

Conception d'un moteur de recommandation de films

Données :

IMDb

Problématique

à partir d'1 film : recommander 5 films similaires et intéressants
Fournir une API en ligne pour remonter les recommandations

Contraintes

Utiliser le dataset : imdb 5000 movies
Pas de données d'historique de consultation utilisateurs

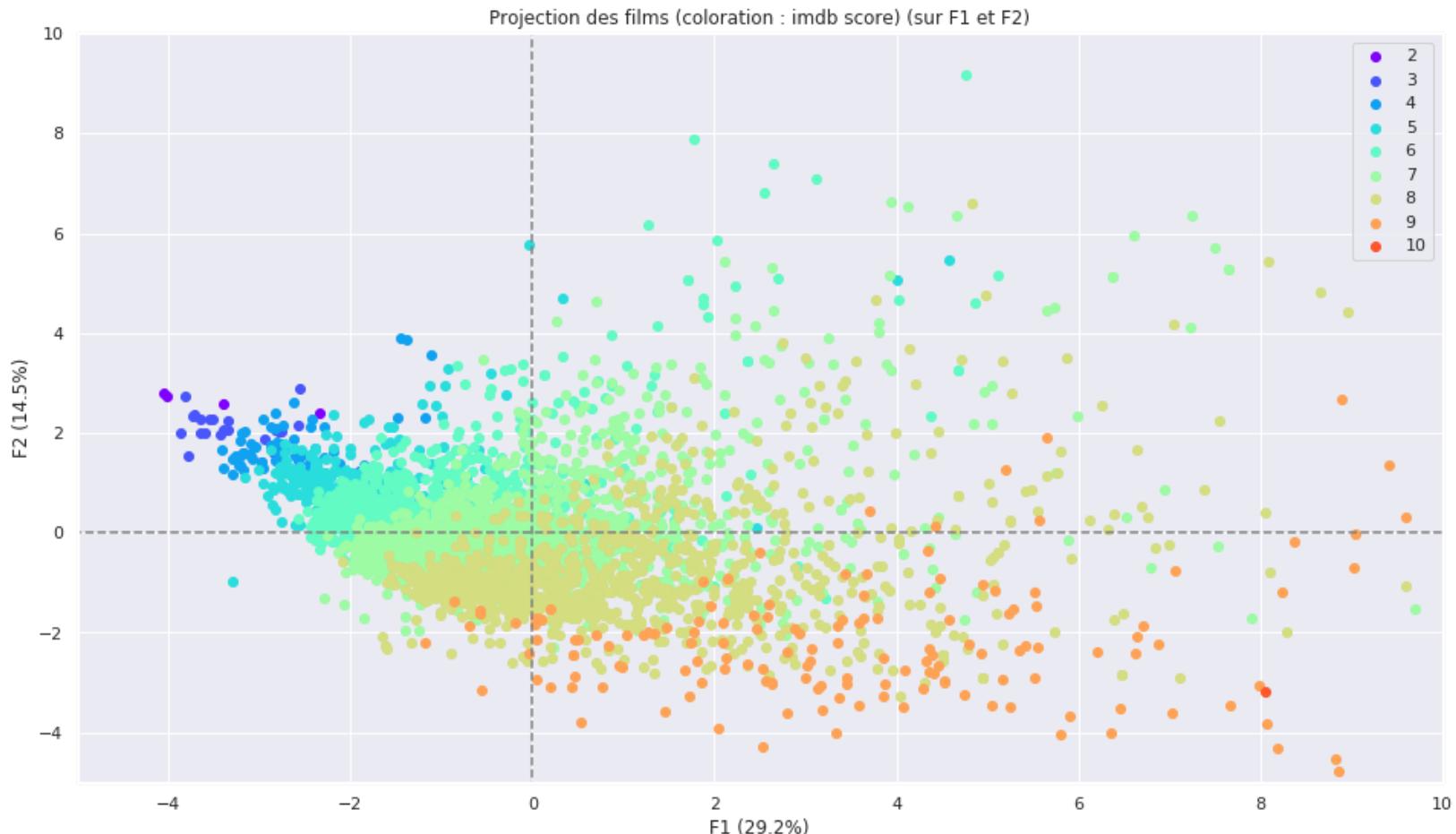
Démarche suivie

- 1/ Quelle méthode pour trouver des films similaires / intéressants ?
- 2/ Exploration des données disponibles
- 3/ Contrôle de la qualité des données et choix sur les données manquantes
- 4/ Encodage des features (numériques et catégorielles)
- 5/ Conception d'un premier modèle de recommandation
- 6/ Les perspectives pour aller plus loin
Expérimentation d'un nouveau modèle
Test de différents modèles avec pipelines + métriques d'évaluation
- 7/ Choix du modèle final et construction de l'API



