Développez un moteur de recommandations de films

Base de données : IMDB 5000 movie dataset

Version figée : lien de téléchargement

Objets de la mission

- Réaliser une analyse exploratoire :
  - Relation entre les variables + analyse des cas extrêmes
  - Préparation de la base à l'analyse

- Tester différentes approches :
  - Clustering
  - Calcul de distance

- Développer une API, 5 films recommandés :
  - Par id
  - Par nom

Problématique

Comment identifier, pour chaque film, les 5 films qui sont les plus ressemblants avec une approche content-based pure, c'est-à-dire sans utilisateur ?

- Quelles sont les variables pertinentes ?
- Evaluer des proximités entre les films d'après des critères de nature différente
- Calculer un score de ressemblance entre les paires de films

Problématique

Mon approche :

- Un mélange entre des films très proches par nature (suite ou prequel) et des films qui ont la même construction (genre, thème abordé)

- Equilibre à trouver et être sûr qu'on a suffisamment d'éléments pour effectuer des rapprochements en fonction de l'importance des critères

1. Analyse exploratoire

a. Nettoyage des données

- Des données en double : 241 films sont des doublons d'après le couple titre/réalisateur ou le lien IMDB et 90 possèdent ont toutes leurs variables identiques à une autre ligne.

⇒Des valeurs numériques diffèrent très légèrement (nombre d'avis ou de likes Facebook). Suppression de tous les doublons.

- Valeurs textuelles corrigées (espace après le titre par exemple)
- Les budgets ne sont pas tous en dollars

⇒Web scrapping pour extraire les valeurs depuis la version live d'IMDB et conversion en USD, budget et recettes ajustée à l'inflation

1. Analyse exploratoire

b. Transformation des valeurs textuelles

- Colonne ‘color’: deux valeurs possibles ‘Color’ ou ‘Black & white’.

