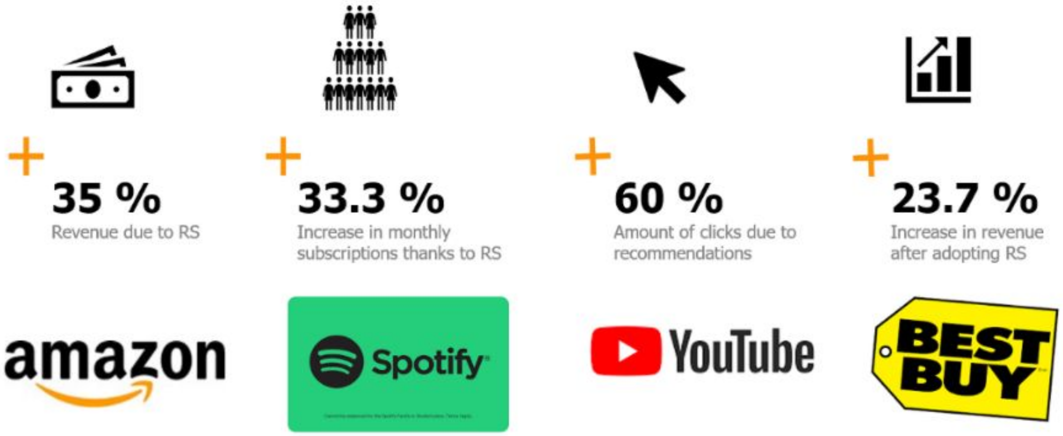
VISIUM

User clustering for movie recommendation

Quick POC for Technical Test

Constraints & Requirements

Executive summary



Recommender system business impact

- Recommender systems aim to predict users' interests and recommend product items that quite likely are interesting for them.

- E-commerce and divertissement companies are **leveraging the power of data and boosting sales/engagement by implementing** recommender systems on their websites.
  - ➢ E-commerce companies that use the recommender system of visium get >**30% increase in the average price of the user basket.**
  - ➢ Visium recommender system provides recommendations that enable to make **optimal earning from customer behaviour.**

Dataset & constraints

- **Movielens 100k Dataset**
  - 100 000 Ratings (1 to 5)
  - 1682 Movies
  - 987 Users
  - User informations : Age, Occupation
  - 18 Movies Categories

title	'Til There Was You (1997)	1-900 (1994)	101 Dalmatians (1996)	12 Angry Men (1957)	187 (1997)
userId					
1	NaN	NaN	2.0	5.0	NaN
2	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	2.0
4	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	2.0	NaN	NaN
6	NaN	NaN	NaN	4.0	NaN

Movielens 100k Dataset

- **Data Constraint**
  - Sparse Dataset

    (Ratings, movies)

