CentraleSupélec

OPENCLASSROOMS

Développez un moteur de recommandations de films

Contenu

MINÉ Nicolas

Synthèse

Contexte :

Dans le cadre de la mise en place d'un site de recommandation de film, une société nous demande de leur fournir une solution de classification non supervisés afin de fournir aux utilisateurs une liste de 5 films similaires à celui vu.

Problème :

Ce problème est un problème d'apprentissage non supervisé. A l'aide de différents modèles de Clustering/Manifolds, une solution doit être proposée au client permettant de regrouper les films par similarité. Une fois regroupés, il sera possible de récupérer par distance les 5 films les plus proche (qui sont donc les plus similaires).

Données :

Les données fournies sont des données issues d'IMDb. Il regroupe 5000 films avec 28 features (titre, date de sortie, note, acteurs, durée, etc…).

Approches :

Après un nettoyage de données ainsi que certaines simplifications, différents algorithmes de Clutering (Kmeans, Agglomerative Clustering et DBScan), de Manifolds (LLE, TSNE, Isomap) ainsi que de décomposition (PCA) vont être testés. Pour chacun d'entre eux, la prédiction sur un même film sera faite afin de l'évaluer. Pour finir, une prédiction sera faite sans modèle afin de comparer la proposition sans avoir aucun modèle.

Performances des modèles :

La performance de ces modèles ne pouvant pas être évaluée par une métrique précise. La comparaison se fera sur un film de référence.

Résultats :

De par la conservation des distances au niveau global, le TSNE est le modèle le plus pertinent sur une série de film. Il se rapproche du modèle simple (juste basé sur la distance euclidienne) grâce à une optimisation de celui-ci par rapport à son kl-divergence.

Une API a aussi été mise en place pour permettre à des utilisateurs de se voir proposer des films.

Introduction

A partir d'une base de données publique concernant des films, un site désire faire un moteur de recommandation de films. Celle-ci ne possédant pas encore d'informations sur les utilisateurs, nous n'avons accès qu'aux informations de films. A l'aide d'algorithmes non supervisés, l'objectif est de fournir une petite API permettant d'avoir 5 films similaires à celui vu.

Ce rapport va donc présenter le nettoyage fait, le choix de la simplification de certaines features, le test de différents modèles avec une comparaison sur un film de référence pour pouvoir comparer les différents modèles, pour finir par la présentation de l'API.

Nettoyage

Le dataset fourni est assez complet. Celui-ci regroupe 5 043 films avec 28 caractéristiques. Dans celui-ci, il ne manque que 2698 points sur 141 000 soit 1.91 % du dataset. De ce fait, une grande partie des données manquantes devraient être ajoutée sans trop biaiser le dataset.

Par la suite, certaines features vont être simplifiées ou supprimé avant de mettre en place les modèles. L'ensemble des étapes est présenté ci-dessous avec les choix faits.

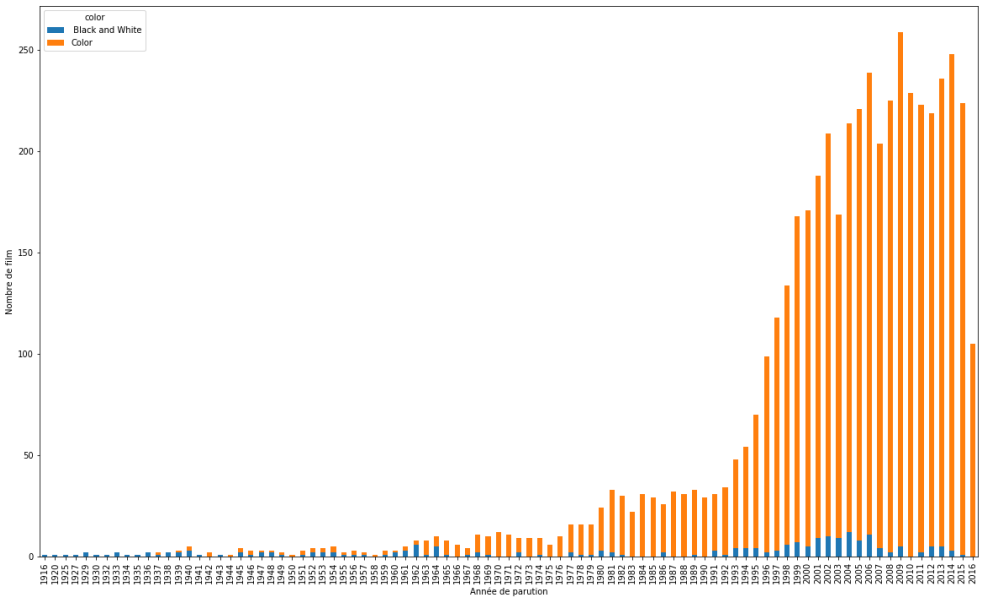
Phase 1 : Ajout des données manquantes

Pour l'ajout de données, chaque feature a été explorée de façon univariée. A l'aide d'histogrammes ou de pies. Cela a permis de faire ressortir les tendances et les façons de compléter le dataset. De cette étude, 5 principaux groupes à compléter sont ressortis :

Groupe 1 : Basé sur la date

Cette sélection n'est utile que pour la feature "Color". En effet, le dataset ne possède que très peu de films en Noir et Blanc comparés aux films en couleurs (209 vs 4815) Parmi l'ensemble des films, seulement 19 ne sont pas labélisés.

Si on regarde les dates des films en Noir et blanc, ils sont en grande proportion jusqu'en 1967 et deviennent rare (toujours en proportion) par la suite.



Au vu de la date des 19 films sans l'information de couleur, il est très probable qu'ils soient en couleur. On peut aussi le confirmer en regardant les titres (bien que ce soit quelque chose de peu faisables sur des plus gros dataset).

Groupe 2 : Données de "comptage":

On appellera pour la suite "features de comptage" toutes les features du type "nombre de likes/votes". Ces features ont parfois des données manquantes. Dans ce cas, on peut considérer qu'il n'y a pas de likes, de visages sur l'affiche, etc. De ce fait, les données manquantes sont remplies avec des 0.

