DataScientest.com

```
fig, ax = plt.subplots(1, figsize=(10,10))

# plot des data
plt.scatter(df_new['Nombre de vues'], df_new['Nombre

# Legende
legend_elements = [Line2D([0], [0], marker='o', colo
                  markerfacecolor=mcolor, markersize=5

# plot de la legende
plt.legend(handles=legend_elements, loc='upper left

# titre et labels
plt.title("Visualisation des utilisateurs", bbox={'
plt.xlabel('Nombre de vues')
plt.ylabel('Nombre de transactions')
```

Projet PyCommerce

Quentin Lubken • Jérémie Guedj • Cédric Bouré

Bootcamp Data Analyst • Mars 2021

Présentation et objectifs

Site ecommerce Retailrocket

- Le projet consiste en l'analyse des données du site Ecommerce Retailrocket disponible sur Kaggle (https://www.kaggle.com/retailrocket/ecommerce-dataset), données partiellement anonymisées.
- On souhaite dans un premier temps comprendre les données que l'on a à disposition en les analysant :
  - Des lignes sont-elles dupliquées ? Faut-il nettoyer le jeu de données ?
  - Combien avons-nous de visiteurs uniques et de produits ?
  - Combien de produits ont été consultés, ajoutés au panier, achetés ?
  - Combien de transactions ? D'abandon ? Quelle popularité des produits ?
  - Avec une **question centrale** : au-delà de l'état des lieux, avons-nous capacité à suggérer des produits qui vont susciter un intérêt à partir de produits précédemment achetés dans un objectif d'améliorer les ventes futures ?

Contexte projet, données

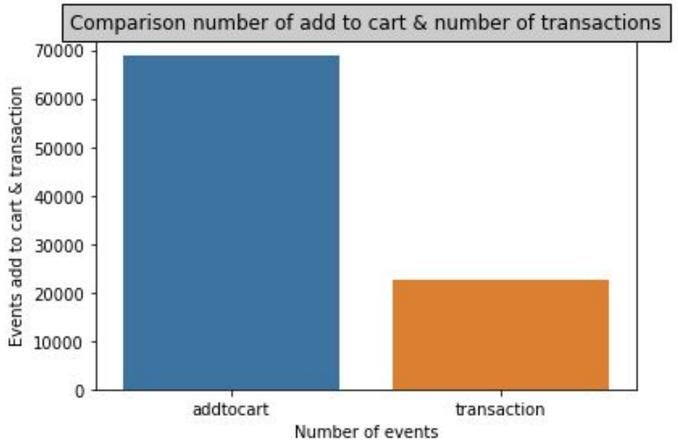
Site ecommerce Retailrocket

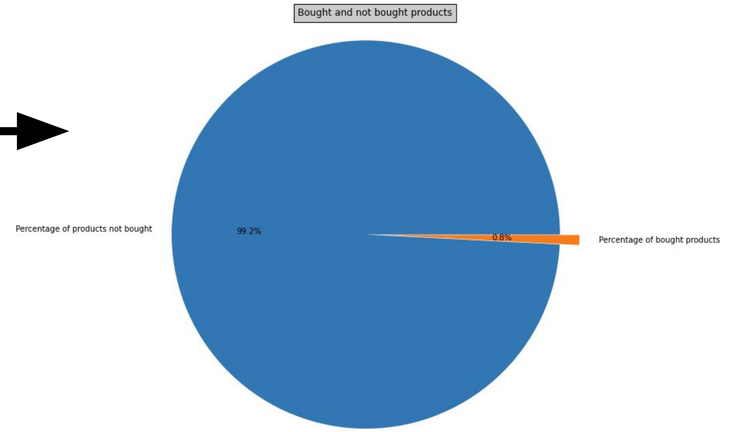
- À notre disposition 4 fichiers CSV :
  - « item_properties_part1.csv » et « item_properties_part2.csv » composés au total de **20 275 902 lignes** et de **4 colonnes** : timestamp, itemid, property (propriété produit), value (ID catégorie)
  - « category_tree.csv » composé uniquement de **2 colonnes** : category et parentid (catégorie et famille)
  - « events.csv » composé de **5 colonnes** : timestamp, visitorid, event (click, add to cart, transaction), itemid, transactionid (si existe).
- Nous nous intéressons rapidement au fichier « events.csv » qui comporte des données visiteurs, comportementales et temps et nous donne un aperçu de données à exploiter pour la suite de l'analyse.
- Une partie des données est **anonymisée** et semble être aléatoire, ajoutant de la complexité à l'analyse.
- La donnée de type **timestamp** (unix), est reconvertie en date/heure et nous donne l'indication d'une période d'agrégation des données disponibles pour l'analyse de 137 jours (02/05/2015 ->17/09/2015).

