Hes·so
Haute Ecole Spécialisée
de Suisse occidentale

BDA

Rapport : Recommending Music and the Audioscrobbler Data Set

Rapport
Version 1.0, 19.06.2020

Samuel Torche
Ayrton Dumas
Marco Mattei
Groupe B

MSE | MASTER OF SCIENCE IN ENGINEERING

Table des matières

1. Introduction

Dans le cadre du cours de Big Data Analytics, nous allons réalisé un mini-projet afin de mettre en pratique les principes vus durant le cours. Notre choix s'est porté sur le dataset Audioscrobbler et sur un mini-projet centré sur la recommandation de musiques.

2. Description du dataset

Le dataset utilisé est appelé AudioScrobbler. Ce dataset contient les artistes musicaux écoutés par plus de 140'000 utilisateurs. Il s'agit d'une archive compressée contenant plusieurs fichiers texte. L'archive fait 135 MB compressée et 500 MB une fois décompressée.

User artist data

Le fichier user_artist_data.txt contient les utilisateurs et les artistes qu'ils écoutent.Chaque ligne correspond contient l'id de l'utilisateur, l'id de l'artiste et le nombre de fois qu'il a écouté une chanson de cet artiste, donc pour chaque artiste écouté par un utilisateur, il y aura une nouvelle ligne. Il y a 140'000 utilisateurs qui ont écouté en tout 24'296'858 artistes. Ce fichier contient donc 3 colonnes: userid, artistid, playcount

Artist data

Le fichier artist_data.txt contient les mappings artistes id -> nom de l'artiste. Ce fichier contient donc 2 colonnes: artistid, artist_name. Il y a 1'848'707 artistes dans ce fichier.

Artist alias

Le fichier artist_alias.txt contient les noms d'artiste qui sont souvent mal orthographiés ou qui sont sujets à variante (Depeche Mode mal orthographié en Depeche Mood par exemple). Ce fichier contient donc 2 colonnes: badid, goodid. Il y a 193'027 alias.

3. Questions d'analyse

1. Peut-on obtenir des résultats intéressants en utilisant des techniques de Market Basket Analysis telles que les règles d'association ?
2. Peut-on déduire les genres des musiques à l'aide d'un clustering des utilisateurs et leurs écoutes ?
3. Est-il possible de recommander non pas des artistes mais d'autres utilisateurs qui partagent les même goût avec ALS ?

Etant donné que la question 3 n'était pas possible à répondre (cf. 5. Algorithmes appliqués), nous l'avons adaptée avec la suivante:

4. Est-il possible de recommander des utilisateurs qui ont des goûts similaires à un certain utilisateur en utilisant du clustering ?

4. Description des features / pre-processing afin d’extraires des features additionnelles

Travail déjà effectué dans le livre

Dans le chapitre donné du livre [1], l’auteur explique comment utiliser ALS avec Spark afin de recommander des musiques à des utilisateurs et ceci selon leurs écoutes.
Dans un premier temps une explication de l’algorithme est réalisé afin d’expliquer son fonctionnement. ALS est une technique de “Collaborative filtering”. Par exemple, décider que deux utilisateurs puissent partager des goûts similaires parce qu'ils ont le même âge n'est pas un exemple de “Collaborative filtering”. Décider que deux utilisateurs pourraient aimer la même musique parce qu'ils jouent beaucoup d'autres chansons identiques en est un exemple. L'un des principaux avantages de l'approche de “Collaborative filtering” est qu'elle ne repose pas sur un contenu analysable par une machine et qu'elle est donc capable de recommander avec précision des éléments complexes tels que des films sans exiger une "compréhension" de l'élément lui-même.[6]
Ensuite l’auteur explique le contenu du dataset et les différentes preprocessing à effectuer pour travailler au mieux avec celui-ci. Finalement, il réalise les différentes étapes nécessaires afin d’obtenir un système de recommandation de musique basé sur ALS.

Market Basket Analysis preprocessing

Afin d’appliquer des techniques de Market Basket Analysis, les données doivent être transformées. En effet, actuellement les données se trouvent sous cette forme:

user	artist	count
1000002	1	55
1000002	1000006	33
1000002	1000007	8
1000002	1000009	144
1000002	1000010	314

On peut voir qu’un utilisateur est présent sur plusieurs lignes, une ligne pour chaque artiste qu’il a écouté. Afin de pouvoir exécuter les algorithmes de Market Basket Analysis (voir chapitre 5. Algorithmes appliqués pour plus d’informations), les données doivent être transformé en format transactionnel, c’est à dire ainsi:

user	items
1000002	[1, 18, 28, 30, 5...]
1000019	[1000010, 1000028...]

