

**Compte rendu**

*8INF919- Apprentissage automatique pour les données massives*

*Membres du groupe :*

* *TEKEM SANDJONG LOIC ARTHUR TEKL05079909*
* *MBOUTCHOUANG KENMOE BRICE YVAN MBOB11060108*

1. **Le contexte de l’étude**

Le consommateur moderne a trop de choix aujourd’hui car les détaillants, les fournisseurs proposent une grande variété de produits, répondant à tous les besoins et à tous les goûts avec un nombre incalculable de possibilités.

Ainsi associer les consommateurs aux produits les plus adaptés est la clé pour améliorer la satisfaction et la fidélité des utilisateurs. C'est pourquoi de plus en plus de détaillants, de fournisseurs s'intéressent aux systèmes de recommandation, qui analysent l'intérêt de l'utilisateur pour le produit afin de fournir des recommandations personnalisées qui correspondent au goût de celui-ci .

Pour comprendre le concept des systèmes de recommandation, prenons un exemple.

Le tableau ci-dessous montre la matrice utilisateur-article, la valeur d’une case entre un utilisateur et l’article représente la note que l'utilisateur a attribué à l’article sur une échelle de 1 à 5. Les valeurs manquantes c’est-à-dire celles qui n'ont pas encore été évaluées par un utilisateur sont représentées par un point d’interrogation.

Une image contenant table

Description générée automatiquement

L'objectif d'un système de recommandation est de prédire les notes de ces éléments pour que les articles les mieux notés soient ensuite être recommandés aux utilisateurs correspondants.

1. **Le problème posé en termes d’apprentissage pour le big data ainsi que les objectifs visés**

Pour mieux comprendre les problèmes qu’ont les systèmes de recommandation classiques dans le cadre du big data nous proposons de faire une analyse sur les deux grandes approches qui existe afin de bien comprendre.

1. **Le filtrage collaboratif**

Le filtrage collaboratif repose sur l'hypothèse que si les utilisateurs étaient d'accord dans le passé, ils le seront aussi dans le futur ce qui signifie que s'ils aimaient les mêmes choses auparavant, la situation dans le futur ne changera pas.

Cette méthode nécessite la collecte et l'analyse d'informations sur les comportements des clients, leurs activités et leurs préférences afin d'identifier des modèles et de fournir des prédictions précises basées sur la similarité avec d'autres utilisateurs.

Prenons un exemple simple pour illustrer cette approche : si toto aime les articles A, B, C et D, et que tom aime les articles A, B et C, il y a de fortes chances pour qu'il aime aussi l'article D.

Cependant, le filtrage collaboratif présente quelques inconvénients surtout dans le cadre du big data. Ces problèmes sont les suivants :

* **Le démarrage à froid**

La mise en place de nouveaux produits se fait de plus en plus rapidement de nous jours ainsi il faudrait que l’on soit capable de pouvoir recommander ces produits susceptibles de répondre aux besoins des utilisateurs mais le problème avec le filtrage collaboratif est que lorsqu’un nouvel élément apparaît, le système ne peut pas le recommander car il ne dispose d'aucune évaluation pour cet élément. Il faudra un certain temps pour obtenir un nombre suffisant d'évaluations pour que le système puisse déterminer quels groupes d'utilisateurs devraient se voir recommander l'élément.

* **Sparsité et L’évolutivité**

Avec des quantités de produits disponibles de plus en plus énormes, il est difficile de s'assurer que suffisamment de personnes explorent toutes les options disponibles. Ainsi, vu qu’un article n'aura pas été évalué par un grand nombre de personnes, le système ne disposera pas de données sur lesquelles baser pour le recommander.

De plus à mesure que le nombre d'utilisateurs augmente, les algorithmes souffrent de problèmes d'évolutivité. Par exemple si vous on a 10 millions de clients et 100 000 films, on devra créer une matrice éparse comportant un trillion d'éléments.