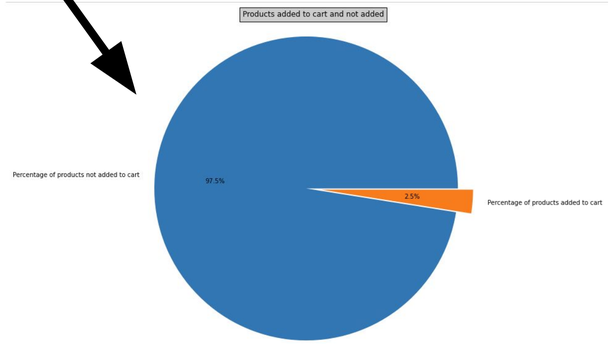
Résultats

Quelques visualisations

- Un taux d'abandon panier de 67,6 %, taux élevé mais 70% taux moyen en B2C.
- 99,2 % des produits ne sont pas achetés, c’est un mauvais score
- Seulement 2,5 % des produits sont ajoutés au panier, score très faible.
- 1 407 580 utilisateurs uniques, 235 061 produits, 17 672 transactions. 460 lignes sont dupliquées.
- 2 664 218 produits ont été consultés, 68 966 produits ont été ajoutés au panier, 22 457 produits ont été vendus.
- « transactionid » représente 99% de valeurs nulles, un taux très élevé.



