stocké dans AWS S3. Plus précisément, ce point de terminaison a été créé à partir d'une requête SQL qui fait appel à la fonction externe " deploy_model " qui a été préalablement définie. Cela nous permet de déployer le modèle en tant qu'API.

```

1 select deploy_model('movielens-model-v1', '<s3_artefact_du_model>') ;

```

À partir de l'interface, lorsque l'utilisateur clique sur le bouton " Note Estimée ", nous faisons appel à la fonction externe " invoke-model " préalablement créée qui fait elle-même appel au modèle en tant qu'API pour générer une prédiction de la note pour l'utilisateur connecté.

5 RÉALISATION DU PROTOTYPE

```
1 select nn.utilisateur_id, nn.film_id, m.titre, invoke_model('movielens-model-v1', nn
.utilisateur_id, nn.film_id) as note_predite from non_note nn, film f where nn.
film_id = f.id;
```

En termes d'amélioration de notre processus, nous pourrions à l'avenir automatiser notre pipeline de machine learning. En effet, en permettant à l'utilisateur de noter des films de façon dynamique mais aussi en ajoutant de nouveaux films dans la base de données, nous pouvons automatiser l'entraînement de notre modèle de machine learning de recommandation.

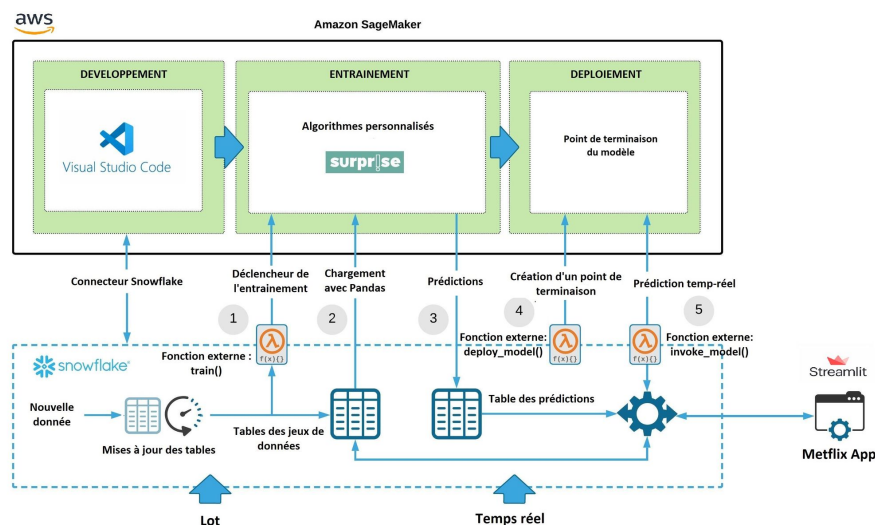


Figure 5 – Architecture du déploiement de bout en bout [11]

5.2 Interface utilisateur

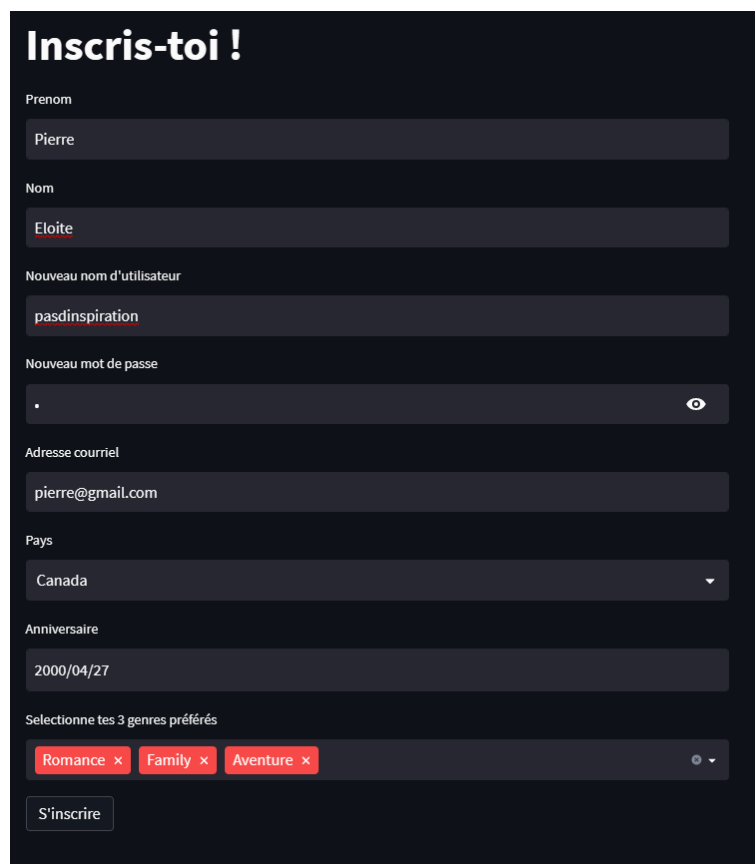
L'interface utilisateur est composée de différentes pages sur lesquelles l'utilisateur pourra se rendre pour interagir avec la plateforme.

