Rapport de projet Machine Learning : ChooseUrMovie

Recommandation de film.

Rédigé par :

CHEBALLAH Jawed

Table des matières

1. Contexte

Le secteur du divertissement, en particulier celui de la diffusion de films, a évolué considérablement avec l'intégration des technologies de recommandation basées sur l'intelligence artificielle, permettant aujourd'hui aux utilisateurs de découvrir des films adaptés à leurs goûts de manière personnalisée et fluide. Notre projet de recommandation de films vise à moderniser la façon dont les utilisateurs explorent le catalogue cinématographique, en leur offrant des suggestions pertinentes basées sur leurs préférences, leur historique de visionnage et des tendances globales.

L'objectif principal est de simplifier la découverte de films pour les utilisateurs et de fournir une interface intuitive et engageante. Grâce à une base de données centralisée et à des algorithmes de Machine Learning, le système gérera efficacement les films, les préférences des utilisateurs et les comptes. La confidentialité des données sera garantie par des protocoles stricts de gestion des identités et des données personnelles.

La plateforme permettra aux utilisateurs de recevoir des recommandations en temps réel en fonction de critères variés tels que le genre, les acteurs, les réalisateurs ou les films similaires qu'ils ont appréciés. Ils pourront également évaluer les films, laisser des commentaires, et gérer leurs listes de films à voir. Le projet inclut un système de recommandation personnalisé qui utilise des techniques de filtrage collaboratif et de filtrage basé sur le contenu, offrant une expérience sur mesure pour chaque utilisateur.

En résumé, ce projet propose une solution complète et innovante, alliant technologie avancée et personnalisation, pour améliorer l'expérience de découverte et d'engagement des utilisateurs dans l'univers du cinéma.

2. Introduction

I. Différents systèmes de recommandation.

Pour créer un système de recommandation efficace pour les films, il est essentiel de prendre en compte plusieurs paramètres afin de proposer des suggestions pertinentes et personnalisées aux utilisateurs. Voici une description enrichie des principaux axes à considérer pour ce type de système :

- **Recommandation basée sur le contenu** : Ce modèle repose sur une analyse approfondie des caractéristiques intrinsèques des films, tels que le genre, le casting, le réalisateur, le synopsis, ou encore le style visuel et narratif. En comparant ces éléments à ceux des films déjà appréciés par l'utilisateur, le système peut identifier des œuvres similaires, et ainsi proposer des films qui correspondent aux préférences de l'utilisateur en matière de contenu. Ce type de recommandation mise donc sur les similitudes thématiques ou stylistiques.

- **Recommandation basée sur la popularité** : Ce modèle s'appuie sur les films qui ont connu un grand succès auprès du grand public, que ce soit en termes de critiques positives, de taux d'audience élevé ou de discussions sur les réseaux sociaux. Le principe ici est de suggérer des films populaires, souvent considérés comme des « valeurs sûres », qui ont plu à un large éventail de spectateurs. Même si ce type de recommandation n'est pas toujours personnalisé, il est souvent pertinent pour les utilisateurs cherchant des films largement appréciés.

- **Recommandation collaborative basée sur des groupes d'utilisateurs similaires** : Ici, le système de recommandation utilise des algorithmes collaboratifs pour suggérer des films en se basant sur les comportements d'un groupe de personnes ayant des goûts similaires. Si un utilisateur a regardé et apprécié certains films, et que d'autres utilisateurs ayant des goûts comparables ont apprécié un film particulier, celui-ci sera recommandé. Cette approche se fonde sur la dynamique de groupe et l'intelligence collective pour proposer des suggestions qui vont au-delà des simples préférences individuelles, en tirant parti des expériences partagées au sein d'une communauté d'utilisateurs.

  Ce modèle de recommandation hybride permet de combiner plusieurs types de données (contenu, tendances populaires et similarités entre utilisateurs) pour fournir des suggestions personnalisées, riches et pertinentes.

II. Workflow.

Pour commencer l'étude des données dans le cadre d'un système de recommandation de films, il est essentiel de procéder par étapes structurées, allant de l'extraction des données pertinentes à leur traitement, afin de pouvoir offrir des recommandations personnalisées. Nous allons utiliser une approche basée sur la **Cosine Similarity** pour comparer les films en fonction de leur contenu, et ainsi recommander ceux qui sont les plus proches des préférences de l'utilisateur. Voici une description enrichie de la démarche :

1. Étude et extraction des données

La première étape consiste à analyser et extraire les informations clés des films à partir d'une base de données. Cela inclut des métadonnées telles que :

- **Titres de films**,
- **Genres** (action, comédie, drame, science-fiction, etc.),
- **Réalisateurs**,
- **Acteurs principaux**,
- **Résumé ou synopsis**,
- **Mots-clés** associés au film (par exemple, thèmes comme "voyage dans le temps", "aventure spatiale", etc.).