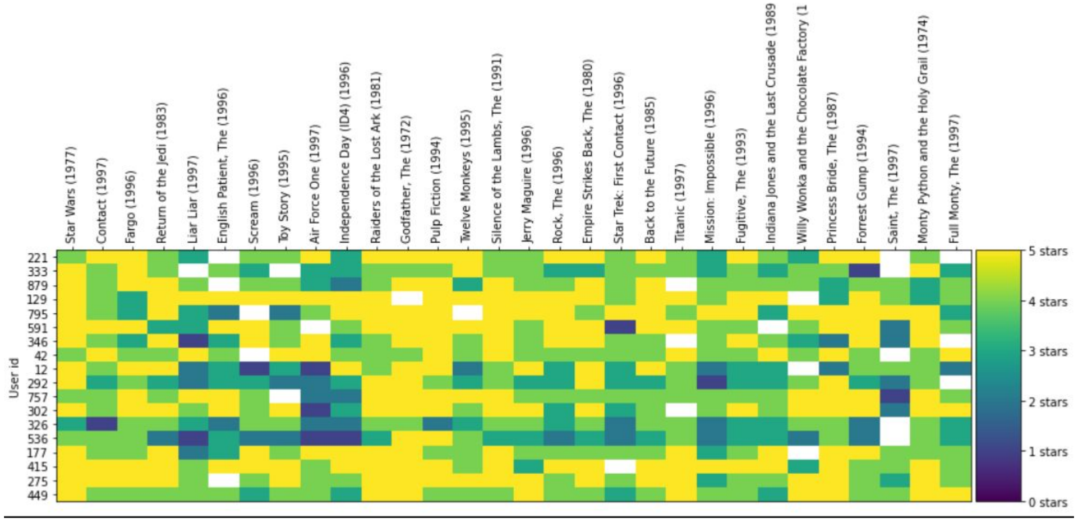
Building a benchmark to compare

	Star Wars (1977)	Contact (1997)	Fargo (1996)	Return of the Jedi (1983)	Liar Liar (1997)	English Patient, The (1996)	Scream (1996)
0	4	1	0	1	1	2	2
1	0	1	0	2	1	0	2
2	1	0	3	0	4	4	2
3	2	1	0	4	1	0	1
4	1	3	2	0	1	1	4
5	0	4	1	3	4	4	2

Random weight distribution

- As a benchmark, we can build **random weights vectors** of ratings
  - ➢ We would compare the efficiency of the recommendation system on **rating weights vs random weights.**
  - ➢ It allows to measure the **efficiency of the recommendation system**.

Rating Features for Clustering

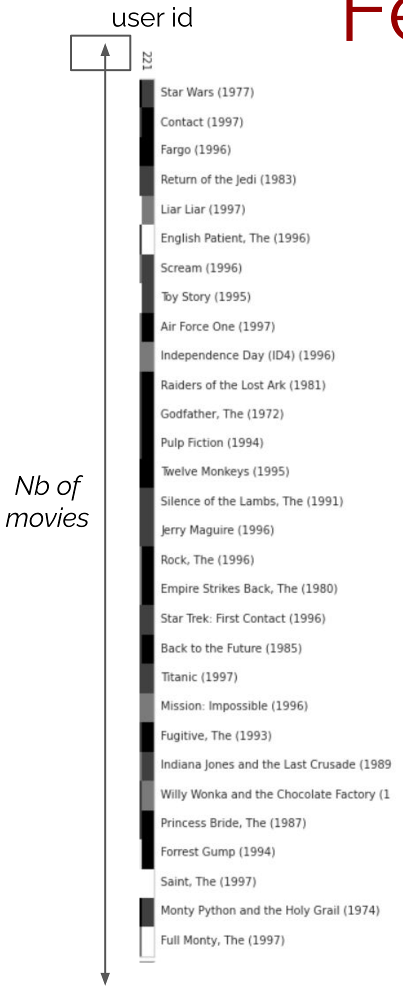


The 30 most rated movies & the top 18 users with most ratings

- We will make clusters based on the the 1000 most rated movies & corresponding users (out of +9000 in the dataset).
  - ➢ **More dense** and **understandable** than the entire dataset
  - ➢ White cells corresponds to not rated movies for corresponding users. We still have to manage the **sparsity**.

