Projet Python

The Movie DataBase & Netflix

Tigran GYURJYAN
Guillaume ROUSTAN
Ibrahim DIABIRA
Marwan HAMZAOUI

Master Économétrie-Statistiques
Langage de Programmation
2024-2025

INTRODUCTION

Dans le cadre de notre cours de Langage de programmation au sein du Master Économétrie-Statistique de l'université Paris 1 Panthéon Sorbonne, nous avons comme devoir de fin de semestre un projet basé sur l'analyse de données. Pour ce faire, nous avions au choix deux sujets : "The Movie Database & Netflix" et " Logements Airbnb dans les grandes villes européennes". Nous avons donc, après mûre réflexion, décidé d'opter pour le sujet 1: "The Movie Database & Netflix".

Pourquoi ce sujet ?

Nous avons choisi le sujet "The Movie DataBase & Netflix" pour plusieurs raisons. La première étant que nous sommes de grands consommateurs de films et de séries. Nous sommes également nombreux à avoir grandi et vécu notre adolescence avec Netflix. C'est donc tout naturellement que nous nous sommes tournés vers le sujet qui met en œuvre la plateforme de streaming leader du marché. Ce projet reflète nos centres d'intérêts et nous permet de travailler sur ce qui nous passionne, ce que nous apprécions et ce que nous connaissons. Ensuite, nous nous sommes toujours demandé "Comment à partir de certaines de nos informations, les plateformes de streaming nous suggèrent des films que nous puissions potentiellement aimer ?". Cette question à été un tremplin vers le choix du sujet n°1.

Objectif général :

L'objectif général de notre projet est donc de construire un moteur de recommandation simple pour les films. Pour cela, nous avons accès à deux bases de données au format CSV : "Full TMDb Movies Dataset 2024" et "Netflix Movies and TV Shows". Nous allons donc procéder à une brève description de nos deux datasets.

La dataset "Full TMDb Movies Dataset 2024" est un extrait de The Movies Database et recense des informations sur plus de 1 000 000 de films. On y retrouve les titres, les dates de sortie, les durées, le budget, les recettes, les genres, les notes moyennes données par les spectateurs, un score de popularité du film, les pays de productions, et pleins d'autres indicateurs sur ces films.

Quant au dataset Netflix, c'est le même principe. C'est une base de données conséquente qui contient des informations sur près de 8 000 films et séries disponibles sur la plateforme Netflix. On y retrouve le nom du film , les réalisateurs, les acteurs principaux, les genres, la date de publication, etc… Il donne moins d'informations que "TMDB". Par exemple, il ne donne pas d'informations sur le budget des films/séries, leurs recettes ou encore les notes moyennes des téléspectateurs. Certaines informations sont communes et d'autres différentes.

Objectifs spécifiques :

Afin de mener à bien notre objectif général, il est nécessaire de diviser notre devoir en plusieurs étapes. Les objectifs spécifiques de ce projet reposaient sur 3 axes dont l'analyse descriptive et la visualisation des données. Puis la mise en place d'un moteur de recherche et de recommandation avec la création d'une interface graphique.
Dans un premier temps, nous avons réalisé une analyse descriptive et une visualisation des données afin d'avoir une vision globale des données et de mieux comprendre les caractéristiques des films. Par la suite, nous avons développé un moteur de recherche et de recommandation simple afin de suggérer des films à partir des préférences des utilisateurs qui ont été filtrés par des fonctions, on peut prendre en exemple les genres, la note moyenne des films.
Enfin pour avoir un aperçu du programme de manière intuitive et accessible à tous le monde, nous avons créé une interface graphique.

Ce projet nous offre l'opportunité d'explorer les coulisses des systèmes de recommandation de films, en concevant et en comprenant le fonctionnement d'un moteur de recommandation.

VISUALISATION ET ANALYSE DES DONNÉES

Pour pouvoir construire un moteur de recherche, il est primordial de procéder, avant tout, à une analyse et une visualisation des données. Il faut regarder “où on met les pieds”, et donc faire une brève analyse de nos datasets, et essayer de voir les variables qui pourraient être pertinentes pour notre moteur de recommandation.
On pourra retrouver toutes les étapes de cette partie dans le Notebook associé.
On commence par importer nos données au format CSV grâce à la librairie Pandas de Python et on obtient nos deux DataFrames. On peut donc maintenant procéder à une description classique de nos données.

Description classiques des données :

Pour cette étape préalable, nous avons décidé d’analyser le nombre d’observations , le nombre de variables, le types de variables de nos DataFrames. Nous allons également analyser si nous avons des valeurs aberrantes et des valeurs manquantes.
Tout d’abord, on va regarder le nombre de d'observations et de variables grâce à la méthode « shape » de pandas qui affiche le nombre de lignes(=observations) et de colonnes(=variables).
TMDB contient 1 130 164 observations(=films) et 24 variables. Ensuite, on regarde le nom de ces variables, et de quel type grâce à la méthode « dtypes ».

id	int64
title	object
vote_average	float64
vote_count	int64
status	object
release_date	object
revenue	int64
runtime	int64
adult	bool
backdrop_path	object
budget	int64
homepage	object
imdb_id	object
original_language	object
original_title	object
overview	object
popularity	float64
poster_path	object
tagline	object
genres	object
production_companies	object
production_countries	object
spoken_languages	object
keywords	object

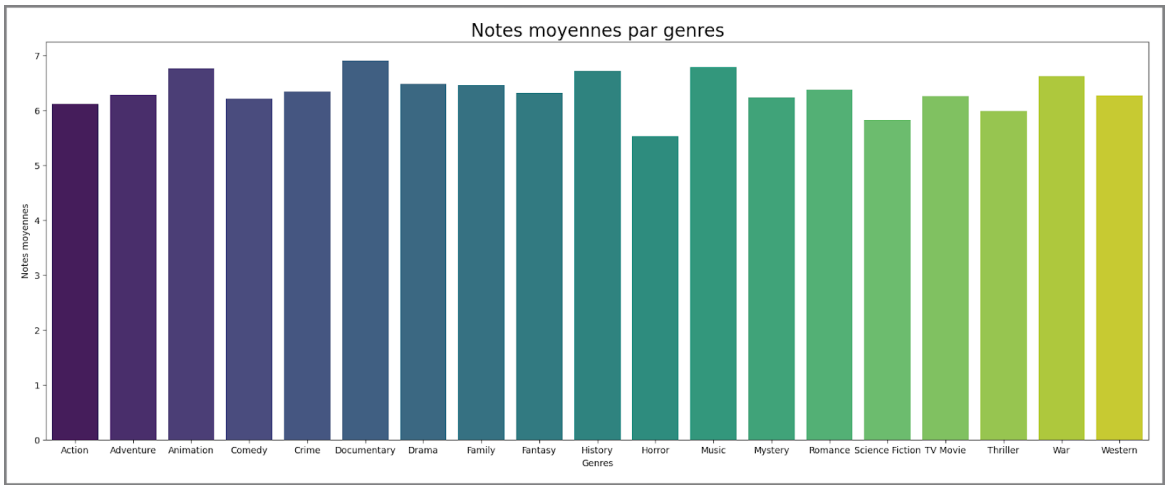
On obtient donc les noms des variables et leurs types. On peut faire plusieurs constats : la majorité des types de nos variables sont des “object” qui est le type de base, nous avons quelques variables qui sont des entiers comme le nombre de votants, les recettes, la durée et le budget du film, et des variables qui sont des flottants (nombres décimaux) comme la note moyenne et le score de popularité. D’ailleurs grâce à la méthode “describe”, on peut avoir les statistiques descriptives des entiers et des flottants, et on remarque des valeurs assez surprenantes pour les minimums et les maximum de chaque variable. Par exemple, on remarque qu’il y a beaucoup de films ayant des notes nulles dû aux nombres de votants qui sont nuls, il faudra donc faire attention lorsqu’on les traite et trouver une astuce pour ne pas prendre en compte les valeurs nulles du DataFrame. Phénomènes semblables pour les durées de films qui ont des valeurs minimales à 0 et des valeurs maximales très grandes. On observe également des recettes de film à 0. Il faudra être vigilant lorsqu’on traite certaines variables.
Ensuite on retrouve un booléen, la variable “adult” qui est une variable binaire qui vaut “True” si le film est dans la catégorie adulte et “False” sinon.
Après avoir regardé nos variables et leurs types, on va essayer de regarder quelles sont les colonnes avec beaucoup de valeurs manquantes. Pour cela, il suffit d’utiliser les méthodes “isna” et “sum”. On pourra directement voir les variables non pertinentes avec beaucoup de valeurs manquantes , et il faudra être vigilant lorsqu’on les traite lors des prochaines étapes de notre projet.