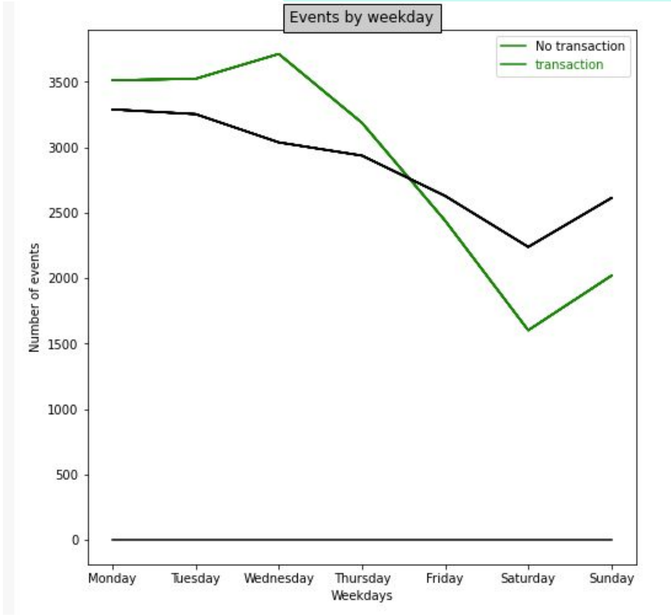


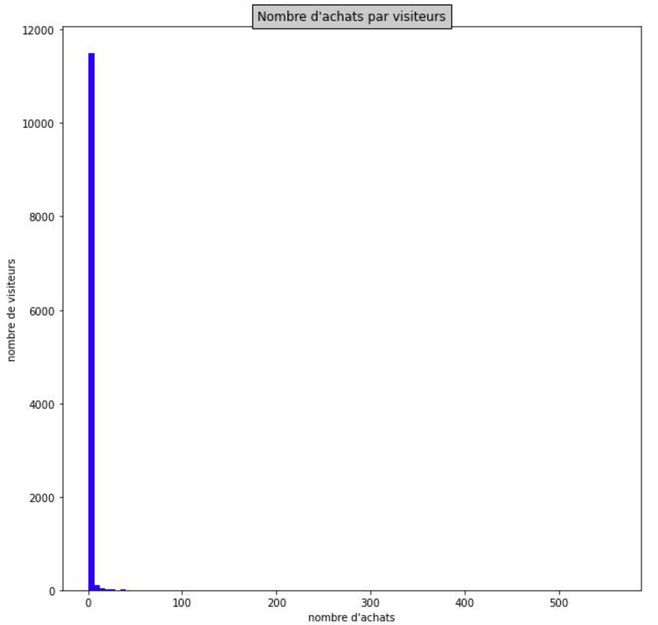


Résultats

Quelques visualisations

- Le nombre de transactions est plus élevée en début de semaine (pic le mercredi), suivi d’une baisse.
- En souhaitant connaître la répartition du nombre d'achats par visiteur, le graphique nous indique bien que le **nombre d'achats par visiteur est anormalement élevé.**
- En utilisant les calculs .count(), .unique(), .value_counts() on constate tout de suite un déséquilibre : le visitorid 1150086 est venu **7 757** fois, le visitorid 530559 est venu **4 328** fois, nous menant rapidement à une suspicion de bots.
- **99,8%** des personnes font moins de **100 achats**. On peut donc considérer que tout le reste sont des outliers, comportements anormaux qu’il faudra traiter à part.
- « transactionid » représente 99% de valeurs nulles, un taux très élevé.





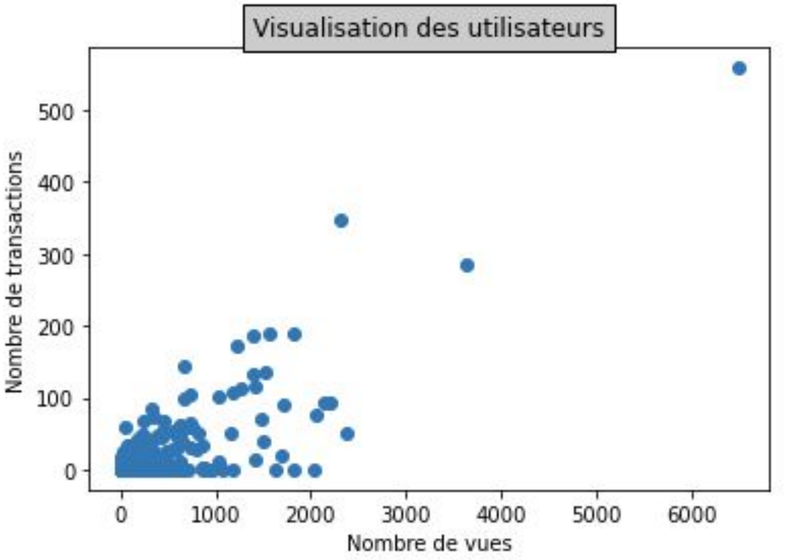
Résultats

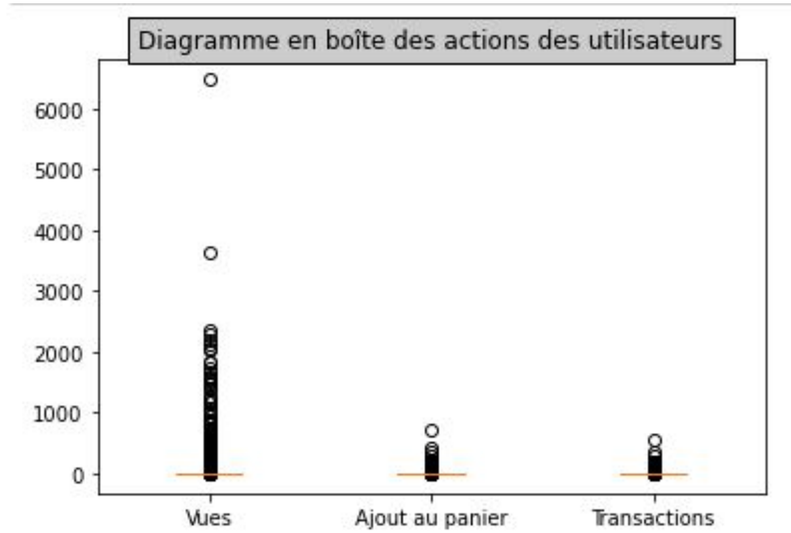
Préparation des données : clustering

- Au préalable du travail de modélisation, nous séparons en clusters les visiteurs selon le nombre et le type d'actions et ainsi **repérer et isoler** les visiteurs aux comportements anormaux.

- Après un essai de clustering hiérarchique non convaincant, on réalise un clustering avec K-means où on cherche à visualiser les données.

