



Réf : 2019/2020

Université Chouaib Doukkali
Ecole Nationale des Sciences Appliquées d'El Jadida
Département Télécommunications, Réseaux et Informatique



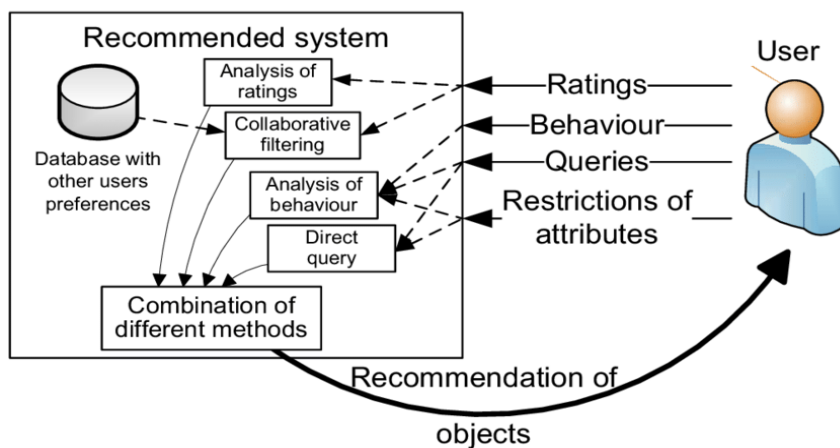
PROJET DE FIN D'ANNEE

Filière : 2ITE

Niveau : 2^{ème} Année

Sujet :

Développement d'une plateforme de recommandation de contenu



Réalisé Par :

BAHLOUL Youssef
NAHED Souhail

Encadré par :

Prof. Nouredine ASSAD

Présenté le **01/07/2020** devant le jury composé de :

Prof. Nouredine ASSAD,
Prof. Mohamed EL Boujnoui,

Année Universitaire : 2019/2020

Professeur à l'ENSAJ
Professeur à l'ENSAJ

Résumé

Depuis quelques années, l'apprentissage automatique (ML) devient de plus en plus important. Il est utilisé partout, notamment dans les systèmes de recommandation. De grandes sociétés de films en streaming comme NETFLIX construisent ces systèmes afin de recommander du contenu à leurs clients et d'augmenter leurs profits. Ce n'est pas seulement NETFLIX qui utilise les systèmes de recommandation mais aussi Amazon, YouTube, Spotify, etc.

Le principal problème de ces entreprises (surtout les entreprises des films) c'est la gestion des goûts et des préférences des utilisateurs, ce problème est extrêmement difficile à gérer car les gens changent au fil du temps [1]. Donc ces entreprises se concentrent sur leur propre contenu. Ils ne recommandent évidemment que le contenu dont ils disposent et ne prennent pas en considération toute l'offre de films existants. L'utilisateur ne peut pas devenir global et objectif de recommandations parce qu'il est au milieu de plusieurs systèmes de recommandation qui ne sont pas complet.

Le but de ce projet est d'abord d'étudier l'apprentissage automatique (ML) et les façons dont il peut être utilisé dans le cas d'un système de recommandation de films. Ensuite, l'objectif est de mettre en œuvre l'un de ces solutions dans un projet d'application web pour des recommandations de films.

Le projet consiste à développer une application web entièrement fonctionnelle qui permet de recommander des films aux utilisateurs en se basant sur le contenu de ce dernier tel que le genre, l'année de sortie.

Remerciements

Ce n'est pas parce que la traduction l'exige ou par l'habitude que cette page est présente dans chaque rapport, mais parce que les personnes auxquelles s'adressent nos remerciements les méritent vraiment.

Nous tenons à remercier dans un premier temps, toutes les personnes qui ont contribué au succès de notre PFA et qui nous ont aidés lors de la réalisation de notre projet.

Tout d'abord, nous adressent nos remerciements À notre Encadrant Mr Nouredine Assad Nous le remercions pour sa gentillesse et la spontanéité avec lesquelles vous avez bien voulu diriger ce travail. Nous avons eu le grand plaisir de travailler sous sa direction, et avons trouvé auprès de lui le conseiller et le guide qui nous a reçus en toute circonstance avec sympathie.

Veillez, cher Encadrant trouver dans ce modeste travail l'expression de notre haute considération, de notre sincère reconnaissance et de notre profond respect.

Enfin, Nous souhaitons ensuite adresser nos remerciements au corps professoral de l'ensaj, et au corps administratif de l'école nationale des sciences appliquée d'El-Jadida.

Table des matières

1	Introduction Générale	12
2	Introduction à la Machine Learning	14
	Introduction	14
2.1	Présentation de l'apprentissage automatique (ML)	14
2.2	Type de l'apprentissage automatique	15
2.2.1	Apprentissage supervisé	15
2.2.2	Apprentissage non supervisé	16
2.2.3	Apprentissage par renforcement	17
2.3	Les algorithmes et les modèles d'apprentissage automatique	17
2.3.1	K-nearest Neighbors algorithm (k plus proches voisins)	17
2.3.2	PCA (Principal Component Analysis)	19
2.3.3	SVD (Singular Value Decomposition)	20
2.3.4	K-Means algorithm	20
2.4	Applications de l'apprentissage automatique	21
2.5	Dans le cas de système de recommandation	22
	Conclusion	24
3	Etat de l'art sur les systèmes de recommandation	25
	Introduction	25
3.1	Définition et utilisation des systèmes de recommandation	25
3.1.1	Définition de SR	25
3.1.2	Utilisation des systèmes de recommandation	26
3.2	Type des systèmes de recommandations	28
3.2.1	Le filtrage basé sur le contenu	28
3.2.2	Le filtrage collaboratif	30
3.2.3	Système de recommandation hybride	33
3.2.4	Les avantages et les inconvénients des différents systèmes de recommandations	34
3.3	Les étapes de construction d'un système de recommandation	35
3.4	évaluation du système de recommandation	40
3.4.1	Évaluation basée sur des métriques	40

3.4.2	Évaluation basée sur les utilisateurs.....	41
3.4.3	Application.....	42
	Conclusion.....	44
4	Conception du projet.....	45
	Introduction	45
4.1	Présentation du projet.....	45
4.1.1	Conception du système	45
4.2	Les outils et l'architecture du projet	47
4.2.1	Langage de programmation	47
4.2.2	Front-End.....	48
4.2.3	Back-End.....	49
4.2.4	Base de données	50
4.2.5	Le modèle d'apprentissage automatique	50
4.2.6	Ensembles de données (Dataset)	51
4.2.7	Architecture du projet	52
	Conclusion.....	53
5	Réalisation du projet	54
	Introduction	54
5.1	Structure globale	54
5.2	Explication du code.....	55
5.2.1	Frontend.....	55
5.2.2	Backend.....	55
	Récupération des films	56
	Model d'apprentissage automatique.....	56
	Architecture Django « MVT »	60
5.3	Les interfaces d'applications	63
	Conclusion.....	71
6	Conclusion générale et perspectives	72
	Bibliographie.....	74

Liste des figures

Figure 1: <i>Processus d'apprentissage automatique [5]</i>	15
Figure 2 : <i>Différent algorithme d'apprentissage automatique.</i>	17
Figure 3 : <i>représentation d'algorithme KNN [8]</i>	18
Figure 4 : <i>KNN avec $K=3$</i>	18
Figure 5 : <i>l'algorithme PCA [10]</i>	19
Figure 6 : <i>K-Mean algorithm [13]</i>	21
Figure 7 : <i>Factorisation matricielle des données de classement des films</i>	23
Figure 8 : <i>Processus de Système de recommandation</i>	26
Figure 9 : <i>les Types de système de recommandation</i>	28
Figure 10 : <i>filtrage basé sur le contenu [16]</i>	29
Figure 11 : <i>le filtrage collaboratif [17]</i>	30
Figure 12 : <i>le filtrage collaboratif basé sur les utilisateurs [18]</i>	31
Figure 13 : <i>filtrage collaboratif basé sur les éléments [18]</i>	32
Figure 14 : <i>système hybride [19]</i>	33
Figure 15 : <i>la matrice utilisateur-article (user-item)</i>	36
Figure 16 : <i>matrice de similarité</i>	37
Figure 17 : <i>classement des éléments</i>	38
Figure 18 : <i>schéma récapitulative [24]</i>	39
Figure 19 : <i>top-10 recommandation en utilisant KNNBasic</i>	39
Figure 20 : <i>train/test Split</i>	42
Figure 21 : <i>résultat de comparaison</i>	44
Figure 22 : <i>Diagramme de cas d'utilisation d'une plateforme de SR</i>	46
Figure 23 : <i>statistiques dans le temps des offres d'emploi sur Indeed.com en ML et science des données [29]</i>	47
Figure 24 : <i>Icon Bootstrap [31]</i>	48

Figure 25 : <i>Icon JavaScript</i>	49
Figure 26 : <i>le Framework Django</i>	49
Figure 27 : <i>logo de postgresQL</i>	50
Figure 28 : <i>architecture du projet</i>	52
Figure 29 : <i>structure globale</i>	54
Figure 30 : <i>structure de backend</i>	55
Figure 31 : <i>fichier movies.csv du dataset movielens</i>	57
Figure 32 : <i>fichier ratings.csv du dataset movielens</i>	57
Figure 33 : <i>Architecture MVT du framework django</i>	61
Figure 34 : <i>Interface d'enregistrement</i>	63
Figure 35 : <i>Interface d'authentification</i>	64
Figure 36 : <i>Interface d'accueil</i>	64
Figure 37 : <i>sliders à la fin de la page d'accueil</i>	65
Figure 38 : <i>souscription pour l'envoi des nouveautés</i>	65
Figure 39 : <i>interface de recommandation</i>	65
Figure 40 ; <i>liste des catégories des films</i>	66
Figure 41 : <i>table contenant l'ensemble des films</i>	66
Figure 42: <i>table des utilisateurs Django</i>	67
Figure 43: <i>l'interface d'admin Django,</i>	67
Figure 44 : <i>interface d'accueil d'admin Django</i>	68
Figure 45 : <i>liste des films de l'interface d'admin</i>	68
Figure 46 : <i>Ajouter un films d'après l'interface d'admin</i>	69
Figure 47 : <i>liste des utilisateurs</i>	69
Figure 48 <i>Ajout d'un nouveau utilisateur</i>	70
Figure 49 : <i>création d'un nouveau groupe avec ces privilèges</i>	70

Liste des tableaux

Tableau 1 : <i>avantage et inconvénient des différents types de SR [20][21][22]</i>	34
--	----

Liste des codes

Code 1 : <i>comparaison entre les algorithmes</i>	43
code 2 : <i>fichier home.html comme exemple des fichiers des front-end</i>	55
Code 3 : <i>recuperation des films a partir de BD IMDB</i>	56
Code 4 : <i>chargement de données</i>	58
Code 5 : <i>le fichier content_based.py</i>	58
Code 6 : <i>importation de vecteur TF-IDF</i>	59
Code 7 : <i>application de TF-IDF au genre de film</i>	59
Code 8 : <i>calcul de similarité</i>	59
Code 9 : <i>les fonctions dans views.py permettant de générer les recommandations</i>	60
Code 10 : <i>classe movie dans le fichier models</i>	62
Code 11 : <i>Liaison avec la base de données</i>	63

Liste des équations

Équation 1 : <i>matrice de factorisation</i>	20
Équation 2 : <i>la formule pour calculer le poids TF-IDF [28]</i>	29
Équation 3 : <i>cousine similarité</i>	36
Équation 4 : <i>distance euclidienne [23]</i>	36
Équation 5 : <i>Corrélation de Pearson [23]</i>	37
Équation 6 : <i>Erreur absolue moyenne [25]</i>	40
Équation 7 : <i>racine d'erreur quadratique moyenne [25]</i>	41

Liste des abréviations

SR	Système de recommandation
ML	Machine Learning
KNN	K-Nearest Neighbors
PCA	Principal Component Analysis
SVD	Singular Value Decomposition
SGD	Stochastic Gradient Descent
ALS	Alternating Least Square
RBM	restricted Boltzmann machine
TF-IDF	term frequency–inverse document frequency
UB-CF	User-Based Collaboratif feltering
IB-CF	Item-Based Collaboratif feltering
HR	Hite Rate
API	Application Programming Interfaces
CBF	Content-Based Filetering

1

Introduction Générale

Une tendance commune qui peut être observée dans les magasins de brique et de mortier, est que nous voyons des vendeurs qui nous guident et nous recommandent des produits pertinents lors de leurs achats, d'autre part, avec les plateformes de vente en ligne, il y a des millions de produits différents disponibles, et nous devons naviguer nous-mêmes pour trouver le bon produit. La situation est que les utilisateurs ont trop d'options et de choix disponibles, mais ils n'aiment pas consacrer beaucoup de temps à parcourir l'ensemble du catalogue d'articles. Par conséquent, le rôle des systèmes de recommandation (SR) devient critique pour recommander des articles pertinents et stimuler la conversion des clients.

Au cours des dernières décennies, avec l'essor de Youtube, Amazon, Netflix et de nombreux autres services Web de ce type, les systèmes de recommandation ont pris de plus en plus de place dans nos vies. Du e-commerce (suggérer aux acheteurs des articles qui pourraient les intéresser) à la publicité en ligne (suggérer aux utilisateurs le bon contenu, correspondant à leurs préférences), les systèmes de recommandation sont aujourd'hui incontournables dans nos déplacements quotidiens en ligne.

De manière très générale, les systèmes de recommandation sont des algorithmes visant à suggérer des éléments pertinents aux utilisateurs (les éléments étant des films à regarder, du texte à lire, des produits à acheter ou autre selon les secteurs d'activité). Ils sont vraiment essentiels dans certaines industries car ils peuvent générer une énorme quantité de revenus lorsqu'ils sont efficaces ou aussi être un moyen de se démarquer de manière significative des concurrents. Pour prouver l'importance des systèmes de recommandation, nous pouvons mentionner qu'il y a quelques années, Netflix a organisé un défi (le «prix Netflix») où l'objectif était de produire un système de recommandation qui fonctionne mieux que son propre algorithme avec un prix de 1 million de dollars à gagner [2] .

Le but de ce projet est d'abord d'étudier l'apprentissage automatique (ML) et les façons dont il peut être utilisé dans le cas d'un système de recommandation de films. Ensuite, **l'objectif** est de mettre en œuvre l'un de ces solutions dans un projet d'application web pour des recommandations de films.

Le projet consiste à développer une application web entièrement fonctionnelle qui permet de recommander des films à regarder en se basant sur le contenu des films tels que le genre et l'année de sortie.

Notre rapport se compose de 5 chapitres :

- Le 1er chapitre " Généralités sur le web", présente une définition générale sur l'apprentissage automatique et les algorithmes utilisés dans les systèmes de recommandation.

- Dans le 2ème chapitre « Etat de l’art sur les systèmes de recommandations », nous allons présenter les types d’un système de recommandation et nous allons voir comment construire un système de recommandation et comment l’évaluer.
- Dans le 3ème chapitre "Conception du projet", nous allons détailler la conception de notre projet par la représentation des différentes phases pour le développement, et nous allons présenter les outils de programmation utilisés pour la réalisation de notre projet.
- Le 4ème chapitre "Réalisation du projet", présente l’implémentation de notre application web basé sur le contenu.
- Le 5ème chapitre "Conclusion générale et perspective", présente une synthèse globale sur le projet et les améliorations qui peuvent être déployés pour aller plus loin.

2

Introduction à la Machine Learning

Introduction

Depuis quelques années, le monde est fou de Machine Learning. Les ordinateurs sont devenus assez puissant pour exécuter des algorithmes d'apprentissage automatique partout et la portée des possibilités sont vaste et illimitées. Le Machine Learning peut être utilisé partout pour tout améliorer. Santé, trading financier, marketing, moteurs de recherche, banque; tout le monde peut en profiter.

Dans ce chapitre, nous présenterons le monde de ML et ces domaines d'applications. Ensuite, nous traiterons ces différents algorithmes et modèles dans le cas d'un moteur de recommandation de contenu (qui est dans notre cas les films).

2.1 Présentation de l'apprentissage automatique (ML)

L'apprentissage automatique (ML) est une branche de l'intelligence artificielle permettant aux ordinateurs d'apprendre sans avoir été programmés explicitement à cet effet [3]. Le but du ML est de créer un système qui réponde des questions. Ce système est appelé **un modèle** et ce modèle est construit par un processus appelé entraînement. Ceci est la phase d'apprentissage. Cette phase se fait en alimentant le modèle avec **des données**. Plus nous avons de données, mieux le modèle répondra aux questions. Le modèle peut être continuellement amélioré en l'alimentant avec plus de données.

Le but principal de ML est de construire des algorithmes qui peuvent obtenir **des données d'entrée** et utiliser ces données pour **prédire** une sortie tout en mettant à jour les sorties à mesure que de nouvelles données arrivent [4].

Essentiellement, il peut être résumé en 3 étapes:

1. Il faut des données.
2. Trouve le modèle à partir des données.
3. Prédit un nouveau modèle à partir des données.

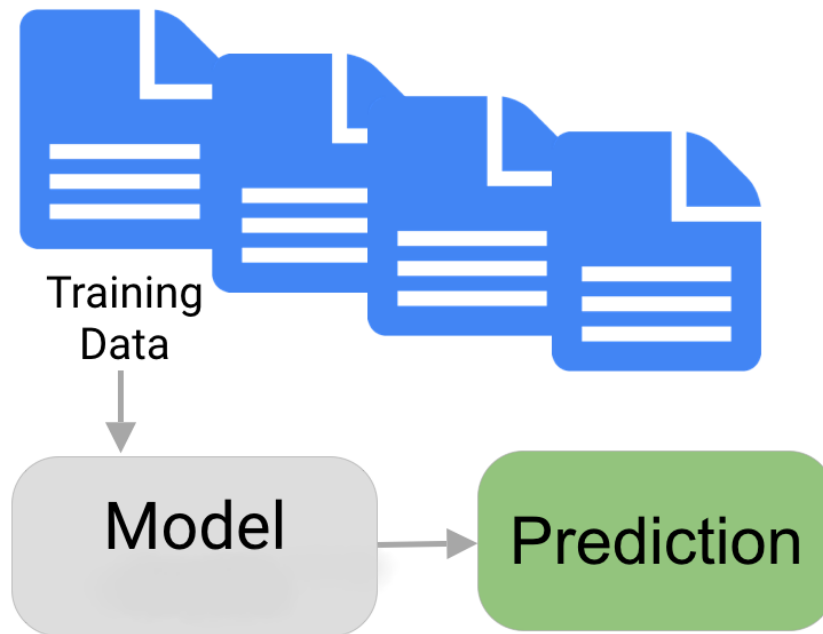


Figure 1: *Processus d'apprentissage automatique [5]*

L'apprentissage automatique diffère de l'informatique traditionnelle car il ne nécessite pas d'être explicitement programmé. Dans l'informatique traditionnelle, nous résolvons les problèmes en expliquant algorithmes quel est le problème et en codant comment le résoudre. Dans l'apprentissage automatique, nous former des modèles en leur fournissant une grande quantité de données, puis leur demander de donner des sorties pour des intrants spécifiques donnés, sur la base de cet apprentissage.

2.2 Type de l'apprentissage automatique

Il existe trois grandes catégories d'apprentissage automatique: l'apprentissage supervisé, l'apprentissage non supervisé et l'apprentissage par renforcement.

2.2.1 Apprentissage supervisé

Les données d'entrée sont étiquetées. L'apprentissage supervisé établit un processus d'apprentissage, compare les résultats prévus aux résultats réels des «données d'apprentissage» (c.-à-d. Les données d'entrée) et ajuste en continu le modèle prédictif jusqu'à ce que les résultats prédits du modèle atteignent la précision attendue, comme **la classification** et **problèmes de régression**. Les algorithmes courants incluent les arbres de décision, la classification bayésienne, la régression des moindres carrés, la régression logistique, les machines à vecteurs de support, les réseaux de neurones, etc. [6].

Exemple de problème de classification

Par exemple, nous voulons savoir pour une image donnée si elle représente un requin ou un océan. Nous besoin de nourrir un modèle avec une grande quantité d'images représentant des requins ou des océans. Chaque l'image est étiquetée comme un requin ou comme un océan. Le système apprendra alors à quoi ressemble un requin comme et à quoi ressemble un océan. À

la fin, nous pouvons donner à la machine une nouvelle image sans étiquette et demandez-lui s'il s'agit d'un requin ou d'un océan [7].

Exemple de problème de régression

Par exemple, nous voulons connaître le prix d'une maison. Nous devons nourrir un modèle avec une grande quantité de données sur les maisons. Chaque élément de maison dans le jeu de données a deux étiquettes qui sont mentionné explicitement: prix et taille. Le système apprendra alors combien coûte une maison selon sa taille. À la fin, nous pouvons donner à la machine une nouvelle maison avec sa taille mais sans son prix et demandez combien cela coûte [8].

2.2.2 Apprentissage non supervisé

Les données d'entrée n'ont pas de balises, mais des algorithmes pour déduire les liens intrinsèques des données, tels que l'apprentissage des règles de **clustering** et **d'association**. L'algorithme de clustering essaie de diviser l'ensemble de données en groupes selon la similitude, comme différentes espèces de plantes. Un algorithme d'association essaie de diviser l'ensemble de données en groupes des éléments qui se produisent fréquemment ensemble, tels que «les personnes qui achètent X ont tendance à acheter Y». Les algorithmes courants incluent l'analyse de composants indépendants, les algorithmes K-Means et Apriori [6].