Pour les données Netflix, on procède aux mêmes étapes. Cependant, lorsqu'on l'importe, on a beaucoup de colonnes superflues composées uniquement de valeurs manquantes, qui apparaissent. On supprime ces colonnes et on constate alors qu'il y a 8809 observations et 12 variables. On voit aussi que les données Netflix sont composées uniquement de variables de type "object" sauf la date de publication, et ne sont donc pas pertinentes pour faire des statistiques descriptives.

Après la réalisation d'une description classique des données. Nous pouvons, à présent, répondre aux questions de la Partie 1.

Quelle est la note moyenne par genre des films ? Représenter graphiquement

Pour répondre à cette question, nous avons commencé par trier les valeurs manquantes de la catégorie "genres". En effet, on observe plus de 400 000 films ayant la mention "Nan" dans la catégorie "genres". On supprime ces films, grâce à la fonction "notnull" car ils ne seront pas pertinents pour notre analyse. Ensuite, on remarque qu'un seul film contient plusieurs genres, donc on va exploser notre catégorie "genres", de sorte à ce qu'on puisse avoir un nouveau DataFrame dans lequel on va retrouver des nouvelles lignes avec plusieurs fois le même film mais avec un seul genre à la fois. Pour ce faire, on va mettre nos genres en une seule liste séparée à l'aide de "str.split" et on pourra ensuite utiliser "explode". Après, il faut s'intéresser aux notes moyennes et ne pas prendre en compte les films avec un nombre de votants très faible car ce n'est pas représentatif. On impose une condition avec un nombre de votants qui doit être supérieur à 20, on regroupe les genres grâce à "groupby" et on calcule la moyenne par genres.
On a donc réarrangé notre DataFrame initiale, et on peut facilement calculer la moyenne et faire un graphique à l'aide des modules Matplotlib et Seaborn.

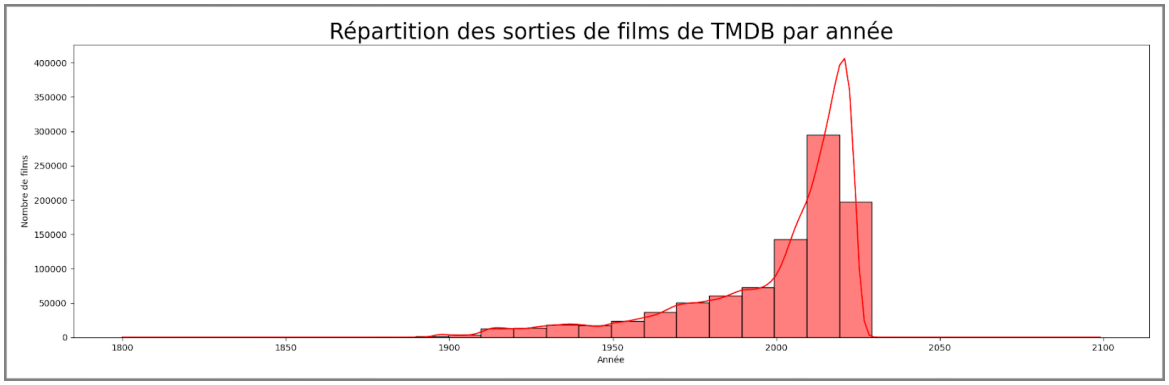


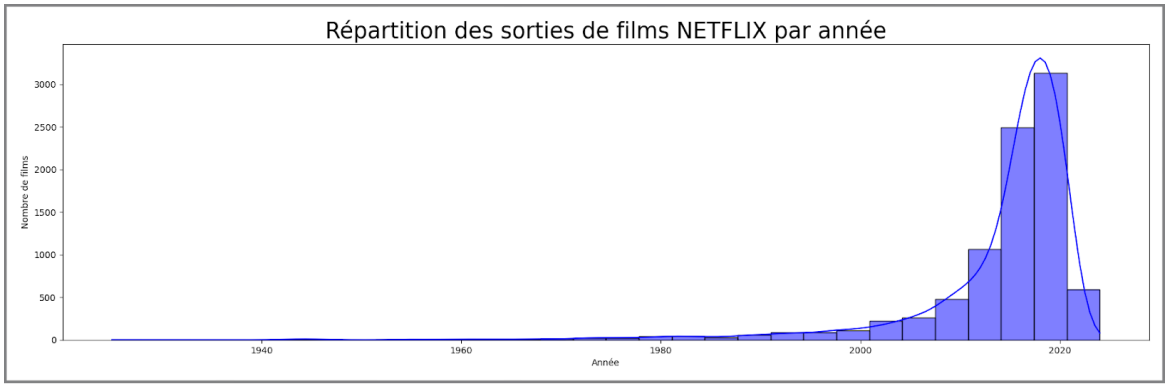
On constate que les notes moyennes pour chaque genre de films sont assez proches, l'écart-type de 3,9 est faible. La note maximale correspond aux films documentaires avec une moyenne de 6,9 et la note minimale correspond aux films d'horreurs avec une moyenne de 5,5.

Quelle est la répartition des années de sortie des films disponibles sur TMDB par rapport aux films ajoutés sur Netflix ? Représenter graphiquement

Pour répondre à cette question, on a récupéré les colonnes des années de publications de nos deux DataFrames. Cependant, pour TMDB il nous a fallu d'abord convertir les dates en "datetime", récupérer uniquement les années, et les transformer en entier avec la méthode "astype". Après ces

étapes, on peut construire les graphiques correspondant aux années de répartition de chacune de nos bases de données.





On remarque alors que les films de la base de données Netflix sont en général plus récents que ceux de TMDB. L'analyse des statistiques descriptives permet de valider notre intuition, avec une année moyenne de 2014 pour Netflix et une année moyenne de 1999 pour TMDB.

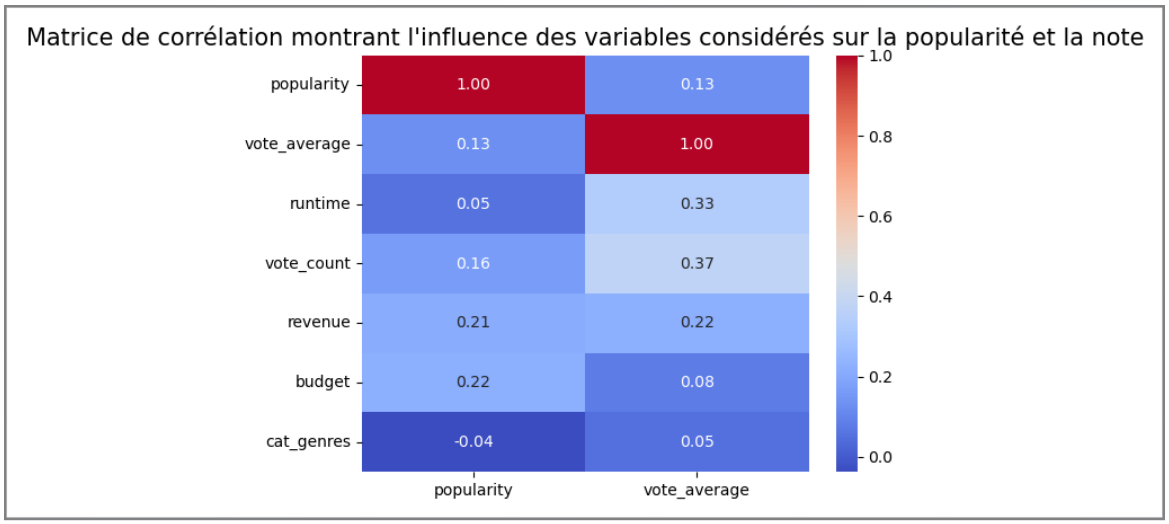
Cependant, il semble y avoir une erreur dans les données TMDB d'après les valeurs de maximum et de minimum. On va donc analyser les dates de TMDB. On remarque alors que TMDB contient en fait des films qui sont annoncés, mais pas encore sortie. D'ailleurs, la variable "status" précise s'ils sont en cours de production, planifié, en post-production... On peut donc les garder pour éventuellement avoir un programme qui permettrait d'afficher les prochaines sorties.

Pour le film dont la date de sortie est annoncée en 2099, il s'agit en fait d'un film tourné en 2015, mais dont l'auteur souhaitait le faire paraître en 2115, soit 100 après son tournage.. Pour les films sortis en avant 1900, il s'agit soit de films pionniers, soit des réadaptations dont la date de sortie de l'histoire était avant 1900.