* **Le temps de calcul**

Avec l’augmentation du nombre d’utilisateurs et d’items, les calculs nécessaires à la recommandation deviennent très coûteux. Souvent des algorithmes de recommandation qui ont une énorme base d’utilisateurs et d’items préfèrent avoir des recommandations moins précises avec un temps de calcul rapide pour pouvoir supporter un nombre d’utilisateurs croissant sans défaillir.

* **Les moutons gris**

Afin de recommander des articles, le système doit regrouper les personnes dont les intérêts sont similaires. De nombreux utilisateurs se retrouveront dans ces groupes et apprécieront les recommandations, mais si certains utilisateurs ne sont pas systématiquement en accord ou en désaccord avec un groupe, ils ne recevront pas de recommandations de qualité.

1. **Filtrage basé sur le contenu**

Le filtrage basé sur le contenu se concentre sur les attributs ou les caractéristiques descriptives des articles pour générer des recommandations de produits. Dans cette approche, des mots-clés sont utilisés pour décrire l'article, et un profil d'utilisateur est établi pour montrer quel type d'articles l'utilisateur aime. L'hypothèse est que si vous avez manifesté de l'intérêt pour un article, vous aimerez également des articles présentant des caractéristiques similaires, qu'il s'agisse du sujet d'un article, d'une marque de produits, de la couleur, de la forme, de la taille, etc. Cette approche est souvent utilisée pour les recommandations d'articles et d'autres documents textuels.

Le filtrage basé sur le contenu présente également certains inconvénients :

* **Description du contenu**

Dans certains cas, il peut être très difficile de fournir une description précise d'un élément. S'il s'agit de musique ou de vidéos que l'on essaie de recommander, la représentation du contenu n'est pas toujours possible.

* **Sur-spécialisation**

Si le comportement antérieur d'un utilisateur ne montre pas qu'il aime quelque chose, le système ne le suggérera pas. Si nous voulons que le système fournisse des recommandations en dehors de ce à quoi l'utilisateur a déjà montré son intérêt, des techniques supplémentaires doivent être ajoutées. Ce qui va augmenter le temps de calcul pour effectuer la recommandation ce qui n’est pas avantageux dans le cadre du big data.

* **Domaine subjectif**

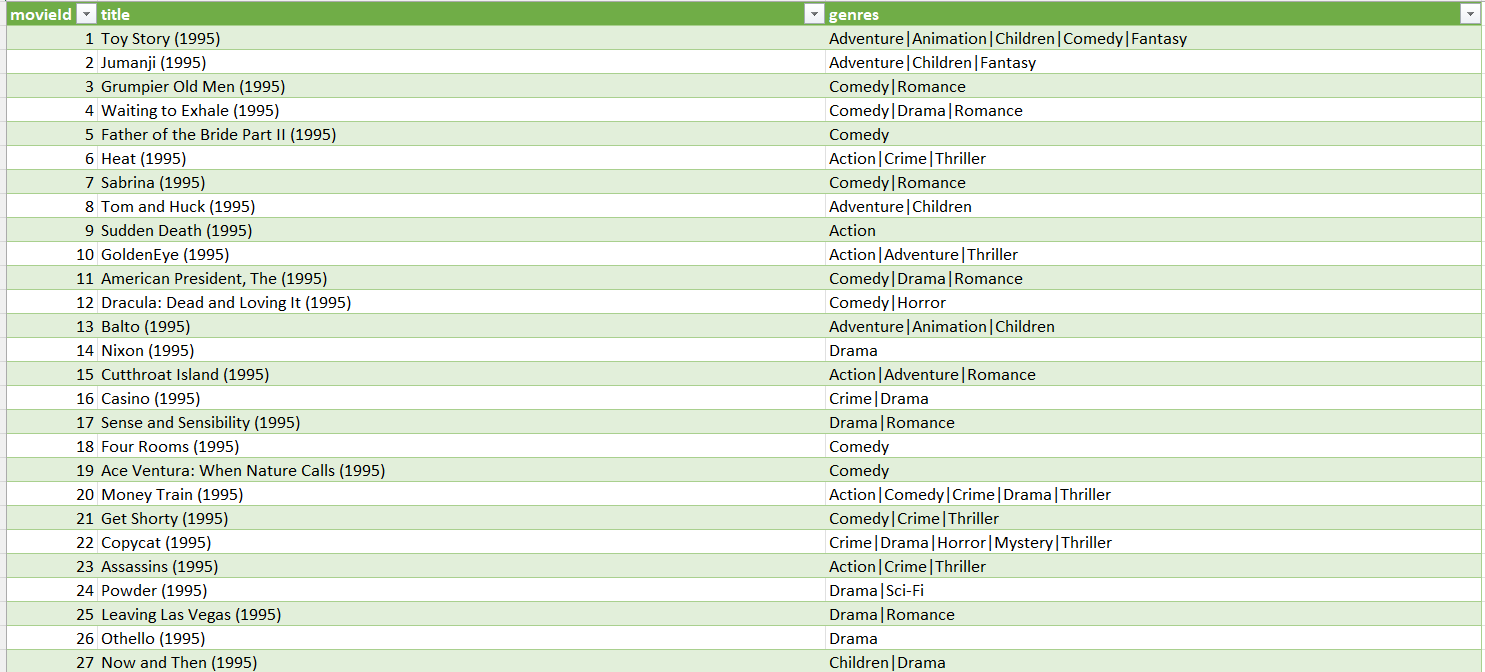
Les techniques de filtrage basées sur le contenu ne traitent pas bien les informations subjectives telles que le point de vue ou l'humour.