Groupe 3 : Données de numériques:

Pour les features numériques suivantes :

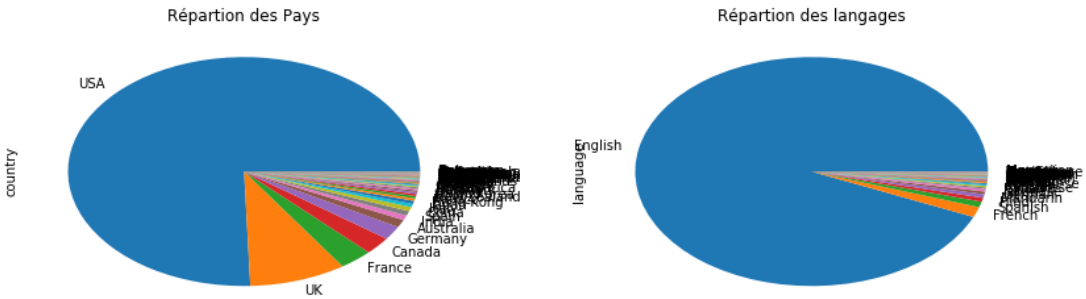
- "Durée"
- "Nombre de ventes"
- "budget"
- "année de sortie"

Le remplissage avec des 0 aurait moins de sens. Cela signifierait que la durée est de 0 minutes, un budget nul, etc. De ce fait, une façon plus logique est d'utiliser la moyenne. Cependant pour le budget, cela n'est peut-être pas le juste car les films dont on ne connait pas le budget sont généralement des plus petits films qui ont donc un budget plus petit que les Blockbuster.

Dans le cas où beaucoup de points seraient manquants, une autre solution aurait été possible. A la place de mettre toujours la moyenne, il aurait été possible de donner une valeur aléatoire répartie suivant une loi Normale centrée sur la moyenne et avec un écart type identique à celui de la feature. Cependant, comme il y a très peu de points à remplir, la moyenne ne fausse pas particulièrement la répartition.

Groupe 4 : Données géographique:

Pour les features "Pays d'origine" et "Langue originale", le choix a été fait en se basant sur la majorité. Les USA produisent 75.5 % des films et 93.5% des films sont en Anglais



De ce fait, la complétion s'est faite avec "USA" pour le pays d'origine et "English" la langue. Concernant ces valeurs, on peut aussi regarder les directeurs des films avec la langue ou le pays d'origine manquant. Si celui-ci a produit d'autres films, on peut avoir la vraie donnée via celle-ci. Cette analyse fait manuellement, renforce cette façon de compléter car sur les 11 directeurs, 5 ont fait plus d'un film et parmi ceux-ci, tous sont fait aux USA et sont en Anglais.

Groupe 5: Noms manquants

Pour ces catégories, il n'est pas possible d'estimer quelques choses. Du coup en attendant la simplification que l'on verra par la suite, les noms seront mis comme "None".

A ce stade, le dataset n'a plus d'informations manquantes, on peut donc passer à la simplification

Phase 2 : Simplification de features

La phase de simplification a été mise en place sur 5 features.

Feature 1 : Les genres de film

Chaque film possède une liste de genre. Le dataset complet possède 21 genres différents mais les films en ont plusieurs. Si on fait un nuage de mots avec leur fréquence d'apparition, cela donne :

Pour la simplification, un OHE a été utilisé sauf qu'à la différence du OHE standard, plusieurs colonnes peuvent être à 1. Ce choix a été fait par rapport au LabelEncoder qui n'a pas de sens en terme de distance. En effet, si Sci-Fi a pour label 2 et Comedy le label 3, les 2 groupes très différents seraient considérés comme proches lors du clustering.

Feature 2 : Les Ratings

Pour cette feature, il y avait déjà des mixes de réglementation, des typos, etc. Du coup, il y a déjà eu une simplification par gamme d'âge (tout âge, 7 ans mini, 13 ans mini, 16 ans …). Suite à cela, il ne restait que peu de valeurs. J'ai donc aussi appliqué un OHE mais cette fois ci de manière un peu plus complexe. En effet, au lieu de mettre 1 dans la colonne correspondant, le 1 se retrouve aussi dans toutes les colonnes suivantes. Par exemple, un film pour tout âge sera valide pour toutes les colonnes de ce dataset. Par contre un film classé pour les adolescents de plus de 13 ans n'aura 1 que avec les colonnes suivantes (16 ans, 17 ans).

Cela permet à une personne ayant regardé un film classé pour les adolescents de 13 ans, de voir aussi des films pour tout âge. Au contraire si quelqu'un regarde un film pour les enfants, les films pour les adolescents seront "plus loin" dans le clustering car les films plus violents seront à 1 au lieu de 0. Il aurait été possible de filtrer sur le site sur ce critère mais un adulte ayant aimé un dessin animé, devrait pouvoir tout de même avoir des propositions de films plus violents s'ils sont similaires sur tous les autres points. C'est pour cela que j'utilise ce type de OHE.

Feature 3-5 : Les Noms

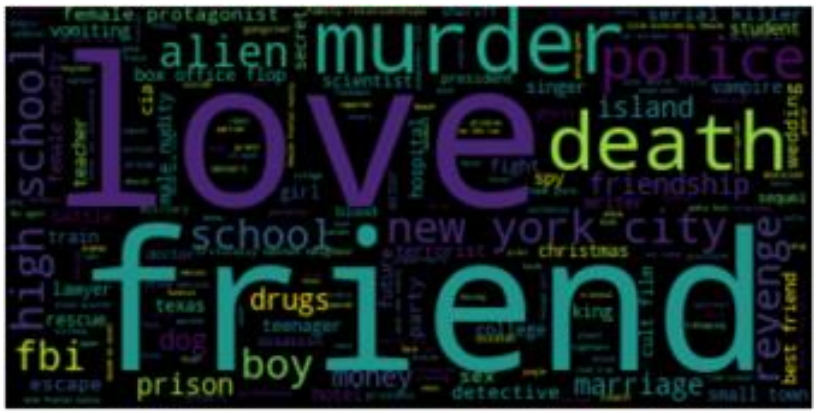
Pour cette dernière feature, on ne peut pas utiliser de OHE car il y a trop d'acteurs possibles. La solution était donc le LabelEncoder. Cependant pour améliorer cet encodage, le label sera égal au nombre d'apparition de cet acteur (en tant qu'acteur 1, 2 ou 3).