Ces informations doivent être soigneusement organisées et nettoyées pour être exploitables dans le cadre de la méthode de recommandation.

2. Transformation des données textuelles en vecteurs

Afin de pouvoir comparer les films entre eux, nous devons transformer les informations textuelles (comme les titres, les genres ou les résumés) en **vecteurs numériques**. Cette étape est cruciale car elle permet de calculer la similarité entre deux films à partir de leurs descriptions respectives.

Utilisation du modèle TF-IDF (Term Frequency-Inverse Document Frequency)

Une des méthodes les plus courantes pour transformer les chaînes de caractères en vecteurs est d'utiliser TF-IDF. Ce modèle attribue un poids à chaque terme (mot ou expression) en fonction de sa fréquence dans le film (TF) et de son importance globale dans l'ensemble de données (IDF). Cela permet d'identifier les mots qui sont représentatifs du film tout en filtrant les termes trop communs.

Une fois les termes représentés sous forme de vecteurs, chaque film est désormais associé à un vecteur dans un espace multidimensionnel.

3. Calcul de la similarité à l'aide de la méthode Cosine Similarity

Pour mesurer la proximité entre deux films, nous utiliserons la **Cosine Similarity**. Cette méthode est largement employée pour comparer la similarité de deux vecteurs en calculant le **cosinus de l'angle** entre eux. Le résultat est une valeur comprise entre -1 et 1 :

- **1** indique une similarité parfaite,
- **0** signifie qu'il n'y a aucune relation entre les deux films,
- **-1** reflète une différence totale (ce qui est rare dans ce cas).

4. Traitement des requêtes utilisateurs

Une fois les films représentés sous forme de vecteurs, nous pouvons alors **prendre en compte les requêtes des utilisateurs**. Par exemple, si un utilisateur saisit un titre ou une description de film, nous transformons également cette entrée en vecteur à l'aide du modèle TF-IDF, puis calculons la **similarité cosinus** entre la requête de l'utilisateur et chaque film de la base de données.

Le système recommandera ensuite les films ayant une similarité élevée avec la requête, garantissant ainsi que les suggestions sont pertinentes par rapport à ce que l'utilisateur recherche.

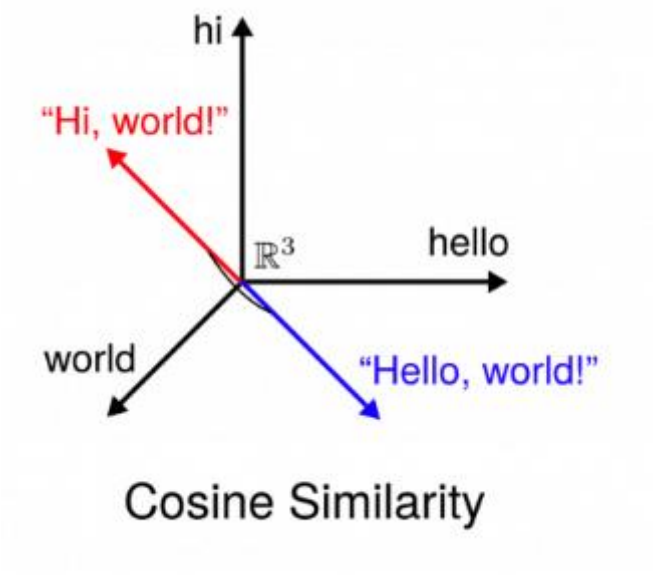
5. Liste des films recommandés

Enfin, après avoir comparé les films à l'aide de la méthode Cosine Similarity, une **liste des films recommandés** sera générée, triée par ordre décroissant de similarité. Cela permet à l'utilisateur d'obtenir des suggestions de films qui sont les plus proches en termes de contenu, assurant ainsi une expérience personnalisée et enrichissante.