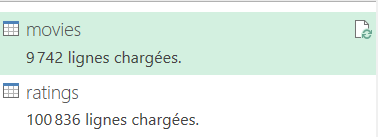
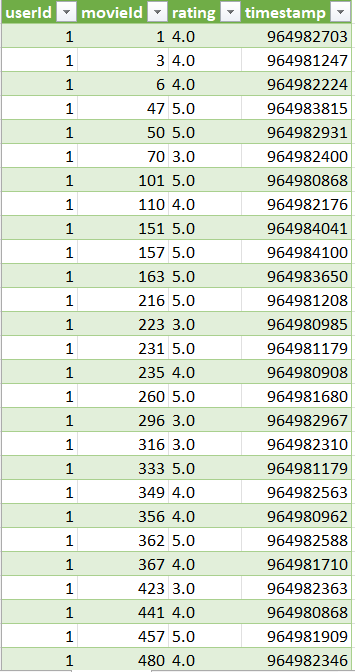
1. **Les objectifs visés**

En raison de l'explosion de la quantité de données numériques sur Internet impliquant des vidéos, des articles, des images et tous les sujets les plus divers, et du large éventail de profils d'utilisateurs qui souhaitent accéder à l'information le plus précisément possible, la nécessité de développer un outil de recommandation efficace systèmes sont apparus naturellement. Ainsi les objectifs visés sont les suivants :

* L’amélioration des performances des système de recommandation.
* Développer une meilleure capacité à manipuler des grandes quantités de données et leur faculté à fournir les meilleures recommandations pour un utilisateur donné.

1. **L’échantillon de données d’apprentissage et de test;**





Le jeu de donnée a été pris sur kaggle il compte plus de 20 millions de votes au total. Le jeu de donné est constitué par deux fichiers dont l’un représente l’ensemble ratings qui caractérisent les notes attribuées à chaque film et l’autre l’ensemble caractérise tous les films qui ont été noté par les utilisateurs.

Dans le cadre de notre étude nous avons choisi de nous limiter à prendre 9742 films et 100k votes

La source est la suivante : <https://www.kaggle.com/datasets/grouplens/movielens-20m-dataset>

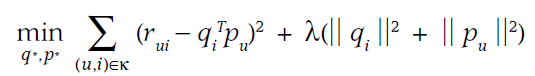
1. **La description détaillée de l’algorithme utilisé**

De nos jours les systèmes de recommandation sont utilisés pour augmenter l’expérience client et constitue donc un processus important pour les entreprises de e-commerce. L’approche du filtrage sur le contenu permet de créer un profil pour chaque utilisateur ou produit afin de les caractériser. Une alternative sur le filtrage de contenu se base sur l’historique de l’utilisateur et est connue sur le nom de filtrage collaboratif. Il analyse les relations entre les utilisateurs ainsi que l’interdépendance entre les produits pour identifier des associations utilisateur-produit.