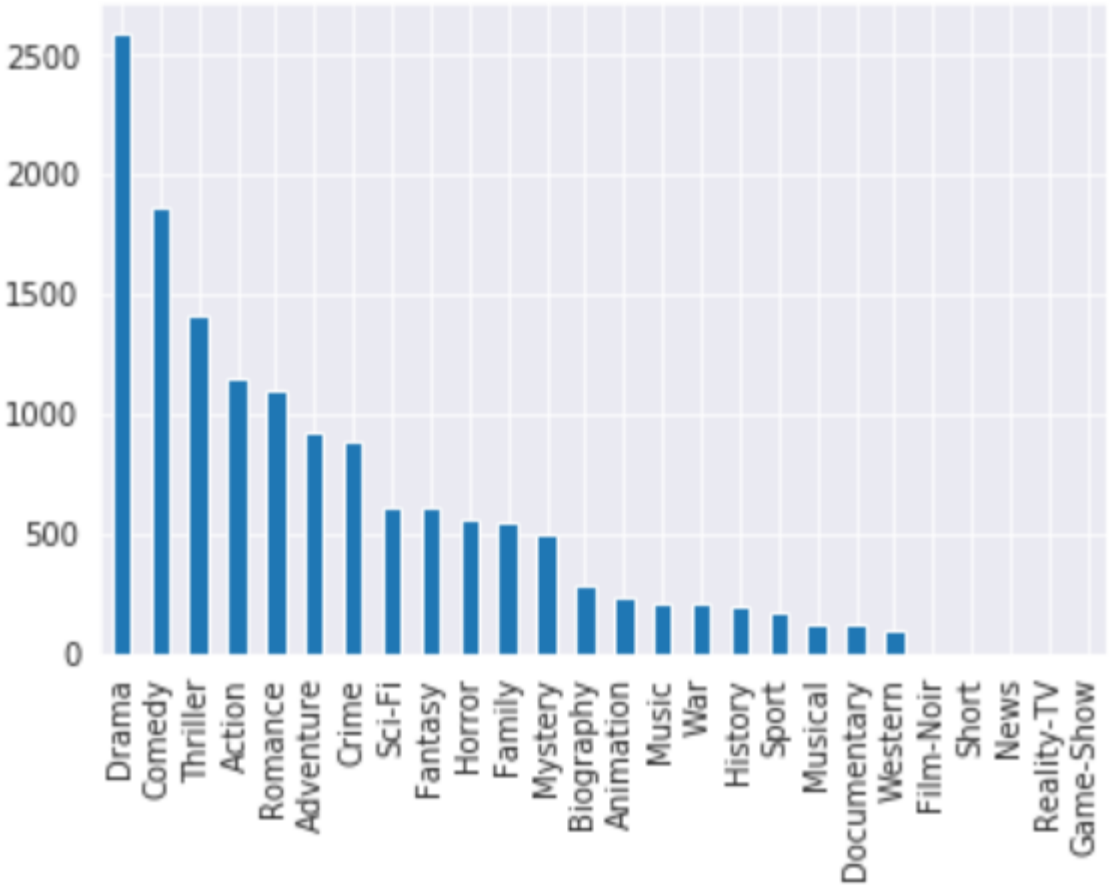
⇒passage en binaire : color=1, b&w=0

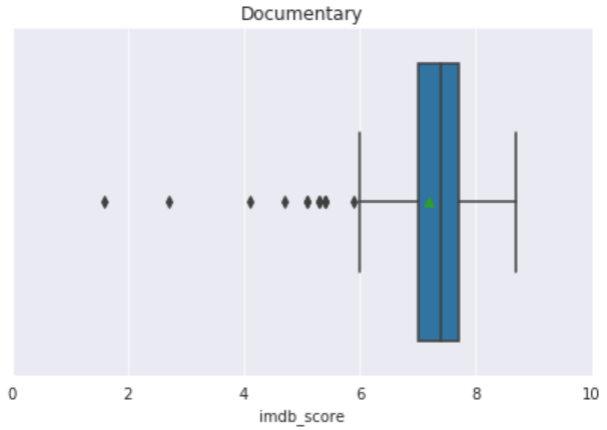
- Colonne ‘content rating’ : variable catégorielle

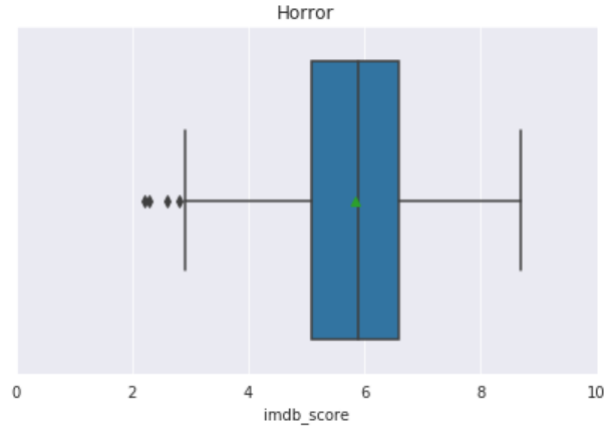
⇒Transformation en trois variables binaires : adapté aux enfants, adapté aux adolescents, film montré au cinéma (il y a des séries TV dans la base)

