Rapport de projet : Prédiction du nombre de likes d'une vidéo YouTube en tendance

I – Choix du dataset et justification

Pour notre projet, nous avons choisi le dataset « Trending YouTube Video Statistics ». En effet, il contient des informations qui nous semblaient majeures pour le bon déroulement de notre projet, telles que le nombre de likes et de dislikes d'une vidéo ou encore les tags (des sortes de balises utilisées pour décrire le contenu d'une vidéo en quelques mots clés). De plus, contrairement aux autres bases de données que nous avons pu trouver, celle-ci a la particularité de remonter assez loin dans le temps, jusqu'en 2009 et de posséder un fichier uniquement consacré aux données françaises.
Chaque pays possède une liste de vidéos en tendance différente. Nous avons donc décidé de nous concentrer sur le fichier des données françaises car ce sont ces vidéos que nous sommes le plus susceptibles de regarder.

II – Description de la tâche à résoudre

Nous souhaitons réussir à déterminer le nombre de likes qu'une vidéo aura suivant plusieurs facteurs tels que son nombre de dislikes, de commentaires, sa date de publication, le nombre de jour passé dans les tendances… Nous souhaitons identifier les paramètres importants, pour cela nous allons étoffer au fur et à mesure du projet notre jeu de données avec tous les attributs à notre disposition dans le dataset. Nous allons procéder plusieurs fois avec la même méthode en rajoutant des informations dans nos données pour essayer de déterminer les attributs importants.

Nous avons plusieurs idées sur la manière de résoudre cette tâche, nous allons dans un premier temps essayer de faire un clustering où chaque classe correspondra à un nombre de likes, ces classes seront dénotées par des labels.
Dans un second temps, nous allons faire une régression pour prédire un nombre de likes précis et nous vérifierons nos prédictions avec une méthode de scoring.

III – Description du dataset, analyse statistique succincte de la distribution des données, éventuel nettoyage des données

Le dataset comprend les attributs suivants :

- `video_id` : l'identifiant unique d'une vidéo (int),
- `trending_date` : la date à laquelle la vidéo est en tendance (Object),
- `title` : le titre de la vidéo (string),
- `channel_title` : le nom de la chaîne qui a posté la vidéo (string),
- `category_id` : la catégorie à laquelle la vidéo appartient (ex : musique, humour, divertissement …) (int),
- `publish_time` : la date et l'heure à laquelle la vidéo est postée (Object),

- `tags` : les tags associés à la vidéos (liste de string),
- `views`, `likes`, `dislikes`, `comment_count` : le nombre de vues, de likes, de dislikes et de commentaires respectivement (int),
- `thumbail_link` : le lien vers l'image de miniature de la vidéo (string),
- `comments_disabled` : commentaires de la vidéo désactivé (booléen),
- `ratings_disabled` : likes et dislikes désactivés (booléen),
- `video_error_or_removed` : erreur de la vidéo ou vidéo supprimée (booléen),
- `description` : description de la vidéo (string).

Dans un premier temps, nous regardons s'il existe des données nulles dans notre dataset. On en trouve dans l'attribut `description`, comme nous ne comptons pas nous servir de cet attribut, nous décidons d'ignorer ces valeurs nulles, elles ne nous poseront pas de problèmes.