On trouve donc par fréquence :

L'intérêt de cette méthode est de regrouper ensemble les acteurs connus et éloigner les acteurs moins connus. Le défaut de cette méthode est que parfois la classification peut estimer que par exemple Nicolas Cage et Johnny Depp sont les mêmes acteurs et donc ne pas proposer le bon film. Cependant, la distinction entre 2 acteurs peut aussi se faire avec le nombre de likes. En opposition, l'intérêt est de grouper les petits films indépendants et les éloigner des blockbusters un peu plus que seulement par le budget.

Phase 3.1 : Suppression des features

Pour les features, cela a été fait initialement basé sur le type de donnés. Par exemple la feature "mot clé", ne peux pas être encodé avec le OHE car il y a 101 mot clés et cela n'est pas très utile car c'est subjectif.



J'ai donc décidé de supprimer la feature.

Le ratio aussi a été supprimé car beaucoup de points étaient manquants et il est difficile d'estimer le ratio.

Le nom du film lui n'est pas nécessaire non plus pour le clustering, c'est comme son ID. C'est la même chose avec le lien IMDb.

Suite aux phases de modélisation, d'autres features vont être supprimées car l'excès de feature rend une feature particulière moins importante (car elle pèse moins). C'est le cas avec la couleur des films, le budget, les likes etc.)

Certaines features ont aussi été supprimées car leur corrélation étant forte, on ne peut garder qu'un des 2 paramètres.

Phase 3.2: Suppression de films

Après ce nettoyage vertical, il y a eu un tout petit peu de nettoyage horizontal. Les films avec les directeurs manquants vont être supprimés. Cependant, ce nettoyage n'a supprimé que 126 films sur 5043.

Phase 4: Encodage

Comme les distances vont être utilisées pour faire le clustering, il ne faut avoir que des nombres dans le dataset. A ce stade, le dataset ne possède plus que peu de features de type Objects. Pour ce faire, un LabelEncoder a été appliqué sur celles-ci.

Par la suite, un scaling est nécessaire pour mettre à l'échelle chaque features. En effet, sans le scaling, l'impact du budget par rapport à la note sur 5 sera énorme. De ce fait, le clustering sera complétement faussé. Pour ce faire, j'ai utilisé le MinMaxScaler. Dans ce cas-ci, le Standard Scaler ne serait pas pertinent car la variance de certains paramètres comme le budget serait conservé et impacterait toujours la qualité du clustering. A ce stade, le dataset est prêt pour la modélisation.

Modélisation

Pour la modélisation, on a accès à 3 principaux groupes dans Scikit-Learn.

Le groupe des Cluster qui a pour objectif de grouper les inputs par similarité dans n clusters. En fournissant un nombre de groupe, l'algorithme cherche un regroupement possible de façon à minimiser la distance entre les éléments du cluster.

Le second groupe est le groupe des Manifolds. Celui-ci regroupe des algorithmes qui ont pour objectif de positionner dans un espace de plus petite dimensions les éléments tout en maintenant les distances inter-éléments de manière locale ou globale. Ce groupe n'a pas pour objectif de regrouper dans un même cluster les points les plus proches mais uniquement de réduire les dimensions afin de les rendre visualisable.

Le dernier groupe est celui des "Décompositions". Dans celui-ci, on trouve le PCA qui a pour objectif de réduire le dataset d'une dimension n à m (m < n) en conservant un maximum de variance. Pour-cela, le PCA cherche une relation linéaire entre les features.

On va donc explorer différents algorithmes de chaque groupe et regarder la prédiction pour chacun d'entre eux sur un film en particulier.

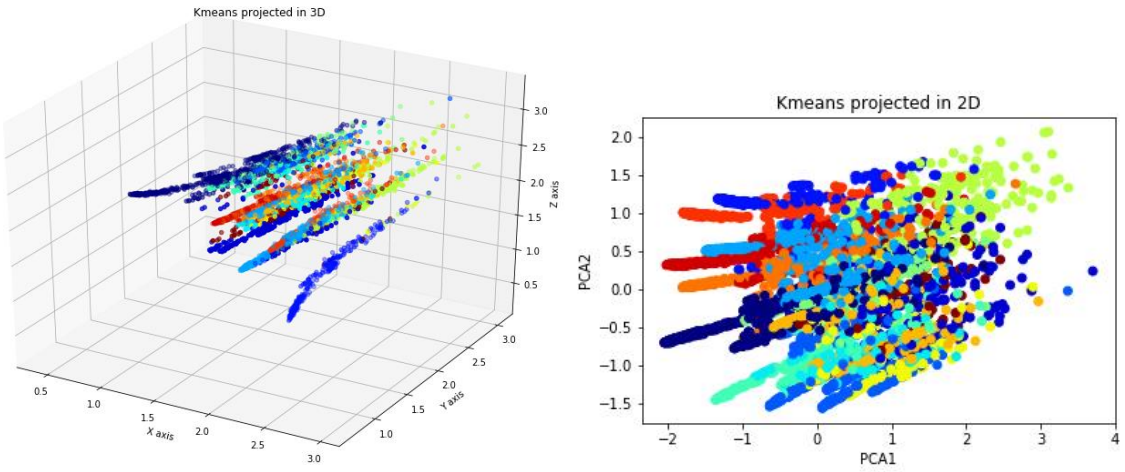
Groupe Cluster

Comme expliqué précédemment, le principal problème de ce groupe est le nombre inconnu de groupes car on n'a pas de labels. On peut cependant essayer de visualiser ce qu'il donne. Pour cela on va passer au travers de 3 algorithmes les plus connus.

K-means

Pour le K-means, le nombre de clusters est fourni en entrée. L'algorithme retourne un label pour chaque film ainsi qu'une nouvelle position dans un espace de dimension réduite. De ce fait on peut visualiser le partitionnement avec les labels et on peut aussi faire un recommandeur de films car on a la position dans l'espace. Il est donc possible pour un point donné de regarder les n points les plus proches qui correspondent aux films similaires.