1. Analyse exploratoire

c. Analyse des genres





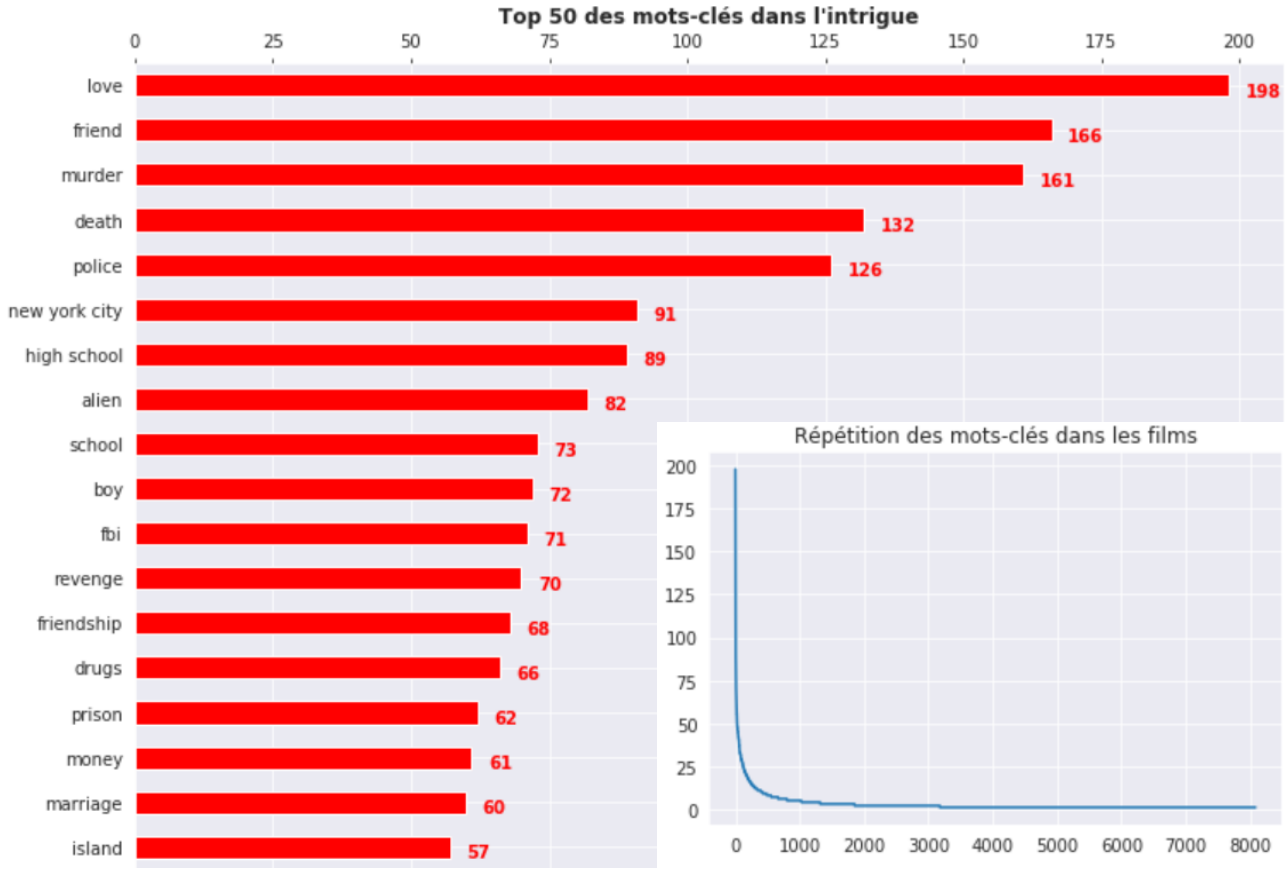


Tous les genres ne semblent pas appréciés de la même manière.

Attention au biais de sélection

1. Analyse exploratoire

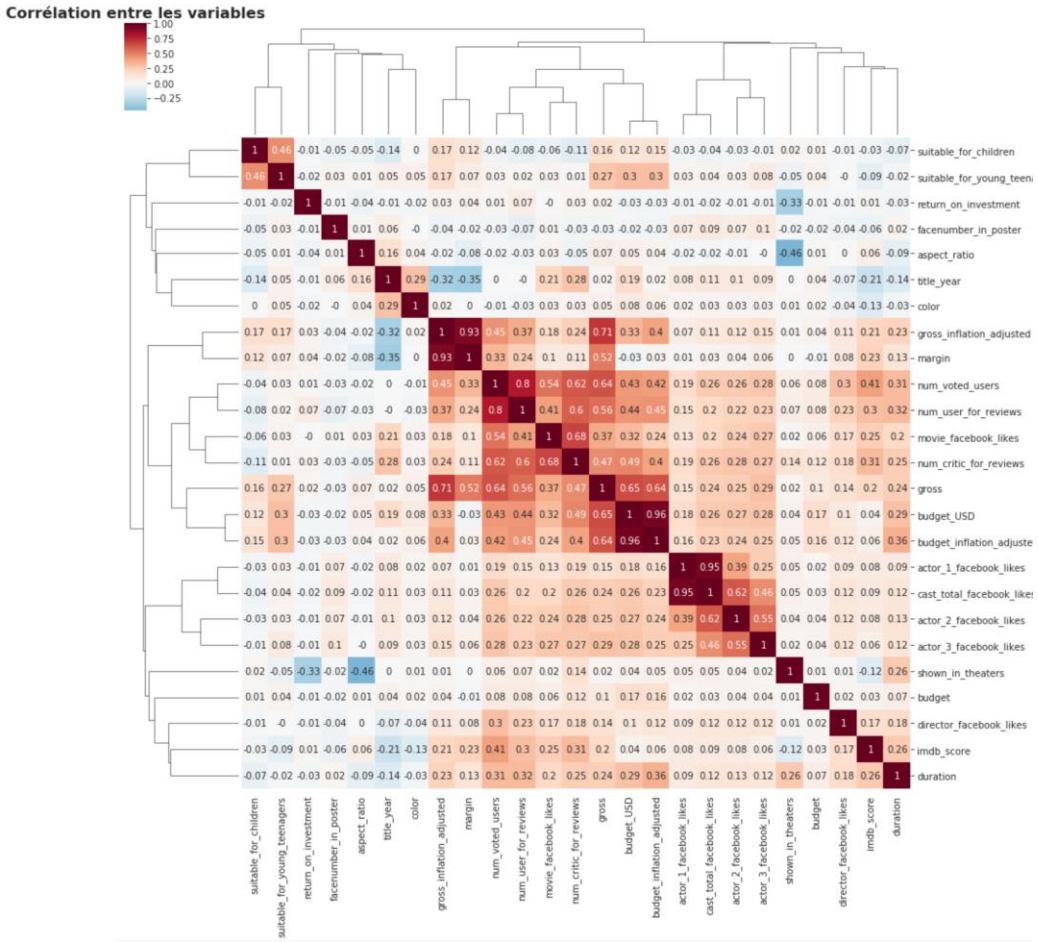
d. Analyse des mots-clés



- Certains mots-clés très populaires qui peuvent constituer des sous-genres : revenge, drugs par exemple.
- Longue queue de comète : 4908 mots-clés uniques