Avantages de l'approche basée sur la Cosine Similarity :

- **Précision dans les recommandations** : Grâce à la transformation des descriptions de films en vecteurs, le système est capable de repérer les similitudes subtiles entre les films, offrant des recommandations plus fines.
- **Rapidité** : Le calcul de la similarité cosinus est efficace même sur de grandes bases de données, garantissant une réponse rapide aux requêtes des utilisateurs.
- **Flexibilité** : Cette méthode fonctionne aussi bien avec des informations textuelles comme les titres, les résumés ou les genres, mais peut être enrichie avec d'autres caractéristiques (acteurs, réalisateurs, etc.).

En utilisant la **Cosine Similarity**, nous construisons ainsi un moteur de recommandation de films qui allie précision et flexibilité, en fournissant des suggestions pertinentes à partir d'une simple requête de l'utilisateur.



3. Bibliothèques utilisées

Voici une présentation détaillée des bibliothèques que nous utiliserons pour le développement de notre système de recommandation de films, enrichie avec des définitions et des liens pour chaque bibliothèque :

1. NumPy :

- **Définition** : NumPy (Numerical Python) est une bibliothèque puissante pour le calcul scientifique et numérique en Python. Elle fournit un support pour des tableaux multidimensionnels (appelés ndarray) et une large collection de fonctions mathématiques pour opérer sur ces tableaux. NumPy est souvent utilisé pour effectuer des opérations sur des matrices et des vecteurs, ce qui le rend essentiel dans le domaine de la manipulation de données numériques et des algorithmes d'apprentissage automatique.
- **Utilité dans notre cas** : Nous utiliserons NumPy pour gérer les vecteurs de données extraits des descriptions de films et effectuer des opérations mathématiques, comme le calcul de la similarité entre films via la Cosine Similarity.
- **Lien** : NumPy Documentation

2. Pandas :

- **Définition** : Pandas est une bibliothèque open-source pour la manipulation et l'analyse des données en Python. Elle fournit des structures de données flexibles, comme les DataFrames, qui permettent de stocker et manipuler facilement des ensembles de données tabulaires. Pandas facilite également le nettoyage, la transformation et l'analyse de données provenant de diverses sources.
- **Utilité dans notre cas** : Pandas sera utilisé pour charger, organiser et manipuler les données des films (titres, genres, résumés, etc.). Nous utiliserons les DataFrames de Pandas pour stocker ces informations et préparer les données pour l'analyse et les recommandations.
- **Lien** : Pandas Documentation

3. Difflib :

- **Définition** : Difflib est une bibliothèque standard de Python qui fournit des outils pour comparer des séquences, comme des chaînes de caractères. Elle peut être utilisée pour trouver des correspondances entre des chaînes similaires et proposer des corrections, comme suggérer des titres de films qui ressemblent à celui entré par l'utilisateur.
- **Utilité dans notre cas** : Difflib sera utilisé pour gérer les erreurs ou fautes de frappe dans les requêtes des utilisateurs. Par exemple, si un utilisateur saisit un titre de film incorrect, Difflib pourra proposer des correspondances proches afin de corriger automatiquement la requête.
- **Lien** : Difflib Documentation

4. Scikit-learn (sklearn) :

- **Définition** : Scikit-learn est une bibliothèque open-source largement utilisée pour l'apprentissage automatique en Python. Elle propose une vaste gamme d'algorithmes de machine learning, de classification, de régression, et d'analyse de données. Elle inclut aussi des outils pour le traitement des données comme le **TF-IDF** (Term Frequency-Inverse Document Frequency) et le calcul de la **similarité cosinus**.
- **Utilité dans notre cas** : Scikit-learn sera utilisé pour transformer les descriptions des films en vecteurs numériques à l'aide du modèle **TF-IDF**. Ensuite, nous utiliserons la méthode **Cosine Similarity** de Scikit-learn pour calculer la similarité entre ces vecteurs et recommander les films les plus pertinents en fonction de la requête de l'utilisateur.
- **Lien** : Scikit-learn Documentation

Récapitulatif de l'utilisation des bibliothèques dans notre projet :

- **NumPy** : Manipulation de vecteurs et de matrices pour les calculs mathématiques.
- **Pandas** : Organisation et manipulation des données des films dans des structures tabulaires.
- **Difflib** : Comparaison de chaînes de caractères pour gérer les fautes de frappe ou erreurs dans les titres de films.
- **Scikit-learn (sklearn)** : Transformation des descriptions en vecteurs (TF-IDF) et calcul de la Cosine Similarity pour mesurer la proximité entre les films.