On aperçoit à présent qu'un seul utilisateur est présent par ligne, et que tous les artistes qu'il a écouté ont été regroupé sur une seule ligne, dans une colonne nommé "items" de type *Array*. Pour faire l'analogie avec un exemple plus parlant, on peut dire, qu'avant, notre dataset contenait une ligne pour chaque item que notre client avait acheté, donc une ligne pour clientA + pain, une deuxième ligne pour clientA + beurre. Maintenant ces lignes ont été regroupé par client donc clientA + pain + beurre.
Pour obtenir ce résultat, il a fallu:

- Grouper le dataframe par user
- Pivoter le champs artiste en tant que colonne
- Grouper toutes ces colonnes d'artistes un une seule colonne "items" de type *Array*

5. Algorithmes appliqués

Market Basket Analysis

L'algorithme utilisé pour miner les frequent items sets est FPGrowth. Il existe deux implémentations de cet algorithme dans la librairie Scala: org.apache.spark.ml.fpm.FPGrowth et org.apache.spark.mllib.fpm.FPGrowth. La première alternative a été utilisé car il était plus aisé de performer les différentes étapes de preprocessing décrites au chapitre 4. L'avantage de l'algorithme de FPGrowth par rapport à l'algorithme de apriori vu en cours de Data Management au semestre passé est qu'il ne génère pas les candidats explicitement, ce qui fait qu'il est largement plus adapté aux grands sets de données, ce qui est notre cas dans ce projet. "*It is found out that the FP-Growth algorithm outperforms the Apriori and ECLAT algorithms for all databases in terms of runtime and usage of memory.*" Sinthuja et al. [3].
FPGrowth nous permet donc de connaître les items sets fréquents, ensuite, à partir des ces items sets fréquents, on peut en déduire les règles d'association.

KMEANS, déterminer les genres

Notre idée pour déterminer les genres de musique est de clusteriser les utilisateurs; un utilisateur écoute principalement qu'un seul type de musique. Nous avons donc sélectionné le top 3 des écoute de chaque utilisateur et effectué une clusterisation.

```
(1005235, 1004983, 1239653),  // ids des 3 artistes préférés d'un utilisateur
(1854, 1000294, 1086437),
(1004701, 1007969, 1001943),
(979, 5837, 1001531),
(1000052, 1007027, 979),
(1016344, 1012594, 1003276),
```

L'utilisateur à la première ligne écoute : the constantine, the dismemberment plan et Q and Not U, les trois groupes font du rock alternatifs. Cela confirme l'hypothèse de trouver les genres de musique avec le top des artistes de chaque utilisateur.

Nous appliquons ensuite un KMEANS avec des valeurs de k de 2 à 10 pour déterminer quel k donne les meilleurs performances (voir la méthode du coude plus bas).

Recommandation d’utilisateurs avec ALS

En premier lieu, nous voulions donc utiliser ALS pour recommander des utilisateurs à un certain utilisateur, en fonction des artistes qu’ils écoutent. C’est-à-dire qu’on veut suggérer à un utilisateur A des utilisateurs qui écoutent des artistes similaires à l’utilisateur A.
Après un certain nombre de recherches, nous nous sommes rendus compte qu’ALS n’était pas adapté à ce genre de problème. En effet, ALS fonctionne sur le principe “user-item” et le problème indiqué ici est plus un problème “user-user” et malgré nos recherches nous n’avons pas trouvé un moyen de répondre à la question avec ALS.
Nous avons donc décidé de tenter de répondre à la question avec du clustering, comme expliqué au chapitre suivant.

