Projet Netflix - Parie 1

Objetcif :

Le but de ce projet est d'analyser un nouveau jeu de données afin de s'entraîner sur les notions acquises précédement. Bien sûr, vous pouvez aller plus loin que les questions qui vous seront posées : elles ne couvrent pas forcément toutes les tendances et informations intéressantes à découvrir dans ce nouveau jeu de données.

Pour chaque question, des indices vous seront donnés, que vous pourrez lire ou non. Ces indices vous permettront de structurer ce que votre code doit faire, voire vous donneront les fonctions, méthodes ou attributs principaux à utiliser. Le but de cet exercice est vraiment de vous faire réfléchir.

N'hésitez pas à chercher la réponse à votre problématique dans un moteur de recherche ou encore à aller voir la documentation des fonctions, méthodes ou attributs proposés en indices. Il est très courant, pour un développeur data, un informaticien ou un analyste de données de chercher des indices via un moteur de recherche ou la documentation de la librairie utilisée.

Présentation du jeu de données :

Le jeu de données proposé regroupe l'ensemble des films et séries disponibles sur Netflix jusqu'en 2019. Il est disponible sur le site kaggle.com : https://www.kaggle.com/shivamb/netflix-shows

Ce jeu de données est sous licence CC0 : Public Domain. Ainsi, il peut être copié, modifié, distribué et analysé, même à des fins commerciales.

Il contient douze variables que l'on pourra étudier :

- **show_id** : correspond à l'ID unique du film ou de la série
- **type** : permet de dire s'il s'agit d'un film ("Movie") ou d'une série télévisée ("TV Show")
- **title** : le titre du film/de la série
- **director** : le directeur de production
- **cast** : les acteurs présents dans le film/la série
- **country** : pays dans lequel le film/la série a été produit
- **date_added** : date à laquelle le film/la série a été ajouté sur Netflix
- **release_year** : l'année de sortie réelle du film/de la série
- **rating** : classement du contenu (TV-MA, TV-14, TV-PG, R, PG-13, NR, TV-Y7, TV-G, TV-Y, TV-Y7-FV, G, UR, NC-17). Pour voir la signification des différents classements de contenu : https://fr.wikipedia.org/wiki/TV_Parental_Guidelines
- **duration** : pour les films - durée du film en minutes. Pour les séries - nombre de saisons
- **listed_in** : la ou les catégories du film/de la série (drame, comédie, documentaire, etc.)
- **description** : la description du film/de la série

Énoncé de l'exercice

1. Lire le fichier

Ici, il s'agit de récupérer le fichier disponible en téléchargement de ce livre : netflix_titles.csv. Vous pouvez aussi le récupérer directement via le site : https://www.kaggle.com

Action : écrire le code qui permet de lire le fichier netflix_titles.csv, de définir la première colonne du fichier comme l'index du dataframe et de stocker ce dataframe dans une variable nommée donnees. Enfin, afficher un aperçu de ce dataframe pour prendre connaissance visuellement des différentes variables du tableau.

Indices :
- N'oubliez pas d'importer la librairie Pandas pour pouvoir utiliser ses fonctions.
- La fonction read_csv() de Pandas permet de lire un fichier.
- L'option index_col permet de définir le numéro de colonne à utiliser comme index.
- La méthode head() permet d'afficher un aperçu.

2. Afficher les dimensions du dataframe

Action : écrire le code pour afficher les dimensions du dataframe, c'est-à-dire le nombre de lignes et le nombre de colonnes. Cela vous permettra de connaître le nombre de films/séries stockés dans ce tableau (nombre de lignes), ainsi que le nombre de variables (nombre de colonnes).
Indice :
L'attribut shape permet d'afficher la dimension d'un dataframe.

3. Compter les films et les séries

Action : écrire le code qui permet d'afficher le nombre de films et le nombre de séries stockés dans ce jeu de données.
Indices :
- La variable utile est la variable type, qui contient les deux modalités Movie (film) ou TV Show (série). Il faut la sélectionner avec l'attribut loc.
- La méthode value_counts() de Pandas permet de compter le nombre d'occurrences de chaque valeur unique (chaque modalité) d'une variable.

4. Générer le résumé statistique du dataframe

Action : générer le code pour afficher le résumé statistique du dataframe pour l'ensemble des variables.
Indices :

- La méthode describe() de Pandas permet de générer un résumé statistique.
- L'option include=all de la méthode describe() permet d'afficher le résumé statistique sur l'ensemble des variables. Par défaut, ce résumé se fait uniquement sur les variables numériques.