First modelling strategy

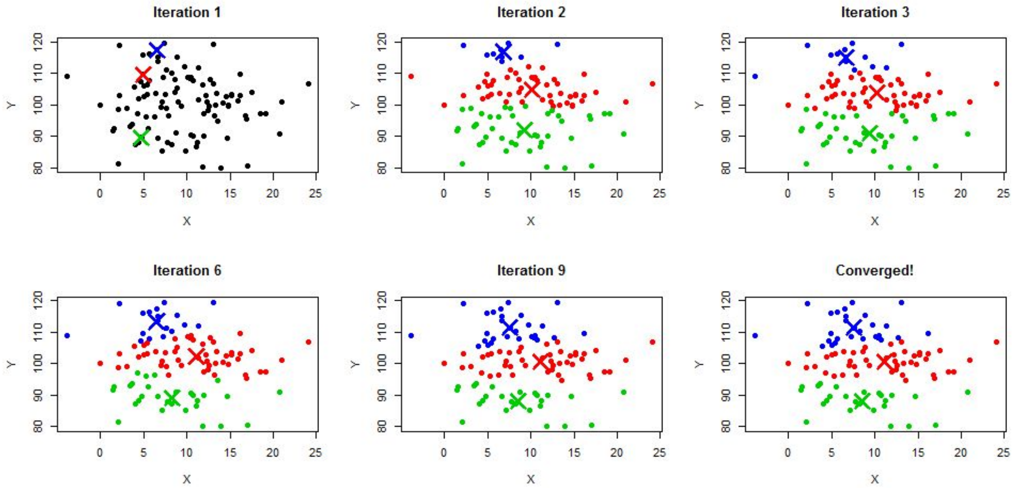
Feature representation



Example of a user vector

- We will base the principle of the clustering algorithm on a **representation of the user as a vector.**
  - ➢ The length of the vector would be **the total number of movies that exists**
  - ➢ The **weights** of the vector would be **0 to 5 (ratings)**

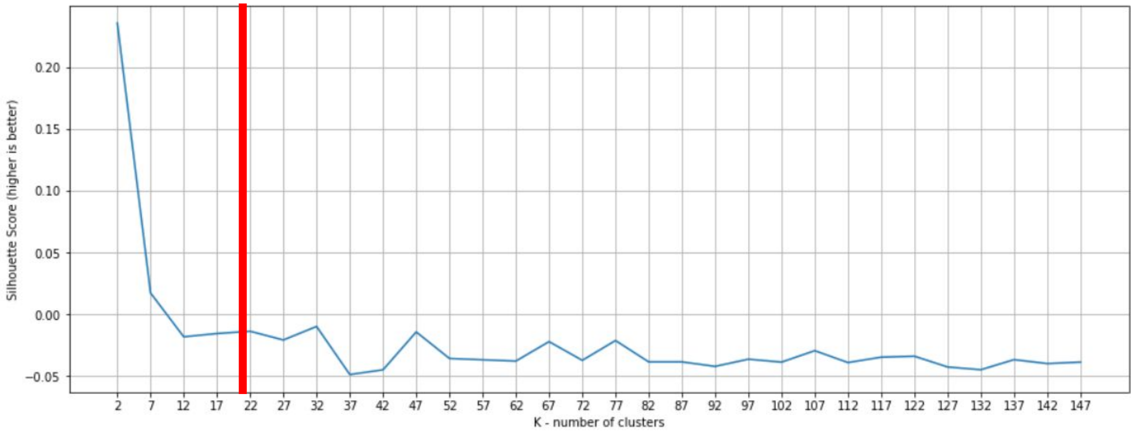
Using Clustering : K-Means to optimize recommendations



K-Means scheme

- The K-Means method put **n observations (users) into k clusters** in which each observation belongs to the cluster with the nearest euclidean metric with centroids.
  - ➢ The second step is to create **new centroids by taking the mean value of all of the samples assigned to each previous centroid.** The border between the old and the new centroids are computed and the algorithm repeats these last two steps until this value is less than a threshold (until the centroids do not move significantly).
  - ➢ The **K-Means** will automatically cluster regarding users ratings..
  - ➢ We measure user distance with a **similarity metric (users = vectors)**