4. Processus de réalisation

I. Récupération de données

Nous allons travailler sur un fichier csv avec le nom des films, les acteurs, les metteurs en scene, synopsis etc etc (fichier dans le zip) stocké dans un dataframe pandas avec la fonction pandas ***: pd.read_csv(`Chemin du fichier`)*** stocké dans une variable ***movies_data***
On utilise la fonction : ***movies_data.head()*** pour récupérer les 5 premières lignes du dataframe :

0	237000000	Action Adventure Fa	http://www.avatarmo	19995	culture clash future s	en	Avatar
1	300000000	Adventure Fantasy A	http://disney.go.com/	285	ocean drug abuse ex	en	Pirates of the Caribb
2	245000000	Action Adventure Cri	http://www.sonypictu	206647	spy based on novel s	en	Spectre
3	250000000	Action Crime Drama	http://www.thedarkkni	49026	dc comics crime figh	en	The Dark Knight Rise
4	260000000	Action Adventure Sc	http://movies.disney.	49529	based on novel mars	en	John Carter
5	258000000	Fantasy Action Adve	http://www.sonypictu	559	dual identity amnesi	en	Spider-Man 3

II. Visualisation et sélection des données

Tout d'abord, pour vérifier la taille du DataFrame, vous pouvez utiliser la propriété ***shape*** de **pandas**. Cette instruction vous donnera un tuple indiquant le nombre de lignes et de colonnes dans votre DataFrame, ce qui vous permettra de voir combien de films sont inclus et combien d'attributs sont disponibles pour chaque film.

Ensuite, pour sélectionner les colonnes pertinentes, vous devez créer un nouveau DataFrame qui contient uniquement les colonnes qui vous intéressent : 'genre', 'mots-clés', 'slogan', 'acteurs' et 'metteurs en scène'. Vous pouvez le faire en indiquant les noms de ces colonnes entre crochets après le nom du DataFrame original. Cela filtrera le DataFrame pour ne conserver que ces informations, vous permettant de vous concentrer sur les données qui sont essentielles pour le système de recommandation.

Après avoir isolé les colonnes nécessaires, vous devez gérer les valeurs manquantes dans ces colonnes. Pour remplacer les valeurs manquantes (NaN) par un espace vide, vous utiliserez la méthode ***fillna(' ')*** sur les colonnes sélectionnées. Cette méthode parcourt chaque valeur manquante dans les colonnes que vous avez choisies et la remplace par un espace. Cela est crucial pour éviter des erreurs dans les étapes ultérieures de traitement des données, comme la création de vecteurs ou le calcul de similarités.

Enfin, pour combiner toutes ces colonnes en une seule structure de données, vous pouvez concaténer les valeurs de ces colonnes pour chaque film. Vous utiliserez une méthode de

concaténation qui combine les valeurs de chaque colonne en une seule chaîne de texte par ligne. Cela signifie que pour chaque film, vous aurez une chaîne de texte unique qui contient toutes les informations pertinentes : le genre, les mots-clés, le slogan, les acteurs et les metteurs en scène. Vous pouvez créer une nouvelle colonne dans le DataFrame pour stocker ces chaînes combinées.

De cette manière, vous obtiendrez un DataFrame nettoyé et simplifié, où chaque ligne représente un film avec toutes les informations importantes combinées en une seule chaîne. Ce DataFrame sera prêt pour les prochaines étapes d'analyse, comme la vectorisation du texte et le calcul de la similarité pour les recommandations.

III. Conversion des datas & « Cosine Similarity »

Pour retrouver les similarités entre les données textuelles (comme les genres, mots-clés, acteurs, etc.), nous devons, comme évoqué précédemment, **transformer les chaînes de caractères en vecteurs numériques**. Cette transformation permet de comparer ces données de manière mathématique. Pour ce faire, nous utilisons l'outil **TfidfVectorizer** fourni par la bibliothèque **SkLearn**.

Tout d'abord, nous devons créer un objet **TfidfVectorizer**. Cet objet va permettre de convertir les données textuelles en vecteurs, en attribuant à chaque mot un poids basé sur sa fréquence dans le document (le film, dans ce cas) et sa rareté dans l'ensemble des films. Cela permet de mettre en avant les mots qui sont particulièrement importants pour un film donné, et de minimiser l'impact des mots trop communs.