Exemple d'algorithme d'association

Par exemple, nous voulons savoir pour quels articles un client peut être intéressé. Nous devons le faire alimenter un modèle avec une grande quantité de données sur les articles achetés par les clients. On raconte alors le système pour trouver des modèles cachés dans les données ou un moyen de représenter et de classer les clients en fonction de leurs achats. À la fin, nous pouvons demander à la machine quels articles pourrait intéresser un client spécifique et la machine pourra répondre en fonction des liste des articles achetés par le client et sur des comparaisons avec d'autres clients [7].

Exemple d'algorithme de clustering

Par exemple, nous voulons classer différentes fleurs en fonction de leur espèce. Nous devons le faire nourrir un modèle avec une grande quantité de données sur les fleurs et leurs caractéristiques (longueur des sépales, largeur sépale, longueur des pétales, largeur des pétales,...). On dit alors au système de regrouper les données en groupes en trouvant un modèle dans l'ensemble de données. À la fin, nous pouvons demander à la machine sont les différents groupes et quelle fleur appartient à quel groupe.

Saisissez les données en tant que rétroaction dans le modèle, en insistant sur la façon d'agir en fonction de l'environnement pour maximiser les avantages escomptés. La différence entre l'apprentissage supervisé est qu'il ne nécessite pas les bonnes paires d'entrée / sortie et ne nécessite pas de correction précise du comportement sous-optimal. L'apprentissage par renforcement est davantage axé sur la planification en ligne et nécessite un équilibre entre l'exploration (dans l'inconnu) et la conformité (connaissances existantes) [6].

Il existe beaucoup d'algorithmes dans le domaine ML mais on va traiter seulement X algorithmes les plus utilisés qui appartiennent aux types présentés dans la section précédente. Le but de cette section est avoir une compréhension générale des techniques utilisées. On ne va pas traiter tous les algorithmes existants parce que ce n'est pas le but de ce projet.



L'algorithme k-voisins les plus proches (KNN) est un algorithme d'apprentissage automatique supervisé simple et facile à implémenter qui peut être utilisé pour résoudre des problèmes de classification et de régression. KNN utilisé dans une variété d'applications telles que la finance, les soins de santé, les sciences politiques, la détection d'écriture manuscrite, la reconnaissance d'image et la reconnaissance vidéo et aussi dans les systèmes de recommandation. Il est basé sur l'approche de **similarité** des fonctionnalités.

Comment fonctionne l'algorithme KNN?

Dans KNN, K est le nombre de voisins les plus proches. Le nombre de voisins est le principal facteur décisif.

La figure ci-dessous montre les données représentées dans l'espace par la fonction X sur l'axe X et par la fonction Y sur l'axe Y. Supposons maintenant que nous ayons pour classer un nouveau point (le point jaune), le modèle calculera la distance des K points les plus proches (où K est un paramètre à choisir) et étiqueté le nouveau point comme un membre de la classe ayant le plus grand nombre de voisins. Dans la figure ci-dessous, le nouveau point sera classé comme membre de la classe A parce que nous mettons le paramètre K à 1. [8]

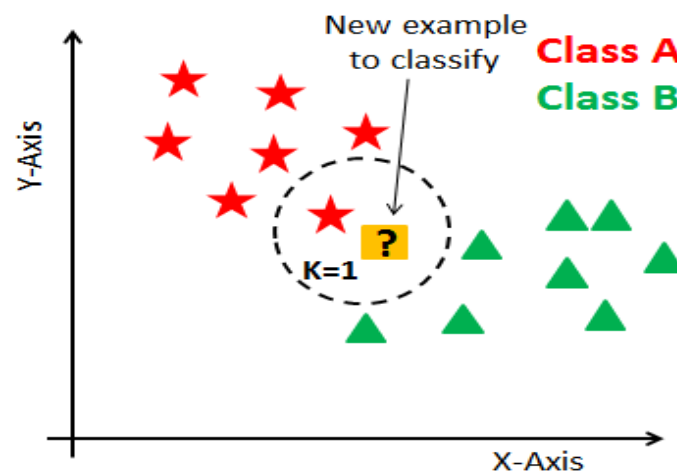


Figure 3 : représentation d'algorithme KNN [8]

Dans la figure ci-dessous, nous fixons le paramètre K à 3. Le point jaune sera alors classé en tant que membre de la classe B.

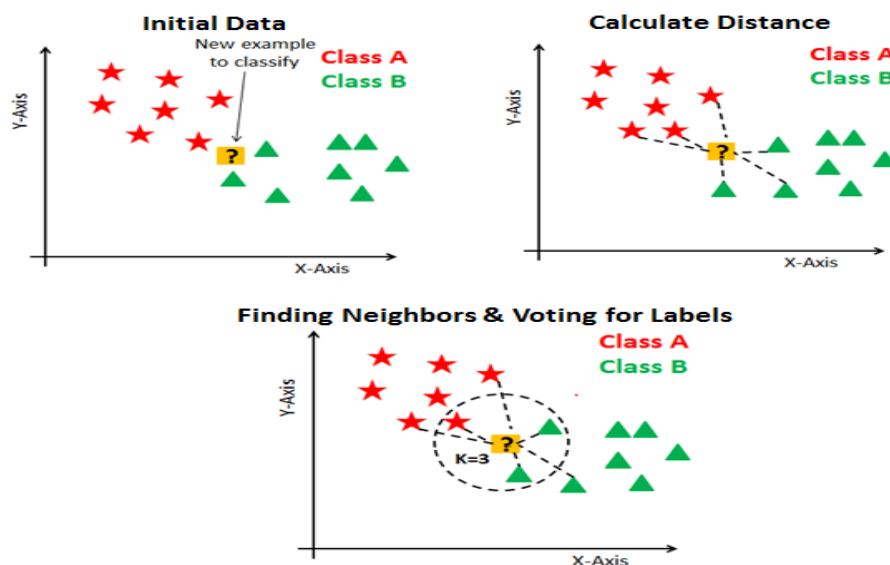


Figure 4 : KNN avec $K=3$

Pour un petit résumé, KNN est les étapes suivantes :

1. Calculer la distance.
2. Trouvez les voisins les plus proches.
3. Votez pour les labels.

Il existe différentes façons pour calculer les distances entre les points, comme la distance euclidienne, distance Hamming, distance Manhattan, similitude cosinus et la distance de Minkowski.

2.3.2 PCA (Principal Component Analysis)

L'analyse en composantes principales (ACP) est une technique statistique non paramétrique non supervisée principalement utilisée pour réduire la dimensionnalité dans l'apprentissage automatique. [9]

Elle est utilisée pour rendre les données faciles à explorer et à visualiser en réduisant le nombre de variables. Cela se fait en capturant la variance maximale des données dans un nouveau système de coordonnées avec des axes appelés «composants principaux».

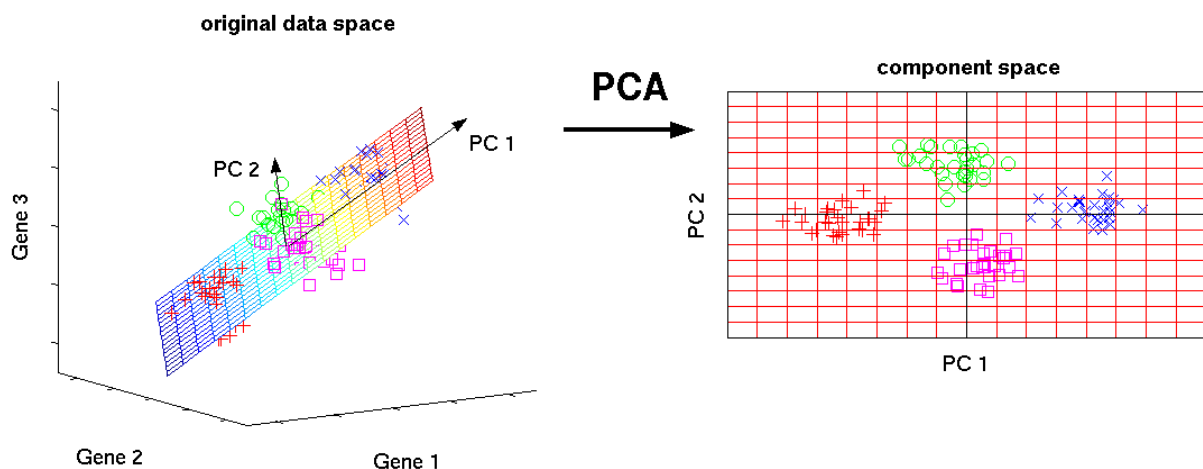


Figure 5 : l'algorithme PCA [10]

Les axes ou nouvelles variables sont appelés composantes principales (PC) et sont classés par variance: la première composante, PC 1, représente la direction de la variance la plus élevée des données. La direction de la deuxième composante, PC 2, représente la plus grande de la variance restante orthogonale à la première composante. Cela peut naturellement être étendu pour obtenir le nombre requis de composantes qui couvrent ensemble un espace de composantes couvrant la quantité de variance souhaitée. [10]

Une dimensionnalité élevée signifie que l'ensemble de données possède un grand nombre de fonctionnalités. Le problème principal associé à la haute dimensionnalité dans le domaine de l'apprentissage automatique est le sur ajustement de modèle, ce qui réduit la capacité de généraliser au-delà des exemples de l'ensemble de formation [9]. L'utilisation d'un nombre inférieur de composants principaux au lieu des données d'origine de grande dimension est une étape de prétraitement courante qui améliore souvent les résultats des analyses ultérieures telles que la classification.

2.3.3 SVD (Singular Value Decomposition)

La décomposition en valeurs singulières (SVD) est l'un des algorithmes d'apprentissage non supervisé les plus largement utilisés, qui est au centre de nombreux systèmes de recommandation et de réduction de dimensionnalité qui sont au cœur de sociétés mondiales telles que Google, Netflix, Facebook, Youtube et autres. [11]

En termes simples, SVD est la factorisation d'une matrice en 3 matrices. Donc, si nous avons une matrice A , alors sa SVD est représentée par:

$$A = U\sigma V^T$$

Équation 1 : *matrice de factorisation*

Où A est une matrice $m \times n$, U est une matrice orthogonale ($m \times m$), σ est une matrice diagonale rectangulaire non négative ($m \times n$) et V est une matrice orthogonale ($n \times n$). [11]

Quelle est la différence entre SVD et PCA? SVD vous offre les neuf mètres de diagonalisation d'une matrice en matrices spéciales faciles à manipuler et à analyser. Il jette les bases pour démêler les données en composants indépendants. PCA ignore les composants moins importants. De toute évidence, nous pouvons utiliser SVD pour trouver PCA en tronquant les vecteurs de base moins importants dans la matrice SVD d'origine. [12]

2.3.4 K-Means algorithm

L'algorithme k-means est un algorithme de clustering **non supervisé**. Il prend un tas de points sans étiquette et essaie de les regrouper en nombre «k» de grappes.

Comment ça fonctionne?

ÉTAPE 1 : Choisissez le nombre K de grappes, nous supposons K de notre choix. Il peut s'agir de n'importe quel nombre (3,5 ou 2, etc.).

ÉTAPE 2 : Sélectionnez au hasard K points, les centroïdes (pas nécessairement à partir de votre ensemble de données) - Centroid est le point central du cluster. Vous pouvez sélectionner deux points dans ce premier nuage de points, ils peuvent être des valeurs

aléatoires X et Y à condition que votre sélectionné le nombre de centroïdes équivaut au nombre de grappes sélectionnées à l'étape 1.

ÉTAPE 3 : Attribuez chaque point de données au centroïde le plus proche> Cela forme le cluster K. Et cela fait par le calcul des distances de chaque point de données aux centroïdes et réaffectez chaque point au centroïde de cluster le plus proche en fonction de la distance minimale

ÉTAPE 4 : Calculez et placez le nouveau centroïde de chaque cluster.

ÉTAPE 5 : Réaffectez chaque point de données au nouveau centroïde le plus proche. Si une réaffectation a eu lieu, passez à l'ÉTAPE 4, sinon arrêtez-la. Il s'agit clairement d'un processus itératif des étapes 4 et 5. [13]

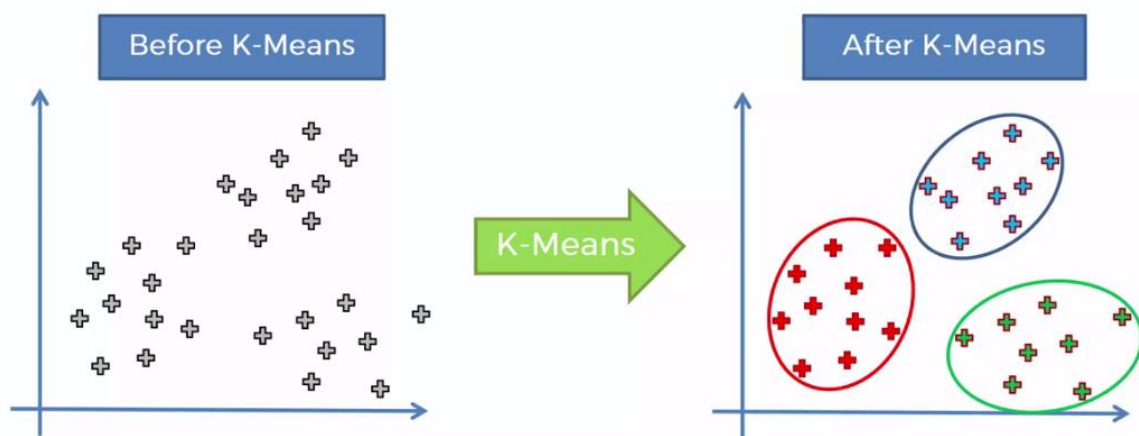


Figure 6 : *K-Mean algorithm* [13]

2.4 Applications de l'apprentissage automatique

ML est un mot à la mode dans le monde de la technologie en ce moment car il représente une étape majeure avancé dans la façon dont un ordinateur peut apprendre. Les possibilités sont grandes et ML peut être utilisé dans de nombreuses situations différentes. Dans cette section, nous allons parcourir les applications ML concrètes les plus connues.

Dans **les moteur de recherche Web**, ML est utilisé pour améliorer la précision des moteurs de recherche. Précision des moteurs de recherche Google utilise ML pour améliorer les résultats du moteur de recherche en analysant les premiers liens cliqués ou ceux esquivés en fonction de l'entrée de la recherche. Si vous avez esquivés la recherche le programme peut tirer des leçons de cette erreur pour obtenir un meilleur résultat à l'avenir. [14]

Dans **le soin de santé**, les algorithmes d'apprentissage automatique sont utilisés pour traiter les informations et repérer modèles pour diagnostiquer et surveiller les patients hospitalisés. Il peut également être utilisé pour comprendre les risques facteurs de maladie dans de grandes populations. [14]

Dans **le trading**, l'apprentissage automatique est utilisé pour prédire ce que le marché boursiers fera un jour donné. Le ML peut être utile pour prédire à tout moment les changements sur le marché de l'année en analysant une grande quantité de données et en exécutant des transactions à des vitesses élevées et élevées le volume. Il peut remplacer les analystes humains avec plus d'efficacité. [14]

Dans **le marketing**, Les algorithmes d'apprentissage automatique sont utilisés pour mieux cibler les clients et générer des e-mails, coupons ou offres personnalisés. Les modèles ML sont alimentés par les métadonnées définir les clients. [14]

Dans les **systèmes d'achat**, des algorithmes d'apprentissage automatique sont utilisés pour *recommander* des produits aux clients. Netflix ou Amazon recommandent des films à regarder ou des articles à acheter à leurs clients en analysant leur activité et en la comparant à d'autres clients. [14]

Dans les **systèmes bancaires**, les algorithmes d'apprentissage automatique sont utilisés pour détecter les fraudes transactions. Paypal a développé un système capable de distinguer entre légitime et transactions frauduleuses entre clients. [14]

2.5 Dans le cas de système de recommandation

Dans cette section, on va voir une vue générale sur les différents solutions et algorithmes qui peuvent appliquer dans le cas de système de recommandation des films. Mais dans le chapitre suivant on va bien détailler et clarifier les définitions et types des systèmes de recommandation.

Un système de recommandation c'est tout simplement un système a pour objectif de fournir à un utilisateur des ressources pertinentes en fonction de ses préférences et ses données historiques.

Parmi les plus populaires algorithmes qui sont utilisés dans la construction des systèmes de recommandation sont : le KNN, SVD (SVD++), SGD, ALS et finalement RBM.

Nous avons vu dans la section précédente que le **KNN** est un algorithme d'apprentissage automatique qui peut être utilisé pour résoudre des problèmes de classification et de régression. Dans notre cas (SR), nous avons utilisés cet algorithme pour trouver des grappes **d'utilisateurs similaires (ou des éléments similaires)** en se basant sur des notes courantes a des films déjà vu par les utilisateurs et faisons des prédictions en utilisant la note moyenne des k premiers voisins les plus proches.

Nous avons aussi vu que **SVD** est un algorithme de ML qui factorise une matrice à 3 matrices. Dans le contexte de système de recommandation, il existe une façon de prédire les notes des éléments (des films) qui s'appelle Matrix Factorization ou en français la factorisation matricielle. Cette dernière c'est une classe d'algorithme de filtrage collaboratif consiste à prédire les notes des utilisateurs en se basant sur leurs historiques. Dans notre cas la matrice A (voir équation 1) est la matrice User-item (utilisateur-élément) et la matrice U représente la matrice User et finalement la matrice V représente la matrice Item. Pour bien comprendre voir l'exemple suivant :

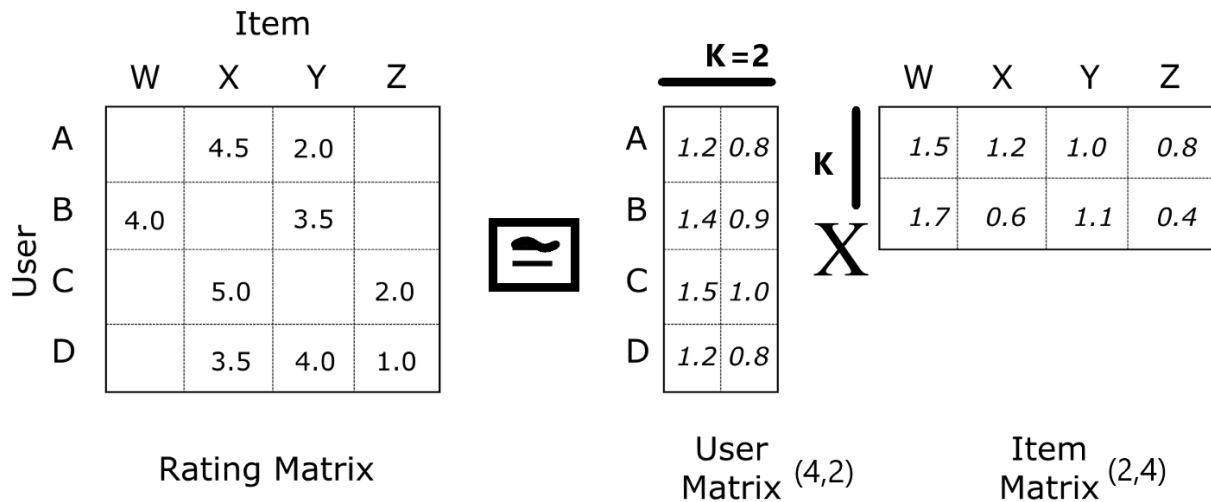


Figure 7 : Factorisation matricielle des données de classement des films

Maintenant notre objectif est de remplir les cases vides par des meilleures notes de film personnalisées pour les utilisateurs. Pour remplir cette matrice nous avons besoin de trouver les deux matrices U et V. mais la question qui se pose ici c'est comment on peut obtenir ces deux matrices ? La repense est simple, en utilisant **SVD** et **PCA** (pour redimensionner la matrice initial user-item en deux matrices de petit taille) on peut ces deux matrices avec la matrice Sigma qui est juste une simple matrice diagonal qui ne sert qu'à mettre les valeurs à l'échelle appropriés. Mais le problème ici c'est que si nous avons **beaucoup des notes manquante** sur la matrice user-item, nous ne pouvons pas utiliser l'algorithme SVD, car SVD ne fonctionne pas lorsque des données sont manquantes ou lorsque la matrice est **très clairsemée** (very sparse). Donc ici viennent des approches plus récentes qui se concentrent sur la minimisation des erreurs quadratiques telles que **SGD** (Stochastic Gradient Descent) ou en français la descente de gradient stochastique et **ALS** (Alternating Least Square) qui est fournie par Apach Spark. Donc en utilisant SGD qui minimise les erreurs en se basant sur les **notes connus** dans la matrice d'origine (user-item matrix), on peut prédire les données manquantes dans cette matrice et par la suite faire des recommandations en se basant sur ces résultats.

Conclusion

Nous avons vu dans ce chapitre une petite introduction à la Machine Learning, et l'application de cette dernière dans différents domaines compris aussi les systèmes de recommandation, et nous avons traité quelques algorithmes qui sont utilisés comme des solutions pour notre cas. Dans le chapitre suivant nous présenterons la façon d'implémenter et d'évaluer un système de recommandation en appliquant l'un de ces algorithmes présentés dans ce chapitre.