Sa principale difficulté nommée « **Cold start »** est liée à la complexité de faire des recommandations pour les nouveaux utilisateurs et les nouveaux produits. Les domaines de base du système collaboratif sont la méthode des voisins et les modèles de facteurs latents. Le premier est centré sur le calcul des relations entre les produits ou les utilisateurs en fonction des notes précédentes. Les modèles de facteurs latents quant à eux tentent d’expliquer une notation en caractérisant les utilisateurs et les produits sur plusieurs facteurs. Les réalisations ayant le plus de succès de ces modèles sont basées sur la factorisation des matrices. Elle caractérise les utilisateurs et les produits par un vecteur de facteur déduit du schéma d’évaluation des produits. L’exemple le plus courant est celui des feedbacks avec les utilisateurs et leur note sur les produits. L’une des forces de cette méthode est que le système est capable de déduire les notes de manière implicite. Chaque produit est représenté par un vecteur qi de facteur qu’il possède ou pas. Les utilisateurs quant à eux sont représenté par un vecteur pu mesurant l’intérêt pour chaque produit possédant le plus un facteur. Le produit de ces 2 vecteurs constituerait alors la note des utilisateurs pour chaque produit.



Ce modèle se rapproche fortement de la décomposition en valeur singulière (SVD). Pour trouver les vecteurs (user and item), minimise l’erreur quadratique régularisé sur l’ensemble des notes connues. Le système apprend en ajustant le résultat précèdent. Le but reste cependant de généraliser les notes connues pour prédire les valeurs manquantes. Une constante déterminée par validation croisée permet de contrôler la régularisation. Nous avons donc l’équation suivante :



Avec K l’ensemble des valeurs non nulles et λ une constante permettant de contrôler la régularisation et est obtenu par validation croisée.

ALGORITHMES D’APPRENTISSAGE

Deux approches sont utilisées pour réduire l’équation précédente : le gradient stochastique décent et les moindres carrés alternés. Nous nous intéresserons ici à la 2e approche.

Les vecteurs étant inconnus, l’équation est convexe. Cependant en fixant une des valeurs, le problème peut être résolu. ALS effectue une rotation en fixant à tour de rôle les valeurs de chaque vecteur (qi, pu). Une fois que toutes les valeurs d’un vecteur sont fixées, le système recalcule les valeurs du second et vice versa. Cette opération est effectuée pour réduire l’équation jusqu’à convergence. L’avantage principale de cette approche sur la première est qu’elle calcule les valeurs de chaque vecteur indépendamment des autres et permet donc une parallélisation.

Plusieurs éléments peuvent être introduit dans l’équation afin de mieux ajuster les prédictions. Il s’agit entre autres :

* Des biais : certains utilisateurs ont tendance à donner des notes très élevées de même que certains produits du fait de leur popularité ont tendance à recevoir des notes plus élevées.
* Des sources de données supplémentaires : Pour palier au problème de cold start, le système utiliser les feedbacks implicites pour déterminer les préférences d’un utilisateur. Un nombre de facteur implicite est alors ajouté au vecteur de produit. Une source de données concernant les attributs de l’utilisateur peut être ajouté(zone géographique, âge, sexe, …)
* Le niveau de confiance varié : Dans de nombreux cas, les notes n’ont pas toutes le même poids. Une grande campagne de publicité peut influencer une note à un moment donné. Une pondération pourrait donc être ajoutée pour chaque note.

PARALLELISATION DE L’ALGORITHME ALS

L'algorithme de recommandation ALS peut être parallélisé pour accélérer son exécution et traiter plus rapidement les données. Cela peut être accompli en divisant les données en plusieurs parties et en exécutant simultanément plusieurs instances de l'algorithme sur différents cœurs ou processeurs.