**Quelle méthode pour trouver
des films similaires /
intéressants ?**

**Chaque film est un point dans l'espace des caractéristiques
⇒ 2 films proches dans cet espace se ressemblent**

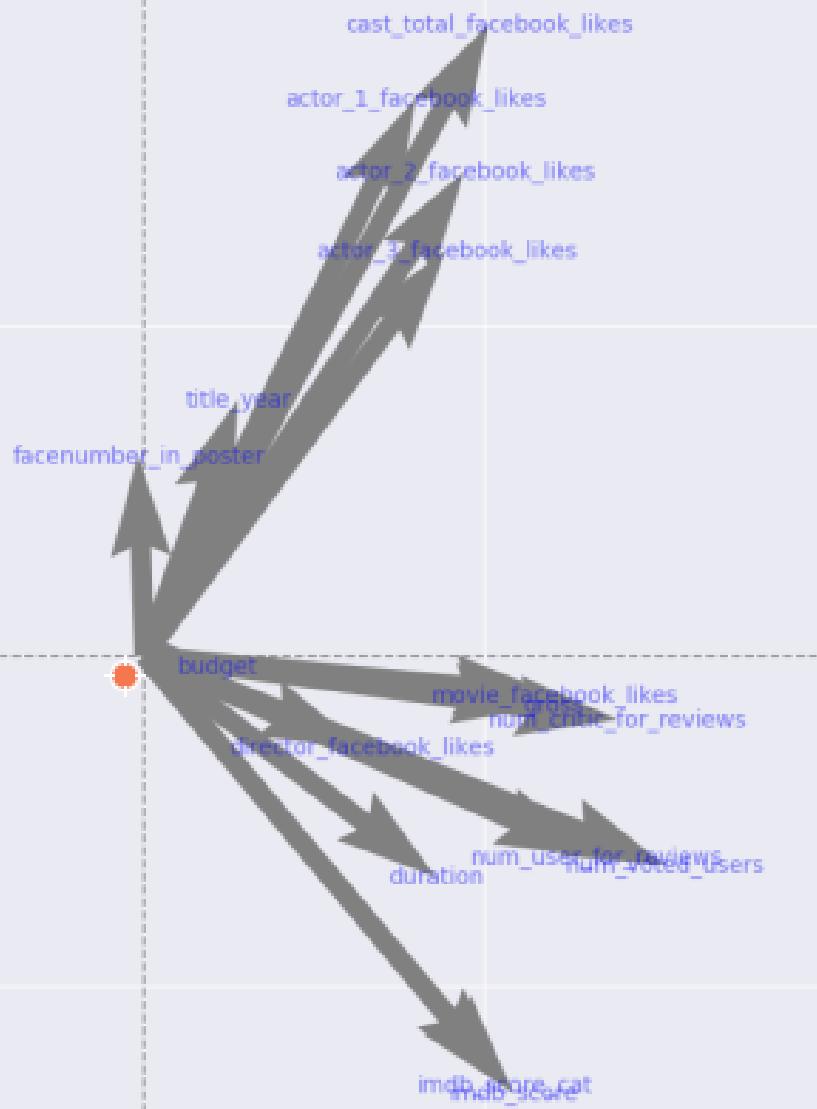




Exploration des données disponibles

Features quantitatives	Features qualitatives	Features non retenues
'movie_facebook_likes	country (ex : France)	aspect_ratio
num_voted_users	director_name (ex : James Cameron)	movie_imdb_link
cast_total_facebook_likes	genres (ex : Action Adventure Drama)	
imdb_score	plot_keywords (ex : amnesia criminal)	
actor_1_facebook_likes	color (ex : Color)	
actor_2_facebook_likes	content_rating (ex : PG-13)	
facenumber_in_poster		
duration		
num_user_for_reviews		
actor_3_facebook_likes		
num_critic_for_reviews		
director_facebook_likes		
budget		
gross		
title		

Les features numériques



⇒ Les features liées aux acteurs
En haut à droite du plan factoriel

⇒ Les features liées au film, au réalisateur, aux votes utilisateurs et au score imdb
En bas à droite du plan factoriel

⇒ Les features très peu corrélées aux autres :
Le budget
Le "face number in poster"

Les features catégorielles

Features catégorielles

country (ex : France)

director_name (ex : James Cameron)

genres (ex : Action|Adventure|Drama)

plot_keywords (ex : amnesia|criminal)

color (ex : Color)

content_rating (ex : PG-13)

Elles seront très utiles pour comparer les caractéristiques des films



**Contrôle de la qualité des données
choix sur les données manquantes**

**Encodage des features pour le
modèle**

Features numériques	% renseigné
'movie_facebook_likes'	100 %
num_voted_users	100 %
cast_total_facebook_likes	100 %
imdb_score	100 %
actor_1_facebook_likes	99.8 %
actor_2_facebook_likes	99.7 %
facenumber_in_poster	99.7 %
duration	99.6 %
num_user_for_reviews	99.5 %
actor_3_facebook_likes	99.5 %
num_critic_for_reviews	99 %
director_facebook_likes	97 %
budget	90 %
gross	82 %
title_year	97 %

Feat. Numériques : bonne qualité des données

Choix : Imputation des données manquantes par régression linéaire par rapport aux autres variables

A noter que les données facebook sont bien renseignées, même pour les années avant la création de facebook

⇒ On peut donc conserver ces données

NB : On a également supprimé quelques doublons

Features catégorielles	% renseigné
country (ex : France)	99.9 %
director_name (ex : James Cameron)	97 %
genres (ex : Action Adventure Drama)	100 %
plot_keywords (ex : amnesia criminal)	96.9 %
color (ex : Color)	99.6 %
content_rating (ex : PG-13)	93.9 %
movie_title (ex : Avatar)	100 %
Actor_n_name (ex : Orlando Bloom)	~ 99 %

Feat. qualitatives :
Bonne qualité des données

Choix :

- Encodage de type « 1 hot »
- Les features non renseignées pour un film donné sont à 0
- Les acteurs 1,2,3 sont regroupés en une seule variable
- Pour le titre du film, les features seront les différents mots clés possibles dans le titre