3

Etat de l'art sur les systèmes de recommandation

Introduction

Les systèmes de recommandation sont utilisés partout aujourd'hui sur Internet. Netflix, Amazon, Youtube ou Pinterest s'appuient sur ces systèmes pour personnaliser des recommandations à leurs utilisateurs. Ils les utilisent pour filtrer des millions de contenus et trouver les plus appropriés pour leurs clients. Les systèmes de recommandation sont bien étudiés et offrent de grandes valeurs aux entreprises.

Dans ce chapitre nous présentons les types des systèmes de recommandation, et comment nous pouvons les implémentés en se basant sur des algorithmes de l'apprentissage automatique (ML). Ensuite, nous présentons la façon d'évaluer les performances d'un système de recommandation en se basant sur des métriques de précision (Accuracy metrics) comme MAE et RMSE.

3.1 Définition et utilisation des systèmes de recommandation

3.1.1 Définition de SR

Selon Wikipédia, Les **systèmes de recommandation** sont une forme spécifique de filtrage de l'information (SI) visant à présenter les éléments d'information (films, musique, livres, news, images, pages Web, etc) qui sont susceptibles d'intéresser l'utilisateur. Généralement, un système de recommandation permet de comparer le profil d'un utilisateur à certaines caractéristiques de référence, et cherche à prédire l'« avis » que donnerait un utilisateur. [15] Donc un système de recommandation est un système a pour objectif de prédire l'intérêt des utilisateurs et de recommander des produits qui sont très probablement intéressants pour eux.

Un système de recommandation est comme tous les modèles d'apprentissage automatique avec lequel vous commencez avec certain **donnée** que vous passez dans un **modèle** et ce dernier génèrent **des prédictions** (voir fig.1). Ainsi dans notre scenario nos données sont les **préférences des utilisateurs**, cela peut être soit **implicitement** avec les cliques des utilisateurs sur des produits, ou par obtention d'une liste d'éléments que l'utilisateur a écoutés ou regardés. Soit **explicitement** avec les évaluations des utilisateurs sur un article ou avec une demande de choisir le meilleur objet entre deux objets différents. Par la suite, en introduisant

ces **préférences** dans notre **système de recommandation**, et en essayant de **prédire** le comportement futur des utilisateurs. Voici un schéma explicatif de notre processus de SR :

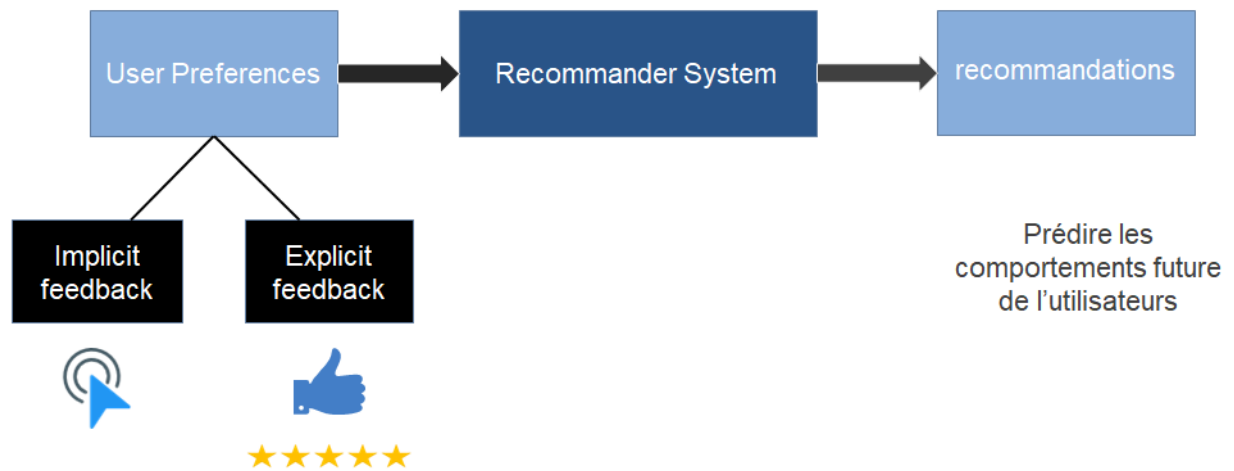


Figure 8 : *Processus de Système de recommandation*

3.1.2 Utilisation des systèmes de recommandation

Les systèmes de recommandation sont principalement utilisés pour suggérer automatiquement le bon contenu ou le bon produit aux bons utilisateurs de manière personnalisée afin d'améliorer l'expérience globale. Les systèmes de recommandation sont vraiment puissants en termes d'utilisation d'énormes quantités de données et d'apprentissage pour comprendre les préférences d'utilisateurs spécifiques. Les recommandations aident les utilisateurs à naviguer facilement à travers des millions de produits ou des tonnes de contenu (articles / vidéos / films) et leur montrent le bon article / information qu'ils pourraient aimer ou acheter. Donc, en termes simples, SR aide à découvrir des informations au nom des utilisateurs. Maintenant, cela dépend des utilisateurs de décider si SR a bien fait les recommandations ou non, et ils peuvent choisir de sélectionner le produit / contenu ou de les jeter et de continuer. Chacune des décisions des utilisateurs (positives ou négatives) aide à recycler le SR sur les dernières données pour pouvoir donner des recommandations encore meilleures.

Les systèmes de recommandation peuvent être utilisés à des fins multiples dans le sens de recommander diverses choses aux utilisateurs. Par exemple, certains d'entre eux pourraient appartenir aux catégories ci-dessous :

- Produits de detail
- Travaux
- Connexion/amis
- Films / musique / Vidéos / Livres / Articles
- Les publicités

La partie « Que recommander ? » dépend totalement du contexte dans lequel SR est utilisé et peut aider l'entreprise à augmenter ses revenus en fournissant les articles les plus probables que les utilisateurs peuvent acheter ou en augmentant l'engagement en présentant le contenu pertinent au bon moment. RS prend soin de l'aspect critique selon lequel le produit ou le contenu recommandé doit être quelque chose que les utilisateurs pourraient aimer mais qu'ils n'auraient pas découvert par eux-mêmes. Parallèlement à cela, RS a également besoin d'un élément de recommandations variées pour le garder suffisamment intéressant. Quelques exemples d'utilisation intensive de RS par les entreprises aujourd'hui comme les produits Amazon, les suggestions d'amis de Facebook, les « gens que vous connaissez peut-être » de LinkedIn, le film de Netflix, les vidéos de YouTube, la musique de Spotify et les cours de Coursera.

L'impact de ces recommandations se révèle immense d'un point de vue commercial et, par conséquent, plus de temps est consacré à rendre ces SR plus efficaces et pertinentes. Certains des avantages immédiats qu'offre RS dans les commerces de détail sont les suivants :

- Augmentation des revenus.
- Commentaires et évaluations positifs des utilisateurs.
- Engagement accru.

Pour les autres secteurs verticaux tels que les recommandations d'annonces et d'autres recommandations de contenu, RS aide énormément à les aider à trouver la bonne chose pour les utilisateurs et augmente ainsi l'adoption et les abonnements. Sans SR, recommander du contenu en ligne à des millions d'utilisateurs de manière personnalisée ou offrir un contenu générique à chaque utilisateur peut être incroyablement hors cible et entraîner des impacts négatifs sur les utilisateurs.

On peut résumer l'utilisation de systèmes de recommandation en ceux-ci :

Valeur pour le client

- Trouver des choses intéressantes
- Limiter l'ensemble des choix
- Aider à explorer l'espace des options
- Découvrir de nouvelles choses
- Divertissement

Valeur pour le fournisseur

- service personnalisé supplémentaire et probablement unique pour le client
- Augmenter la confiance et la fidélité des clients
- Augmenter les ventes, les taux de clics, la conversion, etc.

3.2 Type des systèmes de recommandations

Dans cette section, nous allons présenter les deux principaux types des systèmes de recommandation: le filtrage collaboratif et le filtrage basé sur le contenu. Les paragraphes suivantes décriront ensuite diverses méthodes de filtrage collaborative telles que la factorisation utilisateur-utilisateur, article-article et matricielle. Et le dernier paragraphe sera consacré aux méthodes basées sur le contenu et à leur fonctionnement.

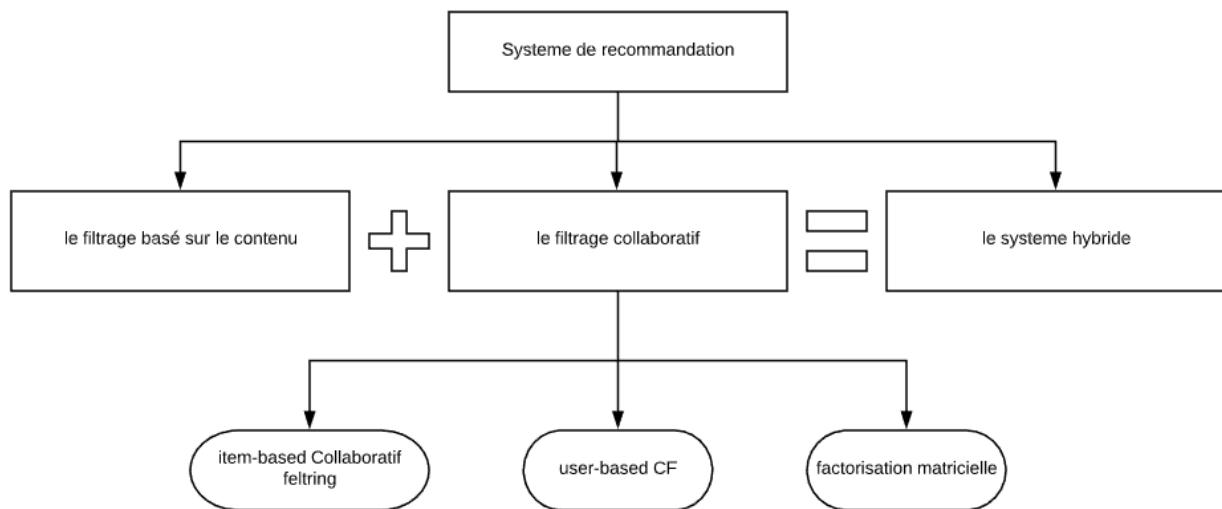


Figure 9 : les Types de système de recommandation

3.2.1 Le filtrage basé sur le contenu

Dans les cette méthode utilise des attributs du contenu pour recommander un contenu similaire. Il n'a pas de problème de démarrage à froid (voir Tableau.1 dans la dernière section) car il fonctionne via des attributs ou des balises (Tags) du contenu, tels que des acteurs, des genres ou des réalisateurs, de sorte que de nouveaux films peuvent être recommandés immédiatement. [27] Ce type nécessite une quantité d'information sur **les fonctionnalités et les caractéristiques** des éléments ou des utilisateurs.

Par exemple, si un utilisateur aime des films tels que «Mission Impossible», nous pouvons lui recommander les films de «Tom Cruise» ou des films du genre «Action». Donc dans ce filtrage, deux types de données sont utilisés. Tout d'abord, les goûts de l'utilisateur, l'intérêt de l'utilisateur, les informations personnelles de l'utilisateur telles que l'âge ou, parfois, l'historique de l'utilisateur aussi. Ces données sont représentées par le vecteur utilisateur. Deuxièmement, les informations relatives au produit sont connues sous le nom de vecteur d'article.

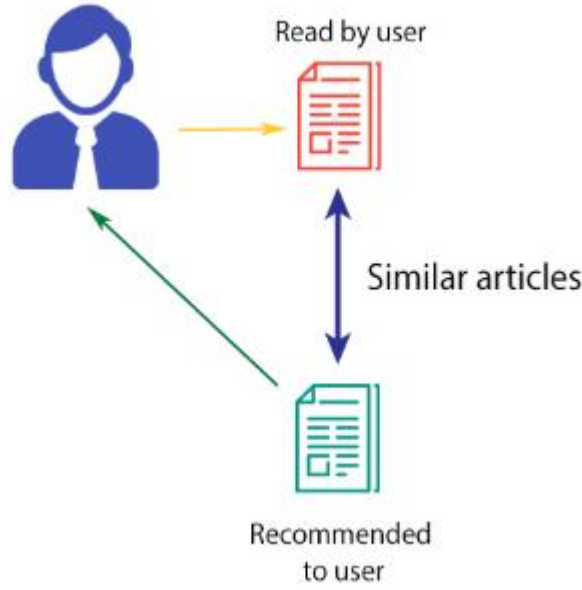


Figure 10 : *filtrage basé sur le contenu [16]*

Il existe différentes manières de calculer les caractéristiques d'un élément et de calculer similitude entre les éléments. Le plus courant est de **vectorisé les éléments** et après calculer la similitude entre les vecteurs.

Un bon moyen de vectorisé un élément est le modèle d'espace vectoriel. Ce modèle extrait des mots clés à partir d'un élément et vectorisé cet élément par **TF-IDF**. Term frequency-inverse document frequency (TF-IDF) est un algorithme destinée à refléter l'importance d'un mot pour un document dans une collection ou un corpus. Plus le terme apparaît fréquemment, plus son poids est important.

$$w_{i,j} = tf_{i,j} \times \log \left(\frac{N}{df_i} \right)$$

Équation 2 : *la formule pour calculer le poids TF-IDF [28]*

$tf_{i,j}$ = nombre d'occurrences de i dans j .

df_i = nombre de documents contenant i .

N = nombre total de documents.

Pour calculer la similitude entre deux vecteurs, une méthode courante consiste à calculer le cosinus similarité (voir équation 3).

3.2.2 Le filtrage collaboratif

Le filtrage collaboratif pour les systèmes de recommandation basé uniquement sur les interactions passées enregistrées entre les utilisateurs et les éléments afin de produire de nouvelles recommandations. Ces interactions sont stockées dans la «matrice d'interactions utilisateur-article». En localisant les utilisateurs / éléments homologues avec un historique de notation similaire à l'utilisateur ou à l'élément actuel, ils génèrent des recommandations à l'aide de ces interaction. [17]

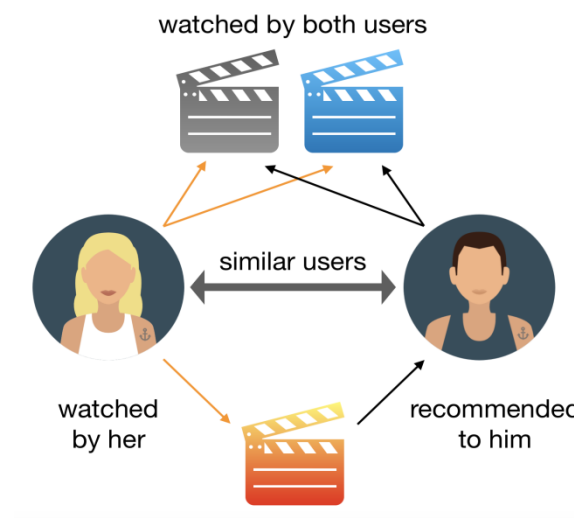


Figure 11 : le filtrage collaboratif [17]

Ce filtrage est décomposé aussi de 3 types : User-Based Collaboratif filtering (UB-CF), Item-based Collaboratif Filtering (IB-CF) et Factorisation matricielle.

Filtrage collaboratif basé sur l'utilisateur (UB-CF)

Le système UB-CF trouve des utilisateurs similaires à un utilisateur donné en fonction de leurs goûts (les éléments qu'ils acheté ou bien noté) et recommander à l'utilisateur donné les articles qu'il n'a pas déjà acheté ou évaluer, mais les autres utilisateurs similaires l'ont fait. Par exemple, l'utilisateur A et l'utilisateur B ont vu les mêmes films et les ont évalués de manière identique mais l'utilisateur A a vu et aimé Titanic et l'utilisateur B non. Le système suppose que cet utilisateur A et B sont similaires et il recommandera Titanic à l'utilisateur B.

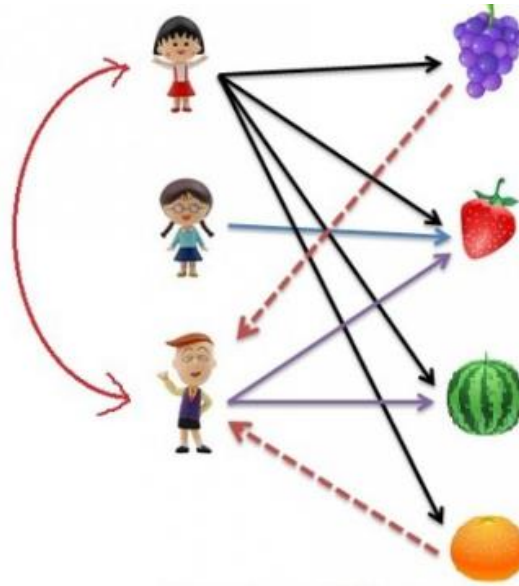


Figure 12 : le filtrage collaboratif basé sur les utilisateurs [18]

Donc le système UB-CF utilise la méthode de similarité entre les utilisateurs pour recommander des éléments. Nous avons déjà vu dans le chapitre précédent un algorithme qui fait ce travail ; c'est l'algorithme **KNN**. L'algorithme KNN trouve les k voisins les plus proches d'un utilisateur donné (similaire) et lui recommande des films basés sur ces voisins similaires. Dans la section de construction de système de recommandation nous allons bien détailler ces étapes.

Filtrage collaboratif basé sur les éléments (IB-CF)

Au contraire d'UB-CF ce système est basé sur la similarité entre les éléments. Il recommande aux utilisateurs des articles similaires à ceux qu'ils ont déjà achetés ou bien notés. Contrairement à *un système basé sur le contenu*, il **n'analyse pas les caractéristiques des éléments** pour définir la similitude entre eux, il se base donc sur la similitude entre les éléments par interactions des autres utilisateurs avec les articles. Par exemple, les utilisateurs A, B et C ont noté 5/5 Spiderman, 5/5 Catwoman et 1/5 Gran Torino. Le système considère que Spiderman et Catwoman sont similaires et Gran Torino est différent d'eux. Enfin, si l'utilisateur D a noté 5/5 Spiderman mais n'a pas regardé Catwoman, le système lui recommande Catwoman car il suppose qu'il devrait l'aimer, en fonction du comportement des autres utilisateurs.

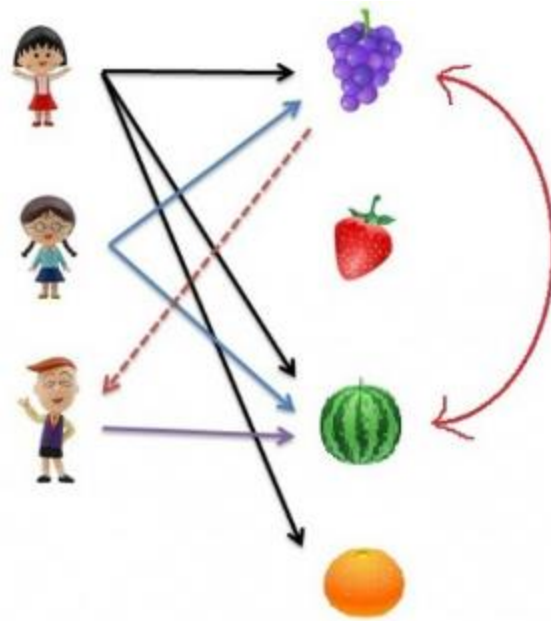


Figure 13 : *filtrage collaboratif basé sur les éléments [18]*

Comme pour les recommandations UB-CF, un bon modèle pour mettre en œuvre un système IB-CF est le Modèle KNN. Il utilise la même matrice que l'UB-CF mais inversé c'est à dire le transposé de la matrice (Item-User matrice au lieu de matrice user-item). Nous verrons plus loin dans la partie suivante comment cela fonctionne concrètement et comment il peut être mis en œuvre.

Factorisation Matricielle

L'hypothèse principale derrière la factorisation matricielle est qu'il existe un espace latent de caractéristiques assez faible dans lequel nous pouvons représenter à la fois les utilisateurs et les éléments et de telle sorte que l'interaction entre un utilisateur et un élément peut être obtenue en calculant le produit scalaire des vecteurs denses correspondants dans cet espace. [2]

Nous avons déjà vu dans le chapitre précédent l'équation de Matrice de factorisation (equation1) et aussi la façon d'appliquer des algorithmes pour recommander un article aux utilisateurs. En bref, cette méthode consiste à trouver les deux matrices (user matrix et item matrix) pour reconstruire la matrice initiale (user-item) en remplissant les cases vides par les notes prédire. Parmi les bons algorithmes qui font ce travail sont SVD, ALS et SGD.

3.2.3 Système de recommandation hybride

[Des recherches récentes montrent](#) que la combinaison de recommandations collaboratives et basées sur le contenu peut être plus efficace. Les approches hybrides peuvent être mises en œuvre en faisant des prédictions basées sur le contenu et basées sur la collaboration séparément puis en les combinant. En outre, en ajoutant des capacités basées sur le contenu à une approche basée sur la collaboration et vice versa; ou en unifiant les approches en un seul modèle. [19]

Netflix est un bon exemple de l'utilisation de systèmes de recommandation hybrides. Le site Web formule des recommandations en comparant les habitudes de visionnage et de recherche d'utilisateurs similaires (c.-à-d. Le filtrage collaboratif) ainsi qu'en offrant des films qui partagent des caractéristiques avec des films qu'un utilisateur a hautement évalués (filtrage basé sur le contenu).