- On observe ce que les calculs nous ont déjà confirmés : la majorité des utilisateurs a consulté et acheté des produits de manière raisonnable, cependant certains d'entre eux ont réalisé **plus d'une centaine d'achats en 4 mois** et ont consulté le site de manière trop fréquente pour un utilisateur lambda. On les isolent.

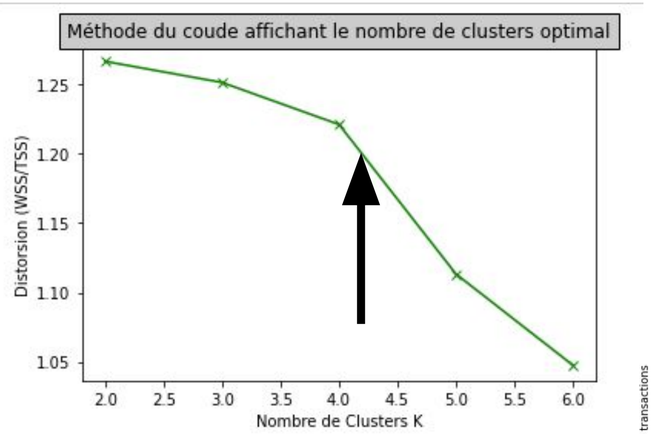


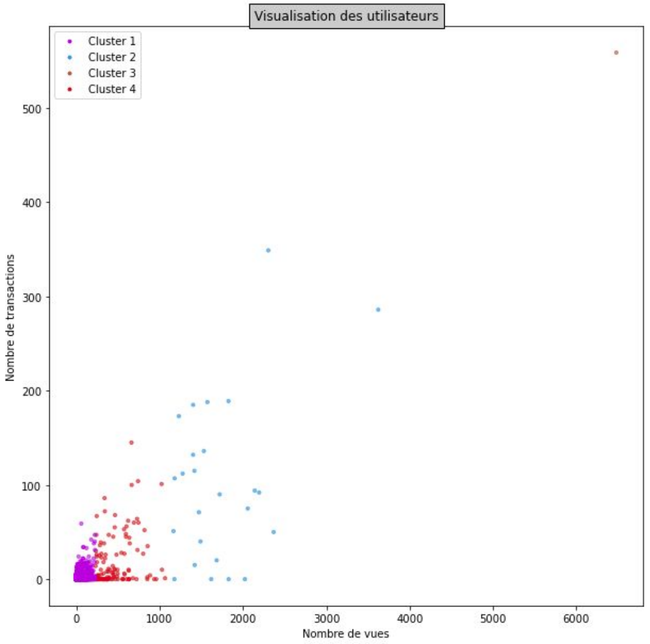


Résultats

Clustering

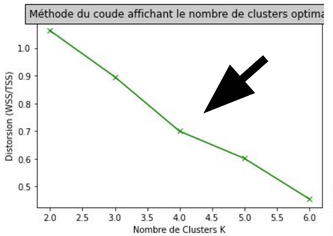
1

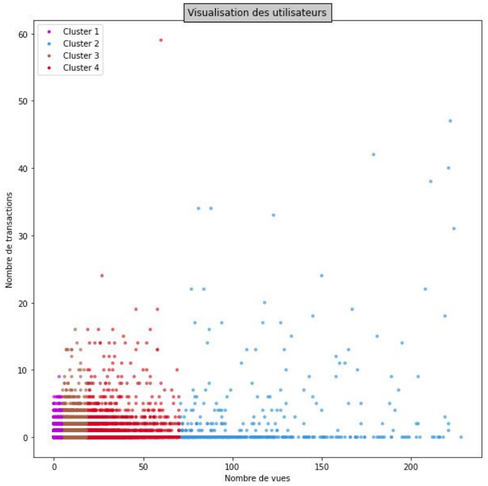




- On réalise alors la méthode du coude qui va nous permettre de déterminer le **juste nombre de clusters** nécessaire puis on visualise les distorsions en fonction du nombre de clusters. **4 clusters distincts** sont alors définis, on les classe.
- Un calcul nous indique que :
  - 3 401 visiteurs, tous clusters confondus, sont venus sur le site **sans regarder** un seul item,
  - 1 369 858 visiteurs, tous clusters confondus, sont venus sur le site **sans ajouter** au panier un seul item
  - 1 395 861 visiteurs, tous clusters confondus, sont venus sur le site **sans acheter** un seul item
- **99,98%** des utilisateurs sont présents dans le cluster 0, on se concentre sur ce cluster que l'on sépare une seconde fois avec la méthode du coude car encore trop volumineux.
- Avec l'algorithme des K-means, on obtient une distribution plus homogène.