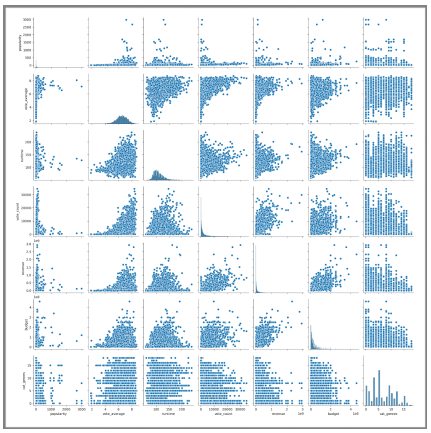
Analyse des facteurs influant sur la note : Analyse des facteurs influant sur la popularité : Quels critères influent sur la note d'un film ? Utiliser des graphiques pour illustrer ces relations, et mettre en avant les tendances importantes identifiées

Dans cette partie, nous allons uniquement nous intéresser à la base de données TMDB, du fait qu'elle contient plus de films et plus de variables numériques. Pour analyser les facteurs influant sur la popularité et la note de films, nous avons opté pour une représentation par une matrice de corrélation. Nous nous sommes alors intéressés aux variables que nous pensons susceptibles d'influencer la popularité ou la note d'un film, à savoir les variables sur les genres, la durée, l'année de publication, les recettes, le budget et le nombre de votants.
Le problème ici se trouve dans le fait que la catégorie "genres" est une variable qualitative, il nous fallait un moyen de la transformer en variable quantitative. Pour ce faire, nous avons utilisé la méthode "factorize" qui nous a permis de convertir la variable "genres" et de l'insérer dans une nouvelle colonne qui se nomme " cat_genres". Cependant, nous nous sommes rendu compte que

cette démarche n'était pas optimale puisque lorsque nous attribuons des chiffres à des genres, la machine comprend seulement que le chiffre attribué à ce genre est plus petit ou plus grand qu'un autre genre. C'est pour cela que dans la suite de notre travail, nous avons exclu cette variable (nous avons laissé la procédure afin que vous puissiez remarquer notre réflexion et nos efforts).
Par la suite, nous avons fait un nettoyage interne qui a consisté à la suppression de toutes les valeurs manquantes (NaN) de la colonne "release_date", puis de convertir cette variable en "datetime" et extraite seulement les années afin de pouvoir faire une heatmap qui représentera notre matrice de corrélation. Nous avons ensuite stocké ce changement dans une variable "Annee" afin de l'appliquer sur les colonnes de "release_date". Pour avoir des données de meilleures qualités, nous avons supprimé les films qui contenait des variables "revenue" et "budget" nulles puisqu'en effet, avoir un budget nul (même très faible) n'est pas cohérent puisque pour produire un film cela nécessite forcément un coût (on ne peut pas le déterminer). De même pour le revenu, si les recettes d'un film sont nulles cela n'est pas pertinent pour notre analyse. Concernant la durée du film, nous avons considéré uniquement les films qui ont une durée comprise entre 30 minutes et 5 heures.
Ensuite, nous avons créé la matrice de corrélation avec toutes les variables qui peuvent potentiellement influencer la popularité et la note des films, avec la méthode "heatmap".



On remarque que la popularité est corrélée positivement avec le nombre de votants, les recettes, le budget et la note moyenne est corrélée positivement à la durée, le budget, les recettes et le nombre de votants. On peut donc penser que ces variables impactent la popularité et la note moyenne des films.
Pour finir, nous avons tracé un pair plot, pour avoir un nuage de points des variables deux à deux et avoir une vision des relations entre elles.



MÉTHODOLOGIE

Dans cette partie, nous allons expliquer notre méthodologie, la description des étapes de développement, les outils et les bibliothèques utilisées et la répartition du travail dans le groupe afin d'atteindre notre objectif final qui est de construire un moteur de recommandation de films à partir de base de données.

Organisation et planification du projet

Dans un premier temps, nous avons analysé les consignes pour avoir une vision globale du devoir. Cela nous a permis d'avoir un premier regard, et d'anticiper les parties qui sont indépendantes les unes des autres et celles qui pourraient nous poser des problèmes. Cela nous a également permis de choisir le sujet que nous trouvons le plus intéressant. Par la suite, on a récupéré nos base de données, et on a fait une première observation à nue sans logiciel. Pour mener à bien notre projet, nous avons procédé à de nombreuses recherches et utilisé plusieurs outils. En effet, pour mieux comprendre ce que nous devions faire nous nous sommes documentés sur le net, nous avons regardé quelques vidéos sur Youtube par exemple afin de mieux cerner le sujet et ne pas se lancer dans une fosse.

Après ces étapes préalables, nous avons commencé dans l'ordre des consignes par la partie analyse et visualisation des données. Tout d'abord, parce que c'est la première étape de tous projets, on ne peut pas travailler sans connaître ses données et deuxièmement car c'est une partie qui est indépendante des deux autres.

Pour cette section, notre méthodologie était assez simple, nous avons essayé chacun de notre côté et ensuite nous nous sommes réunis pour mettre nos éléments en commun. D'après nous, il était important que nous nous approprions individuellement les données et que nous apprenions à faire une visualisation de données, ce qui est le b.a.-ba de notre domaine professionnel.

Ensuite, nous nous sommes attaqués à la suite du devoir, qui correspond à deux sections qui ne sont pas indépendantes. La première est de créer un programme qui nous permet de construire le moteur de recommandation et la seconde est de créer une interface graphique afin de l'utiliser. Pour cela on a donc décidé de scinder l'équipe en deux groupes, avec l'un qui s'occupe de se documenter sur le programme de recommandation et l'autre qui s'occupe de l'interface graphique à l'aide de la librairie Tkinter que l'on ne connaît et ne maîtrise absolument pas. Une fois la répartition effectuée entre les deux groupes, nous avons défini un calendrier précis avec des points de rencontre réguliers sur Discord pour échanger sur l'avancement de nos travaux respectifs. Cette approche nous a permis de rester coordonnés et de garantir que les deux parties (moteur de recommandation et interface graphique) puissent s'intégrer facilement à la fin.

Programme du moteur de recommandation

Pour développer le programme du moteur de recommandation, nous nous sommes avant tout penchés sur la base de données TMDB. Nous avons nettoyé les données en identifiant les variables avec beaucoup de valeurs manquantes et que l'on ne trouvait pas pertinentes pour notre moteur de recommandations à l'aide des bibliothèques pandas et numpy. On a également conditionné certaines variables que l' on a jugé utiles de redéfinir comme la durée des films que l'on à délimité comme expliqué précédemment. Par exemple, les lignes contenant des genres ou des mots-clés incomplets ont été ignorés lorsque cela était nécessaire. Nous avons aussi étalé les films ayant plusieurs genres par exemple en plusieurs lignes.

Nous avons identifié plusieurs critères afin de créer des filtres permettant à l'utilisateur d'obtenir une liste de films similaires selon les critères notés. Par exemple, si on sélectionne un genre, et une année on obtient via le moteur des films du même genre et de la même année. Suite à cela, nous avons décidé d'avoir un moteur de recommandation encore plus précis dans lequel on peut

spécifier également la durée du film, les pays de production , l'âge et qui nous affiche les 5 films les mieux notés correspondant à ces critères.

Par la suite, nous avons créé une fonction qui demande à l'utilisateur les titres des films qu'il a aimés. Cette fonction nettoie et standardise ce que les utilisateurs entrent (suppression des espaces superflus, conversion en minuscules, etc.) pour éviter des erreurs lors de la recherche dans la base de données.

Nous avons identifié un critère principal pour déterminer la similarité entre les films à savoir les mots clés. En effet, après plusieurs essais sur plusieurs variables, nous avons jugés que la variable "keywords" était la seule variable qui était réellement pertinente pour avoir un bon moteur de recommandation. On a également réfléchi à une méthode qui permettrait à l'utilisateur de choisir entre les doublons des films qu'il a saisis. Lorsqu'un utilisateur entre un film, s'il y a des doublons alors il devra entrer l'id du films dont il veut la recommandation. Le détail de toutes ces fonctions sera explicité dans nos scripts. Finalement, nous avons cherché une solution pour que notre programme nous affiche les films qui sont aussi sur Netflix parmi ceux recommandés.