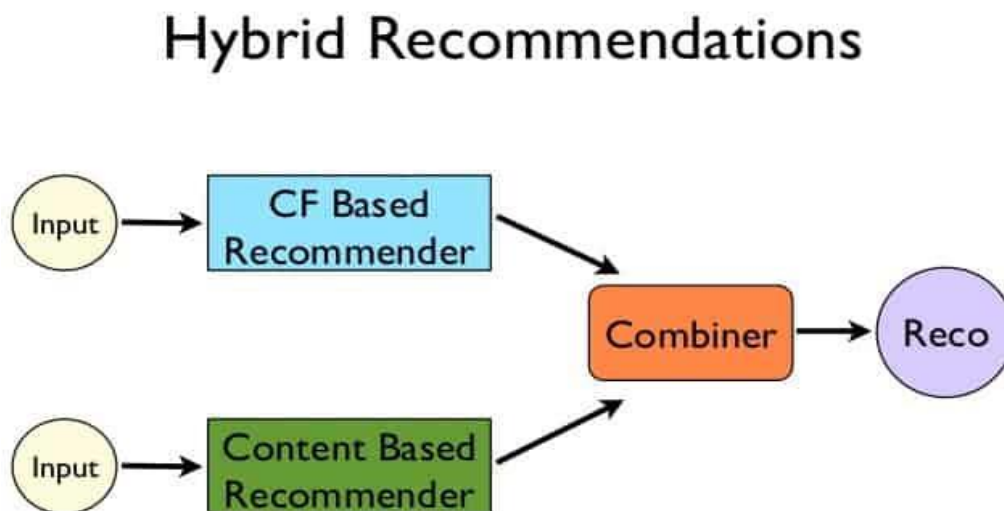


Figure 14 : *système hybride [19]*

3.2.4 Les avantages et les inconvénients des différents systèmes de recommandations

	Avantages	Inconvénients
Le filtrage collaboratif	<p>Nous n'avons pas besoin de connaissances de domaine car les plongements sont automatiquement appris.</p> <p>Comparé à d'autres techniques, telles que celles basées sur le contenu, il est plus précis.</p> <p>De nouveaux produits peuvent être présentés à l'utilisateur si un utilisateur a acheté ou évalué.</p>	<p>Clarté: Le pourcentage de personnes qui évaluent les articles est vraiment faible.</p> <p>Démarrage à froid: les nouveaux utilisateurs n'auront pas ou peu d'informations à leur sujet pour être comparés à d'autres utilisateurs.</p> <p>Le nouvel article ne peut pas être recommandé si aucun utilisateur ne l'a acheté ou évalué.</p>
Le filtrage basé sur le contenu	<p>Le modèle n'a pas besoin de données sur les autres utilisateurs, car les recommandations sont spécifiques à cet utilisateur. Cela facilite la mise à l'échelle d'un grand nombre d'utilisateurs.</p> <p>De nouveaux articles peuvent être recommandés; seules les données de cet élément sont requises.</p>	<p>Si la matrice utilisateur ou la matrice d'élément est modifiée, la matrice de similitude cosinus doit être à nouveau calculée.</p> <p>L'activité ne peut pas être développée car l'utilisateur n'essaye pas un type de produit différent.</p> <p>L'utilisateur ne sera jamais recommandé pour différents éléments.</p>

Tableau 1 : avantage et inconvénient des différents types de SR [20][21][22]

3.3 Les étapes de construction d'un système de recommandation

Construire un système de recommandation efficace et robuste peut être relativement simple si vous suivez les étapes de base pour passer des données brutes à une prédiction. Cela étant dit, il y a certaines particularités à prendre en compte en ce qui concerne les systèmes de recommandation qui sont souvent négligés et qui, pour le processus le plus efficace et les meilleures prévisions, méritent d'être introduites (ou réitérées).

Cette section passe en revue les cinq étapes fondamentales pour construire un système de recommandation :

Étape 1 : Préparation des données

1. Collecte de données

Il s'agit de la première étape et la plus cruciale pour la construction d'un moteur de recommandation. Les données peuvent être collectées par deux moyens: **explicites et implicites**. Les données explicites sont des informations fournies intentionnellement, c'est-à-dire des informations fournies par les utilisateurs, telles que des classements de films. Les données implicites sont des informations qui ne sont pas fournies intentionnellement mais recueillies à partir de flux de données disponibles comme l'historique de recherche, les clics, l'historique des commandes, etc.

2. Stockage des données

La quantité de données dicte la qualité des recommandations du modèle. Par exemple, dans un système de recommandation de films, plus les utilisateurs attribuent de notes aux films, meilleures sont les recommandations pour les autres utilisateurs. Le type de données joue un rôle important dans la décision du type de stockage à utiliser. Ce type de stockage peut inclure une base de données SQL standard, une base de données NoSQL ou une sorte de stockage d'objets.

Étape 2 : Choix et application d'algorithme

Maintenant que nous allons préparer nos données, il est le temps de choisir un algorithme de filtrage (soit le filtrage basé sur le contenu ou le filtrage collaboratif) pour prédire le comportement futur de l'utilisateur et recommander des choses satisfaisantes.

Par exemple, on va choisir le filtrage collaboratif basé sur l'utilisateur (IB-CF) en utilisant l'algorithme KNN. Tout d'abord on va commencer par **la transformation des données sous forme d'une matrice user-item** (figure 15), chaque ligne contiendrait les notes attribuées par un utilisateur et chaque colonne contiendrait les notes reçues par un élément. Donc les cellules dans cette matrice représentent les notes données par les utilisateurs à des films.

	Indiana Jones	Star Wars	Empire strikes Back	Incredibles	Casablanca
Bob	4	5			
Ted					1
Ann		5	5	5	

Figure 15 : la matrice utilisateur-article (user-item)

Dans la plupart des cas, les cellules de la matrice sont vides, car les utilisateurs ne notent que quelques éléments. Il est très peu probable que chaque utilisateur évalue ou réagisse à chaque élément disponible. Une matrice avec des cellules principalement vides est appelée **clairsemée**, et ça parmi les inconvénients de CF.

Ensuite, on va calculer la **matrice de similitude utilisateur-utilisateur** (user-user similarity matrix) à l'aide d'une méthode de **cosinus similarité** (équation 2) en anglais Cosine Similarity pour trouver les top-n utilisateurs similaire à un utilisateur donnée.

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_i A_i B_i}{\sqrt{\sum_i A_i^2} \sqrt{\sum_i B_i^2}}$$

Équation 3 : cossine similarité

La similitude est le cosinus de l'angle entre les 2 vecteurs des utilisateurs(ou des éléments) vecteurs de A et B.

Plus les vecteurs sont proches, plus l'angle sera petit et le cosinus plus grand

Il existe d'autre méthodes qui pouvant être utilisées pour calculer la similitude sont les suivantes:

- **Distance euclidienne** : des éléments similaires se trouveront à proximité les uns des autres s'ils sont tracés dans un espace à n dimensions. Ainsi, nous pouvons calculer la distance entre les articles et en fonction de cette distance, recommander des articles à l'utilisateur. La formule de la distance euclidienne est donnée par: [23]

$$DE = \sqrt{(x_1 - y_1)^2 + \dots + (x_n - y_n)^2}$$

Équation 4 : distance euclidienne [23]

- **Corrélation de Pearson** : Elle nous indique combien deux éléments sont corrélés. Plus la corrélation est élevée, plus la similitude sera grande. La corrélation de Pearson peut être calculée à l'aide de la formule suivante: [23]

$$sim(u, v) = \frac{\sum (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum (r_{ui} - \bar{r}_u)^2} \sqrt{\sum (r_{vi} - \bar{r}_v)^2}}$$

Équation 5 : *Corrélation de Pearson [23]*

r_{ui} = la note donnée par l'utilisateur i à l'article u .

r_{vi} = la note donnée par l'utilisateur i à l'article v .

\bar{r}_u = la note moyenne donnée par tous les utilisateurs à l'article u .

\bar{r}_v = la note moyenne donnée par tous les utilisateurs à l'article v .

u, v = les articles (ou les utilisateurs).

Mais la méthode la plus utilisé c'est la similitude de cosinus.

Maintenant que nous avons calculé la similitude entre les vecteurs, on peut construire la matrice utilisateur-utilisateur (figure 16), puis trier les utilisateurs et récupérer les top-n similaire à l'utilisateur en question :

	Bob	Ted	Ann
Bob	1	0	1
Ted	0	1	0
Ann	1	0	1

Figure 16 : *matrice de similarité*

Dans cet exemple, disons que nous voulons faire une recommandation à Bob maintenant, nous observons qu'Ann est similaire avec Bob donc nous nous retrouvons avec Ann en haut de la liste trier (top-n utilisateurs similaires), et nous éliminerions probablement Ted en appliquant un seuil de score de similarité minimum à ce stade. Nous sautons également le cas de la comparaison de Bob avec lui-même. Donc Ann est le seul utilisateur similaire à Bob. Mais s'il y avait d'autres utilisateurs que nous avons trouvés, nous jetterions également leurs notes dans cette pile.

Etape 3 : Classement des candidats (candidate scoring/rating)

Maintenant que nous avons trouvé les top-n utilisateurs similaire que Bob (ici c'est seulement Ann mais dans le cas réel on peut trouver beaucoup d'utilisateur), nous devons déterminer les meilleurs recommandations à présenter à Bob, parce que nous voulons recommander des choses que les utilisateurs similaires aimaient, pas des choses que les utilisateurs similaires détestaient.

Donc nous devons noter les notes données par utilisateurs similaires d'une manière **normalisée**. Ainsi, un élément pourrait être une sorte de score de notation normalisé ici. Nous traduisons une note de cinq étoiles à 1,0 juste pour garder tout sur la même échelle. Nous devrions également tenir compte de la similitude avec l'utilisateur qui a généré ces notes, alors nous pouvons peut-être multiplier ce score de 1,0 par le score de similitude d'Ann avec Bob, qui est également 1,0. Peut-être qu'au lieu de normaliser les scores de notation sur une échelle de zéro à un, vous pouvez réellement attribuer des scores négatifs à des choses notées une ou deux étoiles pour les pousser activement vers le bas dans les résultats finaux.

Il existe de nombreuses façons de noter les candidats aux recommandations, et il n'y a pas de méthode standard pour le faire. C'est un autre cas où nous devons expérimenter pour voir ce qui fonctionne le mieux pour les données dont vous disposez.



Figure 17 : *classement des éléments*

Etape 4 : filtrage des candidats (éléments)

Après avoir classé les éléments par ordre décroissant, nous devons les filtrer afin d'extraire les top-n recommandations.

Cette étape consiste à filtrer les éléments que Bob a déjà évalués, car il est inutile de recommander des films qu'il a déjà vus. Donc nous devons éliminer Star Wars et recommander The Empire Strikes Back.

Etape 5 : liste de top-n recommandation

Après le filtrage de résultats, nous obtenons une liste de top-n recommandation. Qui est dans notre cas le film The Empire Strikes Back.

Récapitulation :

Pour récapituler, les étapes impliquées dans le filtrage collaboratif basé sur les utilisateurs sont les suivantes: Commencez par créer une table de recherche des utilisateurs pour tous les

éléments qu'ils ont notés et ces valeurs de notation. Construisez ensuite une autre matrice 2D de scores de similitude entre chaque paire d'utilisateurs. Ensuite, générez des candidats à la recommandation en regroupant tous les éléments évalués par des utilisateurs similaires. Puis, évaluons tous ces candidats en examinant comment les utilisateurs similaires les ont évalués, dans quelle mesure les utilisateurs les ont évalués, et classez les. Enfin, nous filtrons les éléments que l'utilisateur a déjà vus et nous avons terminé.

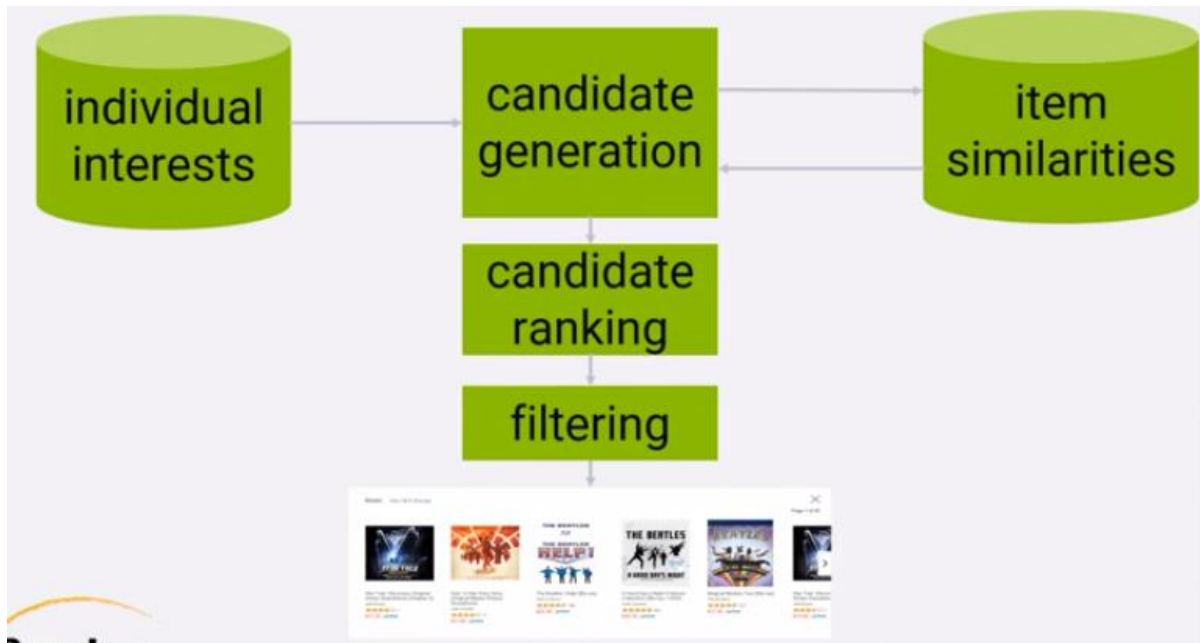


Figure 18 : schéma récapitulative [24]

Exemple de résultat de code :

Voici un exemple de résultat de code d'implémentation d'algorithme KNNBasic sur le cas d'UB-CF :

```
In [1]: runfile('C:/Users/DELL/Desktop/PFA/Formation_SysRec/Realisation_RecSys/RecSys-
Materials/CollaborativeFiltering/SimpleUserCF.py', wdir='C:/Users/DELL/Desktop/PFA/
Formation_SysRec/Realisation_RecSys/RecSys-Materials/CollaborativeFiltering')
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Gladiator (2000) 4.6
Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981) 4.5
Forrest Gump (1994) 4.1000000000000005
Shawshank Redemption, The (1994) 4.1000000000000005
Matrix, The (1999) 3.6
Pulp Fiction (1994) 3.5999999999999996
Seven (a.k.a. Se7en) (1995) 3.5
Groundhog Day (1993) 3.5
Silence of the Lambs, The (1991) 3.3000000000000003
Saving Private Ryan (1998) 3.3
Inception (2010) 3.3
```

Figure 19 : top-10 recommandation en utilisant KNNBasic

3.4 évaluation du système de recommandation

Après avoir construit notre SR, nous avons besoin d'évaluer ces performances de recommandation afin de décider quel algorithme convient le mieux à notre situation. Les méthodes d'évaluation des systèmes de recommandation peuvent principalement être divisées en deux ensembles: les méthodes **hors ligne** et les méthodes **en ligne**. **Les méthodes hors ligne** ne nécessitent pas de vrais utilisateurs, mais une partie des données est utilisée pour former l'algorithme, tandis qu'un autre échantillon est utilisé pour tester les prédictions concernant les goûts des utilisateurs, tout simplement l'évaluation hors ligne est une évaluation basée sur des paramètres bien définis (les métriques de précision ou en anglais accuracy metrics). **Les méthodes en ligne** impliquent d'émettre des recommandations, puis d'interroger les utilisateurs sur la façon dont ils évaluent les articles, comme le A /B Teste en ligne (online A/B test).

Avant de commencer les explications des méthodes, nous voudrions insister sur quelque chose. **L'utilisation d'une seule mesure d'erreur peut nous donner un aperçu limité du fonctionnement de ces systèmes.** Nous devons toujours essayer d'évaluer avec différentes méthodes nos modèles. **La meilleure façon pour évaluer un SR c'est de faire un test en ligne comme le A /B Test.**

3.4.1 Évaluation basée sur des métriques

Les systèmes de recommandation sont des modèles prédictifs avec des algorithmes qui, en général, cherchent à minimiser l'erreur d'une fonction. Il est donc important de mesurer l'erreur de prédiction qu'ils ont en comparant les résultats attendus avec ceux que le modèle donne en sortie. Ici nous présentons deux mesures les plus courantes :

- **MAE (Mean Absolute Error)**

L'erreur absolue moyenne est la moyenne de la différence entre les valeurs d'origine et les valeurs prédites. Il nous donne la mesure de la distance entre les prévisions et la sortie réelle. Cependant, ils ne nous donnent aucune idée de la direction de l'erreur, c'est-à-dire si nous sommes sous-prédits ou sur-prédits. Mathématiquement, il est représenté comme:[25]

$$MAE = \frac{\sum_{i=1}^n |r_{ui} - \bar{r}_{ui}|}{n}$$

Équation 6 : Erreur absolue moyenne [25]

- **RMSE (Root Mean Square Erreur)**

Racine d'erreur quadratique moyenne (RMSE) est assez similaire à l'erreur absolue moyenne, la seule différence étant que RMSE prend la moyenne du **carré** de la différence entre les valeurs d'origine et les valeurs prédites et fait la racine carrée de l'ensemble. L'avantage d'utiliser RMSE sur MAE est qu'il pénalise davantage le terme lorsque l'erreur est élevée. (Notez que RMSE est toujours supérieur à MAE). Il est présenté comme ça :

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (r_{ui} - \bar{r}_{ui})^2}{n}}$$

Équation 7 : racine d'erreur quadratique moyenne [25]

Plus ces deux mesures sont petites, plus le système de recommandation est meilleur.

Il existe d'autres métriques qui sont utiles dans l'évaluation d'un SR mais ne sont pas populaires comme MAE et RMSE. Ces métriques sont Taux de réussite (HR=Hit Rate), Taux de réussite réciproque moyen (ARHR), Taux de réussite cumulé (cHR), Taux de réussite au classement (rHR), la diversité et la nouveauté (novelty). [26]

3.4.2 Évaluation basée sur les utilisateurs

Nous pouvons remarquer qu'il est difficile d'évaluer la qualité d'une recommandation qui n'appartient pas à l'ensemble de données de test: comment savoir si une nouvelle recommandation est pertinente avant de la recommander à notre utilisateur? Pour toutes ces raisons, il peut parfois être tentant de tester le modèle en «**conditions réelles**». L'objectif du système de recommandation étant de générer une action (regarder un film, acheter un produit, lire un article etc...), nous pouvons en effet évaluer sa capacité à générer l'action attendue. Par exemple, le système peut être mis en production, selon une approche de test A / B, ou peut être testé uniquement sur un échantillon d'utilisateurs. De tels processus nécessitent cependant d'avoir un certain niveau de confiance dans le modèle. [2]

Récapitulation :

les systèmes de recommandation sont difficiles à évaluer: si certaines mesures classiques telles que MAE, RMSE, l'exactitude, le rappel ou la précision peuvent être utilisées, il faut garder à l'esprit que certaines propriétés souhaitées telles que la diversité (sérendipité) et l'explicabilité ne peuvent pas être évaluées de cette façon; L'évaluation des conditions réelles (comme les tests A / B ou les tests d'échantillons) est enfin le seul moyen réel d'évaluer un nouveau système de recommandation, mais nécessite une certaine confiance dans le modèle.

3.4.3 Application

Maintenant que nous avons vu les différentes méthodes d'évaluation d'un SR. Passant à appliquer les méthodes hors ligne sur l'algorithme KNN.

Les deux différentes façons de tester notre algorithme en ligne :

1. Train-Test Split

2. Validation croisée K-Fold (k-fold cross validation)

Nous allons utiliser le train/set split. Tout d'abord, on va commencer par diviser notre Data Set en deux ensembles : ensemble d'entraînement et ensemble de test. Ensuite, nous allons appliquer notre Algorithme (KNN) à l'ensemble d'entraînement, puis comparer ces valeurs avec les valeurs d'ensemble du test en mesurant l'erreur entre ces deux dernières avec RMSE ou MAE.