Recommandation d’utilisateurs avec du clustering

Notre idée pour recommander des utilisateurs avec du clustering a été de clusteriser tout le set de données. C’est-à-dire que c’est un cluster 3D avec les axes suivants :

- X : Utilisateur
- Y : Artiste
- Z : Nb d’écoutes

Ceci permet de trouver des clusters d’utilisateurs qui ont écouté beaucoup de fois les mêmes artistes.
Cependant, cette méthode contient un défaut. En effet, nous nous sommes rendus compte qu’il aurait fallu normaliser les axes car actuellement les utilisateurs et les artistes sont des valeurs catégoriques ce qui signifie que leur valeur sur l’axe n’a pas réellement de “sens” pour le cluster. Effectivement, un utilisateur A qui écoute les mêmes artistes qu’un utilisateur B peut se trouver à une très grande distance de ce dernier à cause de son placement dans le graphe. Il s’agit donc d’un problème relativement complexe.

6. Optimisations réalisées

Market Basket Analysis

Les paramètres à optimiser sont les suivants:

- Le support minimal: le support correspond au pourcentage de transactions qui contiennent la totalité des items d'un item set

$$Support = \frac{frq(X,Y)}{N}$$

- La confidence minimale: la confidence minimale correspond au pourcentage de transactions qui contiennent la totalité des items d'un item set sur le pourcentage qui contiennent la partie gauche de l'item set (l'antécédent)

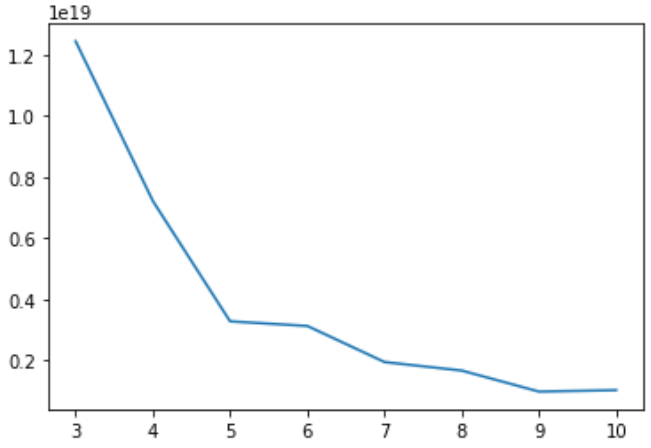
$$Confidence = \frac{frq(X,Y)}{frq(X)}$$

Nous avons choisi les valeurs suivantes pour ces paramètres:

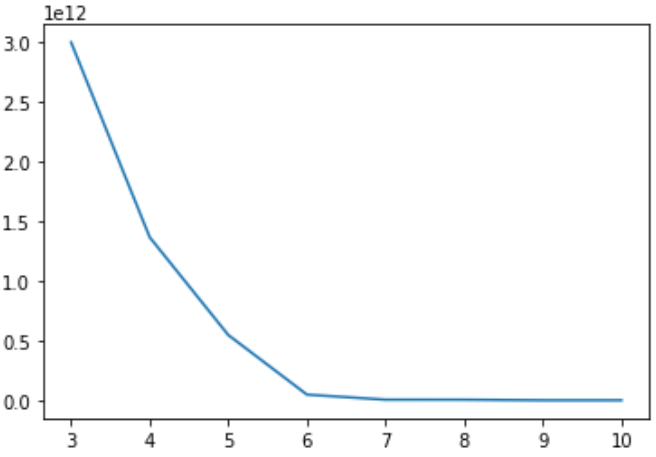
- Support minimal: 0.1 => le dataset étant très grand et contenant énormément d'artistes différents (environ 2 millions), pour seulement 140'000 utilisateurs donc 140'000 transactions, il est nécessaire d'avoir une valeur très basse pour ce paramètre. Si on met une valeur trop haute, on obtient tout simplement 0 règles d'association. Le fait de mettre une valeur trop grande rendrait les règles d'association biaisé envers les artistes très populaire au détriment des artistes un peu moins connu et écouté.
  Rien qu'en passant le support de 0.1 à 0.2, sans tenir compte de la confidence (on met le minimum à 0), on passe de 1470 règles à 0 règles
- Confidence minimale: 0.4 => avec le support minimum mis à 0.1, on passe de 1470 règles à 1277 règles. Pour la confidence, le dénominateur étant les transactions contenant "X", on peut se permettre de mettre une valeur plus élevée que pour le support.
  En passant la confidence à 0.5, on passe à 821 règles comparé au 1277 obtenu avec 0.4, ce qui correspond à une perte trop importante.