2





Résultats

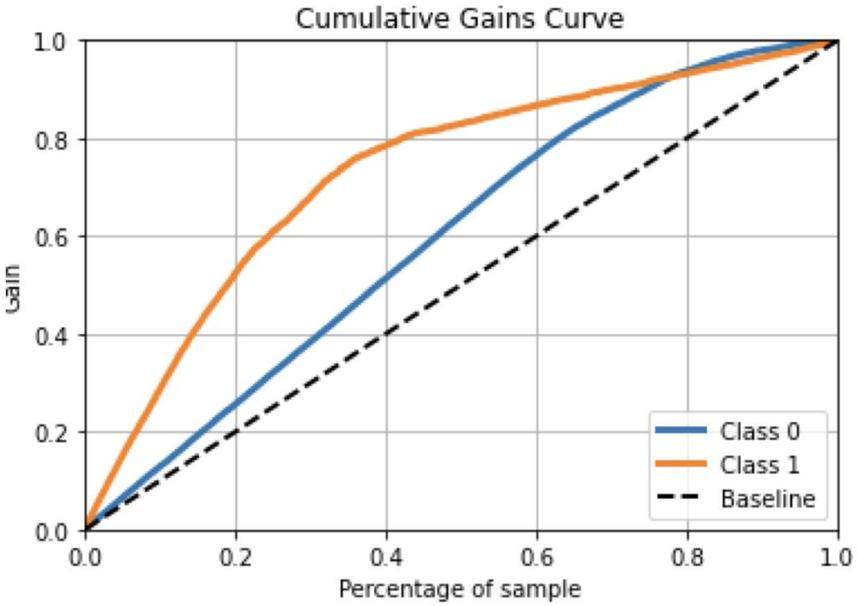
Machine learning

- Nous réalisons un régression logistique avec la fonction LogisticRegression() qui devra prédire le comportement d'achat futur des visiteurs. La précision de notre modèle de prédiction d'achat est de **79,29 %**.

- Nous utilisons la méthode **Random Forest** qui nous renvoi une prédiction d'achat de **81,84 %**.

- La courbe lift nous indique par exemple que si on choisit les 40% des utilisateurs ayant le meilleur score, on réussit à atteindre 80% des utilisateurs qui vont faire un achat.

- Nous utilisons la méthode **SVM** qui nous renvoi une prédiction d'achat de **81,47 %** après amélioration du modèle.



Résultats

Moteur de recommandation

	visitorid	Item acheté	cluster
0	599528	[356475]	0
1	121688	[15335, 380775, 237753, 317178, 12836, 400969,...	0
2	189384	[310791, 299044]	0
3	350566	[54058, 284871, 251130, 268335, 183049, 261940...	1
4	404403	[150100, 50934, 36013, 26210, 118199, 234199, ...	3

- Avec ces données, nous souhaitons réaliser un modèle de prédiction afin de **suggérer** des produits aux précédents acheteurs susceptibles de les intéresser

- On recréer un dataframe qui contient les utilisateurs qui ont réalisé **au moins 10 interactions** afin d'éliminer les utilisateurs les moins significatifs et nous conservons ceux qui ont réalisé **à minima un achat**. On y ajoute la liste des items achetés par chaque visiteur et son cluster d'appartenance.