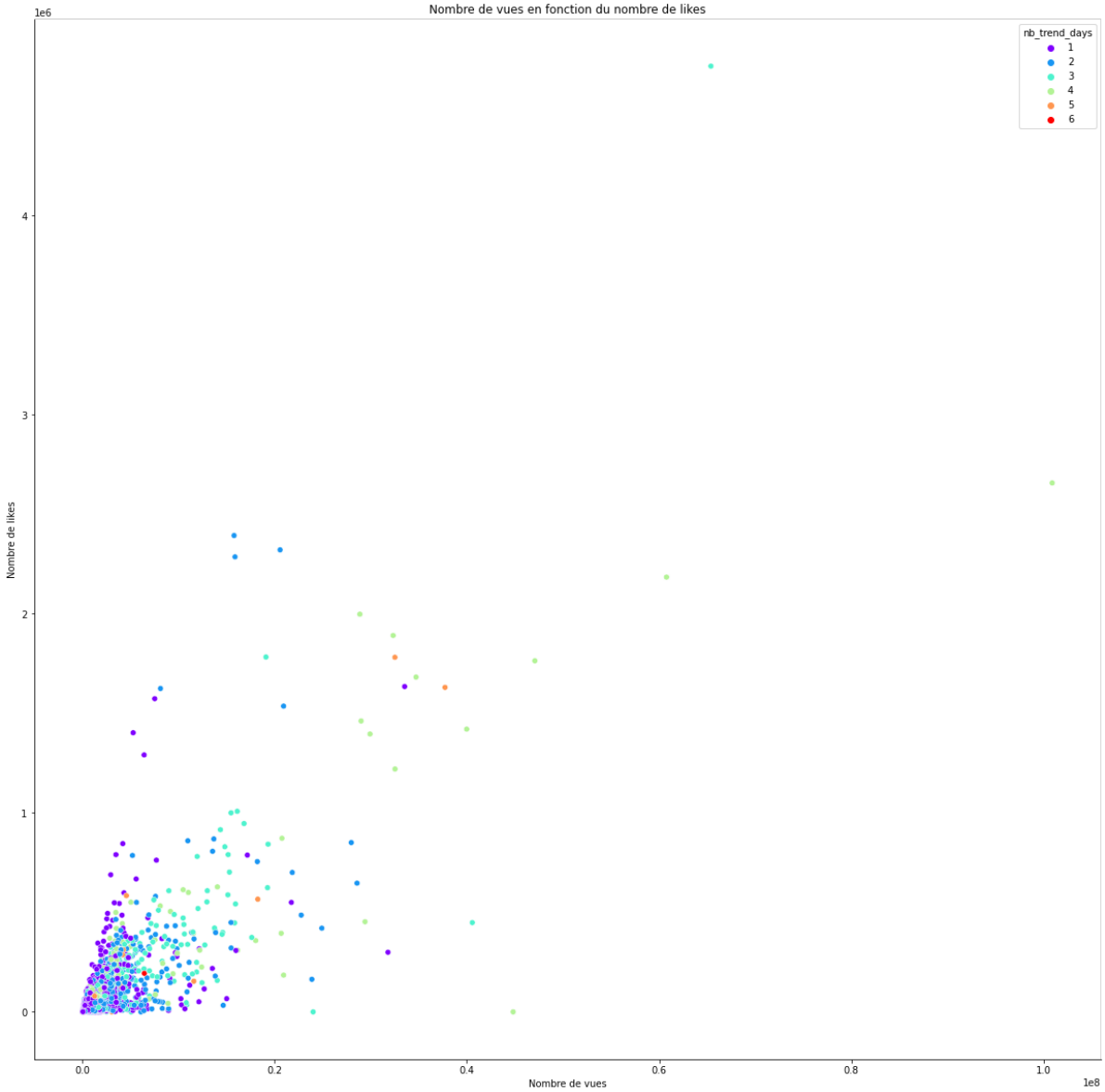
Ensuite, nous décidons de regarder s'il existe des doublons, et nous nous rendons compte que en effet, une même vidéo peut apparaître plusieurs fois dans le dataset. Cela est dû à l'attribut `trending_date`, en effet, une vidéo peut rester en tendance plus d'une journée et se retrouve dans notre dataset autant de fois qu'elle reste de jours en tendance. De plus, on remarque que seuls les attributs `views`, `likes`, `dislikes` et `comment_count` changent.
Nous décidons, pour éliminer les doublons tout en gardant l'information la plus à jour, de créer un attribut `nb_trend_days` pour le nombre de jours qu'une vidéo aura passés en tendance (soit le nombre de fois qu'on la retrouve dans notre dataset) et de ne garder parmi les doublons que celui dont l'attribut `trending_date` est le plus récent, donc celui avec les informations les plus à jour.

Une fois l'élimination des doublons terminée, on utilise la méthode `describe` pour observer nos données.

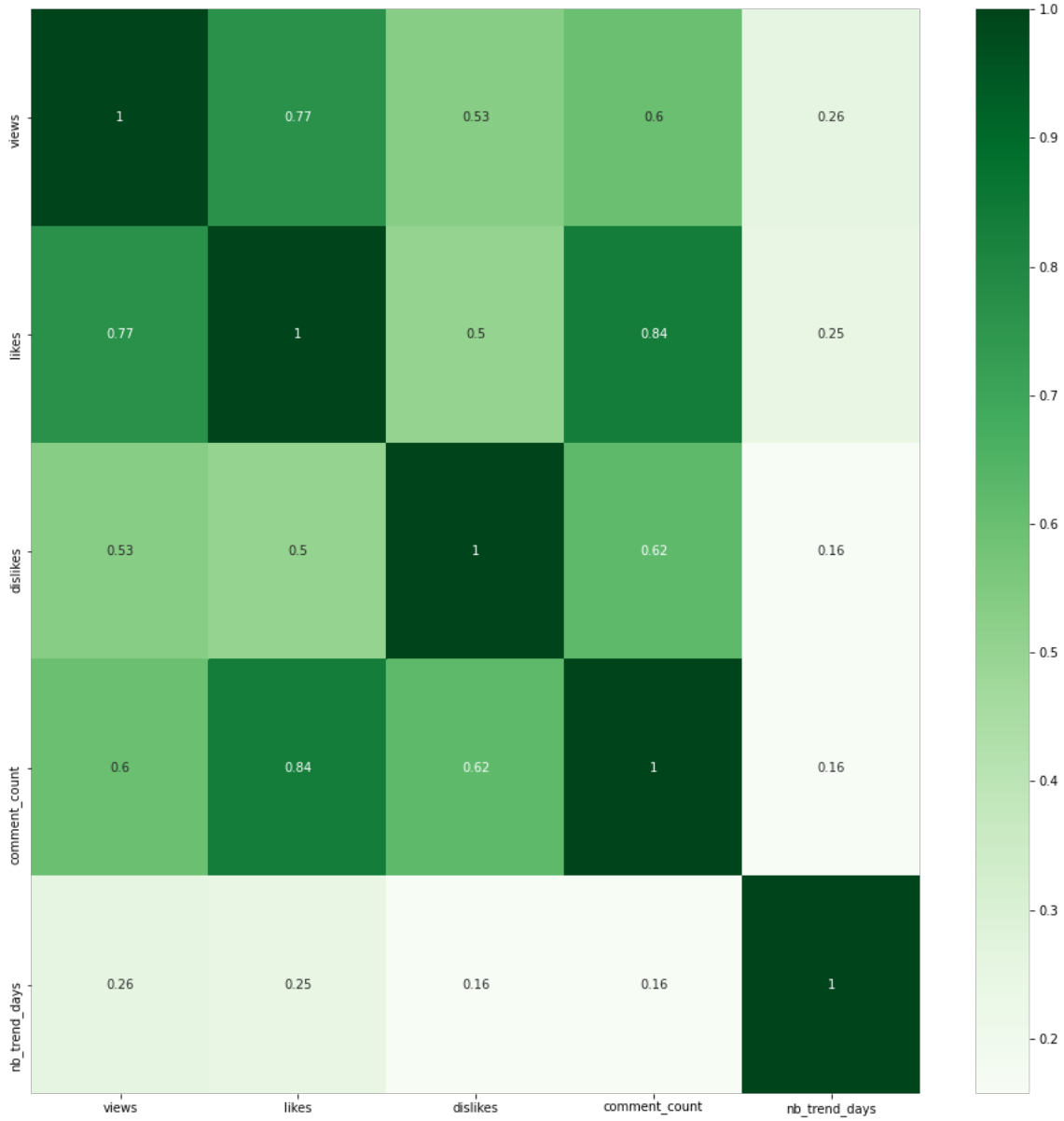
	views	likes	dislikes	comment_count	nb_trend_days
count	3.057800e+04	3.057800e+04	30578.000000	30578.000000	30578.000000
mean	3.384419e+05	1.176385e+04	536.546439	1258.501308	1.331447
std	1.334377e+06	6.176685e+04	4381.706368	8067.874912	0.651483
min	2.840000e+02	0.000000e+00	0.000000	0.000000	1.000000
25%	1.197050e+04	2.500000e+02	13.000000	41.000000	1.000000
50%	5.350300e+04	1.265500e+03	58.000000	174.000000	1.000000
75%	2.159485e+05	4.977500e+03	246.000000	611.000000	1.000000
max	4.707871e+07	2.392595e+06	398361.000000	611327.000000	6.000000