- La page d'inscription : elle permet à l'utilisateur de s'inscrire après avoir renseigné ses informations ;
- La page d'authentification : elle permet à l'utilisateur déjà inscrit de s'authentifier ;
- La page de recommandation : c'est la page principale sur laquelle l'utilisateur voit les films qui lui sont suggérés ;
- La page de recherche : elle permet à l'utilisateur de rechercher manuellement des films dans la base de données.

Ces pages de l'interface utilisateur ont été développées avec *Streamlit* qui est un framework *Python* permettant de créer des interfaces qui ont majoritairement pour vocation de présenter des données et leurs analyses.

5.2.1 Page d'inscription des utilisateurs

La première page sur laquelle arrivera un utilisateur utilisant notre prototype est la page d'inscription. Celle-ci lui demandera plusieurs informations génériques telles que son nom ou son mot de passe. Mais il lui sera aussi demandé d'entrer des informations plus complémentaires comme sa date de naissance, son pays de résidence ainsi que les genres qu'il affectionne le plus en matière de cinéma. Cela permettra par la suite à la plateforme de lui proposer du contenu personnalisé. De cette manière le problème du démarrage à froid est contourné. Par la suite, lorsque l'utilisateur reviendra sur la plateforme il pourra s'authentifier via une page d'authentification et ainsi retrouver les informations de son compte.



The screenshot shows a registration form on a dark background. The title 'Inscris-toi !' is at the top. The form includes the following fields and elements:

- Prenom:** Text input with 'Pierre' entered.
- Nom:** Text input with 'Eloïte' entered.
- Nouveau nom d'utilisateur:** Text input with 'pasdinspiration' entered.
- Nouveau mot de passe:** Password input field with a toggle icon on the right.
- Adresse courriel:** Text input with 'pierre@gmail.com' entered.
- Pays:** Dropdown menu with 'Canada' selected.
- Anniversaire:** Date input field with '2000/04/27' entered.
- Selectionne tes 3 genres préférés:** A row of three red buttons labeled 'Romance', 'Family', and 'Aventure', each with a close icon (x). A dropdown arrow is on the right.
- S'inscrire:** A button at the bottom of the form.

Figure 6 – Page d'inscription

Dans la perspective d'améliorer les fonctionnalités de cette plateforme, il serait possible de demander à l'utilisateur les acteurs et les réalisateurs dont il apprécie le travail, de la même manière que ce qui est fait avec les genres. Ainsi, il serait possible par la suite de faire des suggestions de films en fonction des acteurs et réalisateurs appréciés de l'utilisateur.

5.2.2 Page des recommandations

Une des pages principales de notre prototype est la page des recommandations. Elle permet à l'utilisateur de voir ce que la plateforme lui propose en fonction de différents critères. Les films proposés sont classés en plusieurs catégories selon l'élément qui fait qu'ils ont été proposés à l'utilisateur. Cet affichage permet la transparence : l'utilisateur saura pourquoi un film lui a été proposé et cela permettra aussi de le guider dans son choix.