Méthode du coude

Dans un système de clustering, le nombre de groupe à créer est inconnu.
Dans notre cas, devant faire 2 clustering (un pour la question des genres et un pour la question des utilisateurs), nous ne savons pas combien de genres de musiques ou de regroupement d'utilisateurs existent.
Nous avons alors utilisé la méthode du coude afin de déduire ces valeurs [7]. Ceci permet d'optimiser le temps de traitement car à partir d'un certain "k" les améliorations de précision sont faibles.
Pour le nombre de groupes d'utilisateurs, la figure suivante montre que le "coude" est visible lorsque **K** vaut 5, donc 5 regroupements d'utilisateurs.



Pour ce qui est du nombre de genres, il est possible de voir ce "coude" lorsque **K** vaut 6, 6 genre de musiques :



7. Tests et évaluations

Market Basket Analysis

Malheureusement, il n'a pas été possible de travailler avec l'intégralité des données. En effet, l'utilisation des 25 millions d'écoutes utilisateurs faisait crasher l'application avec une erreur org.apache.thrift.transport.TTransportException qui correspond à un problème dû à un problème de mémoire. Une solution proposée sur internet indiquait d'augmenter la mémoire allouée à l'interpréteur spark dans la configuration de zeppelin [1], conseil également dispensé par les collègues Yann-Ivain Beffa et Jonathan Donzallaz sur le Teams du cours. Nous avons augmenté cette mémoire de 1g à 12g, mais le problème subsistait. Nous avons donc pris la décision arbitraire de ne garder que 5 millions des 24 millions d'écoutes utilisateurs. Cela fait que nous avons uniquement 21'117 transactions (donc utilisateurs), sur les 140'000 de base.
Comme expliqué dans le chapitre 6 Optimisations réalisées, le système final possède 1277 règles d'association.
L'évaluation des règles d'association est une tâche ardue. En effet, Kaliappan Vanitha et al. indiquent que "*However, mining association rules often results in a very large number of found rules, leaving the analyst with the task to go through all the rules and discover interesting ones.*" [2].
Une approche manuelle est donc à faire, et forcément l'évaluation est sujette au biais de l'évaluateur (est-ce qu'il trouve que cette règles est pertinente ?). Nous n'avons bien sûr pas le temps d'analyser ces 1277 règles, et nos connaissances musicales sont plutôt limitées, surtout quand on sait que le set contient 2 millions d'artistes, il est donc inimaginable que nous connaissions ces 2 millions d'artistes.

```
[352,1177,979,0.8645339652448657]
Beck, Coldplay,  -> Radiohead
[352,1307,979,0.843316144387148]
Beck, The White Stripes,  -> Radiohead
[1001646,1177,979,0.8378378378378378]
The Smashing Pumpkins, Coldplay,  -> Radiohead
[352,3327,979,0.8362779740871613]
Beck, Weezer,  -> Radiohead
[1001646,1307,979,0.8283378746594006]
The Smashing Pumpkins, The White Stripes,  -> Radiohead
[1001646,1275996,979,0.8282633808240277]
The Smashing Pumpkins, R.E.M.,  -> Radiohead
[234,1000113,979,0.823134328358209]
Pixies, The Beatles,  -> Radiohead
[1001779,1000113,979,0.8190219484020023]
Modest Mouse, The Beatles,  -> Radiohead
[352,1000113,979,0.8182748039549949]
Beck, The Beatles,  -> Radiohead
```

Ci-dessus les 9 règles d'association ayant le plus de confidence. La première ligne contient les ids des artistes et la confidence, la 2ème ligne le nom des artistes. Sur ces 9 règles, le groupe "Radiohead" est toujours le conséquent de la règle. "Radiohead" est un groupe anglais de rock des

[1] https://stackoverflow.com/questions/36835122/org-apache-thrift-transport-ttransportexception-error-while-reading-large-json-f

années 80. En s'intéressant à la 8ème règle (Modest Mouse, The Beatles -> Radiohead), tous les groupes de la règle font de la musique rock et datent des années 60-90, ce qui fait du sens. C'est le cas de la plupart de ces 9 règles qui concernent des artistes rock en général.
Une manière d'évaluer serait de regarder toutes les règles contenant un artiste apprécié par un des étudiants, ainsi il serait plus apte à juger des recommandations en fonction de ses goûts.
L'artiste choisi est R.E.M:

```
[1275996,4267,979,0.7412831241283124]
R.E.M., Green Day,  -> Radiohead
[1275996,1274,976,0.737450462351387]
R.E.M., Red Hot Chili Peppers,  -> Nirvana
[1275996,1177,1205,0.7180659915060438]
R.E.M., Coldplay,  -> U2
[1275996,976,1274,0.7150176112712135]
R.E.M., Nirvana,  -> Red Hot Chili Peppers
```

Les règles font du sens, l'étudiant en question apprécie les artistes recommandés.