On constate que la plupart des vidéos de notre base ne passent qu'une seule journée en tendance.

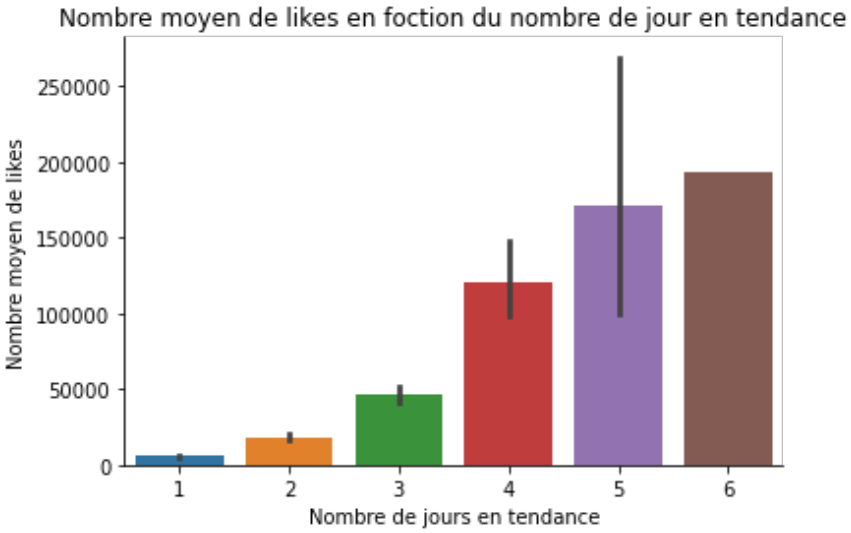
On affiche ensuite quelques graphes qui vont nous donner une idée de la distribution de nos données :



On constate des points très isolés qu'on décide de supprimer, comme celui à plus de 4,5 millions de likes ou ceux à plus de 6 millions de vues. On constate une légère tendance entre le nombre de vues et le nombre de likes, comme on s'y attendait. On affiche une heat map pour mettre encore plus en évidence ces liens, et on en profite pour rajouter le nombre de dislikes, de commentaires et de jours passés en tendance qui nous paraissent aussi importants à première vue.