Interface graphique

Le développement de l'interface graphique a été réalisé à l'aide de la bibliothèque Tkinter, qui nous a permis de concevoir une application interactive et intuitive pour l'utilisateur. L'objectif principal était de créer un moyen simple et efficace d'interagir avec notre moteur de recommandation de films. Tout d'abord, nous avons commencé par réfléchir à la structure de l'interface et aux fonctionnalités essentielles à intégrer, telles que :

- Une zone de saisie pour permettre à l'utilisateur de spécifier ses préférences (titre du film, genre, année, etc.).
- Des menus déroulants et des cases à cocher pour filtrer les films selon divers critères, comme la durée ou la disponibilité sur Netflix.
- Des boutons de validation qui déclenche l'exécution du moteur de recommandation et affiche les résultats.
- Une zone de sortie qui présente les films recommandés, avec des informations complémentaires telles que la note moyenne ou le synopsis, par exemple

Nous avons structuré l'application autour de plusieurs menus, correspondant chacun à des fonctionnalités spécifiques de notre moteur de recommandations. Chaque menu est lié à une fonction différente de notre programme. Voici les principaux menus que nous avons développés :

- Un menu de recherche par filtre simple qui permet à l'utilisateur de recevoir des recommandations basées sur ses préférences en matière de genres de films et d'année de publication.
- Un menu de suggestions personnalisées encore plus précis qui permet à l'utilisateur de recevoir des recommandations basées sur plus de critères et qui recommande des films jugés de meilleure qualité.
- Un menu de recommandation par titre dans lequel l'utilisateur peut saisir directement les titres de films qu'il a apprécié, il reçoit alors des recommandations similaires à ses films et s'ils sont disponibles sur Netflix.

Outils et documentations

Pour mener à bien notre devoir, nous nous y sommes vraiment pris pas à pas, à tatillon. Lorsque que nous avions une erreur ou un problème pour réaliser quelque chose, nous commencions par regarder notre cours, puis nous nous référions à la documentation présente sur internet, les blogs, les forums, les vidéos. Toutes nos sources utilisées seront disponibles dans la bibliographie à la fin

du devoir et dans notre notebook. Pour ce qu'il s'agit des librairies nous avons utilisés les librairies Numpy, Pandas pour traiter nos dataframes, la librairie OS pour créer des fichiers, les librairies Seaborn et Matplotlib afin de réaliser des graphiques et des analyse visuelles, et enfin la librairie Tkinter pour créer une interface graphique pour insérer notre programme.

Tests et validations

Pour garantir le bon fonctionnement de notre projet, nous avons appliqué une méthodologie rigoureuse de tests à chaque étape du développement. Nous avons conçu trois modules distincts pour structurer notre programme, un module dédié à l'importation et au nettoyage des données, un module regroupant toutes les fonctions de filtrage et de logique du moteur de recommandation et un module pour la gestion de l'interface graphique. Chaque module a été testé individuellement pour s'assurer qu'il remplissait correctement sa fonction. Par exemple, nous avons validé que les données issues de la base TMDB étaient bien téléchargées et correctement formatées. Les colonnes et variables nécessaires à notre programme ont été systématiquement vérifiées, et nous avons géré les cas d'anomalies comme les valeurs manquantes ou les genres de films mal définis. Nous avons aussi vérifié que les modules peuvent être appelés les uns par les autres et fonctionner de manière fluide. Cela inclut des tests sur l'emboîtement des fonctions, comme le passage des résultats des fonctions de filtrage directement à l'interface graphique.

Chaque fonction a été testée individuellement pour s'assurer qu'elle renvoyait les résultats attendus. Par exemple, nous avons vérifié que les filtres par genre, année ou durée fonctionnaient correctement. Nous avons également testé ces fonctions sur des cas particuliers, comme des films avec des informations incomplètes,des scénarios où l'utilisateur ferait des erreurs comme entrer un titre inexistant ou oublier de renseigner certains champs, afin d'ajouter des messages d'erreur clairs et utiles dans le but de nous assurer que le programme restait robuste face à des données imparfaites.

Chaque menu de l'interface a été testé indépendamment pour garantir que les interactions avec l'utilisateur se déroulaient sans erreurs. Par exemple, nous avons vérifié que les menus déroulants et les zones de saisie fonctionnaient correctement et que les filtres appliqués reflétaient bien les préférences de l'utilisateur.

Pour nous assurer de la qualité de notre moteur, nous avons analysé la pertinence des recommandations générées. Nous avons comparé les films recommandés avec ceux que nous jugions logiques en fonction des critères donnés par l'utilisateur. Ces tests ont permis d'affiner nos algorithmes et de privilégier les critères les plus pertinents, comme les mots-clés ou les genres.

Enfin, nous avons testé la performance et la vitesse d'exécution du programme. En effet, il était important que notre programme ne soit pas long à mettre en route, et ne retranscrive pas des dysfonctionnement au niveau de l'interface.

CONCEPTION ET IMPLÉMENTATION DU PROJET

Conception :

Le programme 'Searchfilms' doit permettre à un utilisateur d'effectuer des recherches selon certains critères de filtres, comme le genre, la date, la durée, ou encore les langues parlées. Aussi et surtout, il doit pouvoir donner des recommandations simples à partir de titres saisis par l'utilisateur.

Utilisation des deux bases de données TMDB et Netflix :

Définissons d'abord l'utilité que l'on aura des deux bases de données TMDB et Netflix.

La base de données TMDB nous servira pour effectuer les recherches. Elle comprend un très grand nombre de films. Même une fois nettoyé et rendu utilisable pour les différentes intégrations fonctionnelles, nous nous servirons de manière effective d'environ 46000 films.

La base de données Netflix nous servira quant à elle uniquement à donner les disponibilités des titres sur la plateforme, dans la partie recommandation.

L'idée est donc simple : l'utilisateur recherche des films dans la base de données TMDB pour les recherches. Pour les recommandations, le programme devra lui retourner les films recommandés et recherchés ceux disponibles sur la plateforme Netflix en allant chercher ces derniers dans la base de données Netflix.

Critères de choix pour les recommandations :

Quant il s'agit de recommander des films à partir d'une recherche, on pourrait tout de suite penser à l'utilisation de modèles de Machine Learning (que nous ne ferons pas ici).

On pourrait aussi être tenté d'utiliser des caractéristiques tels que le genre des films, les langues originales, les pays de productions, ou encore les dates etc…

Cependant, l'utilisation de ces critères comme premiers choix pour effectuer des recommandations entre plusieurs films semble faible : deux films du genre action peuvent très bien être opposés sur le plan de l'histoire. De même, deux films dont la langue originale est commune ont une forte chance d'être extrêmement différents.

Quels critères faudrait-il dès lors choisir ?

Deux films ayant une histoire similaire, sont trivialement similaires. Dans notre base de données TMDB, vis-à-vis de l'histoire, nous pouvons récupérer les synopsis et les mots-clefs. Les meilleures recommandations possibles se feraient sûrement à l'aide de la comparaison des synopsis entre chaque film. Or, ceci peut être extrêmement laborieux sans l'aide du Machine Learning. Les mots clefs sont quant à eux moins précis, mais apportent tout de même une information pertinente sur la similarité entre plusieurs films. De plus, l'implémentation d'une telle fonctionnalité de recommandation basée sur les mots-clefs uniquement semble plutôt simple : il suffira de voir si chaque mot-clef de chaque film recherché est présent dans l'ensemble des mots-clefs de chaque autre film dans la base de données.

On choisira donc de baser nos recommandations sur le critère unique des mots-clefs, que nous tenterons d'exploiter au maximum.

Pour concevoir un tel programme, dont l'interaction homme/machine doit être facile, on procède ici en 3 étapes, comme suggéré dans les consignes.

Imagination et conception des fonctions brutes utilisés :

On doit avoir 3 fonctions principales :

A. Une fonction de recherche par filtre simple. On réalise deux sous fonctions pour cette fonctionnalité :

- une fonction « df_filtre_genre » qui filtrera la base de donnée TMDB par les genres sélectionné par l'utilisateur et retournera un dataframe filtré selon les meilleures notes des films, et si les notes sont égales, ca sera celui qui possède le plus de nombre de votes ;

- une fonction « df_filtre_annee » qui filtrera le dataframe obtenu avec df_filtre_genre par les dates (années) sélectionnés, qui retournera un dataframe contenant les films filtrés selon tous ces critères, qui servira à l'affichage utilisateur

B. Une fonction de suggestions personnalisées que l'on nommera « infos_perso » qui cette fois à partir de plusieurs filtres entrés par l'utilisateur, genres, dates, durée, pays de production, catégorie adulte, filtrera la base de donnée TMDB selon ces critères, et sur cette base, devra afficher les 5 meilleurs résultats sur des critères de notes et de popularité, et si jamais les notes égales, donner le film qui a reçu le plus de vote. Cette fonction devra retourner un dataframe filtré selon ces critères.

C. Une fonction de recommandation basée sur les mot-clefs que l'on nommera « reco_titres ». Elle devra prendre en entrée une liste de titres de films recherchés, puis devra sur cette base comparer chaque mot-clef de chaque films avec les listes de tous les mots-clefs des autres films dans la base de données. Elle devra retourner un dataframe incluant les informations sur les titres recherchés, un dataframe de recommandation, un dataframe de disponibilité dans la base de données Netflix.