K-Means clustering optimization

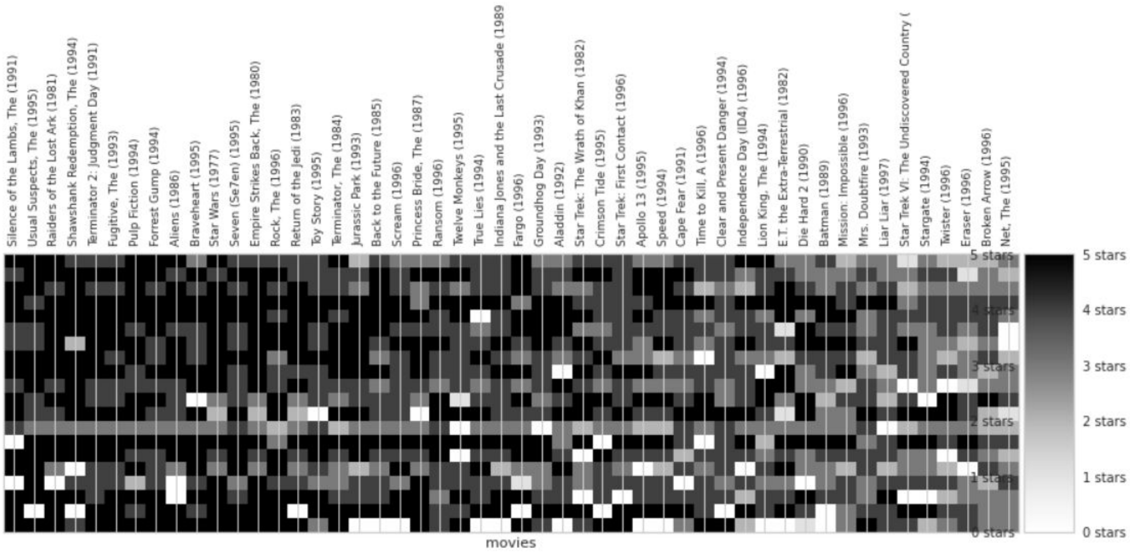


Silhouette score vs number of clusters

- Backtesting the silhouette score of K-Means for a lot of cluster numbers K to evaluate the performance.
  - ➢ **We want to have the one** with the best silhouette score for the highest cluster number.
  - ➢ We will use this K to **have the best clustering algorithm**
  - ➢ We will chose **the optimal K on the inflection point :** here we chose **K = 20**.

Performances

Cluster analysis



Cluster #11 (20 users)

- The **vertical lines represent similarities** in the clusters.
- Looking to the heatmap of the clusters, we can spot **trends in the clusters** :
  - ➢ Some clusters are really black meaning that **it brings together people who really love a certain set of movies.**
  - ➢ It's easy to spot **horizontal lines with similar colors**, these are users without a lot of variety in their ratings. **A rating of four stars means different things** to different people.
  - ➢ We did a few things to make the clusters visible : *(filtering/sorting/slicing)*. This is because datasets like this are "sparse" and most cells do not have a value **because most people did not watch most movies**.

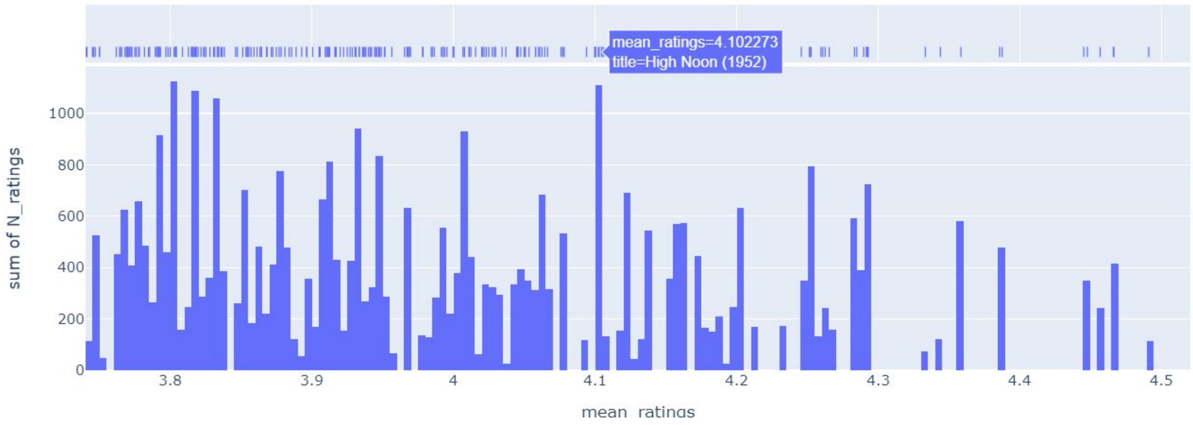
Recommendation system

Glory (1989)	4.500000
Schindler's List (1993)	4.428571
Shawshank Redemption, The (1994)	4.363636
Fugitive, The (1993)	4.357143
Usual Suspects, The (1995)	4.333333
Time to Kill, A (1996)	4.333333
American President, The (1995)	4.300000
Miracle on 34th Street (1994)	4.222222
North by Northwest (1959)	4.222222
Seven (Se7en) (1995)	4.166667

Recommendation system based on clustering

- From our clustering, it is easy to imagine a **recommendation system** that would be based on users data & clustering.
  - ➢ We can **average the ratings of all the other users in the cluster**, and **output the 10 best movies** based on average and not already seen by the user.