On trouve par exemple avec 15 clusters la répartition suivante :



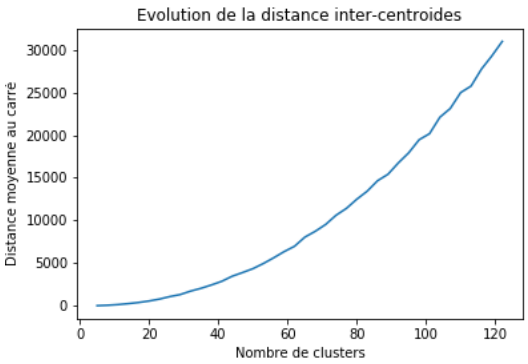
An niveau positionnement, on peut remarquer que les séparations sont assez franches pour un y faible mais se mélange suivant les y élevés. De plus, chaque cluster est très étiré. Cela est un problème car un point d'un cluster pour être plus proche d'un autre cluster qu'un point éloigné de son propre cluster rendant la prédiction moins bonne (sauf si l'on filtre par cluster)

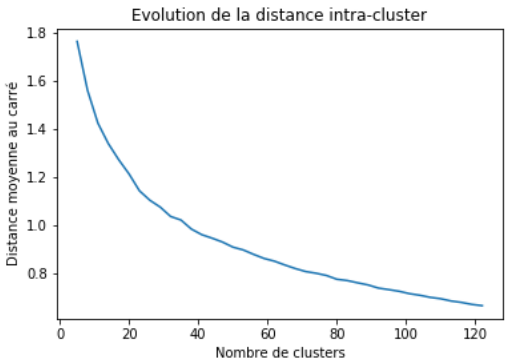
Quant aux résultats, si on regarde les 5 films les plus proches de Spider-Man 3 on trouve :

- Ironclad
- Prince of Persia: The Sands of Time
- Krrish
- The Three Musketeers
- The Musketeer

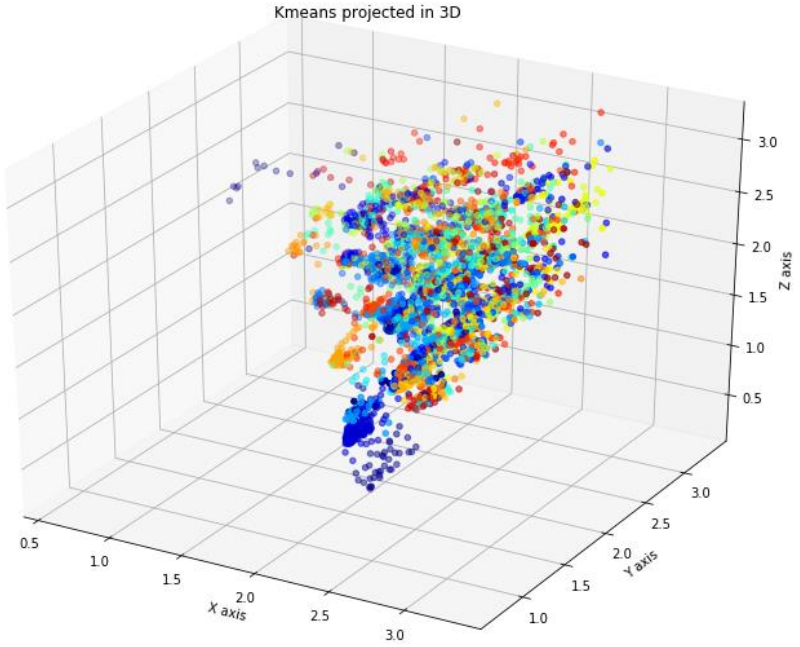
A première vue, le clustering semble curieux car on ne trouve pas d'autres Marvels ou d'autres Spider-man.

Un des bémols avec cet algorithme est que l'on ne connait pas le nombre de cluster. En effet, plus le nombre de cluster augmente, plus la distance inter-cluster augmente et plus la distance intra-cluster diminue. Le partitionnement est donc meilleur mais il n'existe pas de moyen de savoir quel est le nombre de cluster optimal.





De plus, la répartition est très différente en fonction du nombre de cluster comme on le voit ci-dessous avec 125 clusters:



Si maintenant on regarde le choix de 5 films avec ce clustering on a :

- **Ironclad**
- **The Three Musketeers**
- The Lone Ranger
- **The Musketeer**
- **Krrish**

Bien que la répartition soit très différente, 4 films sur 5 sont identiques. Cela "montre" la stabilité de cet algorithme mais le choix reste peu pertinent au vu des films proposés.

DBScan

L'intérêt du DBScan est de ne pas avoir besoin d'un nombre de cluster. Il le détermine lui-même par proximité basé sur 2 seuils que l'on donne en entrée (rayon du cercle et nombre de points voisins). Celui-ci "propage" le label à tous ses voisins. Le problème est qu'il ne fournit pas de distances ou tout autre moyen de trouver les 5 films les plus similaires en utilisant une distance. De plus, le contenu de chaque cluster est variable en taille (certains cluster vont avoir 10 films et d'autres 60). De ce fait, on ne pourra pas faire un recommandeur avec ce modèle.