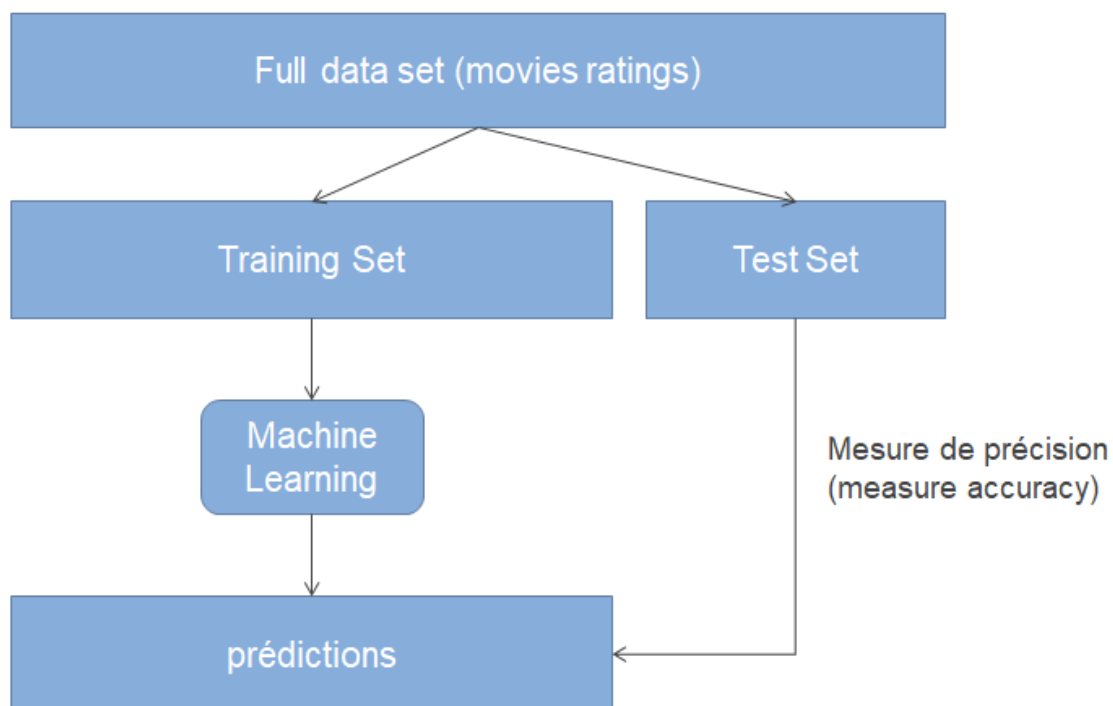


Figure 20 : *train/test Split*

Voici un code qui calcul RMSE et MAE et qui compare entre l'algorithme KNN et l'algorithme aléatoire (Random):

```
from MovieLens import MovieLens
from surprise import KNNBasic
from surprise import NormalPredictor
from Evaluator import Evaluator

import random
import numpy as np

def LoadMovieLensData():
    ml = MovieLens()
    print("Loading movie ratings...")
    data = ml.loadMovieLensLatestSmall()
    print("\nComputing movie popularity ranks so we can measure novelty later...")
    rankings = ml.getPopularityRanks()
    return (ml, data, rankings)

np.random.seed(0)
random.seed(0)

# Load up common data set for the recommender algorithms
(ml, evaluationData, rankings) = LoadMovieLensData()

# Construct an Evaluator to, you know, evaluate them
evaluator = Evaluator(evaluationData, rankings)

# User-based KNN
UserKNN = KNNBasic(sim_options = {'name': 'cosine', 'user_based': True})
evaluator.AddAlgorithm(UserKNN, "User KNN")

# Item-based KNN
ItemKNN = KNNBasic(sim_options = {'name': 'cosine', 'user_based': False})
evaluator.AddAlgorithm(ItemKNN, "Item KNN")

# Just make random recommendations
Random = NormalPredictor()
evaluator.AddAlgorithm(Random, "Random")

# Fight!
evaluator.Evaluate(False)

evaluator.SampleTopNRecs(ml)
```

Code 1 : *comparaison entre les algorithmes*

La méthode `evaluator()` consiste à évaluer notre data Set, elle utilise toutes les fonctions qui sont définies dans la classe `RecommenderMetrics.py` pour mesurer les métriques (RMSE et MAE), HR, Diversité et nouveauté (novelty). La méthode `AddAlgorithm()` consiste à charger les algorithmes pour les évaluer. Et finalement la méthode `evaluate()` consiste à comparer entre les différents algorithmes au niveau de prédiction des notes.

Algorithm	RMSE	MAE
User KNN	0.9787	0.7547
Item KNN	0.9788	0.7610
Random	1.4227	1.1375

Legend:

RMSE: Root Mean Squared Error. Lower values mean better accuracy.
MAE: Mean Absolute Error. Lower values mean better accuracy.

Figure 21 : résultat de comparaison

Nous pouvons remarquer que l'algorithme KNN est performant au niveau de prédiction des notes par rapport à l'algorithme aléatoire, parce que RMSE et MAE sont petits dans l'algorithme KNN. Et nous avons déjà vu dans la section précédente que, plus la valeur RMSE est petit, mieux les recommandations.

Conclusion

Le système de recommandation est un système puissant qui peut ajouter de la valeur à l'entreprise ou aux affaires. Il est composé de deux types principaux : le filtrage collaboratif et le filtrage basé sur le contenu. Chaque type à des avantages et des inconvénients, et la meilleure solution est de combiné les deux (système hybride).

Pour construire un système de recommandation il faut commencer par la collection et la préparation des données, ensuite le choix d'algorithme a appliqué (KNN, SVD, TF-IDF etc.), puis le filtrage des éléments que l'utilisateur a déjà vu, et finalement la génération d'une liste de top-n recommandation.

Lorsqu'on construit un système de recommandation, il est essentiel d'évaluer ses performances. Il existe deux façons pour évaluer un SR : évaluation hors ligne en se basant sur les métrique de précision (MAE, RMSE, HR etc.), et évaluation en ligne en se basant sur le A/B Test online.

Dans le chapitre suivant, nous allons présenter la conception et la structure générale de notre application web. Les outils que nous utiliserons dans l'implémentation de notre plateforme de recommandation de films.

4

Conception du projet

Introduction

Dans les chapitres précédents nous avons vu les différents méthodes et techniques pour construire et évaluer un système de recommandation. Dans ce chapitre, nous allons présenter les différentes fonctionnalités de notre application web, les outils de développement frontend et Backend utilisées, puis nous allons construire l'architecture générale de notre application supportant d'un modèle ML (de type *Content-based filetering*) qui recommande des films aux utilisateurs. Le diagramme de cas d'utilisation sera aussi élaboré afin de modéliser le système.

4.1 Présentation du projet

Notre projet s'agit d'une application web de recommandation de films, a pour objectifs :

- Fournir une interface utilisateur (UI) attrayante, réactive et conviviale.
- Mettre en œuvre un modèle de ML bien adapté afin de recommander des films pertinents utilisateurs.

4.1.1 Conception du système

Le langage de modélisation que nous avons utilisé est Unified Modeling Language (UML) pour présenter la conception de notre application, est un langage de modélisation graphique à base de pictogrammes conçu pour fournir une méthode normalisée pour visualiser, spécifier, construire et documenter la conception d'un système. Nous avons choisi de modéliser notre application par le *diagramme de cas d'utilisation*.

Un diagramme de cas d'utilisation est une représentation graphique des interactions entre les éléments d'un système

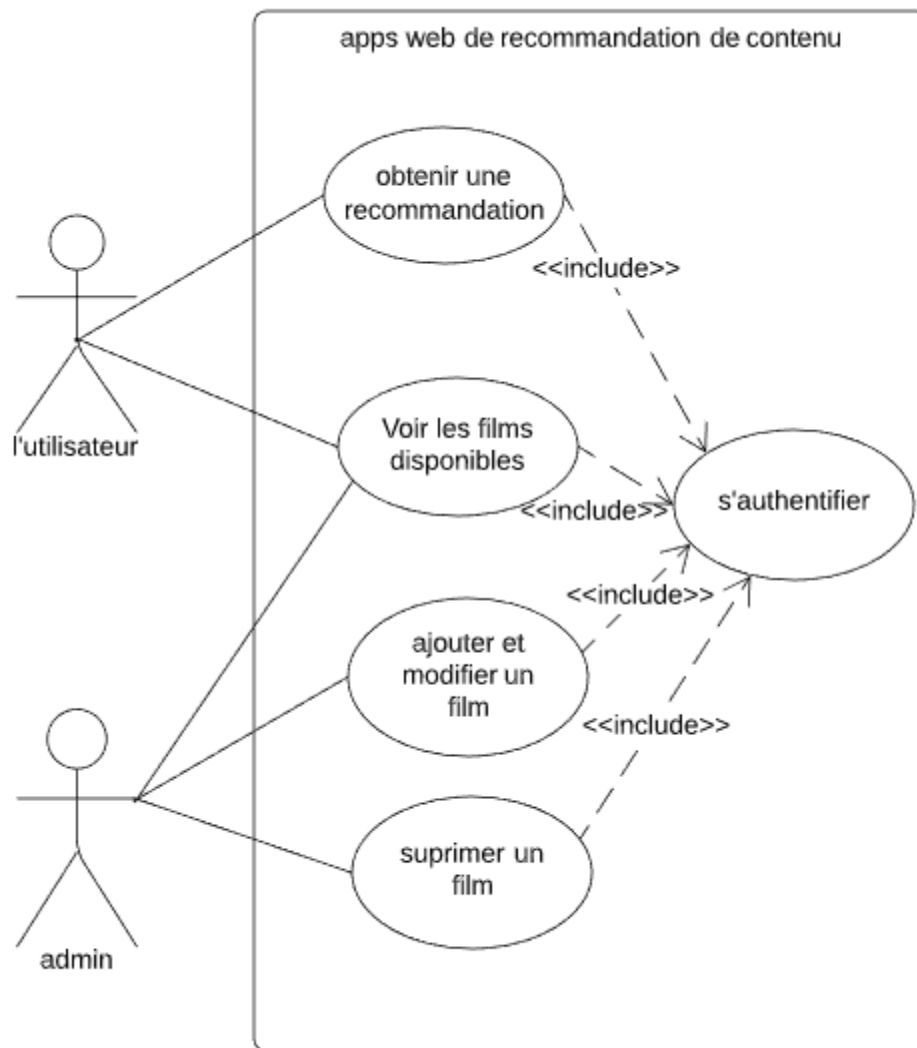


Figure 22 : Diagramme de cas d'utilisation d'une plateforme de SR

L'utilisateur peut utiliser l'application Web pour:

- Créer un compte.
- Connecter à la plateforme.
- Voir les films disponibles dans notre application.
- Obtenir des recommandations de films en fonction de sa liste de films préférés

L'admin peut utiliser l'application Web pour:

- Ajouter, modifier et supprimer des films
- Voir les films.

Pour implémenter tous ces cas d'utilisation, le projet a besoin d'un frontend, d'une authentification, une base de données, un back end, un modèle ML, un ensemble de données et plusieurs programmes d'application Interfaces (API). Ces différentes parties sont décrites dans les sections suivantes.

4.2 Les outils et l'architecture du projet

4.2.1 Langage de programmation

Nous avons déjà vu que le système de recommandation est un modèle d'apprentissage automatique (ML) c'est pour cela nous allons choisir l'un des meilleurs langages qui l'implémente.

La chose principale à considérer lors du choix d'un langage de programmation pour traiter l'apprentissage automatique est le nombre de bibliothèques disponibles et leurs objectifs.

Le langage le plus demandé et le plus utilisé en science des données et en Machine Learning est **Python**. [29]

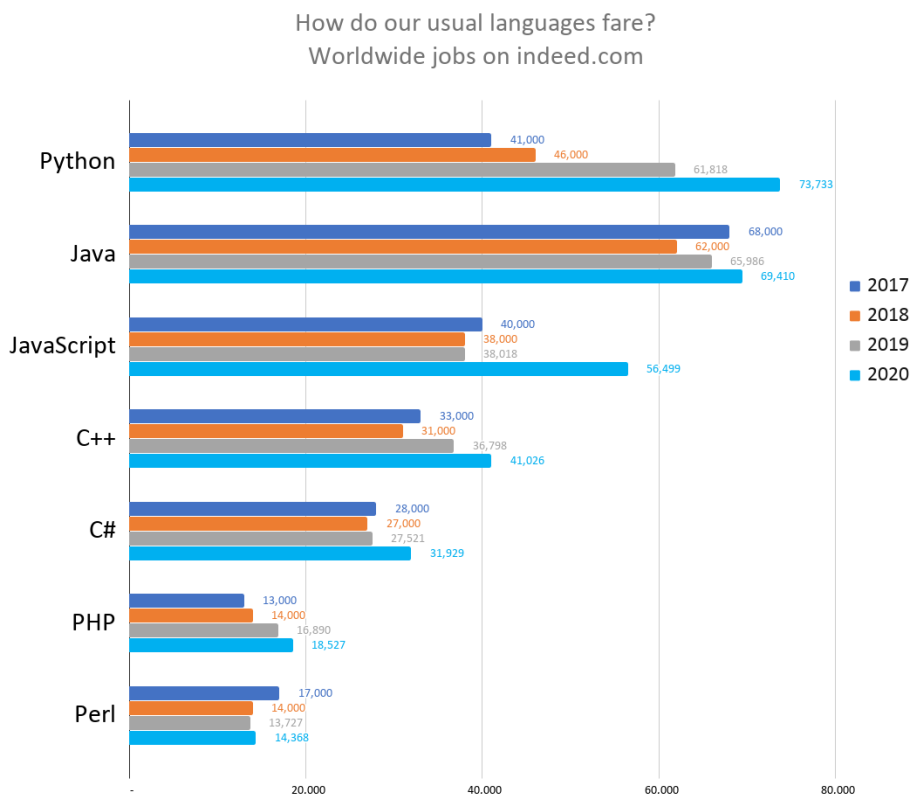


Figure 23 : statistiques dans le temps des offres d'emploi sur Indeed.com en ML et science des données [29]

La figure ci-dessus montre les occurrences dans le temps des termes sélectionnés dans les offres Indeed.com. Python est suivi de R et Java.

Python possède de nombreuses bibliothèques utiles pour traiter ML telles que NumPy, SciPy, Matplotlib et principalement scikit-learn construit sur le dessus d'eux. Il a de bonnes performances pour le prétraitement et travailler directement avec eux. Récemment le développement de bibliothèques Deep Learning comme PyTorch, Keras ou TensorFlow place Python au meilleur endroit dans le domaine des données science. [30]

Pour ces raisons nous avons choisir **Python** comme langage de programmation. Nous avons utilisé dans notre projet les bibliothèques **pandas**, **numpy**, **scikit-learn**, **imdbpy** et **pickle**.

4.2.2 Front-End

L'interface de notre application doit fournir une interface utilisateur responsive, conviviale et efficace. Pour atteindre ces objectifs, nous avons choisi de travailler avec **Bootstrap** et **JavaScript**.

Bootstrap est un Framework utiles à la création du design (graphisme, animation et interactions avec la page dans le navigateur, etc.) de sites et d'applications web. [31]



Figure 24 : *Icon Bootstrap [31]*

JavaScript est un langage de programmation de scripts principalement employé dans les pages web **interactives** mais aussi pour les serveurs avec l'utilisation (par exemple) de Node.js. C'est un langage orienté objet à prototype, c'est-à-dire que les bases du langage et ses principales interfaces sont fournies par des objets qui ne sont pas des instances de classes, mais qui sont chacun équipés de constructeurs permettant de créer leurs propriétés, et notamment une propriété de prototypage qui permet de créer des objets héritiers personnalisés. En outre, les fonctions sont des objets de première classe. [32]



Figure 25 : *Icon JavaScript*

JavaScript est le langage possédant le plus large écosystème grâce à son gestionnaire de dépendances npm, avec environ **500 000 paquets** en août 2017. [32]

4.2.3 Back-End

Dans la premier partie de cette section, Nous avons vu que Python est le meilleur langage de programmation pour mettre en œuvre le Machine Learning. Nous avons choisi d'utiliser le Framework **Django** pour construire notre backend.

Django est un Framework de développement web open source en Python. Il a pour but de rendre le développement web 2.0 simple et rapide. Il est utilisé par plusieurs sites grand public sont désormais fondés sur Django, dont Pinterest et Instagram ou encore Mozilla. [33]



Figure 26 : *le Framework Django*

Dans notre cas, Le backend doit exécuter le modèle Machine Learning et fournir des recommandations.

4.2.4 Base de données

Puisque Nous avons nous avons besoin d'implémenter un modèle ML, nous avons choisi d'utiliser **PostgreSQL** comme base de données.

PostgreSQL est un puissant système de base de données relationnelle objet open source qui utilise et étend le langage SQL combiné à de nombreuses fonctionnalités qui stockent et adaptent en toute sécurité les charges de travail de données les plus compliquées. [34] il est utilisé dans les applications de **BI et ML**.



Figure 27 : *logo de postgresQL*

Nous allons utiliser Posgresql dans le stockage des données et l'authentification des utilisateurs.

4.2.5 Le modèle d'apprentissage automatique

Dans le chapitre précédent, nous avons vu qu'il existe de nombreuses façons de mettre en œuvre le Machine Learning dans un système de recommandation pour les films. Nous avons vu que le système de filtrage basé sur le contenu (Content-Based Filetering) est l'un de ces solutions. Dans ce type il n'a pas de problème de démarrage à froid, mais nous avons vu aussi que le meilleur choix c'est le système hybride.

Nous avons choisi de construire notre système de recommandation avec l'approche **basé sur le contenu** (CBF) car nous avons une contrainte de temps et un système hybride est très complexe et long à mettre en œuvre. Nous utiliserons l'algorithme TF-IDF introduit précédemment pour construire notre système de recommandation basé sur le contenu. Ce système sera intégré dans le backend Django.

Nous avons utilisé la bibliothèque **Scikit-Learn** pour construire ce modèle. **Scikit-learn** est une bibliothèque Python destinée à l'apprentissage automatique (ML). Dans notre cas, scikit-learn nous fournit un vecteur TF-IDF préconstruit qui calcule le poids TF-IDF (équation 2) pour la description de chaque document, mot par mot.

Nous avons également utilisé les bibliothèques :

- **NumPy** est une extension du langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux. [35]
- **Pandas** est une bibliothèque écrite pour le langage de programmation python permettant la manipulation et l'analyse des données. [36]
- **Pickle** est un module de sérialisation des objets. il sérialise les objets afin qu'ils puissent être enregistrés dans un fichier, et chargés à nouveau dans un programme plus tard. [37]
- **IMDbPY** est un package Python pour récupérer et gérer les données de la base de données de films IMDb sur les films et les personnes. [38]

4.2.6 Ensembles de données (Dataset)

Nous avons vu dans les chapitres précédents que la première chose à faire lors du démarrage d'un projet est de décider quels ensembles de données seront pertinents pour notre problème. Cette étape du projet est appelée sélection de données et est très importante car si nous choisissons la mauvaise source de données, nous n'obtiendrons pas de performances réussies.

Chaque fois que nous traitons avec le filtrage basé sur le contenu, nous devons trouver les attributs de notre contenu, qui selon nous sont pertinents pour le problème. De cette façon, nous pouvons ultérieurement classer le contenu pour nos utilisateurs ou leur recommander des parties pertinentes.

Pour ces raisons nous avons décidé à utiliser l'ensemble de données **MoviesLens**. Cet ensemble de données se compose d'une séquence de balises (Tags) telles que les acteurs, les genres, les humeurs, les événements ou les années de sortie pour chaque film.

Nous allons travailler spécifiquement avec l'ensemble des données **ml-1m.zip** qui contient 1M de notes de 6000 utilisateurs sur 4000 films. Publié le 2/2003. [39]

4.2.7 Architecture du projet

Cette partie représente l'architecture de notre projet et les interactions entre les différentes parties de celui-ci. La figure ci-dessous montre l'architecture globale du projet:

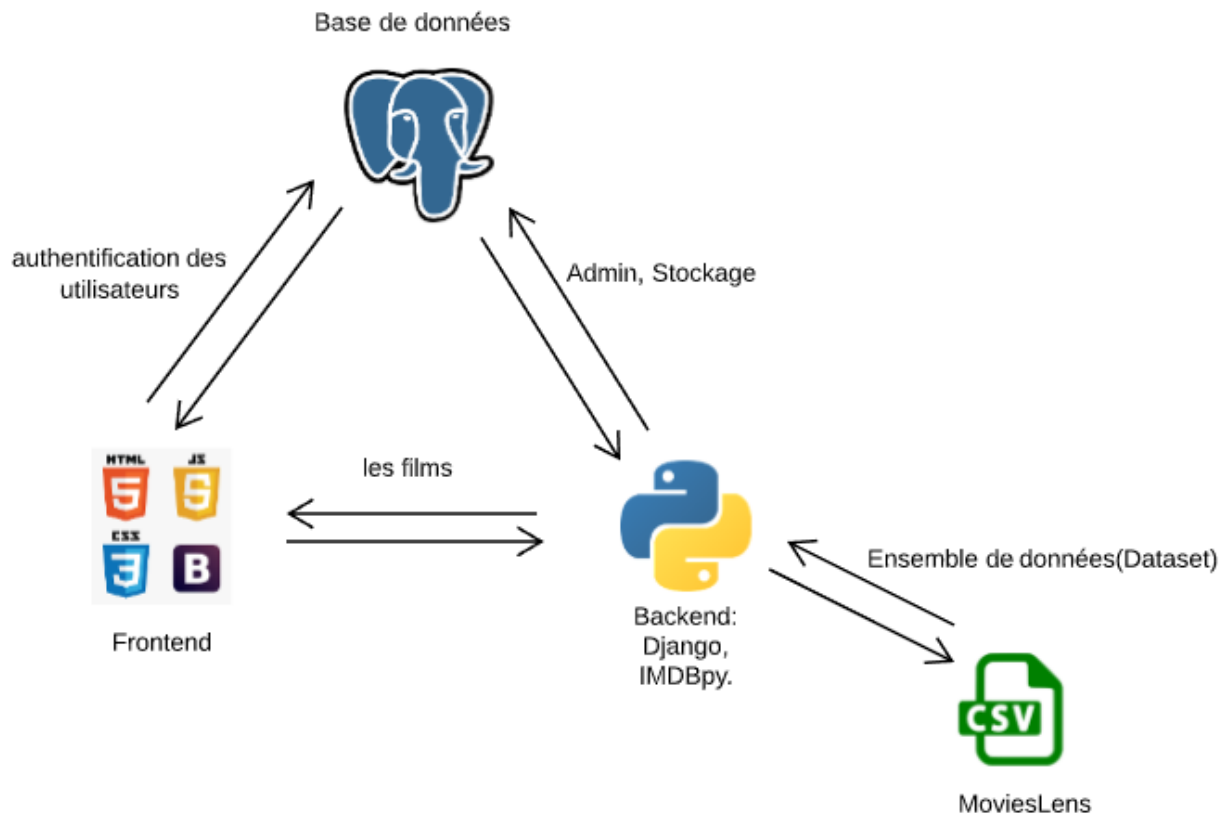


Figure 28 : *architecture du projet*

Le frontend communique avec la base de données PostgreSQL pour que les utilisateurs puissent connecter et déconnecter en leur donnant l'accès à l'application Web. Et pour stocker également les données des utilisateurs.