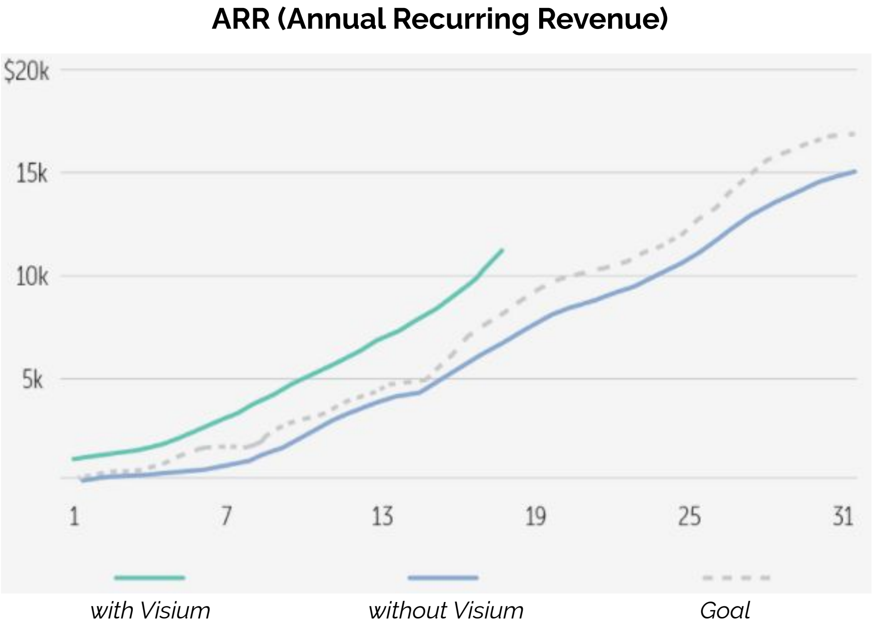
Popularity metric



Popular recommendations

- An interesting metric might be to **look at the number of ratings as a function of the average rating**.
  - ➢ This gives us an idea of which movies are **important to keep on the platform** as they are **very popular.**
  - ➢ **This popularity generate user engagement** and is one reason why **users will stay on the platform.**

ARR Metric



ARR = Current Subs + New Subs + Upgrading Sub - Downgroading Subs - Cancelled Subs

- An other interesting metric might be to **look at the ARR Metric**.
  - ➢ The ARR Metric measures the **total amount gain $ from subscriptions** (taking into account the cancelations)**.**
  - ➢ **It is an essential SaaS metrics** that allow to be aware of the **gains generated by the recommendation system.**
  - ➢ We can measure the ARR with and without the recommender system and **establish a price as a consequence (price to value).**

Possible enhancements/remarks

- **If the cluster had a movie with only one rating. And that rating was 5 stars the average rating of the cluster for that movie is 5**. It can affect our simple recommendation engine.
  - ➢ We could weight the recommendations based on number of rating to address this issue.
- **To deal with the sparsity, we can use the full dataset :** Movielens containing 24 million ratings.
- We have noticed that there is a problem of subjectivity in ratings: the average ratings of users are not the same. **A more objective way of dealing with ratings would be to normalise them.**

- Build a system with others more sophisticated methods :
  - ➢ We could enhance the K-means algorithm using the SVD method : we create two matrices A (users X len feature vector) and B (len feature vector X items) such that **A x B is about equal to the rating matrix**.
  - ➢ In this approach we use the gradient descent to make the product as close as possible to the rating matrix. A = user representation matrix -> K-means.
  - ➢ Good solution to avoid sparsity.

Contact

Saussaye Matthieu, Project Manager
Email : saussayematthieu50@gmail.com
Mob. : +33 6 46 04 69 09