Exemple avant 1 hot encode :

	actor_1_name	actor_2_name	actor_3_name	actors_names	country	genres
0	CCH Pounder	Joel David Moore	Wes Studi	CCH Pounder Joel David Moore Wes Studi	USA	Action Adventure Fantasy Sci-Fi

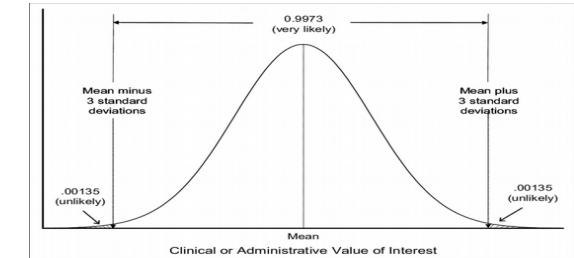
Exemple après 1 hot encode : (toutes les colonnes ne sont pas représentées pour des raisons de place)

	actors_names_Johnny Depp	actors_names_Orlando Bloom	actors_names_Jack Davenport	actors_names_Joel David Moore	country_USA	country_UK	genres_Action	genres_Adventure
0	0	0	0	1	1	0	1	1



Conception d'un premier modèle de recommandation

Standard Scaling : centrage + écart type de 1



Réduction de dimensionnalité (Principal Component Analysis)

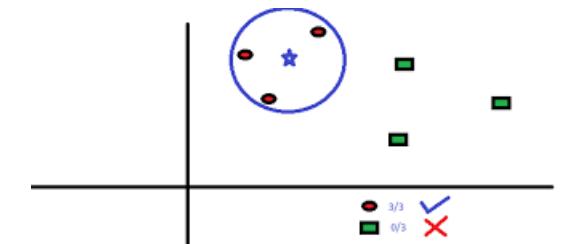
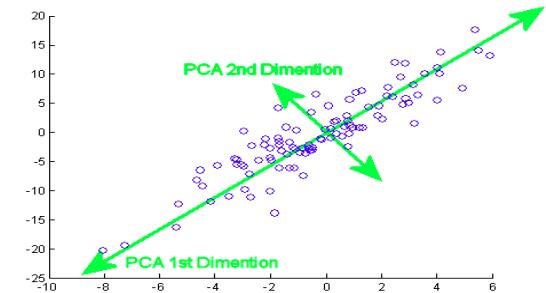
⇒ Réduction à **200 dimensions**

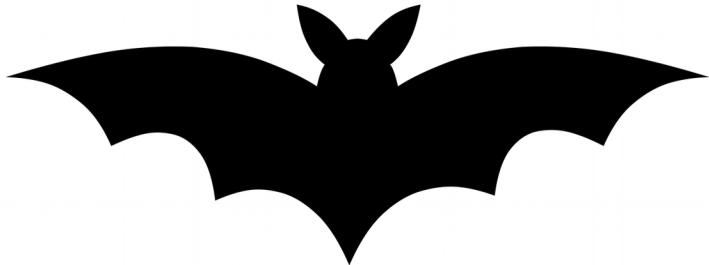


Identification des **5 plus proches voisins** de chaque film
(K nearest neighbours)



5 recommandations pour chaque film





> Film choisi : The Dark Knight Rises - imdb score : 8.5

Films recommandés :

The Dark Knight - imdb score : 9.0

Inception - imdb score : 8.8

Batman Begins - imdb score : 8.3

The Avengers - imdb score : 8.1

Thor: The Dark World - imdb score : 7.1

Exemple de points communs pour 2 films recommandés pour The Dark Knight Rises

Films recommandés :

The Dark Knight - imdb score : 9.0 - http://www.imdb.com/title/tt0468569/?ref_=fn_tt_tt_1

ID film recommandé = 66

Tableau de similarités (1 = point commun catégoriel. 0 = élément catégoriel différenciant)