Déterminer les genres - Clustering

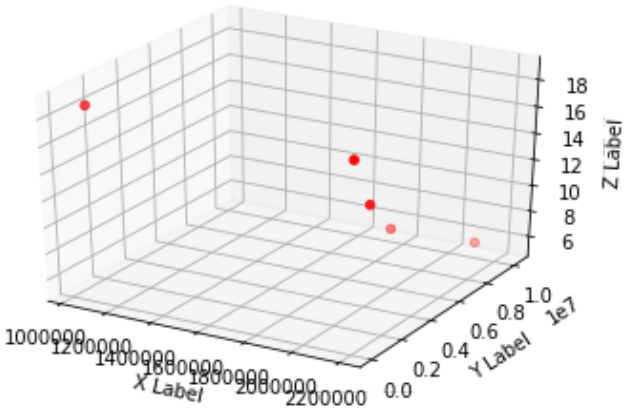
KMeans offre une méthode de calculs de coût permettant d'évaluer la position des centroids, nous l'avons appliqué à plusieurs k pour évaluer lequel était le plus performant.

```
K=1 Within Set Sum of Squared Errors = 1.2445173257535345E19
K=2 Within Set Sum of Squared Errors = 7.2132628707644774E18
K=3 Within Set Sum of Squared Errors = 3.2724949687905096E18
K=4 Within Set Sum of Squared Errors = 3.1214615696898002E18
```

Pour évaluer le résultat du clustering, il faut assigner les centroids à chaque utilisateur, ainsi, on peut comparer si le genre du top 3 des artistes des utilisateurs dans le même cluster correspond entre tous.
Cette évaluation n'a pas malheureusement pas été réalisée.

Recommandation d'utilisateurs avec du clustering

Comme il a été vu dans le chapitre "Méthode du coude", le nombre de groupes d'utilisateurs à utiliser est de 5.
Nous avons alors réalisé un clustering avec 5 groupements d'utilisateurs afin d'obtenir les centroïdes de ceux-ci, ce qui donne le résultat suivant :



Etant donné que le plot de ces graphes est réalisé à l'aide de Python, il nous a été impossible de faire apparaître les points des utilisateurs dans les différents clusters. Effectivement, nous n'arrivions pas à faire passer les données entre les cellules Spark et Python, même en suivant les tutoriels officiels (z.put("centers", model5.clusterCenters) par exemple).
De plus, pour évaluer la pertinence de ces centres, il serait nécessaire de parcourir toutes les données et assigner un centroïde à chacun des utilisateurs. Un manque de temps n'a pas rendu ceci possible. En effet, étant donné que cette question a dû être modifiée, le temps à disposition était réduit et la majorité des traitements et algorithmes lancés avec Zeppelin prenaient un temps non négligeable (des heures pour certaines cellules).

8. Résultats obtenus

Market Basket Analysis

Nous pouvons répondre à notre question: Peut-on obtenir des résultats intéressants en utilisant des techniques de Market Basket Analysis telles que les règles d'association ?
La réponse est oui. Le système mis en place actuellement possède certaines limites, mais il permet de valider l'idée. Les chapitres 7 Tests et évaluations et 9 Améliorations futures décrivent les résultats obtenus, leur pertinence, ainsi que leur limitations.

Détection des genres

Nous pouvons déterminer que le nombre de genre musicaux est au nombre de 6, cependant, comme expliqué dans le chapitre d'évaluation, nous n'avons pas pu vérifier que ces 6 genres sont corrects et correspondent effectivement à l'ensemble des utilisateurs du groupe.

Recommandation d’utilisateurs

Nous pouvons déterminer que le nombre de “groupement” d’utilisateurs est de 5, et qu’on pourrait recommander à un certain utilisateur d’un cluster les autres utilisateurs qui appartiennent au même cluster. Nous n’avons cependant pas pu vérifier la pertinence de cette hypothèse comme expliqué dans le chapitre d’évaluation.