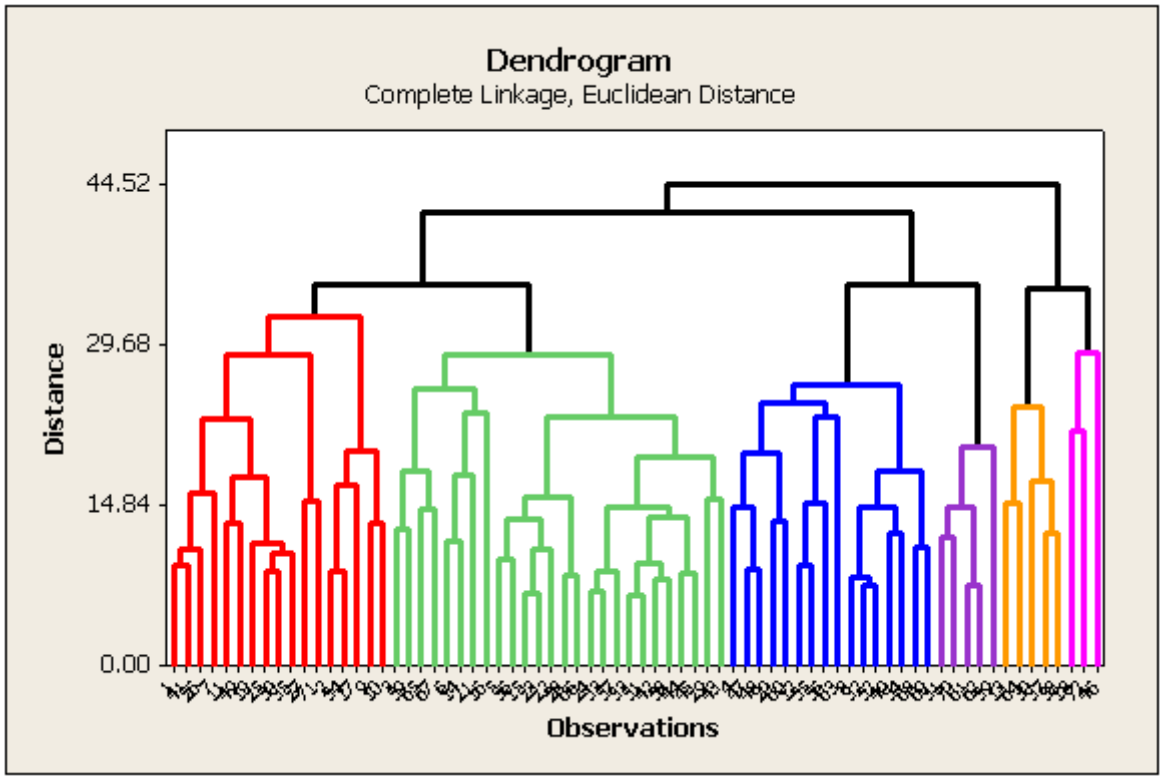
On peut cependant regarder le contenu prédit des clusters pour visualiser le résultat (par exemple avec le Cluster 6).

- Terminator 3: Rise of the Machines
- The Matrix Reloaded
- Hulk
- Total Recall
- Terminator 2: Judgment Day
- Dredd
- Battle Los Angeles
- Æon Flux
- Universal Soldier: The Return
- The Black Hole
- Megaforce
- The Terminator
- Escape from New York
- Escape from the Planet of the Apes
- Battle for the Planet of the Apes
- Conquest of the Planet of the Apes

On remarque tout de même que le groupe est assez homogène. Il regroupe quelques suites (la série de la planète des singes ou Terminator 2 et 3) et il n'a que des films d'actions.

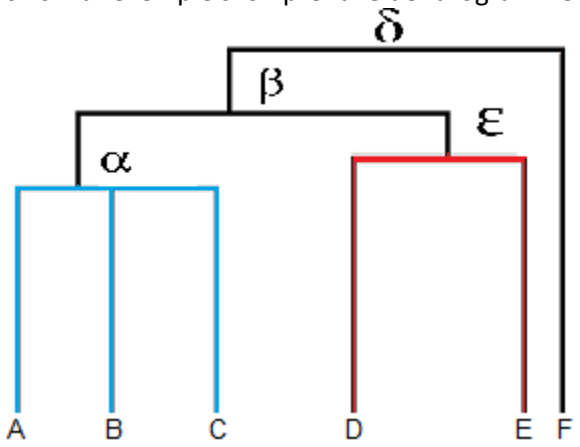
Agglomerative Clutering

Le Clustering Agglomeratif est plus particulier. Il groupe les films 2 par 2 que l'on peut schématiser par un dendrogramme comme ci-dessous.



Ensuite, en fonction du nombre de cluster demandé, il retourne l'ensemble des enfants de chaque branche. Dans l'exemple ci-dessus, chaque couleur correspond à un cluster (on en a donc 6).

De cette manière on peut donc trouver les films similaires en remontant dans le dendrogramme puis en redescendant sur les voisins. Par exemple si on prend le dendrogramme suivant :



Si on recherche les 5 voisins à B, on remonte au lien alpha. Ensuite on redescend sur chaque enfant. On a donc A et C. Comme on n'a pas 5 films, on remonte à beta. Beta nous propose comme enfant alpha et epsilon. Comme on a déjà visité alpha, on passe sur epsilon et on trouve en enfant D et E.

En appliquant cette méthode, on trouve pour Spider Man 3:

- A Knight's Tale
- **The Three Musketeers**
- **The Musketeer**
- Prince of Persia: The Sands of Time
- Spider-Man 2 et Spider-Man

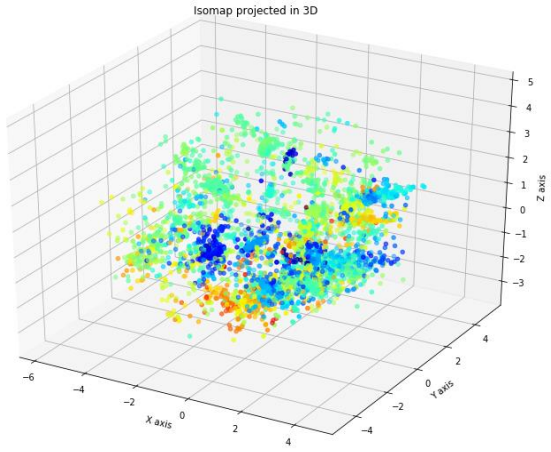
La dernière ligne contient Spider-Man et Spider-Man 2 car ils appartiennent tous les 2 à une blanche externe au film (comme epsilon dans l'exemple précédent). Du coup, les 2 sont à égalité. Coté analyse, on retrouve les précédent Spider-Man ce qui est bon signe ainsi que les 3 mousquetaires et D'Artagnan que l'on a trouvé précédemment aussi.