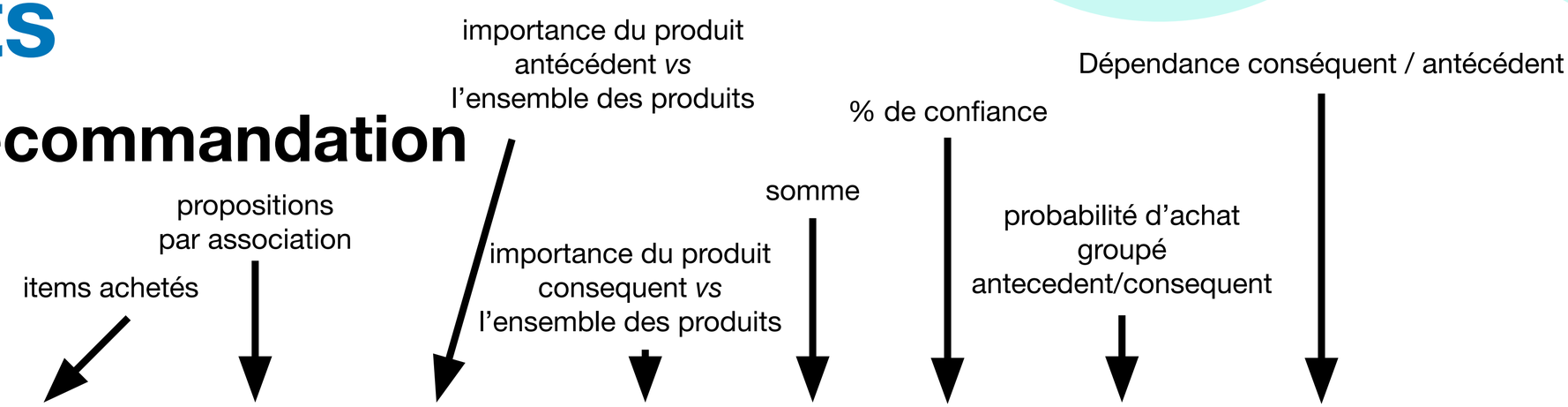
- Pour réaliser ce modèle de prédiction, on applique un algorithme nommé **Apriori** qui exploite ce principe : si un ensemble d'items est fréquent, alors tous ses sous-ensembles sont aussi fréquents. Ainsi, si un ensemble est peu fréquent ou peu éliminer à priori les sous-ensembles qu'il contient et s'il est fréquent on créer des associations.

  - On observe une faible importance des achats groupés sur l'ensemble des achats.

  - On affiche les résultats, avec pour chaque produit acheté (antecedents) une ou plusieurs propositions de produits (consequents) susceptibles d'intéresser les acheteurs.

Résultats

Moteur de recommandation



	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(546)	(119736)	0.003166	0.010554	0.001319	0.416667	39.479167	0.001286	1.696193
1	(546)	(248455)	0.003166	0.004222	0.001055	0.333333	78.958333	0.001042	1.493668
2	(248455)	(546)	0.004222	0.003166	0.001055	0.250000	78.958333	0.001042	1.329112
3	(546)	(338660)	0.003166	0.002639	0.001055	0.333333	126.333333	0.001047	1.496042
4	(338660)	(546)	0.002639	0.003166	0.001055	0.400000	126.333333	0.001047	1.661390
...	...	...	...	...	...	...	...	...	...
141	(384302)	(119736, 338660)	0.002111	0.001319	0.001055	0.500000	379.000000	0.001053	1.997361
142	(213834, 268883)	(445351)	0.001055	0.005805	0.001055	1.000000	172.272727	0.001049	inf
143	(213834, 445351)	(268883)	0.005013	0.004222	0.001055	0.210526	49.868421	0.001034	1.261319
144	(268883, 445351)	(213834)	0.001055	0.011082	0.001055	1.000000	90.238095	0.001044	inf
145	(268883)	(213834, 445351)	0.004222	0.005013	0.001055	0.250000	49.868421	0.001034	1.326649

Résultats

Moteur de recommandation

Démo

Streamlit

Perspectives

Site ecommerce Retailrocket

- Le moteur de recommandation donne des résultats intéressants et directement exploitables sur la base des transactions réalisées.
- Ce modèle est parfaitement transposable à d'autres événements et donc à d'autres visiteurs qui visualisent ou mettent au panier un ou plusieurs items.
- Possibilité d'élargir l'outil de recommandation à d'autres variables. (Timestamp)
- Intéressant de définir combien de produits vont être recommandé pour un seul produit consulté/ ajouté au panier.
- Il est donc intéressant d'intégrer que ce modèle de suggestion est duplicable et facilement modifiable.

Conclusion

Site ecommerce Retailrocket

- Pas de connaissance du contexte. Nous avons donc latitude à nous poser des tas de questions sans pour autant savoir si cela est pertinent en terme d'analyse.

- Notre phase d'exploration était donc conséquente par manque de contextualisation. Notre modèle est assez simple à mettre dans un contexte plus détaillé.

- On aurait souhaité réaliser un calcul complémentaire en ajoutant les clusters de chaque utilisateur comme variable supplémentaire de notre outil de recommandation mais la solution technique ne nous a pas apparu évidente pour réaliser des calculs pertinents.