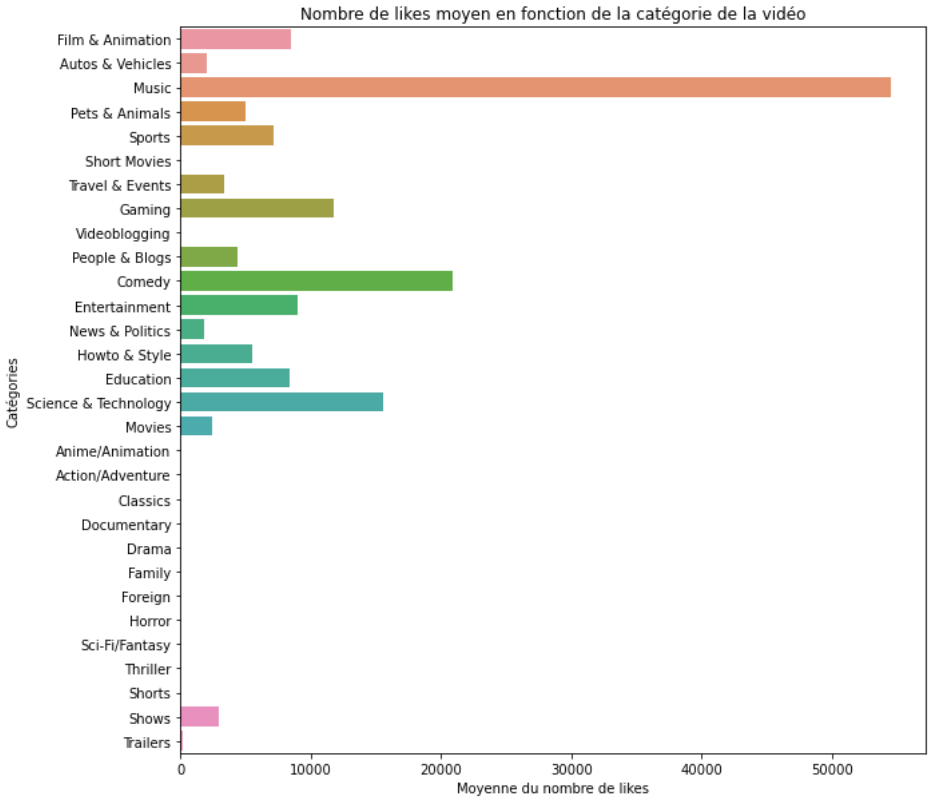


On constate qu'il y a bien une corrélation entre le nombre de likes et le nombre de vues de 77%, on remarque aussi que le nombre de likes est corrélé à 84% au nombre de commentaires. On remarque également que le nombre de jour passé en tendance n'est corrélé qu'à 25% au nombre de likes, on décide de se pencher sur la question car cela nous semble contre-intuitif. En effet, plus une vidéo sera en tendance plus elle sera exposée et donc plus elle aura de vues et par conséquent de likes. Mais le fait que la plupart des vidéos (plus de 75%) ne passent qu'un seul jour en tendance fausse un peu le résultat.



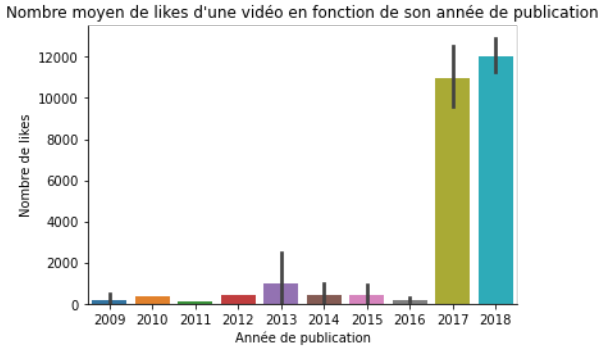
C'est ce phénomène qu'on observe très bien avec ce graphique, plus une vidéo restera de jours en tendance plus elle aura de likes.

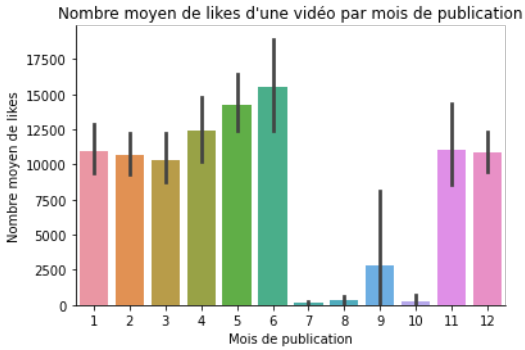
Ensuite, on regarde la distribution du nombre moyen de likes en fonction de la catégorie de la vidéo :

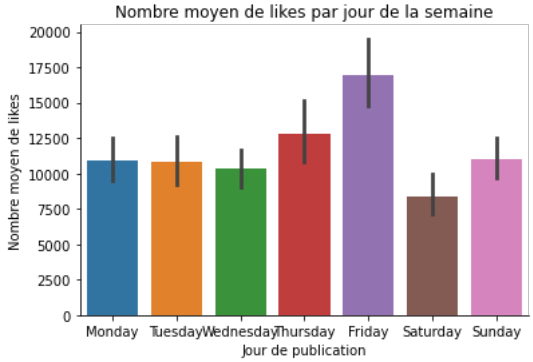


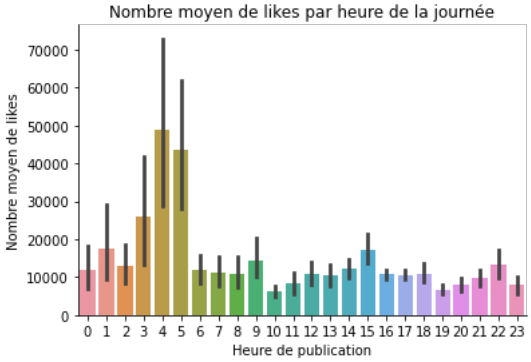
On observe que ce sont les vidéos de la catégorie musique qui obtiennent en moyenne le plus de likes.