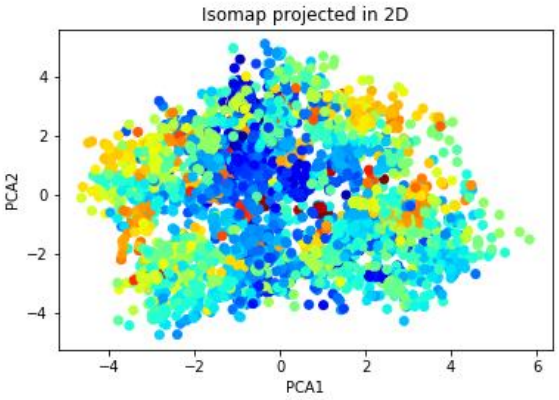
Groupe Manifolds

Pour ce projet, le groupe des manifolds est bien plus logique. En effet, celui-ci cherche à conserver les distances de manière locale ou globale mais dans une dimension plus petite. De ce fait, on peut toujours visualiser le clustering et aussi calculer des distances dessus. De ce fait, on n'a pas besoin non plus d'un nombre de clusters.

Isomap

Avec l'algorithme Isomap, la projection dans un espace 3D donne la représentation suivante en 3D ainsi que sa projection 2D avec le PCA :





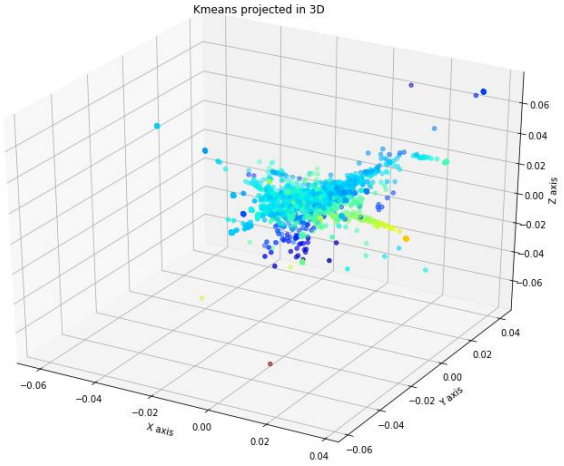
On a tendance à voir des groupements de points mais le groupe reste assez "brouillon". On peut cependant rechercher les 5 films les plus proches à Spider-Man 3 comme on avait fait avec le K-means et on trouve :

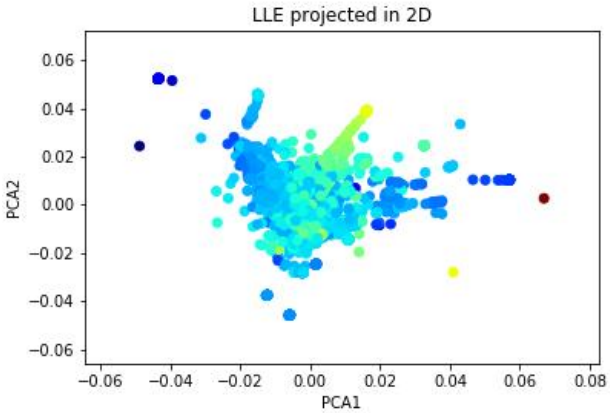
- Repo! The Genetic Opera
- **A Knight's Tale**
- **The Three Musketeers**
- **Ironclad**
- Bram Stoker's Dracula

On retrouve 3 films que l'on avait avec le Kmeans ou le Clustering Hierarchique mais les autres sont nouveaux et parfois assez loin de Spider man en terme de style de film (par exemple Dracula).

Locally Linear Embedding

En appliquant exactement le même algorithme sur le Locally Linear Embedding on trouve:





Au niveau décomposition, l'ensemble des points sont groupés au centre et la sélection des films sera donc mauvaise. Cela est dû au fait que le LLE cherche à maintenir les distances au niveau local uniquement. Pour les films très à l'extérieur, les films proposés seront aussi très différents et ne plairont pas forcément à l'utilisateur. De ce fait, cet algorithme n'est pas particulièrement fait pour cette tâche. Afin de comparer on peut tout de même regarder la prédiction sur Spider Man 3:

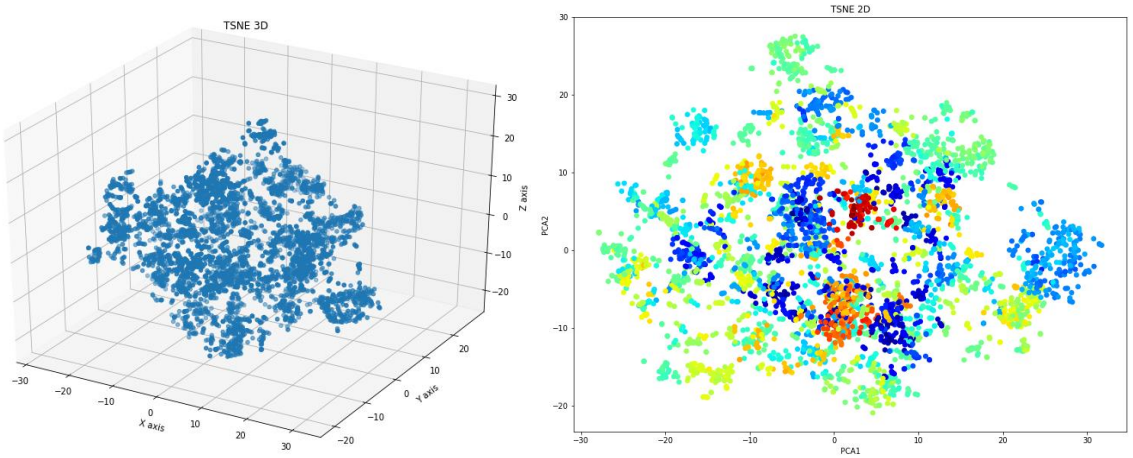
- **A Knight's Tale**

- **The Three Musketeers**
- The Shawshank Redemption
- Pulp Fiction
- It Happened One Night

On retrouve 2 films déjà trouvé précédemment mais les 3 autres sont très différents (par exemple Pulp Fiction et Spider Man n'ont rien en commun).