Il existe plusieurs façons de paralléliser l'algorithme ALS, en fonction de la structure des données et de la façon dont elles sont stockées. Par exemple, l'algorithme peut être exécuté sur des lignes ou des colonnes de la matrice de données en parallèle, ou bien des parties de la matrice peuvent être traitées simultanément.

La parallélisation de la multiplication de 2 matrices 𝑀(n,d),N(d,m) se déroule de la manière suivante avec MapReduce (On sait que les éléments Pik  de P=MN sont Pik  = ijnjk ) :

* Pour chaque fonction Map() de chaque tâche Map sur chaque nœud:
  + Pour chaque élément ij  de 𝑀, on produit une paire de clé-valeur (𝑗, (𝑀, 𝑖, ij  ))
  + Pour chaque élément jk  de N, on produit une paire de clé-valeur (𝑗, (N, k, jk  ))
* La fonction Reduce de chaque tâche Map sur chaque nœud :
  + On fait une sorte de jointure : Pour chaque clé, j, elle examine ses valeurs associées. Pour chaque valeur qui provienne de 𝑀, disons (𝑀, 𝑖, ij ) et chaque valeur qui provienne de 𝑁, disons , (N, k, jk ) produit une paire clé-valeur ((𝑖, 𝑘), ijnjk )
* Par la suite, on effectue un regroupement et une agrégation par une autre opération MapReduce
  + La fonction Map() sera juste une fonction identité qui pour chaque paire ( 𝑖, 𝑘 , v) produira la même paire en sortie
  + La fonction Reduce() pour chaque clé (𝑖. 𝑘) produira la somme des valeurs de la liste associée à la même clé
  + Le résultat est une paire ( (𝑖, 𝑘) . 𝑣) , où 𝑣 est la valeur de l’élément de la ligne 𝑖 et de la colonne 𝑘 de la matrice 𝑃 = 𝑀N

1. **L’analyse comparative, s’il y a lieu, avec au moins une technique qui n’est pas nécessairement celle vue en cours.**

Dans le cadre de notre devoir nous avons essayé de faire une étude comparative entre le filtrage collaboratif classique et les matrice de factorisation afin de montrer que les matrices de factorisation permettent de compenser les difficultés qui sont rencontrées avec le collaborative filtering et réponde mieux aux objectifs visés