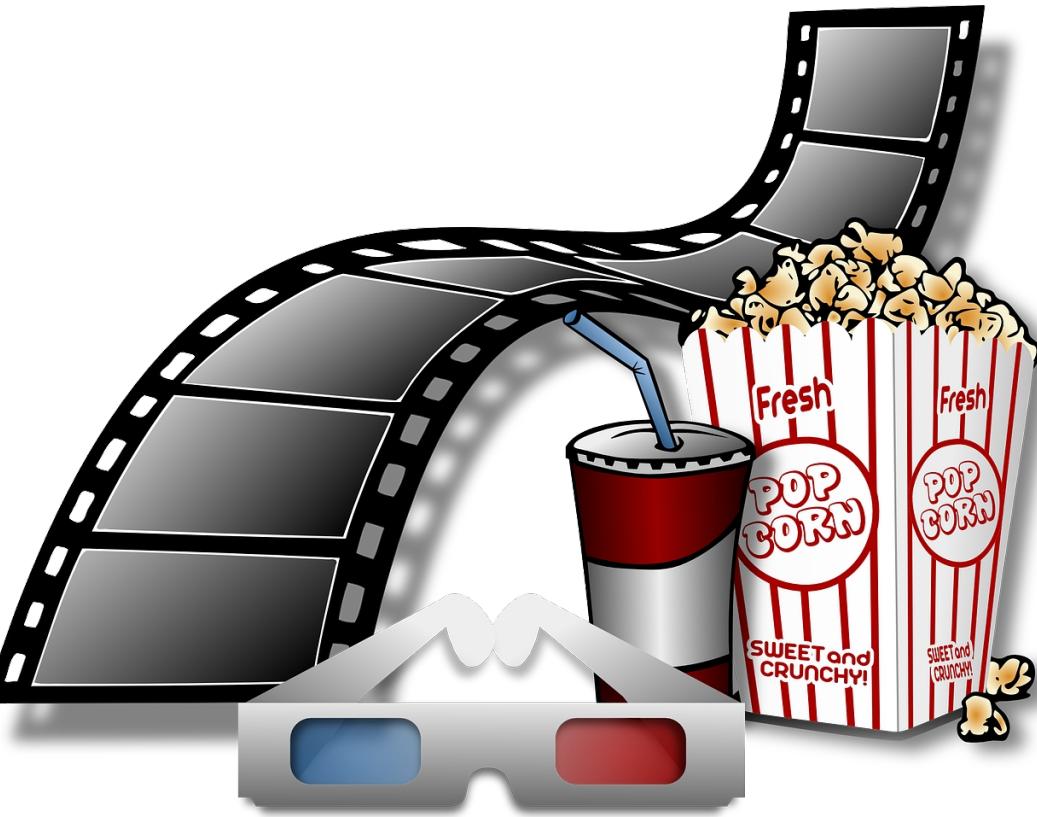
movie_title_the	1.000000
actors_names_Christian Bale	1.000000
director_name_Christopher Nolan	1.000000
language_English	1.000000
country_USA	1.000000
director_facebook_likes	1.000000
color_Color	1.000000
genres_Thriller	1.000000
content_rating_PG-13	1.000000
movie_title_dark	1.000000
genres_Action	1.000000

Inception - imdb score : 8.8 - http://www.imdb.com/title/tt1375666/?ref_=fn_tt_tt_1

ID film recommandé = 97

Tableau de similarités (1 = point commun catégoriel. 0 = élément catégoriel différenciant)

genres_Thriller	1.000000
genres_Action	1.000000
actors_names_Tom Hardy	1.000000
actor_3_facebook_likes	1.000000
actors_names_Joseph Gordon-Levitt	1.000000
director_facebook_likes	1.000000
content_rating_PG-13	1.000000
color_Color	1.000000
country_USA	1.000000
language_English	1.000000
director_name_Christopher Nolan	1.000000



Ce modèle donne déjà de bons résultats !

Mais peut-on aller plus loin ?

Quelles sont les perspectives pour améliorer le modèle ?



Les perspectives pour aller plus loin

Expérimentation d'un autre algorithme de réduction de dimensionnalité

Tester différents modèles avec pipelines + des métriques d'évaluation

Expérimentation de l'algorithme NCA : Neighbour Components Analysis

Cet algorithme est connu pour bien fonctionner avant une
transformation KNN

Source : doc. Scikit learn §1.6.7.1

<https://scikit-learn.org/stable/modules/neighbors.html#nca>

Cet algorithme permet de faire une réduction dimensionnelle tout en **apprenant une métrique qui sera optimisée pour le KNN**

Pour apprendre cette métrique, il a besoin d'étiquettes pour chaque instance (en l'occurrence nous avons utilisé une catégorie de score imdb comme étiquette)

L'objectif de NCA est **d'apprendre une transformation des données** qui maximise la probabilité que les échantillons soient correctement classés

$$\arg \max_L \sum_{i=0}^{N-1} p_i \quad \text{où} \quad p_i = \sum_{j \in C_i} p_{ij}$$

$$p_{ij} = \frac{\exp(-\|Lx_i - Lx_j\|^2)}{\sum_{k \neq i} \exp(-\|Lx_i - Lx_k\|^2)}, \quad p_{ii} = 0 \quad (\text{softmax})$$

Avec :

N le nombre de films

p_i la probabilité que le film soit correctement classé

C_i l'ensemble des points dans la même classe que i

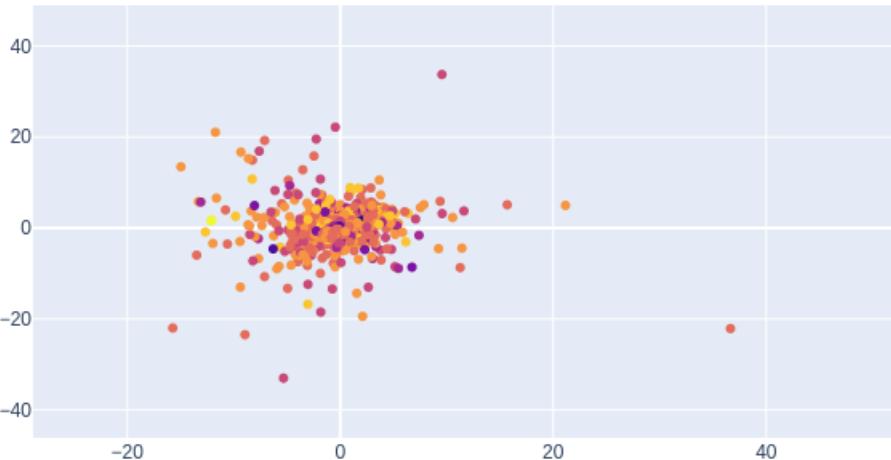
Distance optimisée par l'algorithme : $\|L(x_i - x_j)\|^2 = (x_i - x_j)^T M (x_i - x_j)$