On fait de même pour les années de publication, les mois, les jours et les heures :



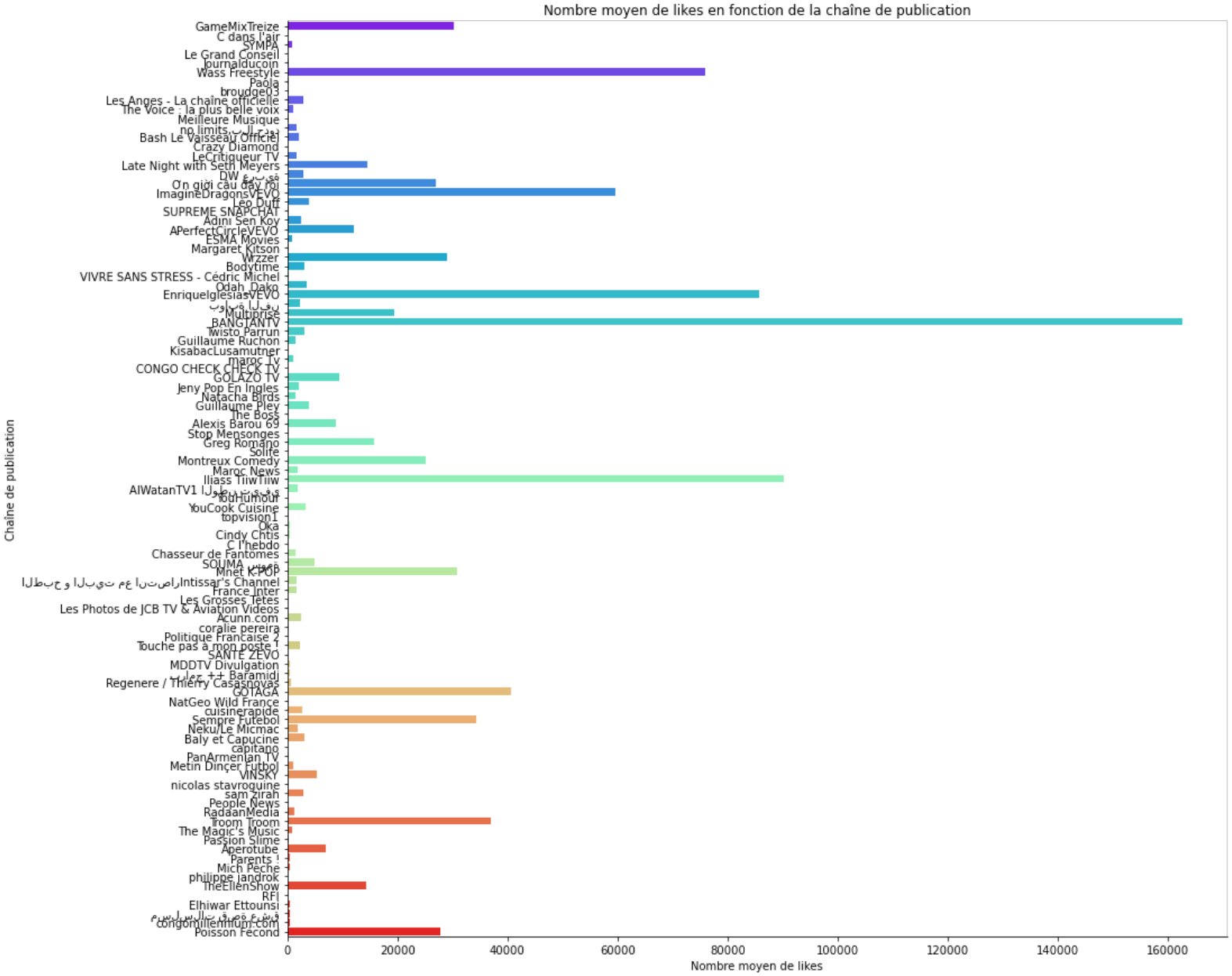




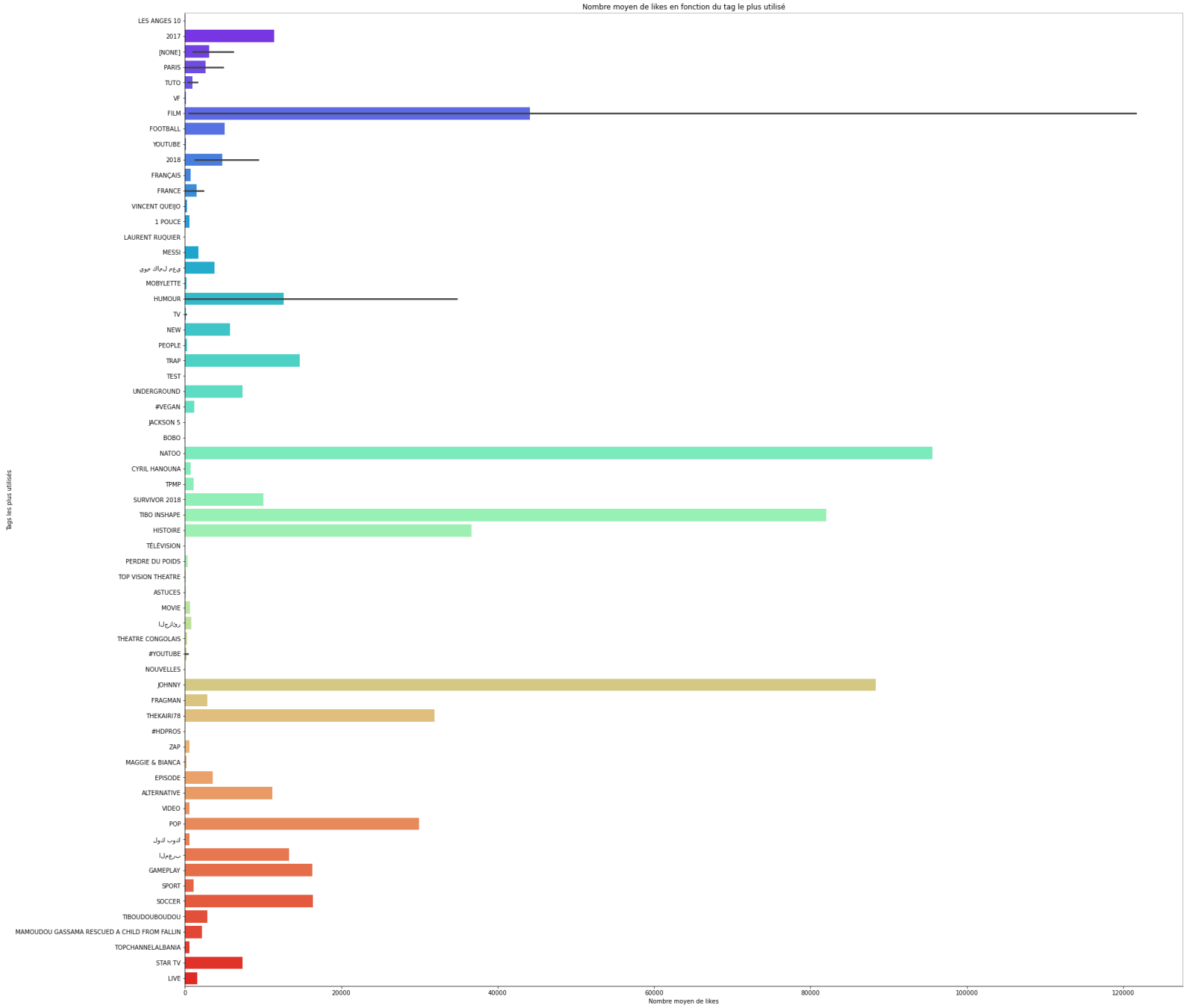


On observe que les vidéos qui ont le plus de likes en moyenne ont été publiées entre 2017 et 2018, ce qui s'explique par le fait que dans notre dataset nous n'utilisons que les informations des vidéos en tendance et que cet outil n'est apparu qu'en 2017.
On remarque également que les heures de publication auxquelles le nombre moyen de likes sur une vidéo est le plus élevée sont entre 3 et 6 heures du matin. Ces horaires peuvent s'expliquer par le fait que, comme on l'a déjà constaté, ce sont les vidéos de musique qui font le plus de likes en moyenne et on peut avancer l'hypothèse de vidéos de k-pop, style de musique coréen qui a beaucoup de fans, en effet le décalage horaire avec la France concorde (+7h). Si les vidéos sont en tendance à 10h en Corée et elles le seront à 3h du matin en France. Ainsi, on constate que le nombre moyen de likes est plus important pour les vidéos publiées en début d'été (mai-juin) le vendredi soit vers 9h soit vers 15h (en ne prenant pas en compte le pic décalage horaire).

Ensuite on s'intéresse au nombre moyen de likes en fonction de la chaîne qui publie la vidéo. On remarque que ce sont encore une fois les vidéos de musique qui génèrent le plus de likes car ce sont les chaînes VEVO qui sont les