2. Modélisation

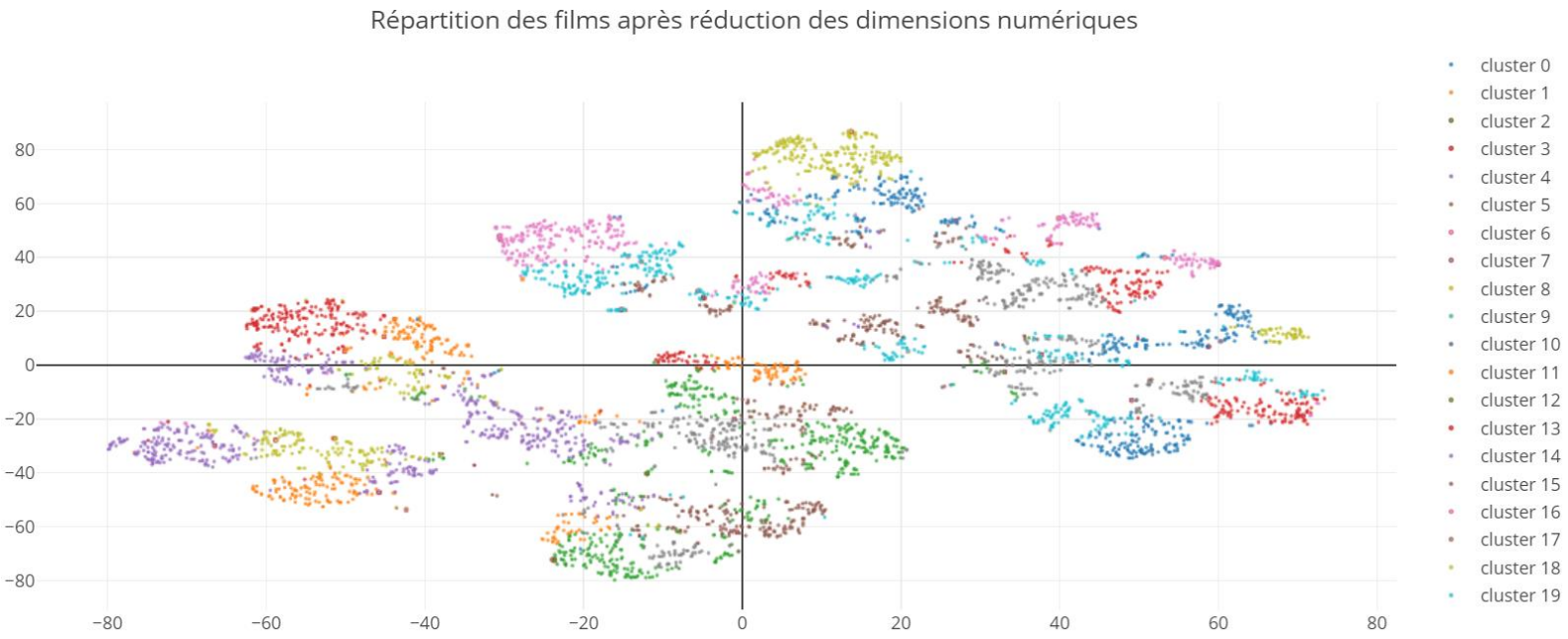
a. Transformation des valeurs numériques



- Nombreuses variables corrélées
- 4 groupes :
  - Investissement
  - Popularité
  - Age du film
  - Type de film
- Réduction des dimensions par PCA

2. Modélisation

a. Transformation des valeurs numériques



Clustering (Kmeans) + TSNE pour visualisation : insuffisant pour bien séparer les films