Interaction Homme/machine avec input :

Une première intégration de ces fonctions pour avoir une interaction homme/machine pourrait d'abord se faire à l'aide d'input natif de python.

Il suffira alors de rajouter ces inputs à chaque fonction pour que l'utilisateur puisse entrer ces filtres/titres, puis lui afficher dans la console python ou le terminal de l'os de l'utilisateur les résultats et les recommandations.

Interaction Homme/machine avec une interface :

Pour réaliser une interface qui permette une bonne intégration des fonctions de filtres et de recommandations, on devra pour cela modifier les fonctions elle-même pour qu'elle puisse bien s'intégrer à l'interface. Inversement, l'interface devra avoir des fonctions qui devront interagir avec les fonctions de filtres et de recommandations.

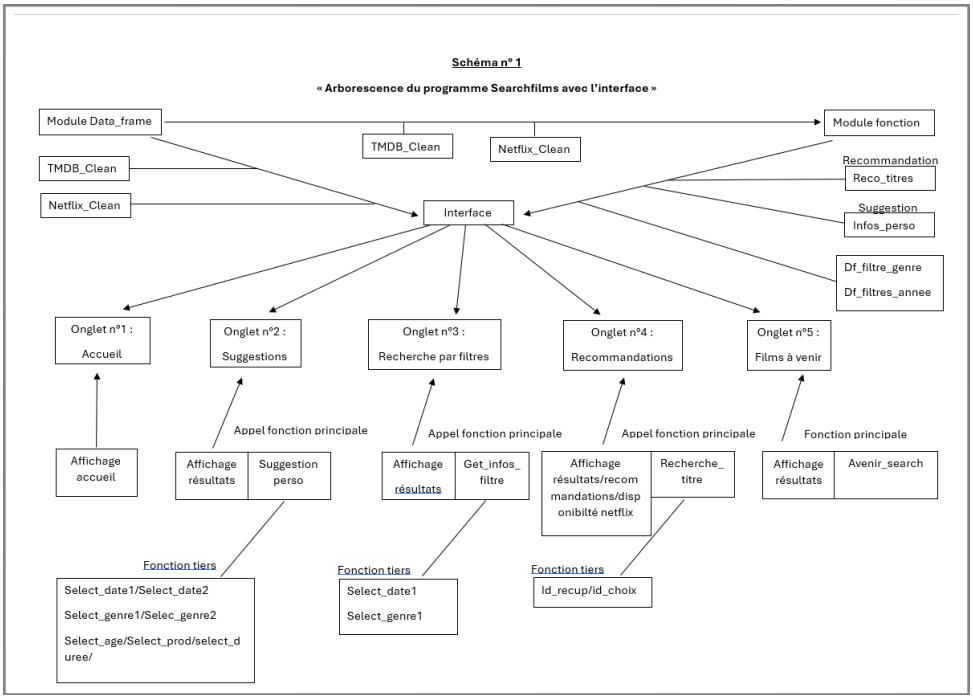
Pour l'interface, on décidera d'avoir 5 onglets :

- Un onglet d'accueil simple
- Un onglet de suggestions personnalisées
- Un onglet de recherche par filtres simples
- Un onglet de recommandations

- Un onglet des films à venir

On utilisera des boutons pour valider chaque choix, qui sauvegardent les entrées utilisateurs dans des variables externes, et qui seront mobilisés dans le cadre des fonctions internes à l'interface, en communication avec le module 'Fonction'.

Voici un schéma d'arborescence du programme comprenant l'interface :



Implémentation : Fonctionnalités développées, exemples de code pertinents et explications des choix effectués

La première étape de l'implémentation du projet a consisté à établir les différentes parties à réaliser. Pour cela, nous avons décidé de suivre la division proposée par le sujet : une première partie consacrée au nettoyage des données, une deuxième partie où nous avons développé les fonctions nécessaires pour répondre aux différentes problématiques, et enfin une dernière partie dédiée à la création de l'interface utilisateur. Nous reviendrons dans cette section sur chacune de ces parties avec une explication claire et détaillée sur chacune des fonctionnalités et des choix effectués.

Tout d'abord, nous avons décidé de travailler sur trois modules distincts :

- Le module "Data_frame" : qui renvoie à la première partie
- Le module "Fonctions": qui renvoie à la deuxième partie
- Le module "Interface" : qui renvoie à la troisième partie

Dans chaque module la première étape a été celle de l'importations des librairies nécessaires pour la réalisations des différentes étapes:

- Pandas : Pour la manipulation des données
- Tkinter : Pour la partie interface
- NumPy : Pour la Manipulation de tableaux
- Os : pour interagir avec le système d'exploitation lorsque cela a été nécessaire

Retour bref sur le module Data_Frame

Ce premier module très simple consiste à importer et nettoyer les données afin de les rendre utilisables par la suite. Il est composé de 3 fonctions :

- La première fonction, **TMDB_clean()**, reprend les étapes de la première partie. Elle prend comme input le DataFrame brut (tel qu'il est téléchargé) et donne comme output une version nettoyée selon les conditions spécifiées précédemment.

Nous avons notamment utilisé la méthode **.dropna()** de pandas pour supprimer les valeurs manquantes (NaN) concernant les variables indispensables à la recommandation des films, la méthode **.to_datetime()** pour convertir la colonne "release_date" en un format date, afin de faciliter les manipulations ultérieures.

Comme notre projet se basera uniquement sur l'année de sortie des films, et non sur le mois et le jour, nous avons décidé de conserver uniquement l'année, en utilisant la méthode **.strftime('%Y').**

- La fonction **Netflix_clean()** prend comme input le dataframe brute et renvoie un dataframe propre sans les colonnes inutiles.

- Enfin la fonction **prod_countries(data)**, qui renvoie comme précisé dans le script la liste des pays de production des films de la base de données

Le module Fonctions

Le module “fonctions” est le cœur de notre projet. Nous avons établi plusieurs fonctions essentielles qui répondent aux différentes problématiques.

Nous évoquerons dans cette partie les fonctions dans leur forme finale, à savoir sans les input() qui étaient présent dans la construction préalable des fonctions.

La recherche par filtre simple

Les deux premières fonctions : df_filtre_genre(), df_filtre_annee() permettent de filtrer le dataframe afin de ne garder que les films les plus pertinents selon les entrées de l’utilisateur. Afin de laisser la liberté à l’utilisateur de ne pas entrer de valeurs, nous avons mis comme valeur par défaut à l’argument de la fonction : None.

Nous avons également fait en sorte de rendre la saisie de l’utilisateur insensible à la casse grâce à “case = False”.

```
if genres is not None: #Si l'utilisateur a bien entré des genres
    df = data_tmdb.copy() #On appelle df la DataFrame pour plus de facilité dans la fonciton
    for genre in genres:
        df = df[df["genres"].str.contains(genre, case=False)]
```

Ainsi le but est de parcourir chaque genre dans la liste des genres donnés par l’utilisateur et de ne garder que les films dans le dataframe qui dans la colonne “genres” ont pour genre ce qui a été entré par l’utilisateur grâce à la méthode .contains().

La fonction nous retournera donc une Data frame contenant les films filtrés, qu’on trie par ordre décroissant selon la note avec la méthode .sort_values à laquelle on met comme argument d’abord “vote_average” puis “popularity” (si la note est la même entre deux films.

Le principe a été le même dans la fonction df_filtre_annee() dans laquelle nous avons filtré les films afin de ne garder que les films dont l’année de sortie correspond à l'entrée de l’utilisateur.

```
#Filtre par année
#La fonction prend comme argument la DataFrame et l'année qu'on initialise à None pour permettre à l'utilisateur de ne rien entrer
def df_filtre_annee(data, year=None):
    if year is not None:   #Dans le cas ou l'utilisateur a bien entré une valeur
        df=data.copy()
        df = df[df["release_date"].isin(year)] #On cherche grâce à la méthode isin qui retourne des booléens pour chaque film selon si la valeur
        #entré par l'utilisateur se trouve dans la colonne release_date
        return df
    else :
        return data #Dans le cas où l'utilisateur n'a pas entré de valeur on retourne simplement la DataFrame
```

Les suggestions personnalisés

La fonction **infos_perso()** est un “développement” des fonctions que nous avons évoqué dans la section précédente. Elle prend comme argument :

- Les genres appréciés par l’utilisateur
- Les années de sortie souhaitées
- Les pays de production souhaités
- La variable binaire adult
- Et enfin le “runtime” souhaité

Le principe a été le même que dans les sections précédentes avec des améliorations permettant de peaufiner la recommandation. Afin de recommander les meilleurs films possibles nous avons décidé de classer dans les premiers rangs de nos recommandations, les films ayant le plus de genre/pays en communs avec les entrées de l’utilisateur.