plus likées d'après le graphe. N'hésitez pas à relancer plusieurs fois cette cellule pour voir les différentes chaînes et le nombre moyen de likes qu'elles génèrent. Des warnings peuvent apparaître, ils sont dus au fait que Python ne supporte pas certain des caractères utilisés, notamment les hanja coréens.



Finalement, on s'intéresse au nombre moyen de likes en fonction du tag de la vidéo. On s'intéresse seulement à un échantillon de 100 vidéos pour que le graphe reste lisible. Encore une fois, n'hésitez pas à relancer plusieurs fois cette cellule pour voir les différents tags et le nombre moyen de likes auxquels ils sont associés. Des warnings peuvent apparaître, ils sont dus au fait que Python ne supporte pas certain des caractères utilisés, notamment les hanja coréens.



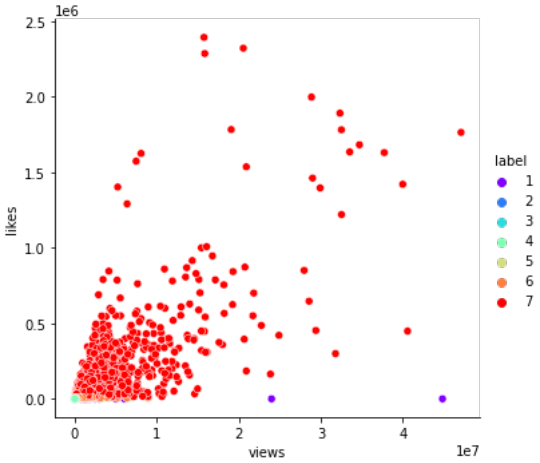
IV – Description de la méthodologie appliquée et des résultats attenus a priori de l'analyse

Dans un premier temps, nous allons effectuer un clustering à l'aide de la régression logistique, comme nous l'avons fait en TP. Il nous faut donc des étiquettes sur nos données.
Nous avons donc décider de les créer en prêtant attention à l'homogénéité des clusters. Pour ce faire, nous avons décidé de diviser notre dataset en un certain nombre de clusters par l'intermédiaire des quantiles. Chaque intervalle entre deux quantiles correspond à une étiquette, on a donc des clusters équilibrés.