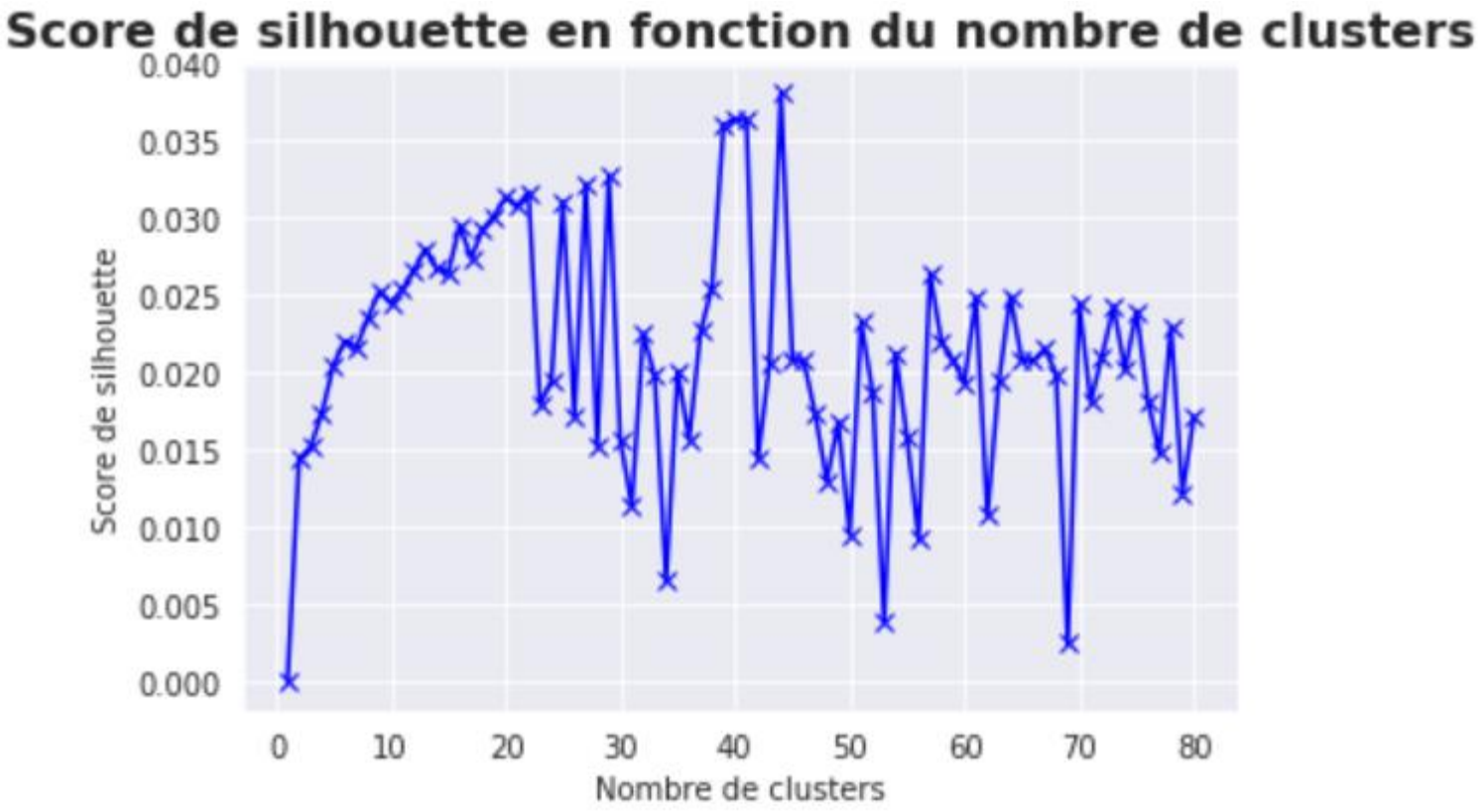
2. Modélisation

b. Transformation des valeurs textuelles : intrigue

Dummy encoding des mots-clés

Séparation des mots-clés et encodage binaire

39 clusters, avec cluster central regroupant plus de la moitié des films (3256).