5. Compter les valeurs manquantes

Action : écrire le code pour afficher le nombre de valeurs manquantes par colonne du dataframe.
Indices :

- La méthode isna() permet de générer un dataframe de booléens avec la valeur False si la valeur n'est pas NaN et True si la valeur est NaN.
- La méthode sum() permet de compter le nombre de valeurs True dans un dataframe et donc le nombre de valeurs manquantes.

6. Explorer les valeurs manquantes

a. Sur la colonne des directeurs de production

Action : écrire le code pour sélectionner les lignes pour lesquelles la valeur est manquante dans la colonne director, puis compter le nombre de films et de séries sans directeur de production.
Indices :

- Créer un sous-tableau contenant uniquement les lignes avec une valeur manquante dans la colonne director grâce à l'attribut loc et à la méthode isna().
- Sélectionner la colonne type sur ce sous-tableau grâce à l'attribut loc et compter les occurrences de chaque modalité avec la méthode value_counts().

b. Sur la colonne des acteurs

Action : écrire le code pour sélectionner les films et séries sans acteurs, c'est-à-dire ceux pour lesquels la valeur dans la colonne cast est manquante. Puis afficher les dix catégories (colonne listed_in) qui ont le plus de valeurs manquantes dans la colonne cast.
Indices :

- Créer un sous-tableau contenant uniquement les lignes avec une valeur manquante dans la colonne cast grâce à l'attribut loc et la méthode isna().
- Sélectionner la colonne listed_in sur ce sous-tableau grâce à l'attribut loc et compter les occurrences de chaque modalité avec la méthode value_counts().
- La méthode head() permet d'afficher un aperçu d'un dataframe, l'option n permet de déterminer le nombre de lignes à afficher.

7. Supprimer les lignes dupliquées

Action : écrire le code pour compter le nombre de lignes dupliquées et pour les supprimer du dataframe.

Indices :

- La méthode duplicated() permet de générer une série de booléens et donc de l'utiliser en indexing pour afficher les lignes dupliquées.
- La méthode drop_duplicates() permet de supprimer les lignes dupliquées.

8. Compter les films/séries produits par les États-Unis et par la France

Action : écrire le code pour compter le nombre de films/séries produits par les États-Unis et par la France.

Indices :

- L'indexation booléenne permet de sélectionner les lignes pour lesquelles une colonne est égale à une certaine valeur. Ici, la variable utile est country.
- La syntaxe suivante permet de faire de l'indexation booléenne : dataframe[(dataframe["variable"]=="valeur")].
- L'attribut shape permet d'afficher la dimension d'un dataframe et donc le nombre de lignes.

9. Afficher le contenu le plus vieux disponible sur Netflix

Action : écrire le code permettant d'afficher la ligne du contenu pour lequel l'année de sortie réelle (release_year) est la plus ancienne.

Indices :

- La méthode min() permet de récupérer la valeur minimale d'un ensemble de valeurs et donc la valeur minimale d'une colonne de dataframe.
- Une fois l'année minimale connue, l'indexation booléenne permet de récupérer la ligne pour laquelle la valeur dans la colonne release_year est égale à cette date.

10. Afficher le film avec la durée la plus longue sur Netflix

a. Nouvelle notion : les méthodes str

Il existe des méthodes Pandas spécifiques pour traiter des données de type chaînes de caractères (str en Python). Généralement, ces méthodes sont précédées du terme str (qui est un attribut) et sont dédiées, comme leur nom l'indique, au traitement des données de type str spécifiquement. Ces méthodes permettent, entre autres, de remplacer des chaînes de caractères par d'autres (str.replace()) ou encore de sélectionner des lignes contenant une sous-chaîne de caractères (str.contains()).

Retenez simplement que si vous avez des chaînes de caractères à traiter, il faut aller regarder spécifiquement la partie de la documentation Pandas dédiée à ce thème : https://pandas.pydata.org/docs/user_guide/text.html

Syntaxe générale de str.replace() :
ma_serie.str.replace("chaine1", "chaine2")
Ici, la chaîne de caractères chaine1 sera remplacée par chaine2 dans l'ensemble de la série.

Syntaxe générale de str.contains() :
mon_dataframe[mon_dataframe["ma_colonne"].str.contains("chaine")]
Dans le cas d'un dataframe, on pourra utiliser la méthode str.contains() pour faire de l'indexation et récupérer les lignes où une colonne spécifique contient le mot chaine.

b. Énoncé

Action : le problème de la colonne duration est que les données qu'elle contient sont des chaînes de caractères, du fait que le mot " min" est écrit à la suite du nombre de minutes. Il n'est donc pas possible de trouver la valeur numérique la plus grande tant que cette colonne est une chaîne de caractères. Ainsi, il faut écrire le code pour convertir les données de la colonne duration en entier puis trier numériquement ces valeurs pour afficher les cinq films avec la durée la plus longue. Ce code n'est pas simple, n'hésitez pas à vous aider des indices ou à aller étudier la solution proposée dans le notebook.