9. Améliorations futures

De manière générale, le dataset aurait besoin d’un énorme nettoyage. En effet, dans le fichier des artistes, bien des artistes sont mal orthographiés, certaines fois il y a l’artiste et le nom d’une de ses chansons ensemble. Le fichier des alias est insuffisant pour pouvoir nettoyer correctement ce fichier. Certains artistes sont présent plus de 50 fois dans le fichier des artistes. Certaines fois cela fait du sens, par exemple lors du featuring de plusieurs artistes.

Market Basket Analysis

Le système possède 1277 règles d’association. Au départ, nous pensions que c’était beaucoup, mais en fait cela est peu. En effet, 1277 règles ne permettent pas de prendre en comptes les artistes les moins bien connus, il faudrait réduire encore le support minimum jusqu’à obtenir plus de règles, 10’000 règles serait une bonne base. En cherchant des artistes moins connus, il est impossible de trouver une règle parmis ces 1277 règles. L’utilisation d’une machine plus puissante pourrait permettre de construire les règles en utilisant toutes les données.

Détection des genres

La première amélioration est d’avoir un système d’évaluation qui fonctionne et automatique. Actuellement, nous utilisons le top 3 de chaque utilisateur, mais quid de 5 ou plus ? L’édition de cet hyperparamètre pourrait changer les résultats. Nous partons aussi de l’hypothèse qu’un utilisateur écoute de préférence qu’un seul genre de musique, mais comment faire si ce n’est pas le cas ?

Recommandation d’utilisateurs

Il serait nécessaire de normaliser les différents axes afin de donner plus de sens à ceux-ci, comme il a été expliqué dans le chapitre “Algorithmes appliqués”.
De plus, comme le clustering réalisé dans ce projet ne contient “que” 5 groupements, il serait intéressant d’explorer d’autres possibilités afin d’être plus précis et réduire le nombre d’utilisateurs dans chaque groupe afin d’avoir des recommandations plus pertinentes pour l’utilisateur.

10. Conclusion

Nous avons pu répondre à nos questions d'analyse avec plus ou moins de succès. La partie évaluation des 3 questions n'était pas facile à répondre. En effet, dans un problème "classique" de Machine Learning, on peut aisément tester la performance d'un modèle grâce à diverses métriques telles que l'accuracy, le recall, la precision, le f1 score, etc… Mais avec nos questions, nous recherchions plutôt à faire du proof of concept, à démontrer qu'il était possible d'utiliser l'une ou l'autre technique afin de prouver la faisabilité de notre idée.

Ce projet était intéressant à réaliser. Pour les 3 membres de l'équipe, c'était le premier projet réalisé à l'aide de Scala et d'une approche de programmation fonctionnelle. Aucun des 3 membres de l'équipe ne suivait en parallèle le cours de programmation fonctionnelle, ce qui aurait probablement permis d'implémenter plus vite.

Zeppelin ne nous a pas facilité la tâche, bien que le principe de notebook reste une très bonne pratique en ce qui concerne l'analyse de données, ses performances en terme de machine learning sont extrêmement faibles, plusieurs dizaines de minutes pour un simple clustering. De plus, il nous a été impossible de "plotter" les données car le transfert des données entre scala et python n'est pas implémenté dans la version zeppelin fournie.

11. Références

[1] Sandy Ryza, Uri Laserson, Sean Owen, Josh Wills, Advanced Analytics with Spark, O'Reilly, May 2017
[2] Kaliappan Vanitha, R. Vijaya Santhi, Evaluating the performance of association rule mining algorithms, 2011
[3] Sinthuja, M. & Puviarasan, N. & Aruna, P.. (2017). Evaluating the Performance of Association Rule Mining Algorithms. World Applied Sciences Journal. 35. 43-53., 10.5829/idosi.wasj.2017.43.53.
[4] Spark Frequent Pattern Mining, https://spark.apache.org/docs/latest/ml-frequent-pattern-mining.html, consulté le 12.06.2020
[5] Spark ScalaDoc pour FPGrowth, https://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.ml.fpm.FPGrowth, consulté le 12.06.2020
[6] https://en.wikipedia.org/wiki/Recommender_system#Collaborative_filtering
[7] Elbow method (clustering), Wikipedia, https://en.wikipedia.org/wiki/Elbow_method_(clustering), consulté le 12.06.2020