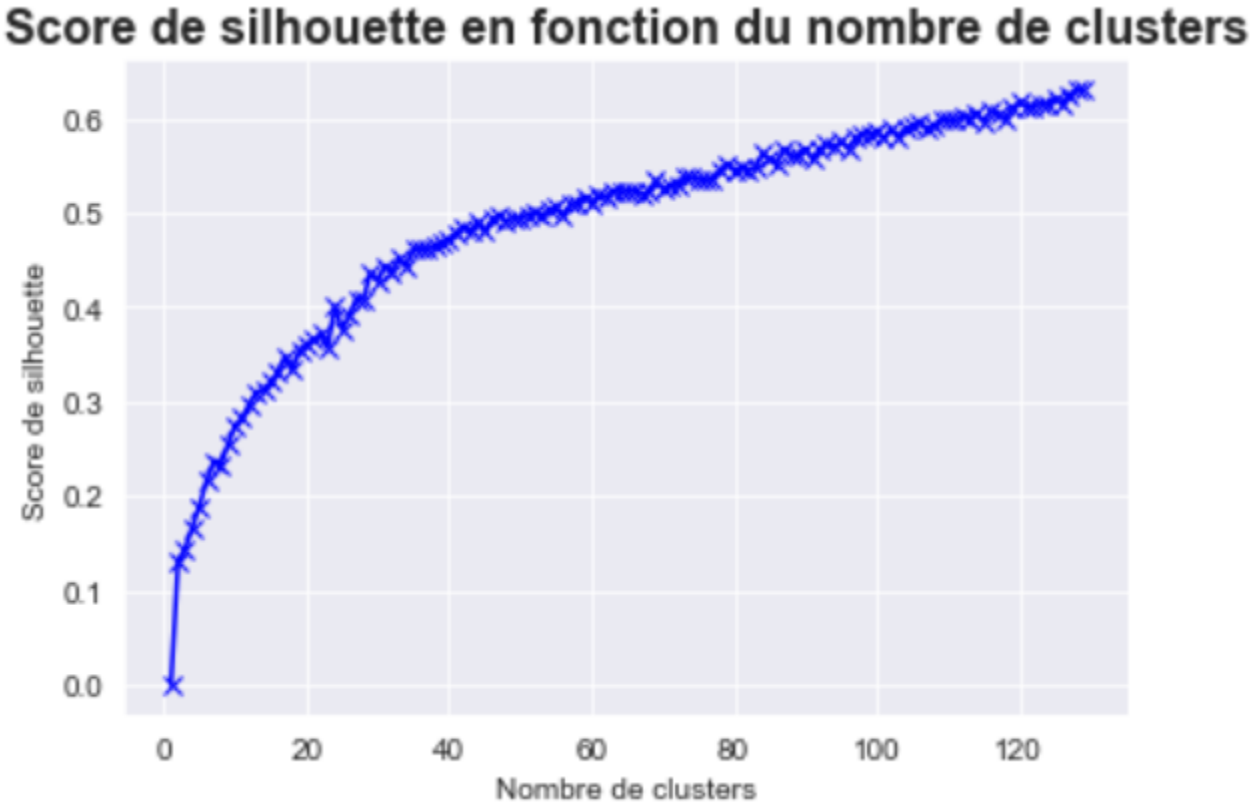


2. Modélisation

b. Transformation des valeurs textuelles : genres

Même logique d'encodage

Score de silhouette croissant, mais risque d'overfit et clusters trop petits



2. Modélisation

b. Transformation des valeurs textuelles : acteurs et réalisateurs

- Regroupement des acteurs puis dummy encodage

⇒Comparaison des acteurs communs, peu importe leur position

- Simple dummy encodage pour les réalisateurs

- Finalement, les champs ‘language’ et ‘country’ sont abandonnés dans la modélisation car peu précis et pas assez bien répartis