Pour cela nous avons créé des fonctions “compteur” dans notre fonction principale qu’on a ensuite appliqué grâce à la méthode .apply() aux lignes de la colonne “genres” et “production countries”. Dans deux nouvelles colonnes nous avons donc stocké le résultat de ces fonctions, à savoir, le nombre de genres/pays en communs. Le classement dans la recommandation a ensuite été basé sur ces variables, car nous avons trié avec la méthode .sort_values() le dataframe obtenu, suite aux différentes étapes de filtrage, par ces variables.

```
    #On crée une fonction qui va compter le nombre de genres en commun (voir utilisateur plus loin)
    def compteur_genres(film_genres):
        compteur = 0   #On initialise la variable compteur à 0
        for genre in genre_ent: #Pour chaque genre dans la liste des genres entrés par l'utilisateur
            if genre in str(film_genres): #Si le genre en question est compris dans les genres du film de la ligne du dataframe,
                compteur += 1 #On ajoute 1 à compteur.
        return compteur

    #On fait la même chose pour compter le nombre de pays en commun
    def compteur_pays(film_pays):
        compteur = 0 #On initialise le compteur à 0.
        for pays in pays_ent: #Pour chaque pays dans la liste des pays entrés
            if pays in str(film_pays): #Si le pays est contenu dans les pays de production du film de la ligne du dataframe,
                compteur += 1 #Le compteur augmente de 1
        return compteur
```

Pour la variable “runtime” , après plusieurs essais, nous nous sommes aperçus que le nombre de recommandations était très faible lorsque la recherche se basait uniquement sur la durée exacte entrée par l’utilisateur. Nous avons donc décidé de filtrer le dataframe sur un intervalle de durée à +-30 min de la durée entrée par l’utilisateur.

```
    #La question de la durée du film.
    if runtime_ent:
        df = df[(df["runtime"] >= runtime_ent - 30) & (df["runtime"] <= runtime_ent + 30)]
        #Si l'utilisateur entre bien une valeur pour le runtime. Nous avons décidé d'afficher les films dans un intervalle de 30 min
    else :
        df = df
```

Enfin nous avons décidé d’afficher le top 5 des films recommandés grâce à la méthode .head() .

La recherche par titre

La fonction reco_titre() est une sorte d'étape finale de la construction de notre moteur de recherche et de recommandations. Elle offre la meilleure recommandation possible basée sur les "keywords" comme nous l'avons vu précédemment.

Elle permet en utilisant une comparaison une à une des mots-clefs de chaque titres recherchés avec chaque liste de chaque titre du dataframe de travail, de sélectionner les films pour lesquelles au moins 3 mots-clefs sont en communs dans un premier temps, puis au moins 1 dans un deuxième temps si la recommandation avait échoué pour certains titres (moins de 3 mots-clefs en communs). On obtient un dataframe de recommandation.

On met ensuite le nombre de mots-clefs communs pour chaque titres recherchés à chaque titres recommandés. Voici un exemple de codes pertinents.

```
"""
4.RECOMMANDATIONS PAR MOT-CLEFS
"""
#Dictionnaire qui à chaque titre lui attribue un dictionnaire contenant ses keywords en liste
titres_dico = {f"{titres[i]}" : list(df[df["title"] == f"{titres[i]}"]["keywords"].str.split(", ").explode().unique()) for i in range(len(titres))}

#Compteur des mot_clefs communs avec chaque titres du dataframe de base
count = 0
#Liste des indices des pour les films ayant en communs plus de 3 mot_clefs avec la liste globale des mot-clefs
indices = []
#Liste du nombre de mot-clefs en communs pour chaque films du dataframe de base
count_val =[]
#L'ensemble des mots-clefs du dataframe de base
key = df["keywords"].str.split(", ")

for i in range(key.shape[0]):
    for j in range(len(keywords)):
        if keywords[j] in list(key.iloc[i]):
            count += 1
        else:
            pass
    #Si il y a au moins 3 mots_clefs en communs en ajoute l'indice du film en question
    #On ajoute aussi le nombre exact de mots-clefs en communs
    if count >= 3 :
        indices.append(i)
        count_val.append(count)
        count = 0
    else :
        count = 0

#Le dataframe de recommandation
df_reco = df.iloc[indices]
```

Par la suite, on met en place un système qui sauvegarde dans un dictionnaire dédié les similitudes entre les titres recherchés et les titres recommandés de manière à informer l'utilisateur, pour chaque titre recommandé, à quelle films recherché est-il associé, par une méthode du maximum de mots-clefs en communs (voir le script du module Fonction pour plus de détails sur ce point).

Enfin, on recherche les disponibilités des films recommandés et recherchés dans la base de données Netflix. Voici un extrait du code qui permet cela :

```
"""
5.DISPONIBILITEES SUR NETFLIX
"""
#On prends les titres des films recommandés que l'on convertit en liste
comparaison_title = df_reco["title"].tolist()
#On ajoute d'abord nos titres recherchés
for i in range(len(titres_bruts)):
    comparaison_title.append(titres_bruts[i])
#On compare enfin avec la base de donnée netflix
df_dispo_netflix = df_netflix[df_netflix["title"].isin(comparaison_title)]


################################################

return dico_comparaison,df_reco,df_titres,df_dispo_netflix,elem
```

On inclura aussi une fonctionnalité de gestion des doublons et de gestion des majuscules/ minuscules dans les inputs utilisateurs.

Le module Interface

Nous utilisons ici Tkinter. Nous avons besoin de plusieurs objets liés à Tkinter pour y intégrer nos fonctions.

D'abord, nous utilisons les widgets Label, qui permettent d'écrire du texte dans un endroit de l'interface. Nous utilisons les widgets Listbox, qui permettent d'insérer dans une fenêtre des barres de sélection avec plusieurs choix possibles, dans lesquels il est d'ailleurs possible de faire plusieurs choix simultanés par l'argument MULTIPLE.

Pour les saisis simples dans une barre de recherche, ce sont les widgets Entry. Pour sauvegarder les sélections des Entry ou des Listbox, nous utilisons les widgets Button qui permettent de communiquer avec une fonction tiers dans Tkinter. Nous utilisons également les widgets Text qui permettent d'afficher les résultats.

Pour calibrer tous ces éléments sur chaque fenêtre, nous utilisons la méthode grid, qui prend en compte la fenêtre comme un tableau à 2 dimensions. On y indique la ligne et la colonne où l'on souhaite placer l'élément. On peut aussi modifier les paramètres de taille avec heights et width, ou encore "coller" les automatiquement les éléments entre eux avec la valeur du paramètre sticky qui prend "nsew".

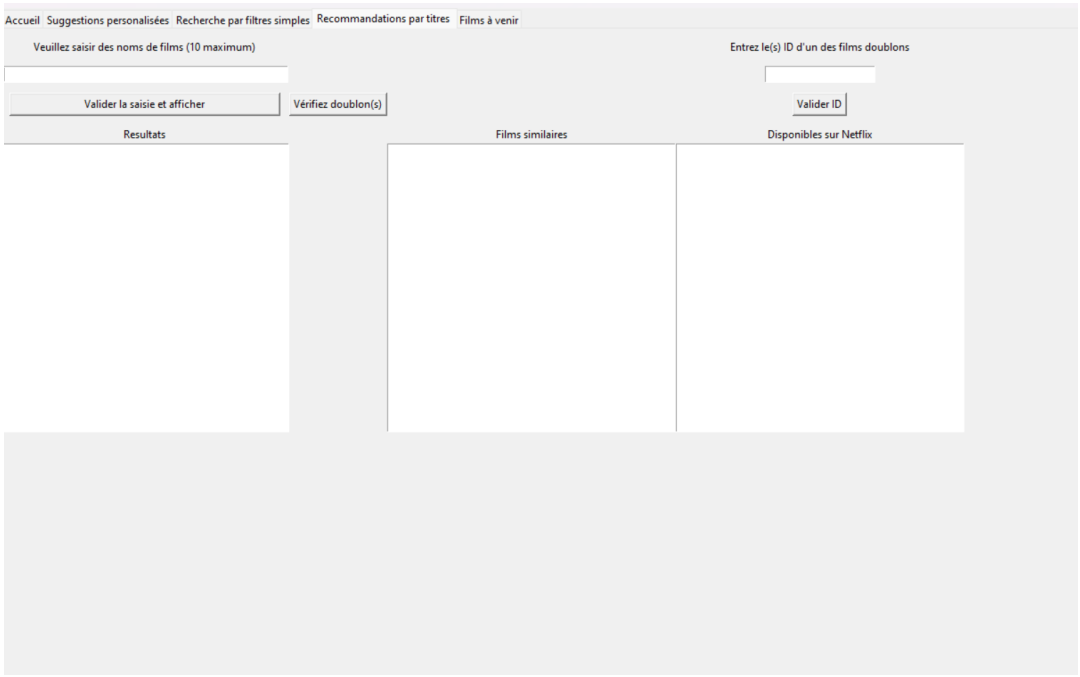
Pour nommer et travailler dans des fenêtres différentes, nous utilisons l'élément Notebook de la classe ttk, qui permet de gérer les fenêtres.