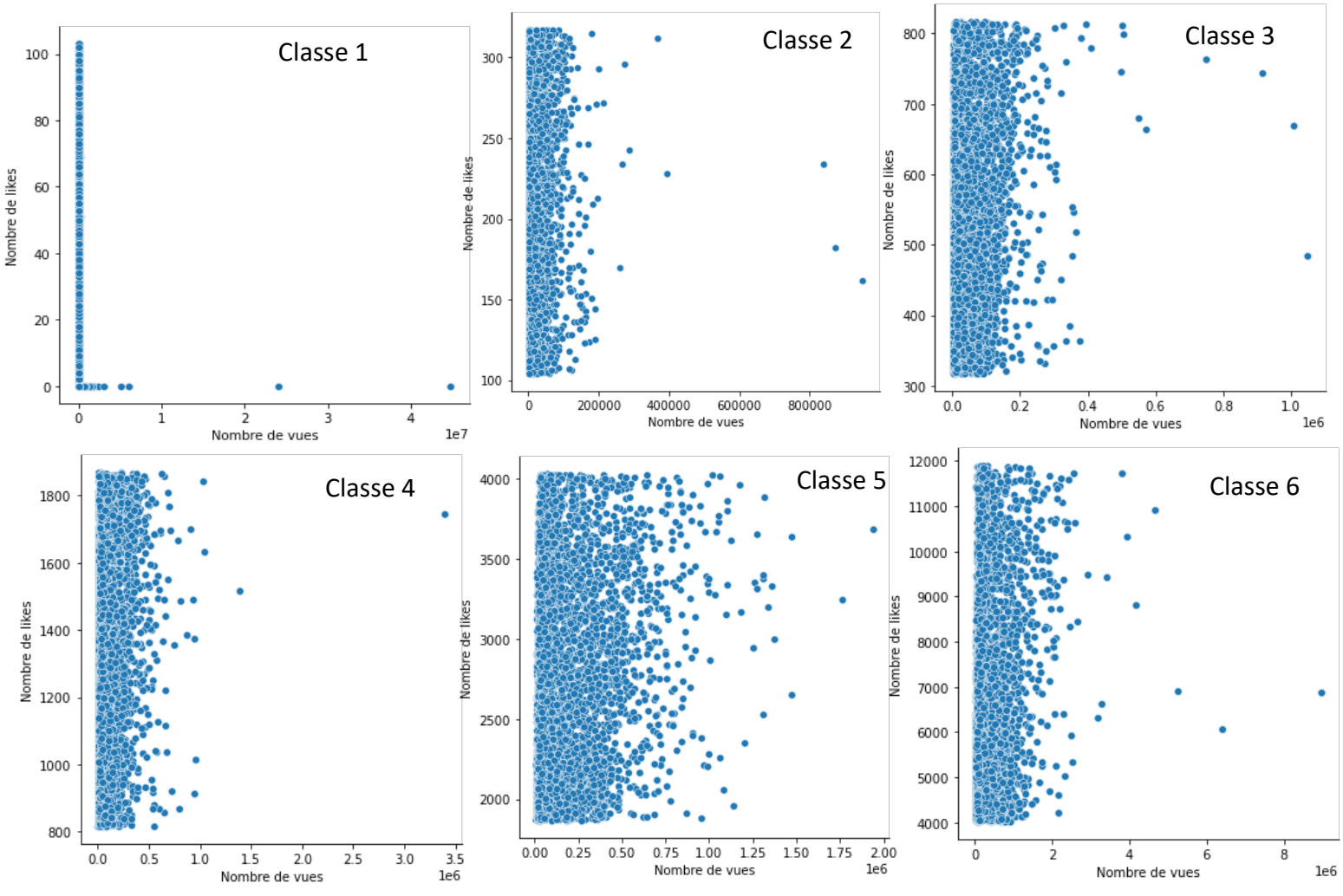
Pour 8 clusters, on a ces quantiles :
`[0.00000000e+00 1.04000000e+02 3.18000000e+02 8.17000000e+02 1.86800000e+03 4.02871429e+03 1.18878571e+04 2.39259500e+06]`
Et on voit bien que les clusters qui en résultent sont équilibrés, comme sur la photo ci-contre.

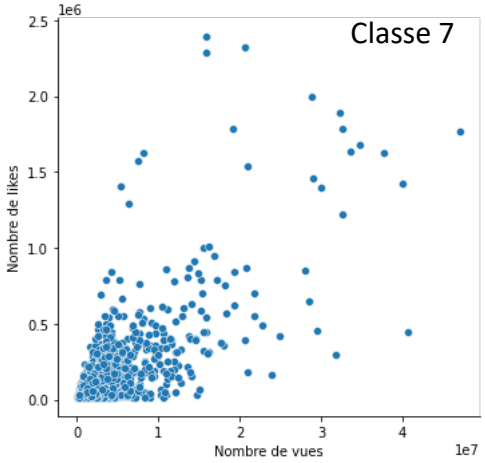
	index	label
0	3	4375
1	7	4369
2	5	4369
3	6	4368
4	4	4368
5	2	4366
6	1	4363

On décide d'afficher les clusters pour voir s'ils ne sont pas trop superposés ce qui rendrait notre tâche assez délicate. Cependant, le graphe est trop grand pour les petits clusters et on ne voit pas grand-chose.