TSNE

Pour finir avec les manifolds, on va utiliser l'algorithme le plus en vogue actuellement, le t-Distributed Stochastic Neighbor Embedding. Celui-ci est très apprécié par sa qualité de partitionnement. Si on applique le même algorithme sur celui-ci on trouve :



On remarque tout de suite qu'il y a comme des nuages de points plutôt distincts, ce qui est bon signe. On peut ensuite regarder sa prédiction sur Spider man 3 et on trouve :

- Spider-Man
- Spider-Man 2
- **The Three Musketeers**
- **The Musketeer**
- **A Knight's Tale**

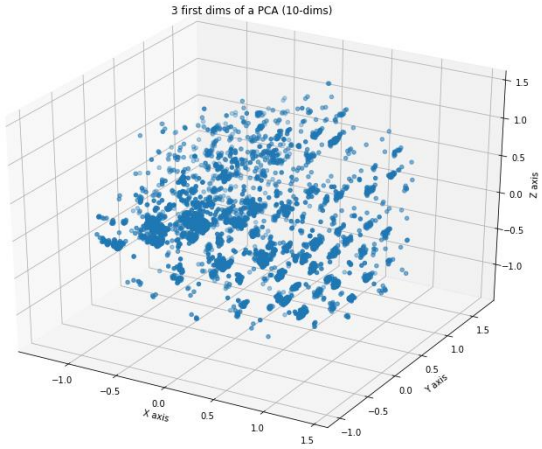
On retrouve 3 films trouvés précédemment mais surtout on trouve les suites de ces films (aussi partiellement trouvé par le Clustering Hierarchique). Ce résultat est très prometteur et est, pour l'instant, le modèle à optimiser et utiliser dans le moteur de recommandations.

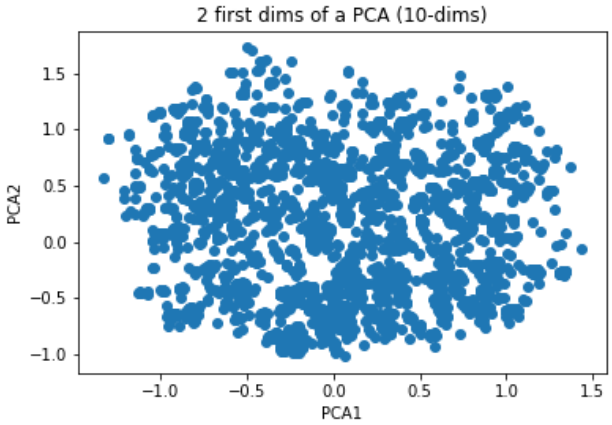
Groupe Decomposition

L'objectif de ce groupe est de réduire le nombre de dimension à l'aide de combinaisons linéaires entre les paramètres. Il n'est généralement pas utilisé pour le clustering.

PCA

On peut appliquer les PCA comme ce que l'on a fait avec les manifolds. Si on l'applique pour passer en 10 dimensions et visualiser uniquement les 3 premières dimensions on a les représentations suivantes :





Cependant ce modèle n'est pas bon au clustering car 10 dimensions n'expliquent que 70% de la variance. Cependant, si on regarde les prédictions sur Spider Man 3 on trouve :

- **The Three Musketeers**
- **The Musketeer**
- The Charge of the Light Brigade
- **Ironclad**
- Red River

On retrouve 3 films déjà trouvés précédemment et 2 nouveaux. En ayant que 70 % de l'information, c'est déjà un bon résultat mais tout de même pas aussi bon que le TSNE.

Modèle Simple – Distance Euclidienne

Le dernier "modèle" testé est basé simplement sur la matrice encodée et scalée. Le calcul de distance est fait directement sur celui-ci sans passer par un clustering/manifold. L'intérêt est d'avoir directement toute l'information. De plus, n'ayant pas d'algorithme autre que ce calcul de distance, il ne nécessite pas de refaire le modèle à chaque ajout de film. Celui-ci fait du *online-learning*.

En appliquant la distance Euclidienne on trouve :

- **Spider-Man 2**
- **Spider-Man**
- **The Three Musketeers**
- **A Knight's Tale**
- **The Musketeer**

On retrouve l'ensemble des films trouvé par le TSNE mais dans un ordre légèrement différent. La variation vient de la perte de données lors de la projection dans un espace de plus petite dimension.

On va donc garder aussi ce modèle pour le site afin de pouvoir visualiser s'il y a des différences plus flagrantes parfois.

Modèle Final

Mise en place

Comme expliqué ci-dessus, on va utiliser le TSNE et le modèle de distance simple. On va dans un 1er temps optimiser le TSNE avec un petit Grid-Search. L'évaluation sera faite sur le kl_divergent fourni avec le TSNE. Cette valeur décrit l'entropie (le désordre) dans le modèle lors de la réduction dimensionnelle. On va donc tester divers paramètre et garder le meilleur (un kl-divergent faible).

Le meilleur paramètre est :

- "n_components":3
- "n_iter":500,
- "n_iter_without_progress":150

Celui-ci donne un kl_divergent de 0.7571.

Pour le fonctionnement, on a 2 datasets, un est encodé pour fitter le modèle et le second est non encodé et ne conserve que les données utiles pour le site (titre, date, acteurs etc..).
Une fois le modèle fitté, on va concaténer les points dans des colonnes additionnelles du dataset non encodé. En effet, on ne peut pas fitter le modèle à chaque demande de l'utilisateur. Une fois fait, le dataset sera sauvegardé pour être utilisé sur le site.

Lorsque le site recevra une requête, il récupèrera l'index du film et avec les positions des points stockés, on utilisera la distance euclidienne pour récupérer les index des points les plus proches et ainsi retourner les données de ces films.

Pour le modèle simple, ce sera la même chose mais avec le dataset encodé.

API

L'API a été faite sur Flask. Elle est très légère (un peu de bootstrap, une page d'accueil et de recommandation). Sur la page d'accueil, on peut choisir soit par ID, soit par titre. Une fois la recherche lancée, le serveur cherche les 5 points les plus proches basés sur les résultats du TSNE et retourne ces données à un template pour l'affichage.