2. Modélisation

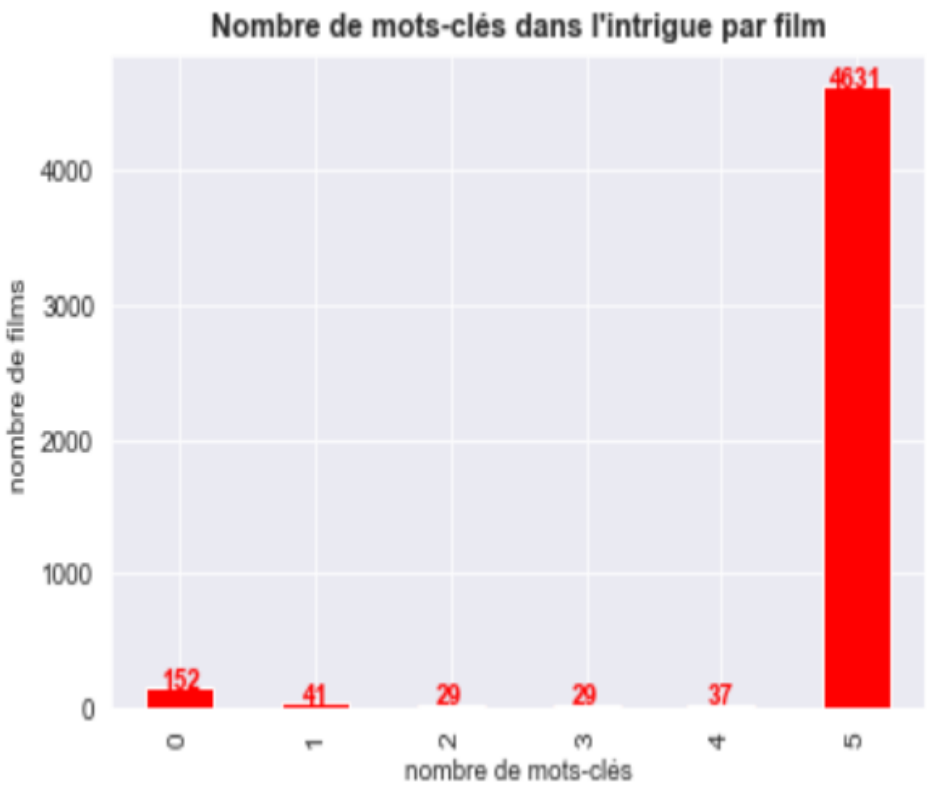
c. Premier regroupement des valeurs

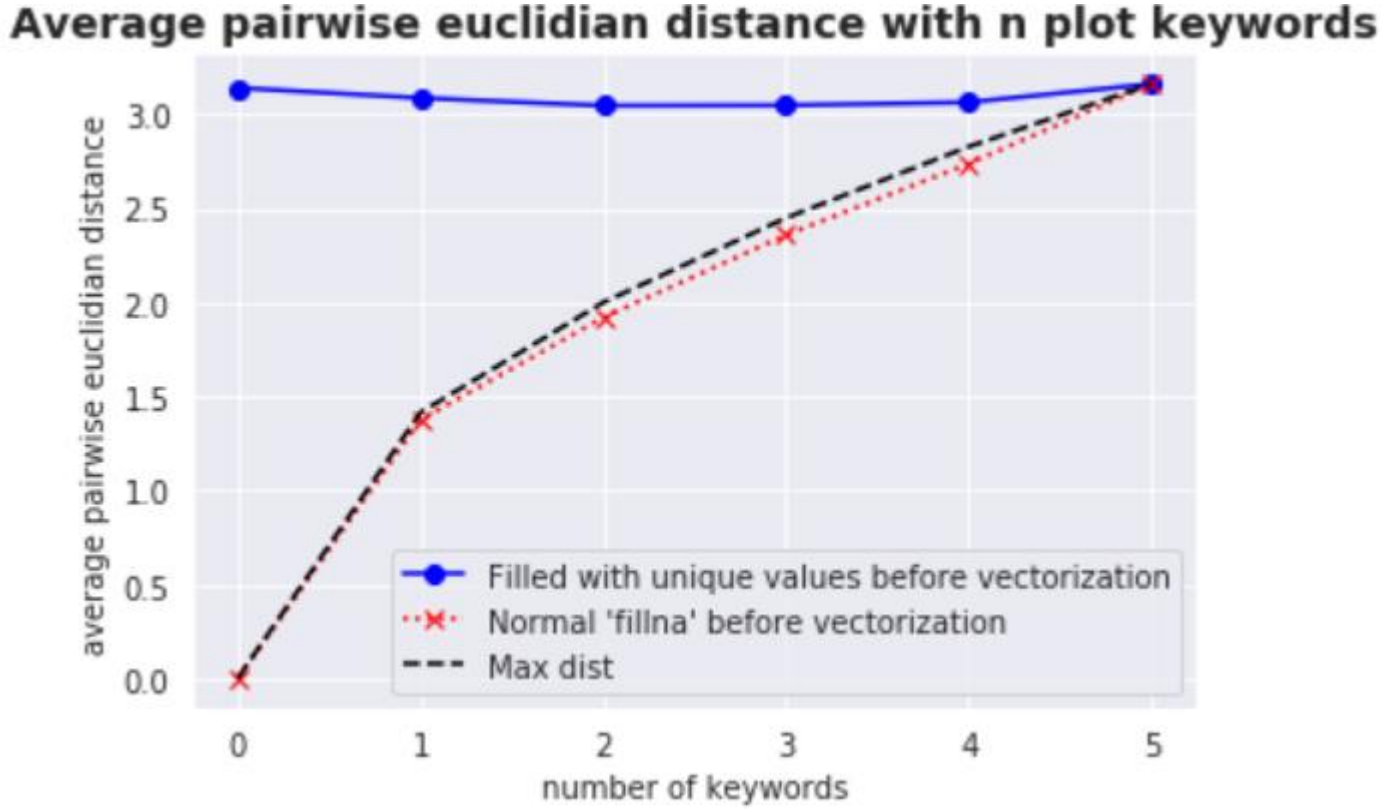
- Toutes les variables catégorielles ont été binarisées

- Première approche calcul de la distance euclidienne :
  - Calcul des plus proches voisins sur toutes les variables et après transformation
  - Premier problème, la distance entre paires est réduite quand il y a des valeurs manquantes

2. Modélisation

d. Justification de la complétion des valeurs manquantes





3. Déploiement et amélioration

a. Première approche

- Pré-calcul des 20 plus proches voisins, sauvegardé en CSV

- Principe utilisé : pour un film i, parmi les 20 plus proches voisins (j) on choisi au hasard 5 films différent. La probabilité dépend de :

$\frac{score\ imdb_j}{dist\ eucl(i,j)^2}$ si $score\ imdb_i \geq 5$, $\frac{1}{dist\ eucl(i,j)^2}$ sinon