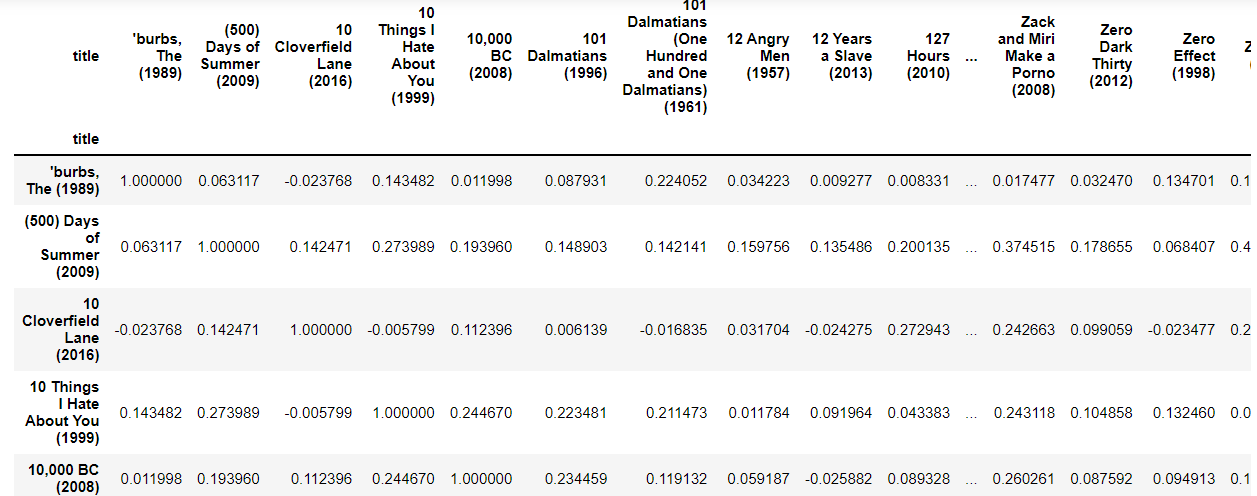
* 1. **Méthodologie de travail**

Dans un premier temps nous avons essayé de mettre en place un système de recommandation qui repose sur le calcul de similarité entre films à partir des notes données par les utilisateurs pour faire des recommandations. Nous l’avons fait pour évaluer les hypothèses théoriques liées aux problèmes qui ont été présentés concernant le collaborative filtering puis par la suite nous avons utilisé l’algorithme als qui utilise les matrices de factorisation évaluer effectivement son efficacité.

* 1. **Observations et analyse des résultats**
     + **Le colaborative filtering avec calcul de similarité**
* **Une image contenant texte

  Description générée automatiquementLe temps d’exécution** : Dans notre cas pour faire la recommandation nous avons utilisé la similarité de Pearson. Nous nous sommes rendu compte qu’elle aide effectivement à faire des recommandations mais présente un énorme problème lié au temps de calcul surtout si on travail le cadre du big data.

Par exemple si on décide de calculer la similarité entre 2 films qui ont chacun eu 610 votes comme dans notre cas on remarque que seul le calcul de la moyenne des valeurs de chaque film sera très long ce qui veut dire que le calcul du coefficient de Pearson le sera aussi lorsque toutes opérations seront faites.

Ainsi l’ensemble des opérations qui permettront d’obtenir la matrice de corrélation entre les films (comme sur l’image suivante) sera extrêmement couteux en temps lorsqu’on aura plein de film et votes.

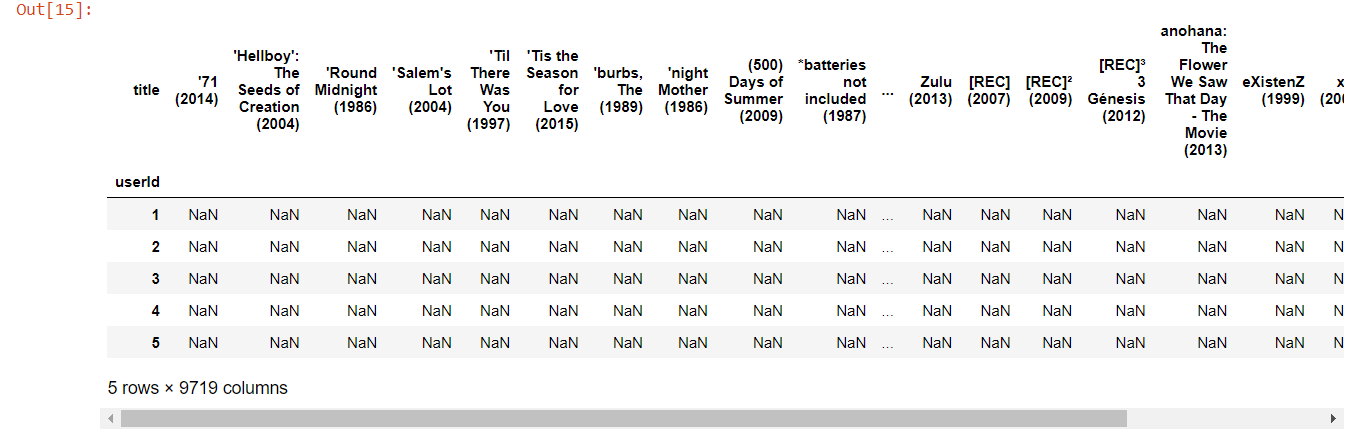
On peut donc conclure que si on travail dans le cadre du big data la mise en place d’un système de recommandation sera très couteuse en temps principalement parce que la quantité d’informations à traiter sera gigantesque ainsi les temps de réponse seront affectés.