Le site est disponible à cette adresse : http://coni57.pythonanywhere.com/

Vous avez vu :

Titre : Spider-Man 3 - 2007
Durée (min) : 156
Type : Action|Adventure|Romance
Origine : USA
IMDB score : 6.2
Facebook likes : 0.0
Directeur : Sam Raimi
Principaux acteurs : J.K. Simmons, James Franco, Kirsten Dunst

Vous aimerez peut-être :

TSNE :

Titre : Spider-Man - 2002
Durée (min) : 121
Type : Action|Adventure|Fantasy|Romance
Origine : USA
IMDB score : 7.3
Facebook likes : 5000.0
Directeur : Sam Raimi
Principaux acteurs : J.K. Simmons, James Franco, Kirsten Dunst

Titre : Spider-Man 2 - 2004
Durée (min) : 135
Type : Action|Adventure|Fantasy|Romance
Origine : USA
IMDB score : 7.3
Facebook likes : 0.0
Directeur : Sam Raimi
Principaux acteurs : J.K. Simmons, James Franco, Kirsten Dunst

Titre : The Three Musketeers - 2011
Durée (min) : 110
Type : Action|Adventure|Romance
Origine : Germany
IMDB score : 5.8
Facebook likes : 19000.0
Directeur : Paul W.S. Anderson
Principaux acteurs : Milla Jovovich, Logan Lerman, Orlando Bloom

Titre : The Musketeer - 2001
Durée (min) : 104
Type : Action|Adventure|Romance
Origine : Germany
IMDB score : 4.7
Facebook likes : 299.0
Directeur : Peter Hyams
Principaux acteurs : Catherine Deneuve, Justin Chambers, Stephen Rea

Titre : A Knight's Tale - 2001
Durée (min) : 144
Type : Action|Adventure|Romance
Origine : USA
IMDB score : 6.9
Facebook likes : 0.0
Directeur : Brian Helgeland
Principaux acteurs : Heath Ledger, Rufus Sewell, Bérénice Bejo

Modèle simple :

Titre : Spider-Man 2 - 2004
Durée (min) : 135
Type : Action|Adventure|Fantasy|Romance
Origine : USA
IMDB score : 7.3
Facebook likes : 0.0
Directeur : Sam Raimi
Principaux acteurs : J.K. Simmons, James Franco, Kirsten Dunst

Titre : Spider-Man - 2002
Durée (min) : 121
Type : Action|Adventure|Fantasy|Romance
Origine : USA
IMDB score : 7.3
Facebook likes : 5000.0
Directeur : Sam Raimi
Principaux acteurs : J.K. Simmons, James Franco, Kirsten Dunst

Titre : The Three Musketeers - 2011
Durée (min) : 110
Type : Action|Adventure|Romance
Origine : Germany
IMDB score : 5.8
Facebook likes : 19000.0
Directeur : Paul W.S. Anderson
Principaux acteurs : Milla Jovovich, Logan Lerman, Orlando Bloom

Titre : A Knight's Tale - 2001
Durée (min) : 144
Type : Action|Adventure|Romance
Origine : USA
IMDB score : 6.9
Facebook likes : 0.0
Directeur : Brian Helgeland
Principaux acteurs : Heath Ledger, Rufus Sewell, Bérénice Bejo

Titre : The Musketeer - 2001
Durée (min) : 104
Type : Action|Adventure|Romance
Origine : Germany
IMDB score : 4.7
Facebook likes : 299.0
Directeur : Peter Hyams
Principaux acteurs : Catherine Deneuve, Justin Chambers, Stephen Rea

Pistes d'évolutions

Actuellement on a en lisse 2 modèles, le TSNE et la distance simple. Selon l'évolution prévue du site, le modèle utilisant la distance simple serait préférable. En effet, l'ajout de film n'impacterait pas le modèle et il n'y aurait aucune maintenance à faire.

Une des évolutions fortes possibles serait aussi d'appliquer des coefficients. En effet à l'heure actuelle avec le MinMaxScaler, le "nombre de like" pèse autant que le budget et que la note. Cette façon de faire pose quelques problèmes. En effet, si un utilisateur regarde un film très mal noté, le moteur proposerait plus probablement des films tout aussi mauvais. De la même façon, si un producteur fait une comédie avec 3 acteurs et un film d'actions avec ces mêmes acteurs. Les propositions ne prendrait pas en compte le genre par exemple car tous les autres critères seraient proches. L'ajout de coefficient permettrait d'avoir des facteurs prépondérants. Celui-ci pourrait être choisi soit par le client (fixe) soit à donner par l'utilisateur. Par exemple un utilisateur pourrait donner un intérêt fort pour les films avec tel acteur plutôt que le genre ou le producteur.

Conclusion

En conclusion, malgré un dataset avec peu de donnée initiales (uniquement 28 features), on arrive à avoir un modèle proposant des films assez similaires. Cependant, il est difficile d'évaluer de manière objective les modèles étant donné que l'on n'a pas de labels. On ne peut donc pas rechercher non plus le nombre de clusters sauf par divers essais et regarder les films proposés.

Cependant, on peut conclure que le Clustering n'est pas fait pour ce type de tâches. Il est plus utilise pour grouper dans n groupes des données par similarité mais non pour fournir les éléments les plus proches. En opposition, le groupe des manifolds est fait pour ce type de tâche mais en fonction du modèle, les résultats peuvent être très différents et parfois incohérents. Une des évaluations possibles serait de l'évaluer sur une centaine de films de manière supervisé (comme cela a été fait avec Spider-Man 3).

A l'instar des autres modèles, le TSNE est le seul à avoir prédit les suites de films ce qui explique son bon fonctionnement. Par contre beaucoup de modèles prédisaient les mêmes films. Peut-être qu'un bagging pourrait rendre le modèle encore plus précis mais une fois de plus on n'a pas de moyen précis d'estimer la qualité du modèle.

Le modèle simple basé sur les distance (comme un KNN) donne lui aussi de très bons résultats et a l'avantage de ne pas perdre d'informations par une réduction de dimensions et d'accepter l'ajout de films sans devoir ré-entrainer le modèle.

Une des possibilités d'ajustement serait de proposer des coefficients sur chaque features soit au client soit à l'utilisateur. Celui-ci permettra par exemple de filtrer d'avantage par genre que par "nombre de like" par exemple (actuellement tous deux pèsent autant).