Le Frontend communique avec le Backend pour afficher la liste des films et leurs informations sur la page d'accueil et la page Content Filter. Les films sont récupérés à l'aide d'IMDBpy (Backend).

Le frontend communique avec le backend pour obtenir des recommandations de films pour les utilisateurs. Le backend envoie une liste de recommandation après avoir trouvé des films similaires à celui que l'utilisateur a déjà aimé.

Le backend utilise l'ensemble de données de MoviesLens, qui est fourni dans un CSV, pour construire le modèle d'apprentissage automatique (ML).

Le backend communique avec la base de données pour stocker, supprimer, mettre à jour et récupérer les données des films.

Le backend utilise le package IMDBpy pour récupérer et gérer les données de la base de données de films [IMDb](#) sur les films.

Conclusion

Dans ce chapitre, nous avons vu la conception et la modélisation de notre application, les outils et le modèle d'apprentissage automatique appliqués. Et finalement nous avons expliqué l'architecture de notre projet. Dans le chapitre suivant, nous allons enfin

5

Réalisation du projet

Introduction

Après avoir vu les outils de développement et l'architecture détaillée de notre application, il est le temps d'implémenter et présenter le coté applicatif de notre travail.

Le chapitre commence par une présentation globale de la structure du projet, ensuite une explication détaillée des importantes parties du code. Finalement, nous allons présenter les différentes interfaces de notre application et nous allons expliquer en détail comment ça fonctionne.

5.1 Structure globale

La figure ci-dessous montre la structure globale de l'application :

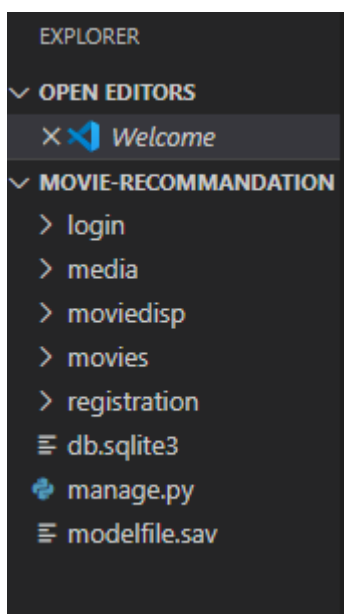
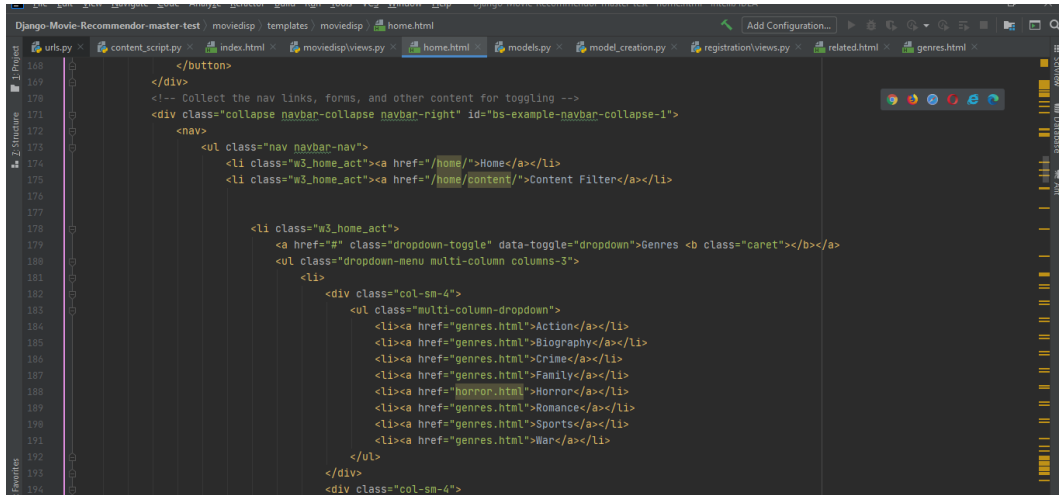


Figure 29 : *structure globale*

5.2 Explication du code

5.2.1 Frontend



```
168 </button>
169
170 </div>
171 <!-- Collect the nav links, forms, and other content for toggling -->
172 <div class="collapse navbar-collapse navbar-right" id="bs-example-navbar-collapse-1">
173   <nav>
174     <ul class="nav navbar-nav">
175       <li class="w3_home_act"><a href="/home/">Home</a></li>
176       <li class="w3_home_act"><a href="/home/content/">Content Filter</a></li>
177
178       <li class="w3_home_act">
179         <a href="#" class="dropdown-toggle" data-toggle="dropdown">Genres <b class="caret"></b></a>
180         <ul class="dropdown-menu multi-column columns-3">
181           <li>
182             <div class="col-sm-4">
183               <ul class="multi-column-dropdown">
184                 <li><a href="genres.html">Action</a></li>
185                 <li><a href="genres.html">Biography</a></li>
186                 <li><a href="genres.html">Crime</a></li>
187                 <li><a href="genres.html">Family</a></li>
188                 <li><a href="genres.html">Horror</a></li>
189                 <li><a href="genres.html">Romance</a></li>
190                 <li><a href="genres.html">Sports</a></li>
191                 <li><a href="genres.html">War</a></li>
192               </ul>
193             </div>
194           </li>
195         </ul>
196       </li>
197     </ul>
198   </div>
199 </div>
```

code 2 : *fichier home.html comme exemple des fichiers des front-end*

Dans le fichier home.html comme dans les autres fichiers genres.html et related.html, il y'a une combinaison de code entre bootstrap css javascript pour le design du site et quelques partie avec python pour générer les films avec des boucle « for » en appelant les fonctions dans le fichier views.py

5.2.2 Backend

La figure ci-dessous montre la structure du backend:

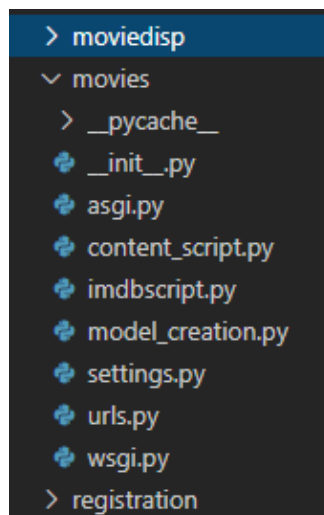
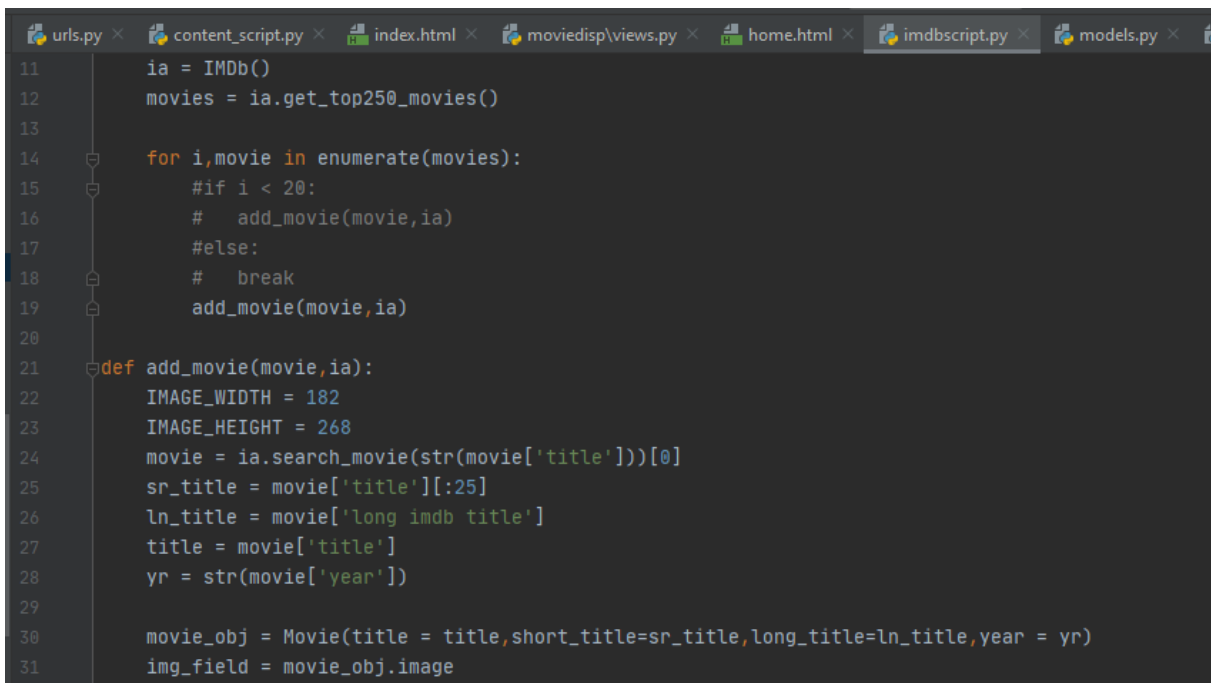


Figure 30 : *structure de backend*

Le dossier *movies* contient la configuration globale du backend:

- Le fichier `urls.py` configure la racine des URL.
- Le fichier `settings.py` contient les définitions de chaque application créée.
- Le fichier `imdbscript.py` récupère les films qui se trouvent dans la base de données IMDB.
- Le fichier `model_creation.py` crée notre modèle de recommandation basé sur le contenu.
- Le fichier `content_script.py` charge les données à partir d'ensemble de données (Dataset).

Récupération des films



```

11 ia = IMDb()
12 movies = ia.get_top250_movies()
13
14 for i, movie in enumerate(movies):
15     #if i < 20:
16     #    add_movie(movie, ia)
17     #else:
18     #    break
19     add_movie(movie, ia)
20
21 def add_movie(movie, ia):
22     IMAGE_WIDTH = 182
23     IMAGE_HEIGHT = 268
24     movie = ia.search_movie(str(movie['title']))[0]
25     sr_title = movie['title'][:25]
26     ln_title = movie['long imdb title']
27     title = movie['title']
28     yr = str(movie['year'])
29
30     movie_obj = Movie(title = title, short_title=sr_title, long_title=ln_title, year = yr)
31     img_field = movie_obj.image

```

Code 3 : *recuperation des films a partir de BD IMDB*

Ce fichier `imdbscript.py` sert à importer les films du site `imdb` à travers son API, et ajoute ces films dans la base de données, il prend aussi l'image de chaque film et la redimensionne pour quel soit intégré facilement dans la base de donnée sans occuper beaucoup d'espace après il ajoute ces films avec la fonction `add_movie()` en affectant chaque attribut avec ses propriétés.

Model d'apprentissage automatique

Nous avons vu dans les chapitres précédents comment construire un système de recommandation, maintenant nous allons suivre les mêmes étapes pour implémenter le SR dans notre projet.

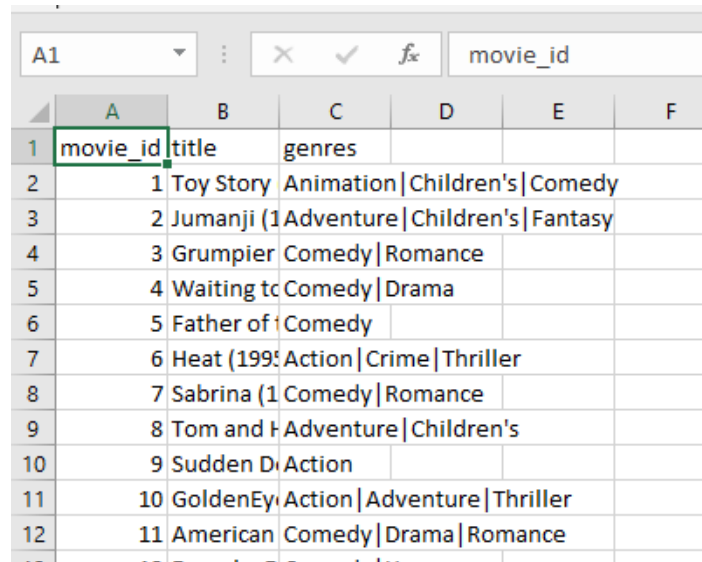
1. Collection et chargement des données :

Nous allons travailler spécifiquement avec l'ensemble des données **ml-1m.zip** qui contient 1M de notes de 6000 utilisateurs sur 4000 films.

L'ensemble de données MoviesLens contient 3 fichiers CSV :

- Movies.csv
- Ratings.csv
- Users.csv

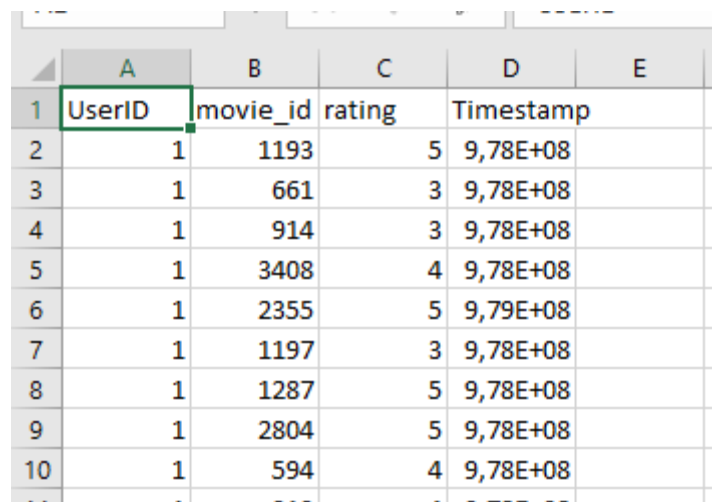
Nous allons seulement travailler avec les deux premiers fichiers :



	A	B	C	D	E	F
1	movie_id	title	genres			
2	1	Toy Story	Animation Children's Comedy			
3	2	Jumanji (1	Adventure Children's Fantasy			
4	3	Grumpier	Comedy Romance			
5	4	Waiting to	Comedy Drama			
6	5	Father of	Comedy			
7	6	Heat (199	Action Crime Thriller			
8	7	Sabrina (1	Comedy Romance			
9	8	Tom and H	Adventure Children's			
10	9	Sudden D	Action			
11	10	GoldenEy	Action Adventure Thriller			
12	11	American	Comedy Drama Romance			

Figure 31 : fichier *movies.csv* du dataset *movielens*

Le fichier *movies.csv* contient des informations sur les films, telles que le rating le titre et le genre.



	A	B	C	D	E
1	UserID	movie_id	rating	Timestamp	
2	1	1193	5	9,78E+08	
3	1	661	3	9,78E+08	
4	1	914	3	9,78E+08	
5	1	3408	4	9,78E+08	
6	1	2355	5	9,79E+08	
7	1	1197	3	9,78E+08	
8	1	1287	5	9,78E+08	
9	1	2804	5	9,78E+08	
10	1	594	4	9,78E+08	

Figure 32 : fichier *ratings.csv* du dataset *movielens*

Le fichier *ratings.csv* contient les évaluations des films par l'utilisateur. Chaque utilisateur a au moins 20 évaluations.

Maintenant, nous devons charger ces données dans le backend. Tout le processus de chargement se fait dans un *content_script.py* que nous créons.

```

filepath = "C:/Users/DELL/Desktop/PFA/Movielens/"

movies_df = pd.read_csv(os.path.join(filepath,"movies.csv"),
# print("Dataframe created!")

ratings_df = pd.read_csv(os.path.join(filepath,"ratings.csv")
# print("Dataframe created!")

```

Code 4 : *chargement de données*

Les paramètres `movies_df` et `ratings_df` sont tous deux des chemins système vers les Fichiers CSV.

2. Création d'un vecteur TF-IDF (etape2 : choix d'algorithme)

Le script ci-dessous nous génère les movies pour les charger dans la base de donnée. Ce processus se fait dans la classe `content_script.py`.

```

import pandas as pd
from moviedisp.models import Movie
import pickle
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel

def create():
    movies = Movie.objects.all().filter(top_movie=False)
    genrelist = []

    for movie in movies:
        genrelist.append(movie.genres)

    genres = pd.Series(genrelist)
    print(genres.size)

    #genres = genres.apply(lambda x: x.split(" "))
    print(genres.head(n=20))

    vectorizer = TfidfVectorizer(analyzer="word",min_df=0,ngram_range=(1,2),stop_words="english")
    tfidf_matrix = vectorizer.fit_transform(genres)
    cosine_sim = linear_kernel(tfidf_matrix,tfidf_matrix)
    cosine_pd = pd.DataFrame(cosine_sim)

    with open("modelfile.sav","wb") as f:
        pickle.dump(cosine_sim,f)

    print(cosine_pd.head(n = 20))

```

Code 5 : *le fichier content_based.py*

Nous avons déjà vu que l'algorithme **TF-IDF** est utilisé pour peser un mot clé dans n'importe quel document et attribuer l'importance à ce mot clé en fonction du nombre de fois qu'il apparaît dans le document.

Maintenant nous devons vectoriser nos données. A l'aide de bibliothèque *scikit-learn* nous pouvons obtenir un vecteur TF-IDF préconstruit qui calcule le score TF-IDF

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
```

Code 6 : *importation de vecteur TF-IDF*

Le code ci-dessous calcul le poids TF-IDF pour **le genre** de chaque film :

```
vectorizer = TfidfVectorizer(analyzer="word",min_df=0,ngram_range=(1,2),stop_words="english")
tfidf_matrix = vectorizer.fit_transform(genres)
```

Code 7 : *application de TF-IDF au genre de film*

Ici, le `tfidf_matrix` est la matrice contenant chaque mot et son score TF-IDF par rapport à chaque genre de film, ou élément dans ce cas.

Maintenant, nous avons une représentation de chaque film en termes de son genre, nous devons calculer la pertinence ou la similitude d'un film (document) à l'autre.

3. *Comparez la similitude du vecteur TF-IDF*

Maintenant que nous avons stocké chaque élément (film) en tant que vecteur de ses attributs, nous devons calculer les angles entre les vecteurs pour déterminer la similitude entre ces derniers.

```
cosine_sim = linear_kernel(tfidf_matrix,tfidf_matrix)
cosine_pd = pd.DataFrame(cosine_sim)
```

Code 8 : *calcul de similarité*

Ici, nous avons calculé la similitude en utilisant cosinus similaire de chaque film avec tous les autres films d'ensemble de données, puis nous transformons ces résultat sous forme d'une trame de données à l'aide de la fonction `DataFrame()` de bibliothèque *pandas*. Et finalement les stockés dans `cosine_pd`.

4. *Faire une recommandation*

Maintenant que nous avons trouvé les films similaires, nous pouvons les filtrer. Finalement, les envoyer au front end.