Les films vont être classés en fonction :

- Des genres favoris entrés par l'utilisateur à la création de son compte : si un utilisateur a indiqué qu'il aimait les comédies et les films d'action, ces deux parties lui seront proposées ;
- Du filtrage collaboratif des notes qui ont été données par d'autres utilisateurs : les notes données aux films par la communauté d'utilisateur vont permettre grâce aux techniques d'apprentissage machine de faire des prédictions sur ce que l'utilisateur est susceptible d'apprécier ;

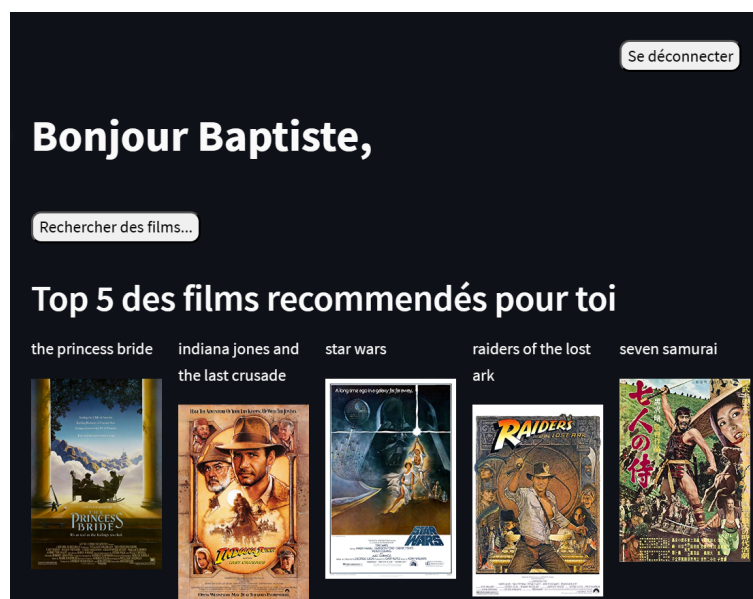


Figure 7 – Page d'accueil avec les recommandations

- De la période de l'année dans laquelle l'utilisateur se trouve à un instant donné : en effet, il est pertinent par exemple de proposer des films de Noël en période de Noël et des films d'horreur en période d'Halloween, les différentes périodes qui sont prises en compte par le prototype sont Noël, Halloween, les vacances d'été, la Saint Valentin, l'action de Grâce, l'Armistice de la Première Guerre mondiale, l'Armistice de la Seconde Guerre mondiale ect. ;

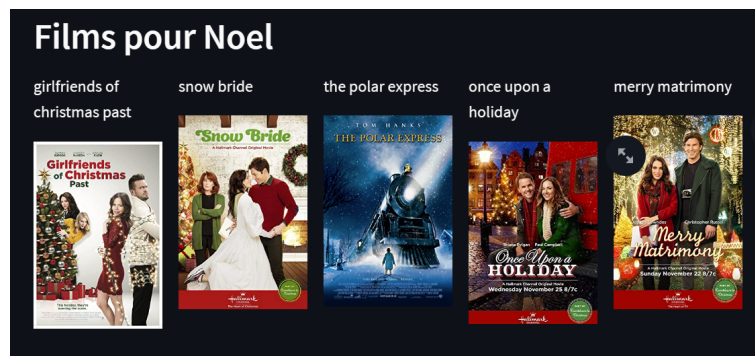


Figure 8 – Recommandation par rapport à la période de l'année, Noël dans ce cas

- Du pays de résidence de l'utilisateur : l'utilisateur se verra proposer des films qui ont été produits complètement ou en partie dans son pays de résidence. En plus de permettre à l'utilisateur de découvrir d'autres films que ceux associées à ce qu'il préfère habituellement, cela lui permet de découvrir ce qui est fait chez lui ;
- Du succès qu'ont connu les films : les films qui ont eu du succès un jour, qu'ils soient récents ou anciens, sont proposés à l'utilisateur, on retrouve dans cette catégorie aussi bien les grands classiques que les blockbusters ;
- De la date de naissance de l'utilisateur : des films sortis durant l'année de naissance de l'utilisateur lui sont proposés. Cette catégorie permet essentiellement à l'utilisateur de découvrir des films qu'il n'aurait pas découverts autrement, il n'y a effectivement aucune raison pour qu'il préfère les films sortis l'année de sa naissance. Cette catégorie peut également satisfaire la curiosité d'un utilisateur curieux de voir ce qui passait en salle l'année où il est né ;