Une fois l'objet **vectorizer** créé, nous utilisons la fonction **fit_transform()**. Cette fonction va :

- Analyser le texte dans les colonnes que nous avons sélectionnées (comme les genres, les acteurs, etc.).
- Créer une matrice où chaque ligne correspond à un film et chaque colonne correspond à un terme (mot) du vocabulaire extrait des descriptions des films.
- Transformer les descriptions textuelles en vecteurs numériques en fonction du poids de chaque mot.

Ces vecteurs seront stockés dans une nouvelle variable, que nous pouvons appeler **feature_vectors**. Chaque ligne de cette matrice correspond à un film sous forme de vecteur de caractéristiques (une représentation numérique basée sur son contenu textuel).

Après cela, pour comparer ces films entre eux, nous utilisons la méthode **cosine_similarity()**. Cette méthode calcule la similarité entre chaque paire de films en utilisant la distance cosinus. Elle renvoie un tableau carré où chaque cellule (i, j) indique le score de similarité entre le film i et le film j.

Les scores de similarité vont de 0 à 1, où :

- 1 signifie une similarité parfaite (le film est comparé à lui-même, donc ils sont identiques).
- Des valeurs proches de 0 indiquent peu ou pas de similarité.
- Des valeurs intermédiaires, comme 0.072, indiquent un certain degré de ressemblance (par exemple, 7,2 % de similarité).

Ce tableau des similarités aura une dimension égale au nombre de films présents dans le DataFrame. Cela signifie que, pour chaque film, il est comparé à tous les autres films, y compris lui-même, ce qui explique pourquoi la diagonale de ce tableau est composée de 1 (chaque film est identique à lui-même).

Le résultat final sera un tableau ou une matrice de forme **(nombre de films, nombre de films)**. Cela nous permet de visualiser comment chaque film se compare à tous les autres films, en termes de similarité de contenu, et de choisir les films les plus similaires à recommander.

Ainsi, grâce à **TfidfVectorizer** et **cosine_similarity()**, nous pouvons transformer les descriptions textuelles en vecteurs et calculer les similarités entre tous les films pour les recommandations.

IV. Récupération du film de l'utilisateur

La première étape consiste à demander à l'utilisateur quel est son film préféré. Ensuite, nous devons comparer ce film avec les autres pour trouver les films les plus similaires à recommander. Cependant, un problème courant survient : **les fautes d'orthographe** dans la saisie du titre par l'utilisateur. Pour contourner cela, nous utiliserons **Difflib** pour corriger les éventuelles erreurs de saisie et trouver le film correct.

Lorsque l'utilisateur entre son film préféré, nous utilisons la fonction **get_close_matches()** de la bibliothèque **Difflib**. Cette fonction permet de comparer la chaîne de caractères donnée par l'utilisateur (le titre du film) avec la liste des titres de films que nous avons récupérée dans le DataFrame principal. En fournissant la liste complète des titres de films à la fonction, nous pouvons obtenir une correspondance qui est la plus proche du titre entré, même en cas d'erreur d'orthographe ou de faute de frappe.

Après avoir utilisé **get_close_matches()**, nous obtenons une liste de suggestions de titres similaires. Nous récupérons simplement le **premier film** de cette liste, car il correspond généralement à la meilleure correspondance trouvée.

Ensuite, pour procéder à l'analyse, nous devons récupérer l'index de ce film dans notre DataFrame. Cet index nous permettra d'extraire les informations de ce film et de le comparer aux autres films en fonction des scores de similarité que nous avons déjà calculés avec la méthode **cosine_similarity()**.

Une fois l'index du film préféré récupéré, nous utilisons les **scores de similarité** pour comparer ce film avec tous les autres films dans la base de données. Nous trions ces scores dans l'ordre décroissant pour obtenir une liste des films les plus proches (ayant les scores de similarité les plus élevés).

Pour finir, nous sélectionnons le **top 10 des films les plus similaires** en fonction de leur score de similarité. Ces films sont alors affichés à l'utilisateur comme une liste de recommandations basées sur son film préféré. En combinant **Difflib** pour gérer les erreurs de saisie et **cosine_similarity()** pour identifier les films similaires, nous assurons une recommandation fluide et précise.

Ainsi, même en cas de fautes d’orthographe ou d’inexactitudes dans la saisie du titre, l'utilisateur pourra toujours recevoir des suggestions de films proches de celui qu'il aime, grâce à une correction automatique et un tri des scores de similarité.