```

70 def related(request, movie_id=-1):
71     template = loader.get_template('moviedisp/related.html')
72     # new_obj = Movie.objects.all().filter(top_movie=False)[:20]
73
74     new_obj = return_related_movies(movie_id)
75     for movie in new_obj:
76         movie.range = range(movie.ratings)
77         movie.white_star = range(4-movie.ratings)
78     # return HttpResponse("Related!")
79     return HttpResponse(template.render({"movies": new_obj}, request))
80
81
82 def return_related_movies(movie_id):
83     with open("modelfile.sav", "rb") as f:
84         cosine_sim = pickle.load(f)
85     new_obj = Movie.objects.all().filter(top_movie=False)
86     movie_list = []
87     index_list = []
88     for index, movie in enumerate(new_obj):
89         movie_list.append(movie.id)
90         index_list.append(index)
91
92     indices = pd.Series(index_list, index=movie_list)
93     print(indices.head(n=20))
94     idx = indices[movie_id]
95     sim_scores = list(enumerate(cosine_sim[idx]))
96     sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)[1:25]

```

Code 9 : les fonctions dans views.py permettant de générer les recommandations

Architecture Django « MVT »

Le MVT (Model, View, Template) est un modèle de conception logicielle. Il s'agit d'une collection de trois composants importants, la vue du modèle et le modèle. Le modèle aide à gérer la base de données. Il s'agit d'une couche d'accès aux données qui gère les données.

Le modèle est une couche présentation qui gère complètement la partie de l'interface utilisateur. La vue est utilisée pour exécuter la logique métier et interagir avec un modèle pour transporter des données et restituer un modèle.

Bien que Django suive le modèle MVC mais conserve ses propres conventions, Ainsi le contrôle est géré par le framework lui-même.

Il n'y a pas de contrôleur séparé et l'application complète est basée sur la vue du modèle et le modèle. C'est pourquoi on l'appelle application MVT.

Voir le graphique suivant qui montre le flux de contrôle basé sur MVT.

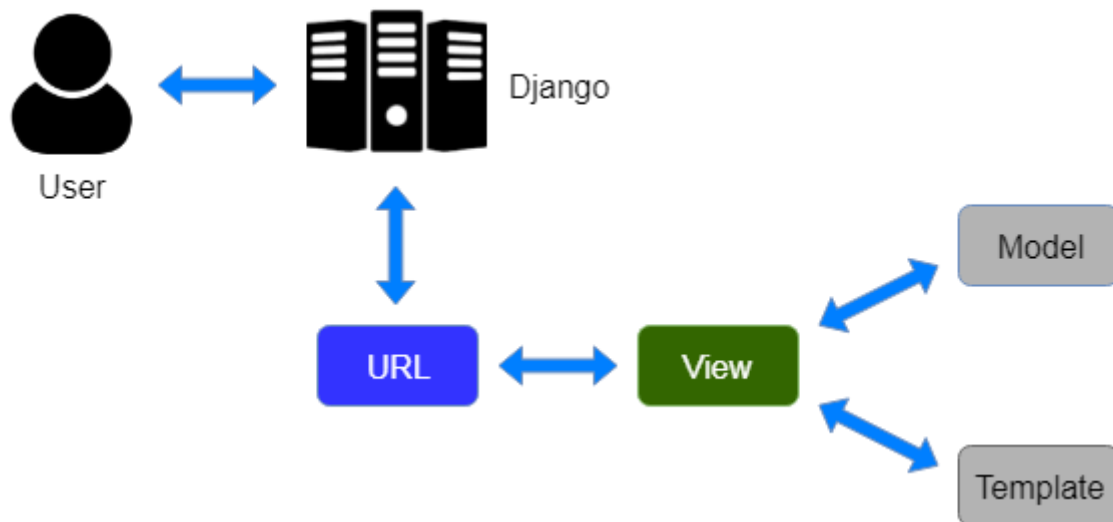


Figure 33 : Architecture MVT du framework django

Voir le graphique qui montre le MVT : un utilisateur demande une ressource au Django, Django fonctionne comme un contrôleur et vérifie la ressource disponible dans l'URL.

Si des mappages d'URL, une vue est appelée qui interagit avec le modèle et le modèle, elle rend un modèle.

Django répond à l'utilisateur et envoie un modèle en tant que flux de contrôle "response".

Création Des Modèles : après avoir créé les modèles des classes avec les commandes de Django (python manage.py startapp moviedisp) qui nous a générer un ensemble de fichier

```

__init__.py
admin.py
apps.py
migrations/
    __init__.py
models.py
tests.py
views.py
  
```

Nous allons dû ajouter les attributs du moviedisp dans le fichier models.py.

```
urls.py × content_script.py × index.html × home.html × models.py ×
3      # Create your models here.
4
5      class Movie(models.Model):
6          title = models.CharField(max_length=150, default="")
7          short_title = models.CharField(max_length=25)
8          long_title = models.CharField(max_length=200)
9          year = models.CharField(max_length=4)
10         image = models.ImageField(upload_to='movieimg')
11         ratings = models.IntegerField(default=4)
12         half_rating = models.BooleanField(default=False)
13         top_movie = models.BooleanField(default=True)
14         genres = models.CharField(max_length=200, default="")
15
16         def __str__(self):
17             return self.title
18
```

Code 10 : *classe movie dans le fichier models*

Pour lier la base de données avec notre projet il suffit de configurer le fichier settings dans le projet et indiquer le nom de l'administrateur son mot de passe et la base de données utilisé ainsi il faut installer les modules nécessaires pour la connexion entre Django et PostgreSQL.