Projection des films de l'espace encodé (~ 22000 dimensions) vers 2 dimensions

Comparaison PCA et NCA

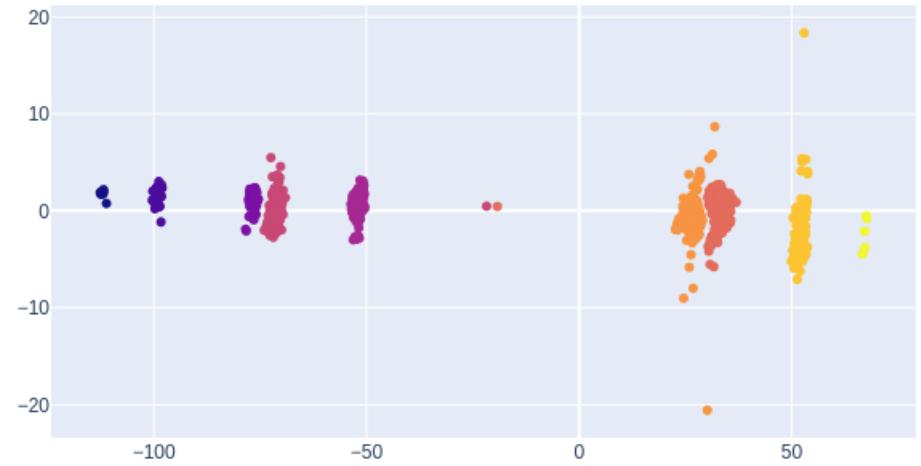
Projection PCA

Réduction dimensionnelle des features encodées (22000 dimensions) avec PCA, coloration par IMDb score



Projection NCA

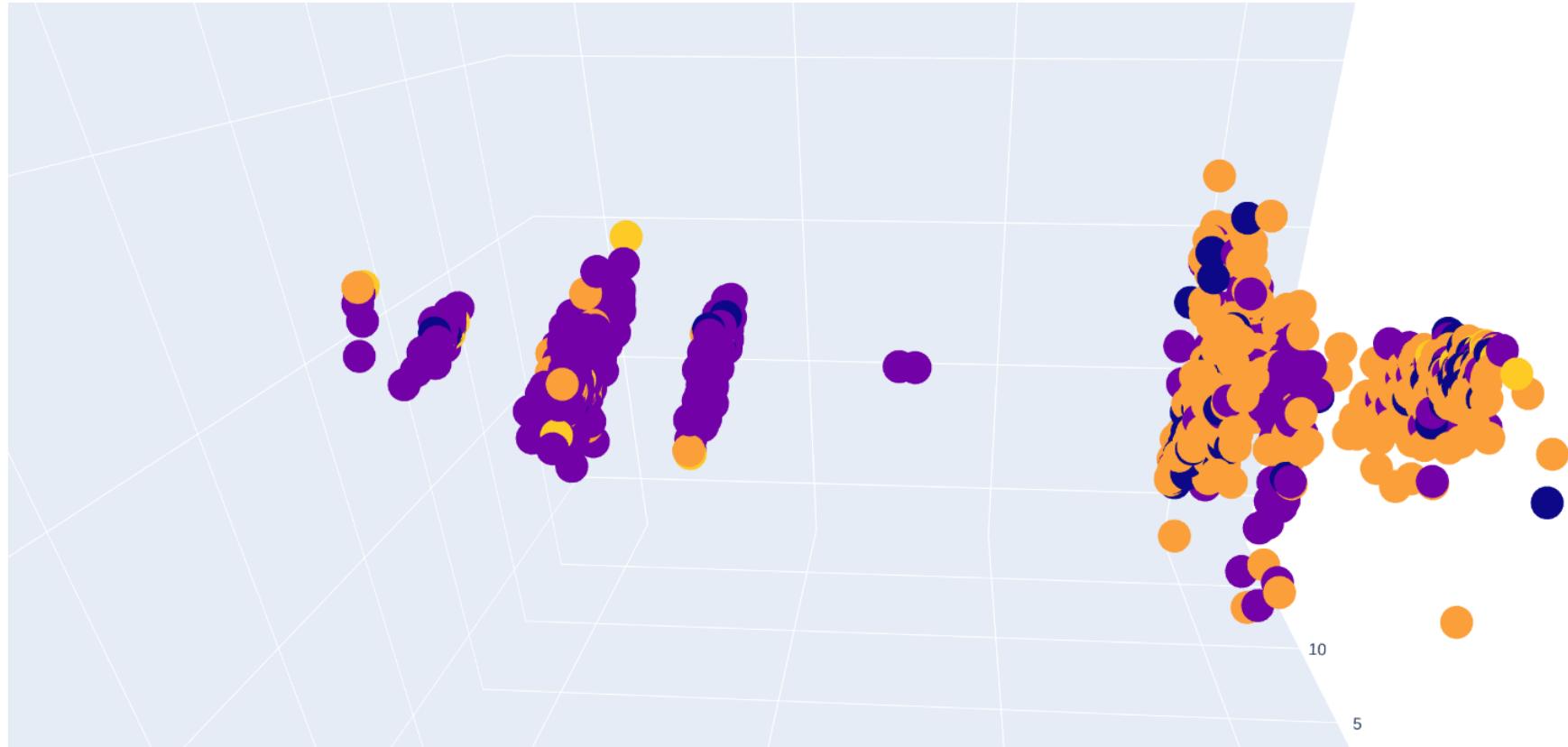
Réduction dimensionnelle des features encodées (22000 dimensions) avec NCA, coloration par IMDb score