Indices :

- Créer un sous-tableau contenant uniquement les contenus de type Movie grâce à l'indexation booléenne.
- À partir du sous-tableau, créer une série correspondant à la colonne duration, et la ranger dans une variable nommée, par exemple, duree.
- Utiliser la fonction str.replace() sur la série duree pour remplacer la chaîne de caractères " min" par une chaîne vide "", ce qui permet d'enlever le texte des valeurs contenues dans la série.
- Utiliser la méthode astype() pour transformer le type des valeurs contenues dans la variable duree en un autre type, ici on souhaite des entiers (int).
- Trier les données de cette série avec la méthode sort_values, dans l'ordre décroissant.
- Utiliser les index de la série contenant les valeurs de durée triées, grâce à l'attribut index, pour récupérer les lignes du dataframe originel, afin que celles-ci soient triées selon la durée. Enfin, afficher les cinq premières lignes.

11. Étudier les catégories avec le plus de contenu

Ici, nous allons étudier les catégories avec le plus de contenu, c'est-à-dire que nous allons nous intéresser à la colonne listed_in. La difficulté des colonnes listed_in, director et cast est qu'elles peuvent contenir des listes de valeurs.

Nous allons voir ensemble comment analyser des listes de valeurs en prenant comme exemple la colonne listed_in. Ensuite, vous devrez adapter ce code pour les colonnes director et cast.
Voici un aperçu de la colonne listed_in.

```
import pandas as pd
donnees=pd.read_csv("../datasets/netflix_titles.csv",
index_col=[0])
donnees["listed_in"].head()
```

Ici, on lit le fichier et on affiche un aperçu de la colonne listed_in.

```
show_id
81145628    Children & Family Movies, Comedies
80117401                        Stand-Up Comedy
70234439                                Kids' TV
80058654                                Kids' TV
80125979                                Comedies
Name: listed_in, dtype: object
```

On constate que chaque film/série peut appartenir à une ou plusieurs catégories. Dans le cas où il appartient à plusieurs catégories, celles-ci sont séparées par une virgule et un espace ", ". Nous devons splitter, couper, ces catégories pour pouvoir les étudier, sinon Python considérera "Children & Family Movies, Comedies" comme une unique catégorie, alors qu'en réalité il y en a deux ici : "Children & Family Movies" et "Comedies".

Pour faire ça, nous allons voir une méthode Pandas que vous connaissez peut-être, car cette méthode existe plus généralement en Python. Il s'agit de la méthode join(), qui permet de joindre tous les éléments d'un tuple, d'une liste ou même d'une série, séparés par un caractère que nous lui donnons en début de code.

Pour illustrer ce propos, utilisons la méthode join() sur la colonne listed_in de notre dataframe stocké dans la variable donnees et joignons l'ensemble des éléments de la colonne avec le séparateur ", " (une virgule et un espace). On ignore les valeurs manquantes grâce à la méthode dropna().

```
", ".join(donnees["listed_in"].dropna())
```

<u>Résultat :</u>

```
['Children & Family Movies',
'Comedies',
'Stand-Up Comedy',
"Kids' TV",
"Kids' TV",
'Comedies',
...
]
```

Ici, on obtient une liste contenant toutes les catégories du jeu de données Netflix, films et séries confondus. On constate que "Children & Family Movies" et "Comedies" sont maintenant deux catégories bien séparées. Ainsi, nous allons transformer cette liste en série, pour pouvoir ensuite utiliser la méthode value_counts() que nous connaissons bien, afin d'avoir le nombre d'occurrences de chaque catégorie dans le dataframe. En effet, la méthode value_counts() est une méthode Pandas, qui fonctionne donc sur une série et non sur une liste.

```
categories=pd.Series
(", ".join(donnees["listed_in"].dropna()).split(", "))
categories.value_counts().head()
```

Résultat :

```
International Movies      1926
Dramas                    1622
Comedies                  1113
International TV Shows    1001
Documentaries              668
dtype: int64
```

À présent, on peut conclure que la catégorie contenant le plus de contenu est la catégorie "International Movies", suivi par "Dramas" et "Comedies". Si nous n'avions pas fait tout ce traitement, les résultats auraient été grandement biaisés.