Pour chaque catégorie citée plus haut, cinq films sont proposés à l'utilisateur et ces films changent au rechargement de la page. Ainsi, si un utilisateur n'est pas convaincu par ce qui lui est proposé, il peut tout simplement recharger la page. Dans les cas des utilisateurs mineurs, c'est-à-dire ayant moins de 18 ans, les films ayant leur attribut "Adult" égal à *True* ne leur seront pas proposés. Ainsi, le prototype ne risque pas de proposer à ces utilisateurs du contenu qui ne leur est pas destiné et qui pourrait les heurter. Dans la perspective d'améliorer ce point, nous aurions pu demander à l'utilisateur lorsqu'il s'inscrit sur le site d'indiquer s'il y a des types de film ou des films associés à certains mots clefs qu'il ne veut absolument pas voir.

Dans le but d'améliorer les recommandations faites aux utilisateurs nous aurions pu également entrainer un modèle du filtrage basé sur le contenu en plus du filtrage collaboratif.

5.2.3 Page de recherche de films

La page de recherche permet à l'utilisateur de faire une recherche de film par rapport à un mot clef qu'il rentre. S'il inscrit dans la barre de recherche, par exemple, le mot clef "Batman", les films contenant le nom "Batman" dans leur titre dans la base de donnée seront proposés. La page contient également une case à cocher qui propose à l'utilisateur de faire un filtrage supplémentaire en fonction du genre du film, du réalisateur, de la moyenne des notes ainsi que de la date de sortie. Pour chaque film affiché dans le cas où l'utilisateur ne l'a pas noté, une note générée par le modèle de filtrage collaborative pourra être affichée. De cette manière, l'utilisateur aura un aperçu de la note qu'il pourrait donner et saura s'il est susceptible d'aimer le film. Dans la perspective d'améliorer les fonctionnalités de cette plateforme, il serait nécessaire d'empêcher la duplication des films hors de la recherche. En effet, durant la recherche de film, l'utilisateur se voit proposer les mêmes films mais avec des notations différentes.