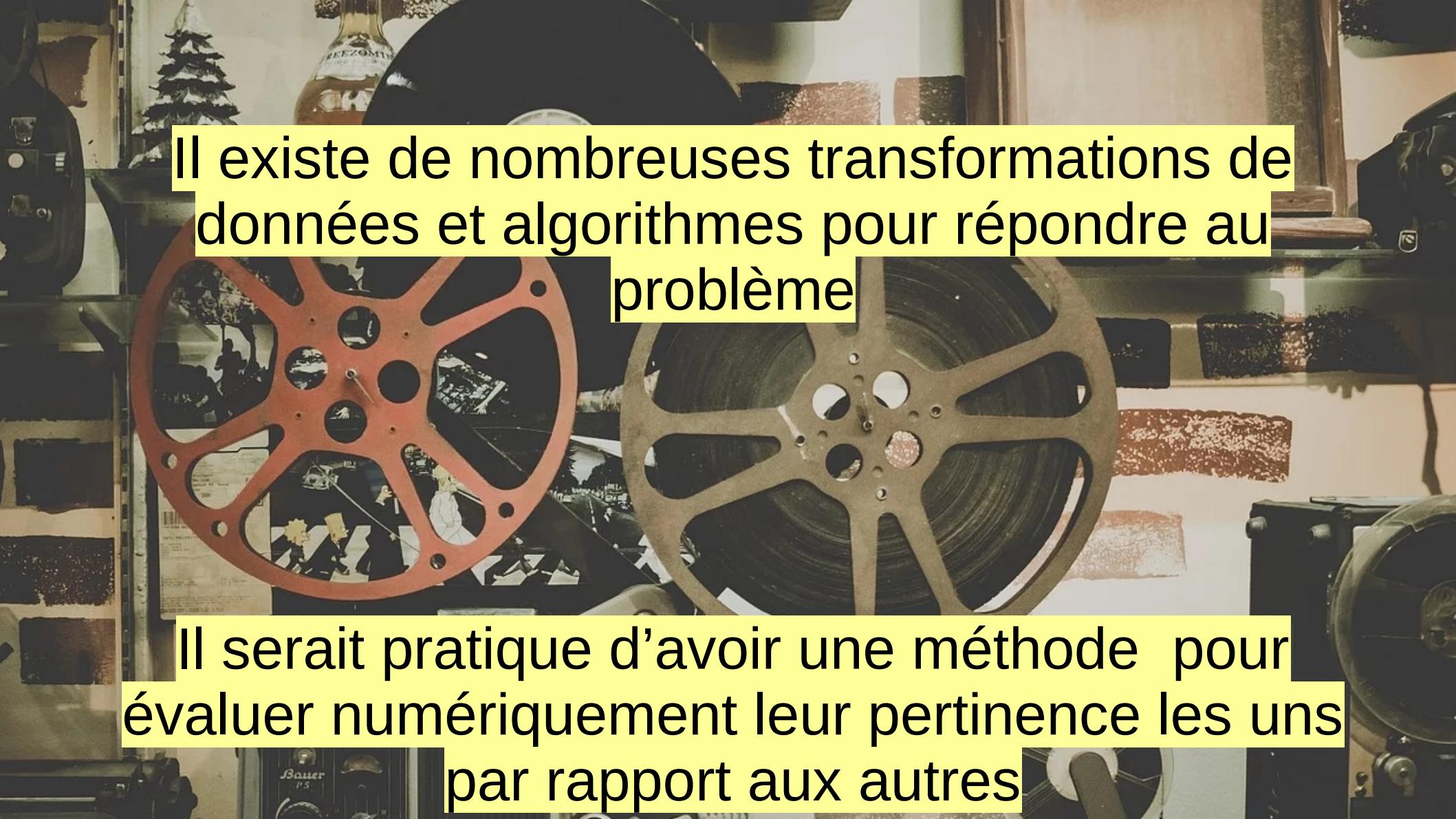


Légende : la couleur correspond au score imdb
Les couleurs chaudes (orange, rouge..) = score élevé
Les couleurs froides = score faible

Une idée intuitive de ce qui se passe en plus haute dimension.....

Réduction de 22000 à 3 dimensions avec NCA,
et représentation en 3 dimensions colorée par un clustering KMeans (effectué sur la base d'une réduction à 10 dimensions puis d'un clustering à 10 dimensions)

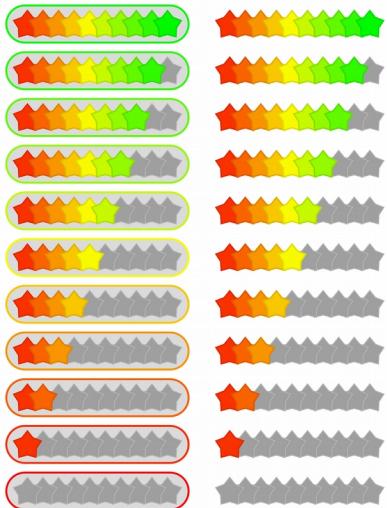


A collage of various items related to cinema, including several large film reels in the foreground, some in red and gold, and others in black and silver. Behind them are several movie posters for films like "The Beatles' Abbey Road" and "Star Wars". There are also some bottles and a small Christmas tree in the background.