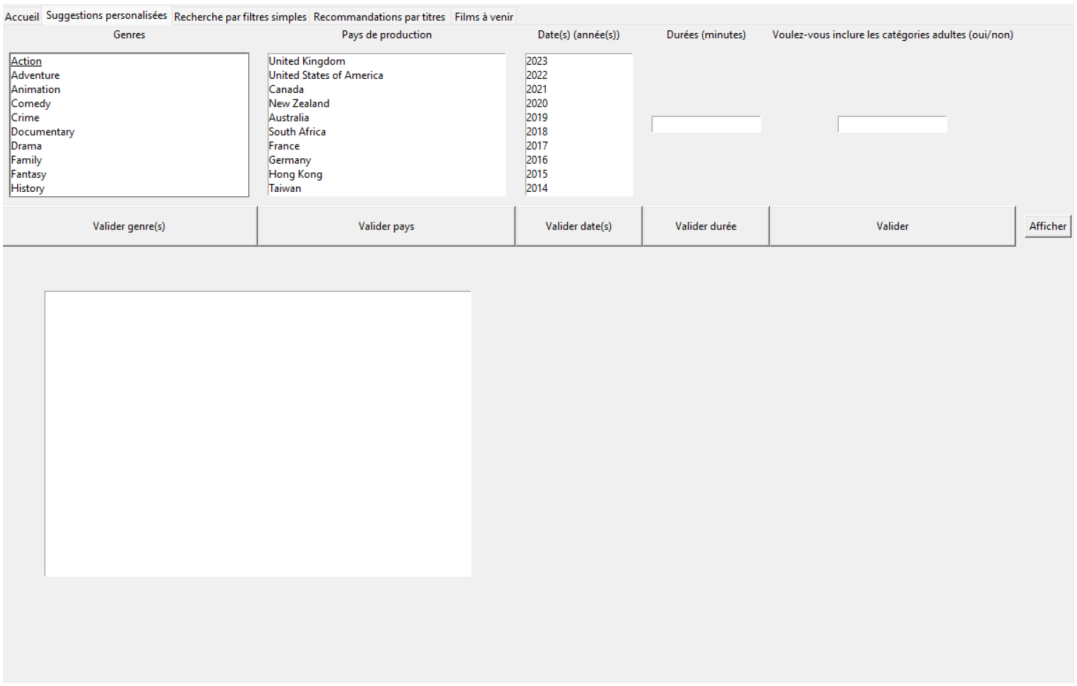
Les boutons doivent communiquer avec des fonctions tiers qui servent à récupérer les sélections par la méthode get, et les sauvegarder dans des listes.

Les fonctions tiers servent soit à la sauvegarde des sélections, soit à la communication avec le module Fontions pour ensuite afficher les résultats obtenus dans les widgets Text, en actualisant ces derniers par la méthode config.

Nous vous invitons à consulter le script "Interface.py" pour plus de détails, en consultant les commentaires liés à chaque élément.

Voilà quelques capture d'écrans de l'interface :





Fonctionnalités supplémentaires développées :

Gestion des doublons dans la partie recommandation :

Pour gérer les doublons, nous implémentons dans le module Fonctions, dans la fonction reco_titres, une gestion des doublons par la saisie manuelle par l'utilisateur dans l'interface ou un input dans la console python ou le terminal. Deux listes vont servir à la communication entre l'interface et le module Fonctions : ID et elem.

La liste elem est la liste des éléments doublons. Elle servira dans la fonction tiers de l'interface recherche_titre, par l'intermédiaire de la fonction id_choix, qui va récupérer ces doublons par

l'appel de reco_titres, et afficher à l'utilisateur la listes des doublons et leurs id associés pour l'inviter à saisir dans la barre de recherche associée dans l'onglet recommandations par titres.

La liste ID va servir à récupérer les id choisis par l'utilisateur par la fonction tiers id_recup, qui permettra ensuite de mettre cette liste dans la fonction reco_titres par la fonction tiers recherche titre, pour enfin afficher les résultats de recommandations et de disponibilités dans Netflix.

Gestion des majuscules et des minuscules :

Dans la fonction reco_titre du module Fonctions, nous avons décidé de gérer ce problème pour que l'utilisateur puisse entrer ses titres préférés avec n'importe quel mélange de minuscules et de majuscules.

Pour cela, on convertit systématiquement chaque string dans la liste des titres choisis en minuscules. on créer ensuite une nouvelle colonne dans le dataframe de travail mettant les titres en minuscules. On récupère le bon caractère des titres en comparant la liste des titres saisis minuscules avec la colonne des titres minuscules du dataframe par la méthode isin de Pandas.

Affichage des films à venir :

Dans l'onglet des films à venir dans l'interface, on affiche simplement par l'utilisation du bouton associé les films à venir à partir de la date d'aujourd'hui, en utilisant le module datetime.

RETOUR D'EXPÉRIENCES ET PERSPECTIVES D'AMÉLIORATION

Dans cette partie nous ferons un retour individuel et personnel sur le projet par rapport à ce que nous avons pris, les difficultés que nous avons rencontré, ce que nous aurions pu améliorer.

Ibrahim Diabira : Ce projet m'a permis d'apprendre énormément, autant sur le plan technique que personnel. J'ai découvert l'importance de structurer un programme en modules : c'est vraiment ce qui a simplifié notre travail et rendu notre code plus clair et facile à gérer. J'ai appris à faire preuve de persévérance et à ne pas abandonner au moindre problème, à faire des recherches et à me documenter. J'ai aussi appris à quel point les tests sont indispensables : tester chaque fonction, l'intégration des modules et les menus de l'interface m'a permis de comprendre que c'est en vérifiant chaque détail qu'on peut éviter des bugs majeurs. J'ai également gagné en compétences sur des outils que je ne maîtrisais pas avant, comme Tkinter, et cela m'a donné une vision plus complète de la création d'une application. J'ai également appris à travailler et collaborer dans une équipe, ce n'est pas toujours évident de concorder ses pensées, ses idées et ses résultats.
La plus grande difficulté pour moi a été de me familiariser avec Tkinter, un outil que je ne connaissais absolument pas du tout, et il est clair que je ne le maîtrise toujours pas. J'ai souvent eu l'impression de tourner en rond, surtout au début, et chaque petite erreur semblait être un obstacle immense. Il y a aussi eu des moments où le projet avançait plus lentement que prévu, et c'était frustrant. Parfois, j'avais l'impression de ne pas avoir les compétences nécessaires pour résoudre certains problèmes, ce qui a été source de stress. Mais ces moments m'ont aussi appris à chercher des solutions de manière autonome et à m'appuyer sur mes coéquipiers quand j'avais besoin d'aide.

Si je devais améliorer quelque chose, ce serait sans aucun doute la préparation en amont. Avec un peu plus de temps consacré à me former sur les outils et à planifier les étapes, j'aurais pu éviter pas mal de stress et avancer de manière plus efficace. J'aurais aussi aimé mieux gérer mon temps , j'ai mettais parfois trop d'énergie sur des détails techniques, alors que d'autres parties du projet auraient peut-être nécessité plus d'attention.

Marwan Hamzaoui : Ce projet m'a été très utile en termes d'apprentissage du code Python notamment à beaucoup manipuler les library Pandas, Seaborn, Numpy, Matplotlib grâce à toutes les recherches et documentations que j'ai pu réaliser. Ce travail m'a également permis de bien organiser mon code notamment à travers les multiples interactions entre les codes grâce aux différents modules mais aussi de créer un moteur de recherche et une interface graphique grâce à Tkinter. J'ai également appris à bien collaborer et à communiquer pour établir le projet de manière efficace.

Les difficultés que j'ai rencontrées étaient principalement la mise en force de l'interface graphique de Tkinter et l'aboutissement de des fonctions qui fonctionne correctement. En collaborant avec mes camarades, on a donc pu tous s'aider et établir le projet petit à petit en se complétant mutuellement. Si le projet était a refaire, j'aurai investit plus de temps dans l'exploration et le nettoyage des données afin d'anticiper les problèmes liés aux valeurs manquantes.