* Une image contenant texte, table

  Description générée automatiquement**La sparsité :**

Comme on le constate sur l’image précédente on a plein de films qui ont été voté une seule fois ce qui veut dire que sur 610 utilisateurs au total on a 609 utilisateurs qui n’ont pas donné de notes à ces films soit 609 films avec la valeur Nan.

Ainsi on se retrouve avec une matrice sparse ce qui reflète la réalité dans le domaine du big data comme on peut le voir ici :

Le problème est que pour pouvoir exploiter les Nan on est obligé d’attribuer la valeur 0 en supposant que si l’utilisateur ne vote pas cela veut dire qu’il n’a pas apprécié le film mais cela ne reflète pas la réalité car un utilisateur peut avoir apprécié un film sans pour autant voter.

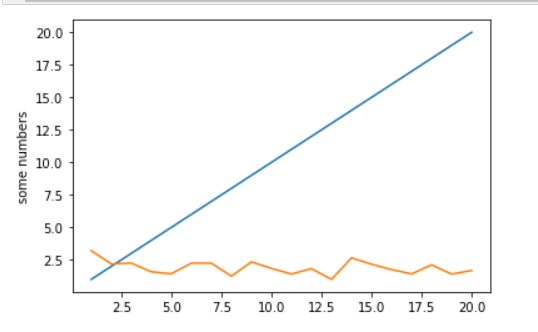
On peut également décider de supprimer les films qui n’ont pas un minimum de 10 votes comme sur l’image suivante mais le problème qui se pose est qu’on perd une quantité considérable d’informations.

Une image contenant texte

Description générée automatiquement

* **L’évolutivité** : à partir des problèmes cités précédemment on peut facile conclure que l’évolutivité sera difficile notamment avec les temps de calculs qui vont augmenter et la sparsité aussi
  + - **Recommandation avec als**
* **Le temps d’exécution et** évolutivité : pour évaluer le temps nous avons décidé de segmenter un jeu de donné de 20M de lignes en plusieurs jeu de donnée allant de 5 à 100% du jeu de donnée total.

Puis on a fait des entrainements et des prédictions successives et nous avons obtenu la courbe de prédiction suivante :



Comme on peut le constater le temps de prédiction c’est-à-dire de recommandation est très rapide et en moyenne tourne autour de 2.5 secondes peu importe la taille du jeu de donnée.

On peut donc conclure que grâce aux matrices de factorisation les problèmes liés à l’évolutivité et au temps d’exécution sont largement compensés.

* **La sparsité** : concernant la sparsité on n’a pas eu besoin de faire des transformations sur les données la machine grâce à l’algorithme à pu estimer seule quel serait les notes que ces films pourraient avoir en fonction des utilisateurs.

**Conclusion**

Travailler sur ce projet nous a permis de mieux comprendre la notion de matrice de factorisation et de nous rendre compte de son importance dans le cadre du big data. En effet elle permet de résoudre beaucoup de problème pour offrir des recommandation rapide et pertinante

Source

<https://spark.apache.org/docs/2.2.0/ml-collaborative-filtering.html>

<https://github.com/codeheroku/Introduction-to-Machine-Learning/blob/master/Collaborative%20Filtering/Movie%20Lens%20Collaborative%20Filtering.ipynb>

<https://github.com/dataquestio/project-walkthroughs/blob/master/movie_recs/movie_recommendations.ipynb>

<https://github.com/snehalnair/als-recommender-pyspark/blob/master/Recommendation_Engine_MovieLens.ipynb>

<https://towardsdatascience.com/build-recommendation-system-with-pyspark-using-alternating-least-squares-als-matrix-factorisation-ebe1ad2e7679>

<https://medium.com/@patelneha1495/recommendation-system-in-python-using-als-algorithm-and-apache-spark-27aca08eaab3>

<https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-2-alternating-least-square-als-matrix-4a76c58714a1>

<https://www.youtube.com/watch?v=ZspR5PZemcs>

<https://www.youtube.com/watch?v=n3RKsY2H-NE>

https://www.youtube.com/watch?v=v\_mONWiFv0k

<https://www.youtube.com/watch?v=Fmtorg_dmM0>