Il existe de nombreuses transformations de données et algorithmes pour répondre au problème

Il serait pratique d'avoir une méthode pour évaluer numériquement leur pertinence les uns par rapport aux autres

Nous avons expérimenté 2 méthodes :

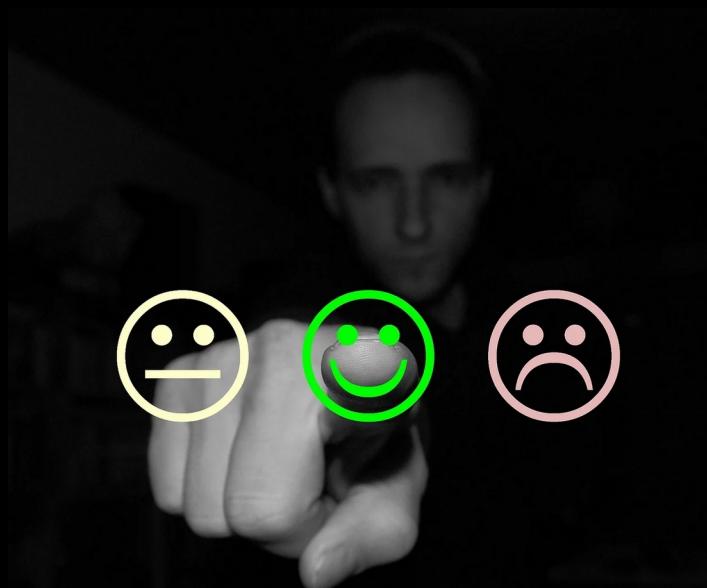


- La métrique imdb : Calculer l'erreur moyenne de prédiction du score IMDB pour chaque film, et
⇒ évaluer l'algorithme sur sa capacité à prédire le score imdb
- La métrique similarité : Calculer un score de similarité pour chaque prédiction : plus les films prédits ont d'attributs en commun, plus ils sont similaires.
⇒ Evaluer l'algorithme sur sa capacité à maximiser le score de similarité

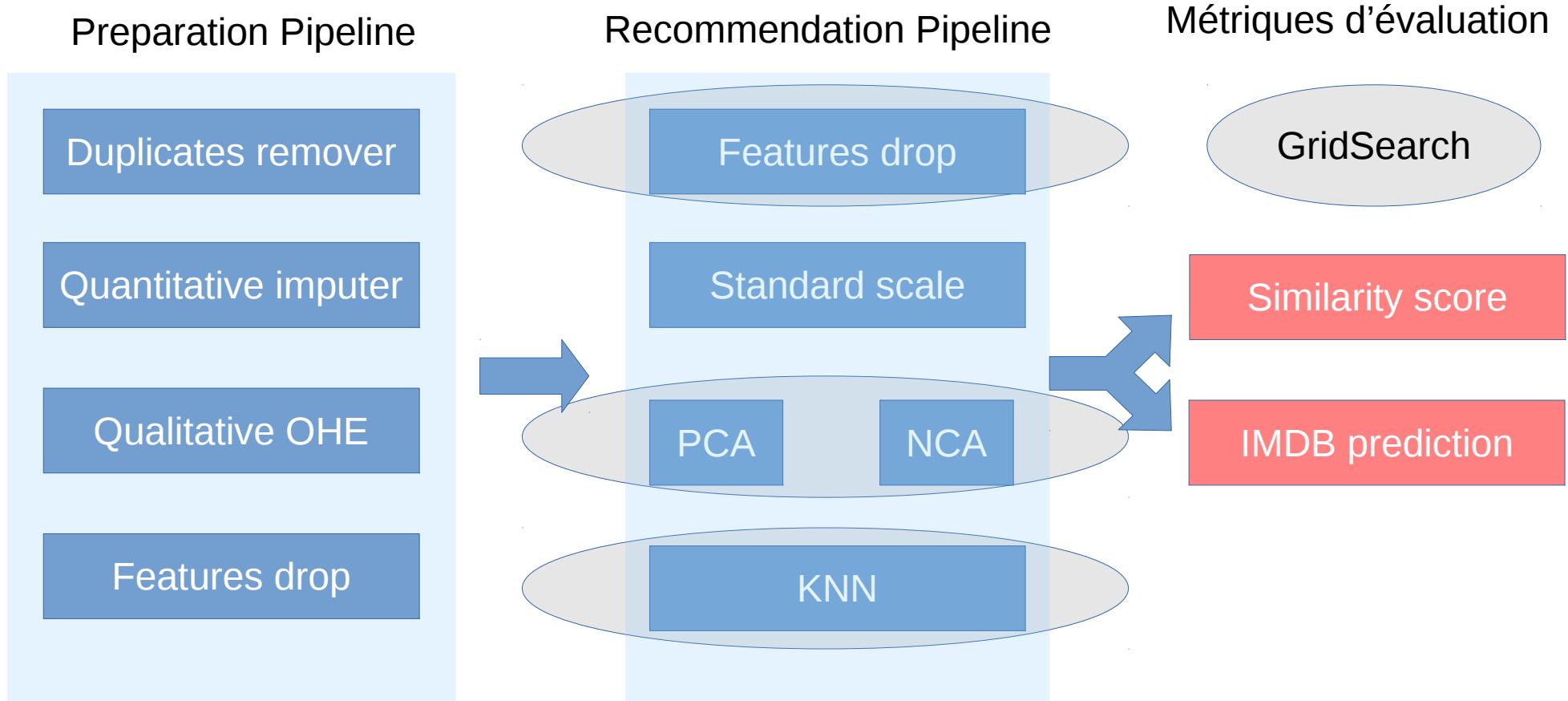
Ces méthodes ne sont pas idéales et ne remplacent pas l'évaluation humaine !