Guillaume Roustan : Ce projet m'aura permis de m'améliorer dans le codage en python : surtout dans la gestion d'interaction entre les différents modules, et la gestion des variables globales entre ces modules. J'ai pris plaisir à rechercher les bugs que nous avons rencontrés, les corrigés un à un, et de modifier le code en fonction.

Dans ce projet, j'ai été mis en difficulté par la gestion du temps, notamment dans la rédaction du rapport. Je suis aussi quelque peu frustré de ne pas avoir eu le temps de mettre en place un modèle de machine learning pertinent pour la partie recommandation de films. Certes les mots-clefs sont

assez pertinents, mais les erreurs se rencontrent parfois dans les recommandations, même (vous allez le voir) elles restent rares.

Tigran Gyurjyan :

Les difficultés rencontrées :

- La qualité des données : Faute d'avoir assez bien étudié le contenu des données au départ (plusieurs films avec le même nom par exemple) nous nous sommes rendu compte au fur et à mesure, lors de l'établissement des fonctions, qu'elles n'étaient pas compatibles avec les données. Il a donc fallu réajuster nos fonctions afin de les adapter et construire un moteur de recommandation pertinent.

- La gestion du temps: C'est selon moi la plus grande difficulté car en cherchant à obtenir le meilleur résultat possible, on s'est aperçu au fur et à mesure qu'on avait des erreurs et il fallait parfois recommencer à 0.

Ce que j'aurai fait différemment :

- Prendre le temps de bien étudier les données avant de commencer à développer les fonctions

BIBLIOGRAPHIE

https://www.data-bird.co/blog/data-cleaning-methodes,

https://www.youtube.com/watch?v=H_pVrg1_49w&t=4s : Tout d'abord nous nous sommes informés sur le nettoyage de données, la manipulation des donnés mais aussi la visualisation qui est l'étape la plus importante du travail.

***https://docs.python.org/3/library/stdtypes.html#str.split* :** Sur ce site, on a trouvé la méthode « .str.split(' , ') qui nous a permis de nous retourner une liste de chaîne de caractère séparé par des virgules.

***https://www.datacamp.com/tutorial/pandas-explode* :** On a utilisé la méthode « .explode() » afin d'éclater la liste de la colonne « genres », qui a été convertie en liste de chaine de caractère (méthode ci- dessus), pour avoir uniquement un genre par ligne pour tous les films.

***https://seaborn.pydata.org/generated/seaborn.barplot.html* :** Nous nous sommes renseignés sur ce site afin d'établir les différents graphiques tels que les histogrammes notamment avec l'estimation de la densité par noyau qui permet de lisser les données et d'avoir un meilleur aperçu de la répartition. Mais aussi pour représenter la matrice de corrélation avec la méthode « .heatmap »**.**

***https://pandas.pydata.org/docs/reference/api/pandas.factorize.html* :** Initialement nous avons pensé à associer à chaque genres un chiffre en grâce à de l'encodage afin d'avoir des valeurs numériques pour établir la matrice de corrélation, grâce à la méthode « .factorize ».

***https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.corr.html* :** Utilisation de la méthode « .corr » pour créer la matrice de corrélation.

***https://pandas.pydata.org/docs/reference/api/pandas.Series.str.contains.html* :** La méthode « .str.contains() » nous permet de vérifier si une chaine de caractère inséré entre parenthèse est bien présent dans la colonne que nous considérons.

***https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.isin.html* :** La méthode « .isin() » nous permet également de vérifier si un élément insérer entre parenthèse figure bien dans la colonne considéré mais cette méthode est plutôt utilisé pour trouver des valeurs précises.

***https://pandas.pydata.org/docs/reference/api/pandas.Series.to_list.html* :** Lorsqu'on veut mettre un DataFrame sous forme de liste, il faut utiliser la méthode « .tolist() ».

***https://www.youtube.com/watch?v=N4M4W7JPOL4*,**

https://docs.python.org/fr/3/library/tkinter.html,

https://python.doctor/page-tkinter-interface-graphique-python-tutoriel ,

Pour créer l'interface graphique, nous nous sommes beaucoup documentés notamment avec des vidéos, des sites qui nous ont permis de comprendre comment fonctionne Tkinter et de réaliser l'interface graphique.

ANNEXES

Instructions pour exécuter le programme :

Module « Data_frame » :

Dans un premier temps, il est nécessaire d'ouvrir le module Data_frame, qui contient les dataframes issus de nos deux datasets : TMDB et Netflix.

1. Nettoyage du DataFrame TMDB :

Il faut commencer par exécuter la première fonction de ce module. Celle-ci nettoie le DataFrame TMDb en supprimant les valeurs manquantes des colonnes considérées.Puis une condition est également appliquée pour filtrer les films « atypiques ». Ce traitement est essentiel pour garantir la qualité des données utilisées dans le moteur de recommandation de films, notamment en filtrant les films en fonction de certaines caractéristiques.

2. Nettoyage du DataFrame Netflix :

La deuxième fonction se charge de nettoyer le DataFrame Netflix en éliminant les colonnes contenant des valeurs manquantes. Ce nettoyage permet de se concentrer sur les informations pertinentes et dont nous avons besoin.

3. Traitement des pays de production :

La dernière fonction de ce module traite la colonne des pays de production. Les valeurs manquantes sont supprimées, et on applique la méthode qui permet d'avoir uniquement un pays de production par lignes.

Ce module est un pilier puisqu'il sera importé dans les autres modules qui nous permettront de faire des fonctions qui filtrent les films afin de créer le moteur de recherche et de recommandation tout en ayant des données de films qui ne sont pas « aberrantes ».

Module « Fonctions » :

Ce module contient toutes les fonctions nécessaires à la réalisation de la partie 2 du projet, qui est la création du moteur de recherche/recommandations.

1. Importation des données nettoyées :

Commencez par importer les DataFrames TMDb et Netflix nettoyés depuis le module Data_frame.

2. Application des filtres :

Cet section regroupe plusieurs fonctions de filtrage, telles que :

- Filtrage par genres : permet de sélectionner les films en fonction de leur genre.
- Filtrage par années de sortie : permet de filtrer les films en fonction de l'année indiquée.

Avec ces fonctions, on recommande des films par rapport à leurs notes et leur popularité en affichant seulement quelques caractéristiques des films.

3. Collecte des préférences utilisateur :

Il y a une fonction qui permet de récupérer les préférences de l'utilisateur en lui demandant des informations sur les genres qu'il apprécie, la durée du film… En récupérant toutes ces informations et en les combinant dans une fonction, on suggère à l'utilisateur des films en prenant en compte ses préférences.

4. Recommandations basées sur les mots-clés :

Enfin, il y a la dernière fonction qui permet de donner des recommandations similaires aux films que l'utilisateur a saisis, qui est basée sur les mots clés. Par la suite, elle met en évidence les films suggérés qui sont disponibles sur Netflix.

Toutes ces fonctions présentes dans ce module vont être ajouté dans l'interface graphique Tkinter afin d'avoir un rendu compréhensible et facile à l'utilisation

Module « Interface » :

Il faut lancer tout le module de l'interface graphique qui comporte les fonctions du module « Fonctions ». Il y aura donc une page qui s'ouvrira avec des onglets, il suffira d'insérer ce que vous souhaitez et vous trouverez les films suggérés qui seront très proche de vos préférences.

Il serait préférable d'ouvrir l'interface graphique depuis le terminal de l'ordinateur

Guide d'utilisation pour l'interface graphique :

Lorsque vous souhaitez avoir des recommandations de films en passant par l'onget "Suggestions personnalisées" il faut sélectionner les genres que vous désirez puis appuyer sur "valider genre(s)" et faire de même pour les "pays de production" et les "date(s)". Pour la durée et la catégorie du film il faut insérer directement dans la case également valider les critères. Si vous n'indiquez aucun genre il faut quand même valider genre(s). Cela s'applique pour toutes les caractéristiques.

Pour l'onglet "Recherche par filtres simples", la procédure est identique à celle présentée pour l'onglet "Suggestions personnalisées" (ci-dessus).

Pour l'onglet "Recommandations par titres", il faut indiquer les noms des films que vous appréciez puis appuyer sur le bouton "Valider la saisie et afficher". Il faut ensuite appuyer sur "Vérifiez doublon(s)", dans le cas où il y en a il faut récupérer l'ID du film que vous souhaitez et l'insérer dans la case "Entrez le(s) ID d'un des films doublons" et les films similaires et ceux qui sont disponibles sur Netflix apparaîtront.