```
index.html x home.html x related.html x genres.html x settings.py x
78 # https://docs.djangoproject.com/en/3.0/ref/settings/#databases
79
80 DATABASES = {
81     'default': {
82         'ENGINE': 'django.db.backends.postgresql',
83         'NAME': 'moviedatabase',
84         'USER': 'postgres',
85         'PASSWORD': 'admin',
86         'HOST': 'localhost',
87     }
88 }
89 }
```

Code 11 : Liaison avec la base de données

5.3 Les interfaces d'applications

Pour éviter la duplication des données il est nécessaire de ne pas avoir deux utilisateurs avec le même username c'est pour cela qu'on doit vérifier à chaque fois est ce que le username ajouté par l'utilisateur existe déjà dans la base de données ou non.

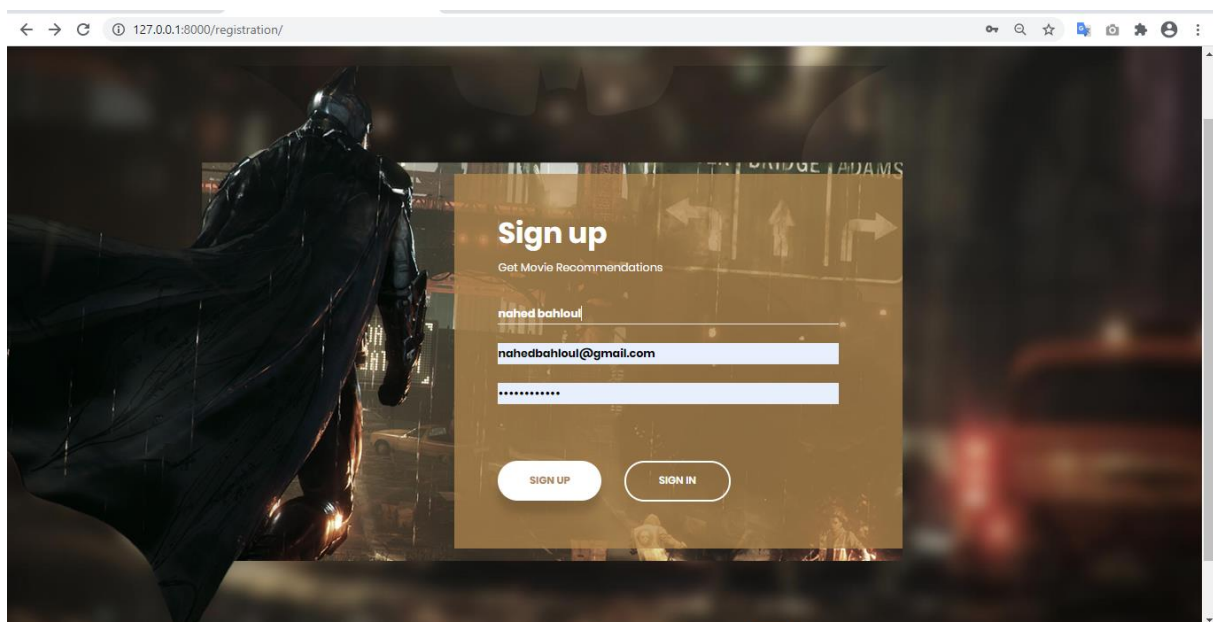


Figure 34 : Interface d'enregistrement

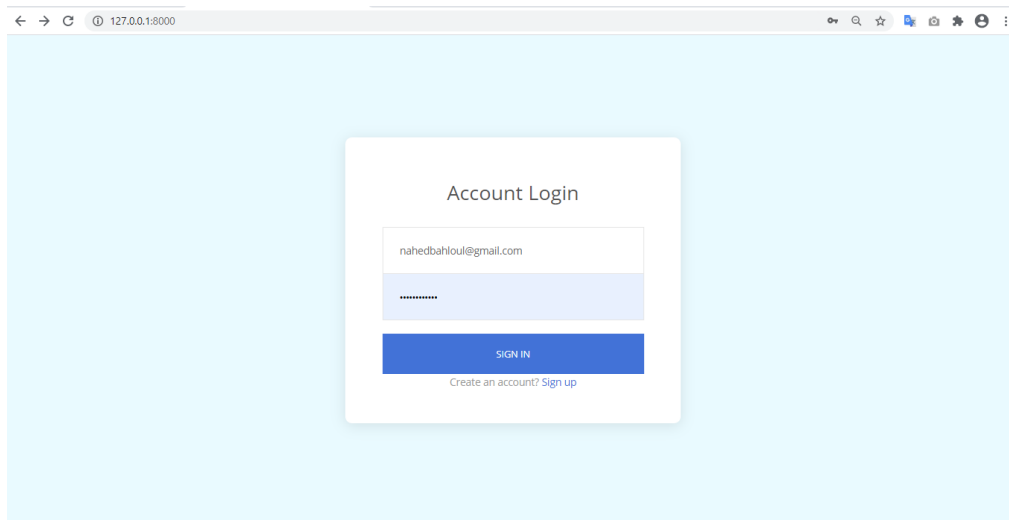


Figure 35 : Interface d'authentification

Pour l'interface de login l'utilisateur doit entrer un email valide et un or de passe entre 8 et 20 caractères sinon un message d'erreur va s'afficher lui demander de fournir des données valide respectant les conditions citées.

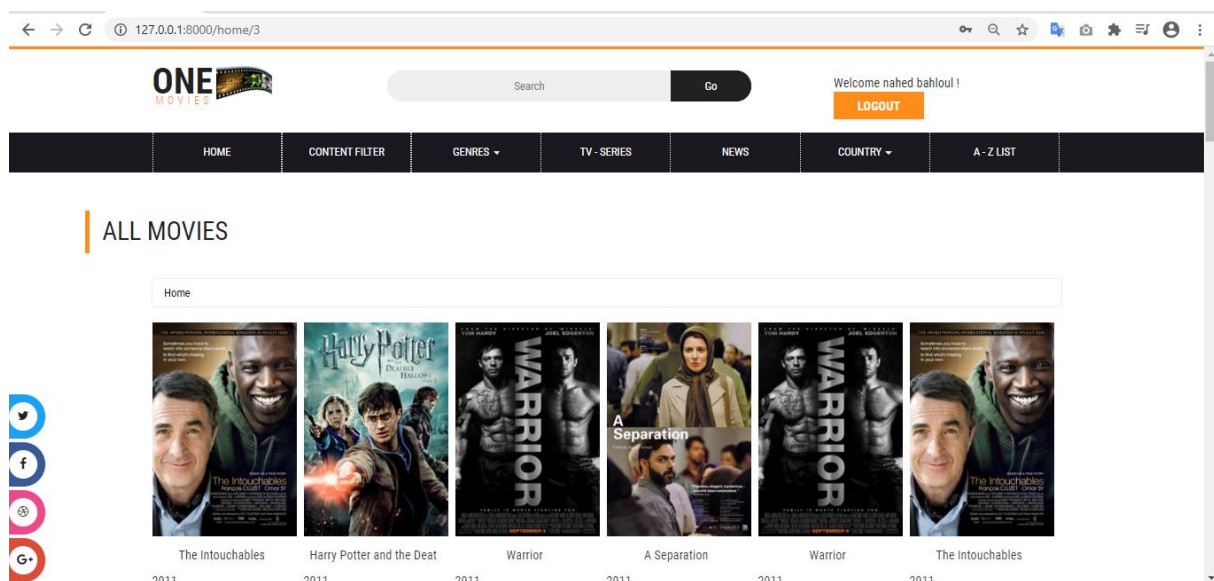


Figure 36 : Interface d'accueil

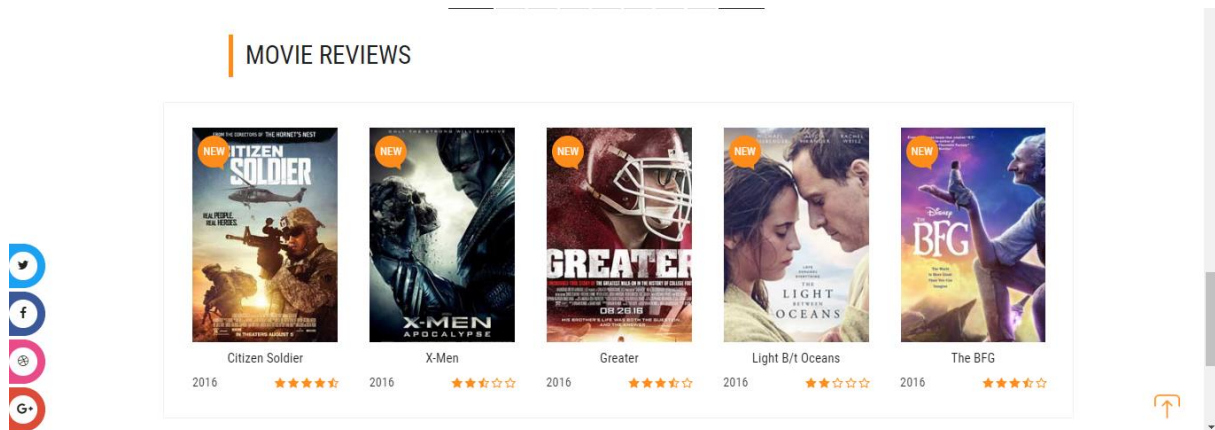


Figure 37 : sliders à la fin de la page d'accueil

Dans cette partie on a ajouté une sliders pour les films les plus visités (d'une manière statique)

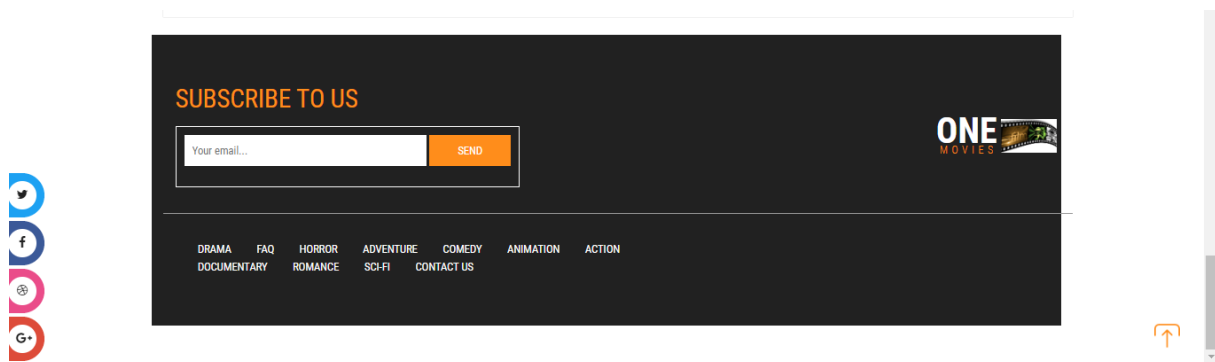


Figure 38 : souscription pour l'envoi des nouveautés

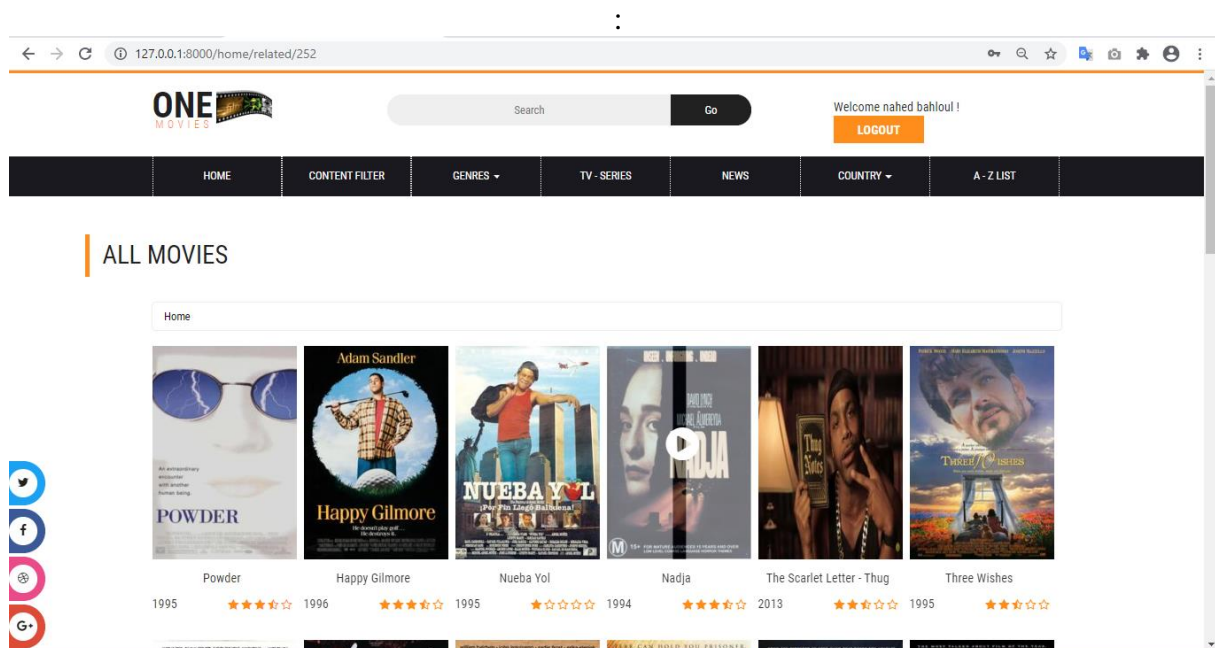


Figure 39 : interface de recommandation

Interface de recommandation, à travers cette interface les algorithmes appliquées derrière génère des recommandations spécifiques pour chaque film choisi parmi la liste des recommandations, la plateforme est programmée pour générer 24 films recommandé pour chaque film proposé.

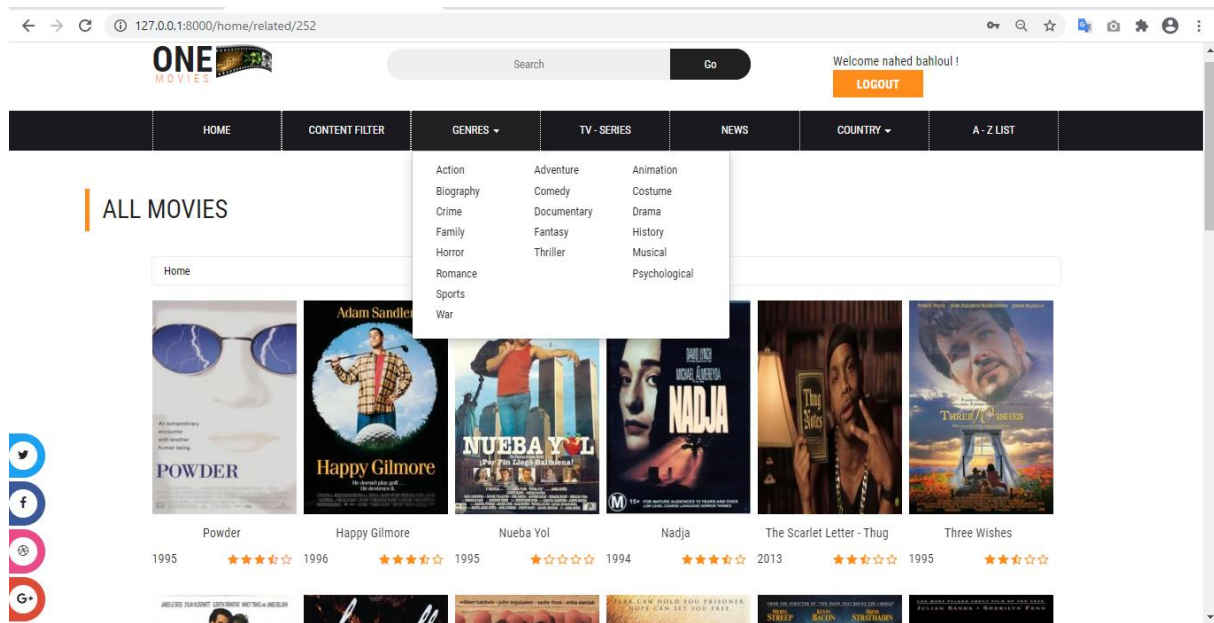


Figure 40 ; liste des catégories des films

La liste des genres la barre de recherche ainsi que d'autres éléments de la plateforme ne sont pas encore fonctionnelles nous essayons de développer le site encore, même après l'envoi de notre rapport.

The screenshot shows the PgAdmin interface with a table of movies. The table has columns for id, short_title, long_title, year, image, ratings, title, and top_movie. The data is as follows:

id	short_title	long_title	year	image	ratings	title	top_movie
2055	Gremlins	Gremlins (1984)	1984	movieimg/Gremlins_Q9j8DnJ...	3	Gremlins	false
2056	Gremlins 2: The New Batch	Gremlins 2: The New Batch (19...	1990	movieimg/Gremlins_2_The_Ne...	2	Gremlins 2: The New Batch	false
2057	The Goonies (1985)	'Oliver Harper's Retrospectives...	2013	movieimg/The_Goonies_1985...	3	The Goonies (1985)	false
2058	The Mask of Zorro	The Mask of Zorro (1998)	1998	movieimg/The_Mask_of_Zorro...	3	The Mask of Zorro	false
2059	Polish Wedding	Polish Wedding (1998)	1998	movieimg/Polish_Wedding_D0...	3	Polish Wedding	false
2060	This World, Then the Fire	This World, Then the Fireworks...	1997	movieimg/This_World_Then_sh...	2	This World, Then the Fireworks	false
2061	Soylent Green	Soylent Green (1973)	1973	movieimg/Soylent_Green_1973...	3	Soylent Green	false
2062	Metropolis	Metropolis (1927)	1927	movieimg/Metropolis_RouJx2...	4	Metropolis	false
2063	Back to the Future Part I	Back to the Future Part II (1989)	1989	movieimg/Back_to_the_Future...	3	Back to the Future Part II	false
2064	Back to the Future Part I	Back to the Future Part III (1990)	1990	movieimg/Back_to_the_Future...	3	Back to the Future Part III	false
2065	The Poseidon Adventure	The Poseidon Adventure (1972)	1972	movieimg/The_Poseidon_Adve...	3	The Poseidon Adventure	false
2066	Freaky Friday (1977)	'Disneyember' Freaky Friday (...)	2013	movieimg/Freaky_Friday_1977...	3	Freaky Friday (1977)	false
2067	The Absent Minded Profess	The Absent Minded Professor (...)	1961	movieimg/The_Absent_Minde...	3	The Absent Minded Professor	false
2068	The Apple Dumpling Gang R	The Apple Dumpling Gang Ride...	1979	movieimg/The_Apple_Dumplin...	2	The Apple Dumpling Gang Ride...	false
2069	Babes in Toyland	Babes in Toyland (1961)	1961	movieimg/Babes_in_Toyland_I...	3	Babes in Toyland	false
2070	Bambi	Bambi (1942)	1942	movieimg/Bambi_Fz1rXy.jpg	3	Bambi	false

Figure 41 : table contenant l'ensemble des films

id	password	last_login	is_superuser	username	first_name	last_name	email	is_staff
1	pbkdf2_sha256\$150000\$F5EJ...	2020-06-23 ...	false	Souhail Nahed			souhailnahed@gmail.com	false
2	pbkdf2_sha256\$150000\$Fgfm...	2020-06-23 ...	false	youssef bahloul			youssefbahloul@gmail.com	false
3	pbkdf2_sha256\$100000\$EU1...	2020-06-24 ...	true	souhail			souhailnahed@gmail.com	true
4	pbkdf2_sha256\$100000\$K4gx...	2020-06-24 ...	false	amine			amine@gmail.com	false
5	pbkdf2_sha256\$150000\$5ih4E...	[null]	false	hassan			hassan@gmail.com	false
6	pbkdf2_sha256\$150000\$G08c...	[null]	false	mustapha			mustapha@gmail.com	false
7	pbkdf2_sha256\$150000\$9egm...	[null]	false	aziz			aziz@gmail.com	false
8	pbkdf2_sha256\$100000\$8ZNJ...	2020-06-24 ...	false	bahloul			bahloul@gmail.com	false
9	pbkdf2_sha256\$150000\$REQH...	[null]	false	nahed			nahed@gmail.com	false
10	pbkdf2_sha256\$150000\$SzOQ...	2020-06-23 ...	false	youssef			youssef@gmail.com	false
11	pbkdf2_sha256\$100000\$0x0...	2020-06-25 ...	false	nahed bahloul			nahedbahloul@gmail.com	false

Figure 42: table des utilisateurs Django

Pour les mots de passe ils ne sont pas stockés en clair dans la base de données mais ils sont cryptés à l'aide du hashage sha256 utilisé par Django.

Par défaut, Django utilise l'algorithme PBkDF2 avec un hachage SHA256, un mécanisme d'étirement de mot de passe recommandé par le NIST. Cela devrait être suffisant pour la plupart des utilisateurs : il est assez sécurisé, ce qui nécessite un temps de calcul énorme pour se casser.

Django génère une interface admin avec une simple ligne de commande

(python manage.py createsuperuser)

Figure 43: l'interface d'admin Django,

Pour l'interface d'admin Django, l'administrateur aura la possibilité d'ajouter modifier supprimer des films facilement, avec C

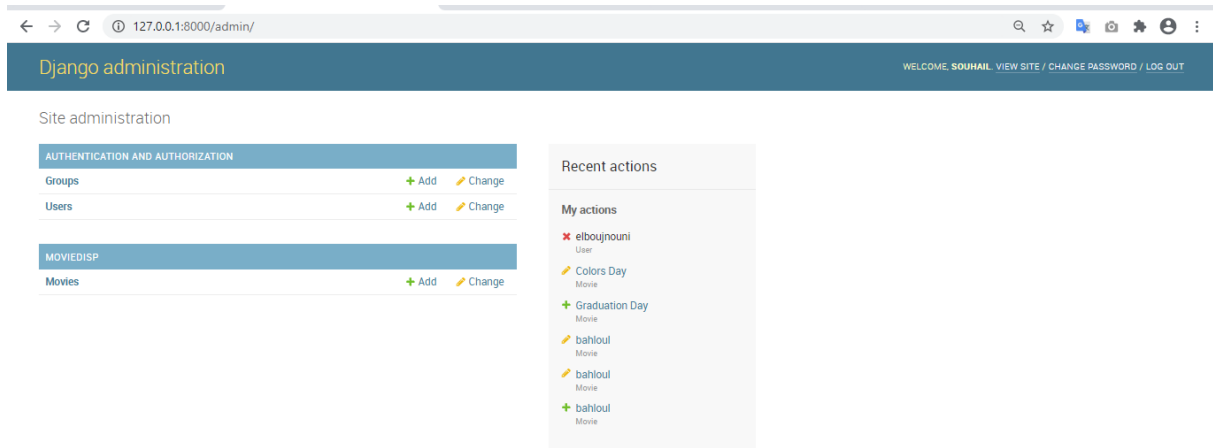


Figure 44 : interface d'accueil d'admin Django

Dans l'interface admin toutes les modifications déjà faites sur la plateforme sont enregistrés est gardés comme trace, pour permettre à un groupe lui travaillent ensemble sur un site de suivre l'avancement et savoir en cas de problèmes, la source de ce problème est celui qui était derriere et pouvoir la régler facilement.

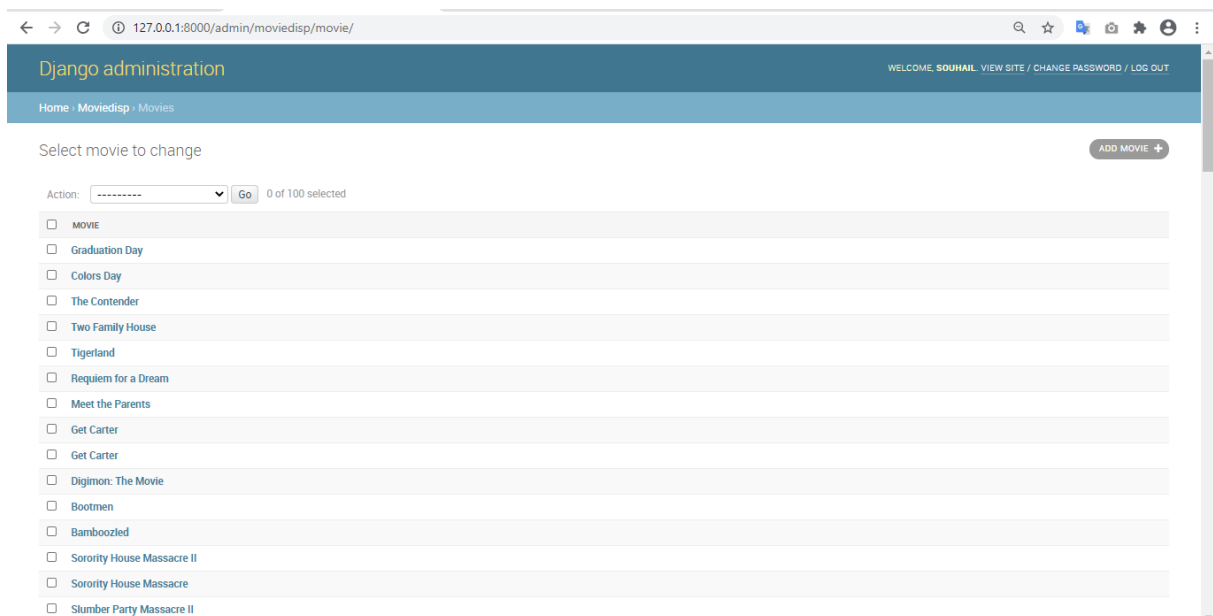


Figure 45 : liste des films de l'interface d'admin

La suppression facile des films ajouté précédemment dans la base de données d'après l'interface d'admin

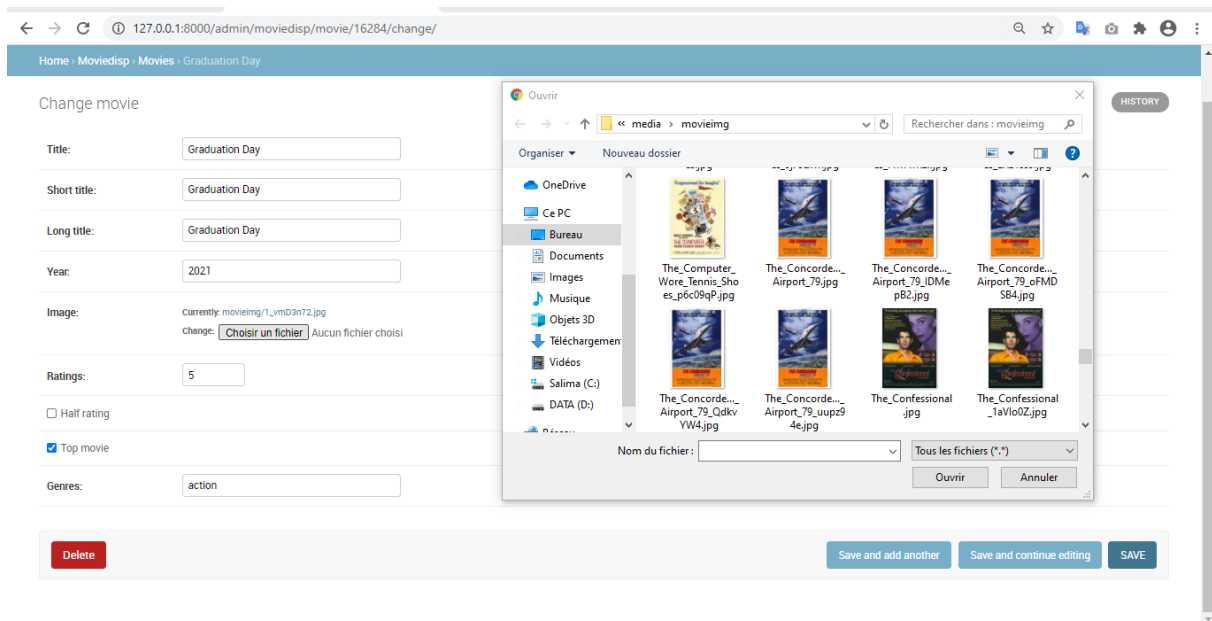


Figure 46 : Ajouter un films d'après l'interface d'admin

Django interprète les variable donnée dans le code et crée des champs pour chaque variable facilitant la manipulation des données.

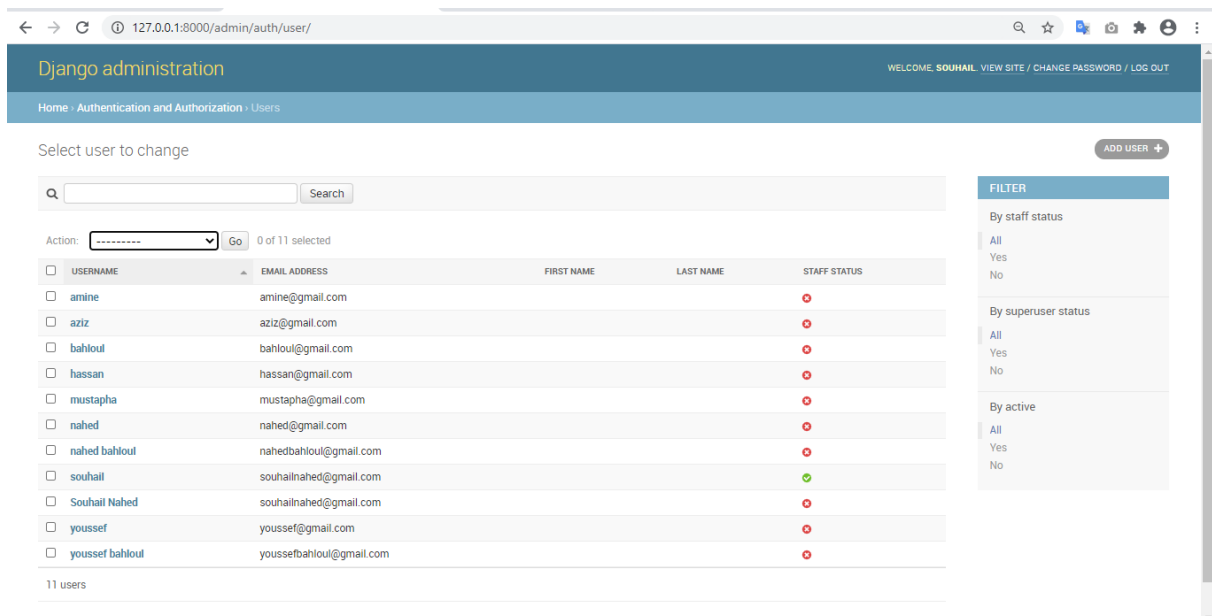


Figure 47 : liste des utilisateurs

L'ajout la suppression des utilisateurs et facile d'après l'interface d'admin sans aller à l'interface de login.

Figure 48 Ajout d'un nouveau utilisateur

Figure 49 : création d'un nouveau groupe avec ces privilèges

Dans cette interface il y'a aussi la possibilité de créer un groupe d'utilisateurs et lui attribuer un ensemble de privilège pour chaque application (partie) du projet (site web).

Conclusion

Nous avons appris à créer un système de recommandation entièrement fonctionnel en Python avec un filtrage basé sur le contenu, et nous avons implémenté ce système dans notre application web de recommandation des films. Mais comme nous l'avons vu, le filtrage basé sur le contenu n'est pas pratique, ou plutôt, pas très fiable lorsque le nombre d'éléments augmente avec le besoin de descriptions claires et différenciées.

Pour surmonter tous les problèmes abordés précédemment, nous pouvons mettre en œuvre des techniques de filtrage collaboratif à notre système pour obtenir finalement un système hybride, qui est avéré meilleures et plus évolutives.

6

Conclusion générale et perspectives

Dans ce projet, nous avons introduit les bases d'apprentissage automatique (ML) et nous avons expliqué et étudié comment cela fonctionne dans le cas d'un système de recommandation des films. Nous avons mentionné l'importance d'utiliser un système de recommandation dans votre site, et comment les grandes entreprises les utilisent pour augmenter leurs profits. Nous avons vu les avantages et les inconvénients de chaque type de système de recommandation et nous avons présenté les différentes étapes d'implémenter et d'évaluer un SR.

Dans notre application, nous avons choisi d'implémenter un système de filtrage basé sur le contenu avec l'algorithme TF-IDF, car il est l'un des systèmes de recommandation qui n'a pas de problème de démarrage à froid c'est-à-dire les nouveaux articles peuvent être recommandés, plus il n'a pas besoin de données sur les autres utilisateurs pour recommander un film. Le projet devra fournir aux utilisateurs des bonnes recommandations pour chaque film qu'ils veulent sur une interface responsive.

Nous pouvons maintenant dire que l'objectif de ce projet est atteint puisque nous avons construit un système de recommandation basé sur le contenu sur une interface conviviale, responsive et attrayante. Ce n'est donc pas un résultat parfait mais c'est assez bon. Nous pourrions aller plus loin pour améliorer notre **interface d'application** en ajoutant le filtrage par genre (action, romance, drame, documentation...), un stepper à la première connexion sur l'application avec un guide virtuel (pop-ups...) expliquant le but de l'application et montrant les différentes étapes successives pour une meilleure utilisation de site afin d'obtenir les bonnes recommandations et tout cela pour offrir une parfaite expérience des utilisateurs. Nous pourrions enfin améliorer les informations sur les films en ajoutant sur chaque film des liens vers Netflix ou d'autres sociétés de streaming qui offrent cet article.

C'est vrai que les recommandations dans notre application sont assez bien et pertinentes mais pas très fiables lorsque le nombre d'éléments augmente avec le besoin de descriptions claires et différenciées. Cela est dû à l'utilisation d'un système de filtrage basé sur le contenu. Nous avons vu que ce système basé seulement sur les caractéristiques d'un film, donc l'utilisateur ne sera jamais recommandé pour différents éléments, s'il aime les films d'action le système recommande seulement les films d'action. Nous pourrions aller plus loin pour améliorer les **recommandations** en ajoutant un système basé sur le contenu et en les fusionnant dans un système hybride. Le système collaboratif équilibrera les recommandations en recommandant les différents genres des films (Action, Document, etc.) aux différents utilisateurs. Le système hybride pourrait détecter si un film est populaire ou non en analysant le nombre d'avis ou de notes. Il pourrait aussi à chaque fois fusionner les recommandations issues des deux systèmes

et donner aux utilisateurs des résultats équilibrés. Donc c'est la meilleure façon d'implémenter un SR.

Concernant l'auto-apprentissage, Nous avons appris comment utiliser les algorithmes d'apprentissage automatique (Machine Learning) dans le cas de système de recommandation, comment construire un système de recommandation et comment l'évaluer. Nous avons appris l'application de langage de programmation Python dans le domaine d'apprentissage automatique et comme nous ne savons rien à ce sujet au début. Nous avons également appris à développer un serveur principal avec le Framework Django et à intégrer un modèle ML à l'intérieur. Enfin, nous avons appris à construire un système entièrement fonctionnel et une architecture capable d'exécuter l'application complète du projet.

Bibliographie

- [1]: <https://research.netflix.com/research-area/recommendations>
- [2]: <https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada>
- [3] : <https://www.lebigdata.fr/machine-learning-et-big-data>
- [4] : <https://searchenterpriseai.techtarget.com/definition/machine-learning-ML>
- [5] : <https://towardsdatascience.com/the-7-steps-of-machine-learning-2877d7e5548e>
- [6] : <https://towardsdatascience.com/a-beginners-guide-to-machine-learning-5d87d1b06111>
- [7] : <https://www.digitalocean.com/community/tutorials/an-introduction-to-machine-learning>
- [8] : <https://www.geeksforgeeks.org/regression-classification-supervised-machine-learning/>
- [9] : <https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn>
- [10]:<https://medium.com/apprentice-journal/pca-application-in-machine-learning-4827c07a61db>
- [11] : http://www.nl pca.org/pca_principal_component_analysis.html
- [12]:https://medium.com/@jonathan_hui/machine-learning-singular-value-decomposition-svd-principal-component-analysis-pca-1d45e885e491
- [13]:<https://towardsdatascience.com/k-means-clustering-identifying-f-r-i-e-n-d-s-in-the-world-of-strangers-695537505d>
- [14]:<https://www.forbes.com/sites/bernardmarr/2016/09/30/what-are-the-top-10-use-cases-for-machine-learning-and-ai/#3531327394c9>
- [15] : https://fr.wikipedia.org/wiki/Syst%C3%A8me_de_recommandation
- [16]: <https://medium.com/@bhabukpokharel20/books-recommendation-system-74443f3c88f2>
- [17] : <https://towardsdatascience.com/brief-on-recommender-systems-b86a1068a4dd>
- [18]:<https://medium.com/@cfpinela/recommender-systems-user-based-and-item-based-collaborative-filtering-5d5f375a127f>
- [19] : <http://dataconomy.com/2015/03/an-introduction-to-recommendation-engines/>
- [20]:<https://developers.google.com/machine-learning/recommendation/content-based/summary>
- [21]:<https://developers.google.com/machine-learning/recommendation/collaborative/summary>

- [22] : <https://towardsdatascience.com/brief-on-recommender-systems-b86a1068a4dd>
- [23] : <https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-recommendation-engine-python/>
- [24] : <https://www.udemy.com/course/building-recommender-systems-with-machine-learning-and-ai/>
- [25] : <https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234>
- [26] : <https://medium.com/fnplus/evaluating-recommender-systems-with-python-code-ae0c370c90be>
- [27] : <https://www.offerzen.com/blog/how-to-build-a-content-based-recommender-system-for-your-product>
- [28] : <https://www.link-assistant.com/news/tf-idf-tool-for-seo.html>
- [29] : <https://community.ibm.com/community/user/legacy?lang=en>
- [30] : <https://www.digitalocean.com/community/tutorials/an-introduction-to-machine-learning>
- [31] : [https://fr.wikipedia.org/wiki/Bootstrap_\(framework\)](https://fr.wikipedia.org/wiki/Bootstrap_(framework))
- [32] : <https://fr.wikipedia.org/wiki/JavaScript>
- [33] : [https://fr.wikipedia.org/wiki/Django_\(framework\)](https://fr.wikipedia.org/wiki/Django_(framework))
- [34] : <https://medium.com/we-build-state-of-the-art-software-creating/why-should-i-use-postgresql-as-database-in-my-startup-company-96de2fd375a9>
- [35] : <https://fr.wikipedia.org/wiki/NumPy>
- [36] : <https://fr.wikipedia.org/wiki/Pandas>
- [37] : <https://www.datacamp.com/community/tutorials/pickle-python-tutorial>
- [38] : <https://imdbpy.github.io/>
- [39] : <https://grouplens.org/datasets/movielens/1m/>