⇒ Mais elles permettent d'éliminer les algorithmes les moins bons : on peut se dire qu'un algorithme peu performant pour prédire le score imdb ou maximiser la similarité des recommandations a de bonnes chances d'être médiocre

⇒ Ainsi cela peut être une méthode pour shortlister les meilleurs algorithmes, avant de les évaluer par d'autres méthodes faisant intervenir l'évaluation humaine (A/B testing, étiquetage par des humains, etc)



Enchaînement des traitements d'évaluation



Résultats obtenus

Les meilleurs modèles apparaissent en haut :

Métrique similarité						Métrique score imdb					
Features drop	NCA or PCA	Dimensions	KNN algo	KNN distance	Score	Features drop	NCA or PCA	Dimensions	KNN algo	KNN distance	Score
B	NCA	200	ball_tree	Minkowski	8,768236885	Aucune	NCA	10	ball_tree	Manhattan	-1,015439856
B	NCA	200	ball_tree	Euclidean	8,768236885	Aucune	NCA	10	ball_tree	Minkowski	-1,019797679
imdb_score	NCA	100	ball_tree	Manhattan	8,753227734	Aucune	NCA	10	ball_tree	Euclidean	-1,019797679
A	NCA	100	ball_tree	Minkowski	8,750021893	Aucune	NCA	100	ball_tree	Manhattan	-1,051385594
A	NCA	100	ball_tree	Euclidean	8,750021893	Aucune	NCA	100	ball_tree	Minkowski	-1,059188876
imdb_score	NCA	100	ball_tree	Minkowski	8,73769623	Aucune	NCA	100	ball_tree	Euclidean	-1,059188876
imdb_score	NCA	100	ball_tree	Euclidean	8,73769623	Aucune	NCA	200	ball_tree	Manhattan	-1,069869628
B	NCA	200	ball_tree	Manhattan	8,732366155	Aucune	NCA	200	ball_tree	Minkowski	-1,088261224
B	NCA	100	ball_tree	Manhattan	8,724163979	Aucune	NCA	200	ball_tree	Euclidean	-1,088261224
B	NCA	100	ball_tree	Minkowski	8,721136992	Aucune	PCA	100	ball_tree	Manhattan	-1,12390004

A = Drop de tout ce qui est en bas à droite sur le plan factoriel : Drop de : ['movie_facebook_likes', 'num_critic_for_reviews', 'director_facebook_likes', 'num_user_for_reviews', 'num_voted_users', 'duration', 'imdb_score']

B = Conservation de cast_total_facebook_likes (en haut à droite du PF) + imdb_score (en bas à droite du PF) + les features OHE

On obtient des résultats assez différents via 2 méthodes d'évaluation différentes

Le modèle retenu au final pour l'API est celui surligné en vert

⇒ C'est le meilleur modèle avec la métrique similarité, mais seulement le 29ème avec la métrique imdb

⇒ Ainsi la métrique de similarité semble la mieux adaptée pour traiter la problématique

Pour le détail des résultats, voir fichiers :
grid_search_results_metric_imdb_20200229.csv
grid_search_results_similarity_metric.csv

Mise en place de l'API

> **Endpoint URL :**

<https://francoisboyer.pythonanywhere.com/recommend>

> **Data exchange Protocol :**

JSON

> **Data transfer method :**

POST

> **Example de commande de test :**

```
curl -X POST -H "Content-Type: application/json" -d  
"{"id_film": 1}"
```

<https://francoisboyer.pythonanywhere.com/recommend>

Résultat de l'API

```
curl -X POST -H "Content-Type: application/json" -d "{ \"id_film\": 100 }" https://francoisboyer.pythonanywhere.com/recommend
```

```
{
  "_results": [
    {
      "id": 1321,
      "name": "The Fast and the Furious "
    },
    {
      "id": 208,
      "name": "Fast Five "
    },
    {
      "id": 45,
      "name": "Furious 7 "
    },
    {
      "id": 930,
      "name": "22 Jump Street "
    },
    {
      "id": 509,
      "name": "2 Fast 2 Furious "
    }
  ]
}
```

Perspectives possibles pour aller encore plus loin ...

On a maintenant un framework qui permet de tester différents algorithmes

⇒ Expérimenter d'autres algorithmes notamment à base de réseaux de neurones
(par exemple : auto encodeurs)

⇒ Améliorer encore le 1-hot encoding des mots clés (exemple : enlever le « The »)

⇒ Utiliser une base de données utilisateur en entrée du modèle
Le modèle se baserait sur la somme des films dans l'espace des utilisateurs

⇒ On pourrait filtrer les imdb scores en dessous d'un certain seul (exemple : 7/10)

⇒ On pourrait filtrer les contenus offensants / pour adultes

⇒ Parmi les 5 recommandations, on pourrait en faire une ou deux un peu plus éloignée
que les autres du film d'origine (mais pas trop non plus)
⇒ Objectif : faire découvrir à l'utilisateur de nouvelles choses !

FIN