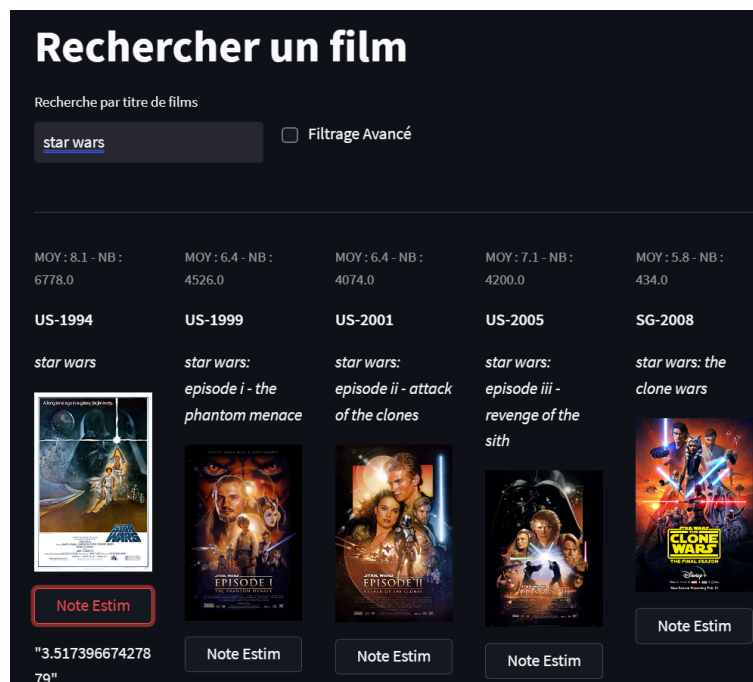


Figure 9 – Page de recherche

6 Contributions

Etudiant	Description des contributions
Baptiste Viera	<p>Déploiement de bout en bout sur Microsoft Azure (stockage des données, entraînement et déploiement du modèle en tant qu'API)</p> <p>Chargement de l'ensemble des données et création des tables dans Snowflake</p> <p>Connexion et intégration de Snowflake avec Streamlit sur l'ensemble des pages de l'application</p> <p>Déploiement de bout en bout sur Amazon AWS (entraînement et déploiement du modèle)</p> <p>Amélioration de l'interface utilisateur sur Streamlit</p> <p>Rédaction du rapport</p> <p>Gestion de projet</p>
Emma Ollivier	<p>Revue de la littérature</p> <p>Choix des sources de données</p> <p>Extraction des données</p> <p>Création de l'architecture des données</p> <p>Implémentation de la page de recommandations</p> <p>Rédaction du rapport</p> <p>Gestion de projet</p>
Kenia Pineda Mesa	Interface utilisateur de la page des recommandations sur Streamlit
Mohamed El Amine Meguenani	<p>Détection des anomalies</p> <p>Nettoyage de données</p> <p>Création du tableau de bord</p> <p>Rédaction du rapport</p>
Paul Deschildre	Pages d'authentification et d'inscription
Vanick Djamien Djofang	<p>Création de la page de recherche</p> <p>Rédaction du rapport</p>

Table 1 – Tableau résumant les contributions des membres du projet

7 Conclusion

Arrivés au terme de la réalisation de ce prototype, nous pouvons affirmer que nous avons réalisé un système de recommandation de films de bout en bout, de la récupération des données en format .csv jusqu'à la réalisation d'une plateforme de recommandation en passant par la création d'une base de données infonuagique et le déploiement dans le nuage d'un modèle d'apprentissage machine. Dans l'implémentation de ce prototype, nous avons tenté de répondre aux défis de performance et d'éthiques posés par les systèmes de recommandations, notamment en suggérant des films très personnalisés pour l'utilisateur mais aussi des films qui le sont moins pour l'empêcher d'être enfermé dans ses propres préférences, tout en veillant à ne pas lui proposer des films pouvant le déranger.

Toutefois, des améliorations sont encore possibles afin de rendre ce prototype plus performant. Nous pourrions, en effet, permettre à l'utilisateur de noter des nouveaux films, utiliser la méthode d'apprentissage machine de filtrage basé sur le contenu ou encore d'automatiser la pipeline d'apprentissage pour réapprendre automatiquement un nouveau modèle.

8 Bibliographie

- 1 Roy, D., Dutta, M. A systematic review and research perspective on recommender systems. *J Big Data* 9, 59 (2022). <https://doi.org/10.1186/s40537-022-00592-5>
- 2 Schedl, M., Zamani, H., Chen, CW. et al. Current challenges and visions in music recommender systems research. *Int J Multimed Info Retr* 7, 95–116 (2018). <https://doi.org/10.1007/s13735-018-0154-2>
- 3 Milano, S., Taddeo, M. Floridi, L. Recommender systems and their ethical challenges. *AI Soc* 35, 957–967 (2020). <https://doi.org/10.1007/s00146-020-00950-y>
- 4 N. Seaver, “Captivating algorithms : Recommender systems as traps.” *Journal of Material Culture*, 24(4), 421–436, 2019, <https://doi.org/10.1177/1359183518820366>
- 5 Gomez-Uribe, C-A., Hunt, N., The Netflix Recommender System : Algorithms, Business Value, and Innovation. *ACM Trans. Manage. Inf. Syst.* 6, 4, Article 13 (January 2016), <https://doi.org/10.1145/2843948>
- 6 Roh, Y., Heo, G., Euijong Whang S., A Survey on Data Collection for Machine Learning : a Big Data – AI Integration Perspective, 2019,1811.03402, arXiv, <https://doi.org/10.48550/arXiv.1811.03402>
- 7 Rounak Banik, The movie Dataset, Kaggle www.kaggle.com/datasets/rounakbanik/the-movies-dataset
- 8 CrowsFlower, Blockbuster Database, Data world, <https://data.world/crowdfower/blockbuster-database>
- 9 Documentation de Streamlit, <https://docs.streamlit.io/>
- 10 Microsoft recommenders, Miguelgferro, Github, <https://github.com/microsoft/recommenders>
- 11 Snowflake-Labs, Filanthropic, Github, <https://github.com/Snowflake-Labs/sfguide-recommender-pipeline>

Table des figures

1	UML de l'architecture de données	10
2	Modèle de relation entre les tables	13
3	Visualisation de notre tableau de bord	14
4	Exemple de publication du tableau de bord	15
5	Architecture du déploiement de bout en bout [11]	22
6	Page d'inscription	23
7	Page d'accueil avec les recommandations	24
8	Recommandation par rapport à la période de l'année, Noël dans ce cas	25
9	Page de recherche	26