3. Déploiement et amélioration

b. Déploiement

- Flask + Version gratuite de PythonAnywhere

- Accessible de deux manières :
  - Par l'id :

  => jeromehoen.pythonanywhere.com/recommend/<id>
  - Par le nom du film :

  => jeromehoen.pythonanywhere.com/recommend/fuzzy/<movie title

3. Déploiement et amélioration

c. Amélioration

- Problème constaté : distances euclidiennes souvent trop proches les unes des autres et peu compréhensibles => similarité cosinus

- Tests effectués :
  - Tous les coefficients des composantes à 1 = simple regroupement des colonnes
  - Coefficients personnalisés et personnalisables
    - 'popularity': 0.7,
    - 'investment': 0.7,
    - 'age': 0.6

    } pas toujours super pertinent pour comparer les films
    - 'plot': 0.5, car mix entre des thèmes communs et très présent "love" et de vrais sous-genre "vampire", "time travel"
    - 'director': 0.8 rejoint déjà un peu le genre du film
    - 'actors': 0.5 car pas 100% fiable

3. Déploiement et amélioration

c. Amélioration

Matrice des genres modifiée à plusieurs reprises :

1. Encodage binaire simple
   - ✅ Approche identique aux mots-clés,
   - ❌ Tous les thèmes ont la même valeur, certains sont plus que d'autres
2. Approche par TfIdf (logarithmique) ajustée
   - ✅ Thèmes rares mieux représentés
   - ❌ Des films rapprochés alors qu'Action|Drama|Thriller différent de Drama|Thriller (Ocean's Eleven et Side Effects)
3. Réduction des dimensions par PCA
   - ✅ Les thèmes similaires sont rapprochés
   - ❌ Perte d'information (80% de la variance retrouvée)

Conclusion

a. De bons résultats

- Approche multicritère et globale: genres, acteurs, réalisateurs, budget, âge sont pris en compte avec un score de similarité unique

- Des résultats qui semblent assez équilibrés : aucun groupe d'éléments ne prédominent entièrement sur les autres

- Testé par moi-même, permet des découvertes

Conclusion

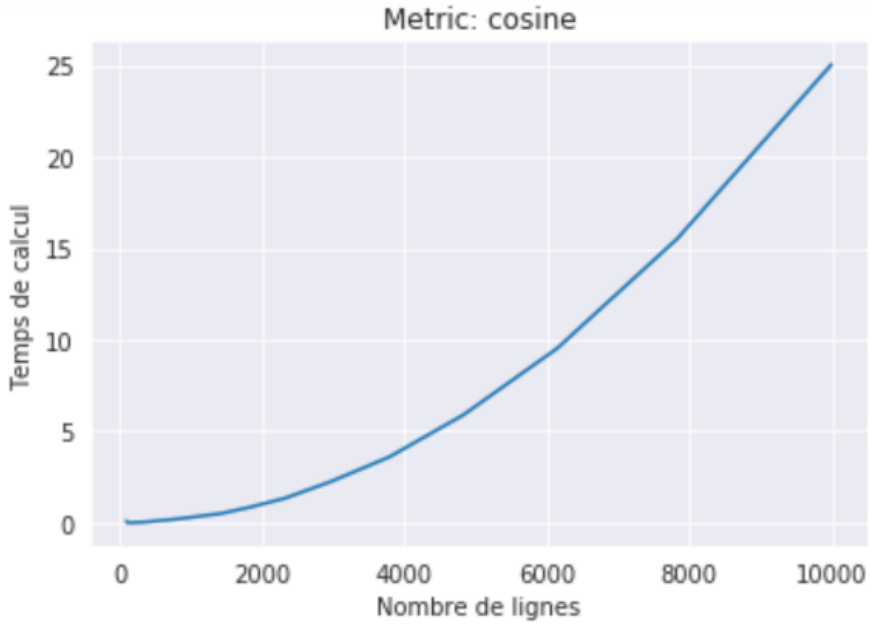
b. Limites

- Scalabilité : Complexité de l'ordre de **O(n²)**

⇒2 sec pour les voisins des 5000 films

⇒70 jours pour les 6 millions de films d'IMDb

Calcul à faire à la demande



- Testabilité : content-based vs. collaborative filtering

⇒Impossible de trouver une métrique pour qualifier les résultats

⇒Subjectivité

Conclusion

c. Pistes d'améliorations

- A/B testing : déploiement simultané de plusieurs variantes

- Validation par les utilisateurs (système d'upvote/downvote)

⇒Identifier les cas où les résultats ne sont pas bons

- Suggestion des utilisateurs :

⇒Laisser les utilisateurs compléter les listes fournies