On décide d'afficher les clusters sur différents graphes pour mieux les mettre en évidence.





On voit bien que les différents seuils des quantiles sont respectés mais les classes sont très proches les unes des autres et la dernière part dans tous les sens. On se dit que le clustering va être difficile puisque les différents clusters se superposent beaucoup.

On prévoit donc une seconde méthode : la régression ridge. On se rend compte que le clustering va être difficile, et on décide de prédire un nombre de likes plutôt que la classe associée. De plus, comme nous comptons rajouter des données au fur et à mesure pour affiner notre prédiction et se rendre compte de quels facteurs sont importants, la régression ridge nous paraît comme le meilleur candidat si le clustering ne nous satisfait pas.

Pour ce qui est des données, toutes les données numériques sont passées directement dans notre régression après avoir été standardisés (nous avons décidé qu'il fallait standardiser les données numériques après avoir effectué des tests avec et sans l'outil de standardisation grâce à notre pipeline).
Nous avons décomposé les dates de publication en plusieurs attributs :

- Pour l'année de publication, nous avons simplement crée un attribut `publish_year`
- Pour les mois, les jours de la semaine et les heures de la journée correspondant à la date de publication de chaque vidéo, nous avons à chaque fois crée deux attributs pour ajouter ces informations à nos données. Nous avons fait cela comme indiqué dans le cours afin de respecter le fait que lundi est très proche temporellement de dimanche, ce qui n'est pas du tout retranscrit par des valeurs entières (1 pour lundi et 7 pour dimanche). Nous avons donc créé des attributs de la forme : `sin_publish_day` et `cos_publish_day` pour l'ensemble des valeurs que nous avons donné au-dessus. Pour les valeurs de ces deux nouveaux paramètres, pour chaque nouvelle information nous lui avons attribué les valeurs suivantes (en modulant évidemment en fonction de si on regardait les jours, les mois ou les heures) :

$$\texttt{sin\_publish\_day} = \frac{\sin(2\pi * (jour\ de\ la\ semaine\ \text{à}\ convertir))}{nombre\ de\ jour\ dans\ une\ semaine}$$

$$\texttt{cos\_publish\_day} = \frac{\cos(2\pi * (jour\ de\ la\ semaine\ \text{à}\ convertir))}{nombre\ de\ jour\ dans\ une\ semaine}$$

Après avoir ajouté toutes les valeurs numériques de notre base de données, nous avons décidé d'ajouter le nom des chaînes qui ont publié les vidéos en tendance pour essayer d'améliorer toujours plus notre efficacité. Pour cela nous avons utilisé un processus de one hot encoding que nous avons intégré directement dans notre pipeline sans faire de nettoyage sur les données des chaînes.

Pour les tags, nous avons également utilisé un one hot encoder, mais avant de l'utiliser, nous avons fait en sorte de ne retenir qu'un seul tag pour chaque vidéo de la base. Nous avons retenu celui qui parmi tous les tags de la vidéo en question apparaît le plus dans les autres vidéos de la base de données. Après avoir fait cette manipulation nous avons pu appliquer le one hot encoder (avec le pipeline) pour ajouter l'information qu'on avait gagné avec les tags.

V – Analyse & choix des méta-paramètres

On décide d'utiliser un pipeline comme en TP pour pouvoir moduler assez efficacement nos paramètres, et notamment notre scaler, et on applique la validation croisée.
L'exécution de notre régression logistique n'est pas très concluante, les résultats ne sont pas bons. Nous sommes à environ 43% d'exactitude, pour un C à 100, ce qui ne nous satisfait pas. Comme nous le craignions, les clusters sont trop superposés pour que l'algorithme soit efficace.
Nous essayons ensuite la régression ridge, toujours avec le pipeline et la validation croisée, et pour un paramètre alpha à 7, nous obtenons un score de 86% bien meilleur que celui de la régression logistique. On utilise la méthode de `r2_score` pour mesurer la qualité de notre régression, en effet la méthode `accuracy_score` n'était pas adaptée.
Plus on rajoute de données, plus notre score augmente, ce qui nous fait un score de 90% une fois toutes les données exploitées.

VI – Conclusion

Nous sommes plutôt satisfaits des scores que nous avons obtenu avec notre régression et de la manière dont nous avons traité les différentes données qui étaient à notre dispositions dans cette base de données. Nous sommes conscients que l'exécution de notre régression entraîne un warning mais il est dû au fait que très peu de vidéos ont le même nombre de likes nous avons donc choisi de les ignorer, nous faisons de la régression donc essayons de prédire une valeur numérique, il est donc normal d'avoir des vidéos avec un nombre de likes uniques.
Si nous avions voulu allez plus loin, nous aurions pu ensuite faire de l'analyse de sentiments sur les descriptions des vidéos de la base ou sur les titres pour déterminer si certaines tournures de phrases ou manière de rédiger (capitalisation importante et beaucoup de points d'exclamation par exemple) ont une influence sur le nombre de likes. Nous aurions également pu construire des réseaux de neurones analysant les images des vignettes de vidéo pour déterminer si certains éléments graphiques de ses vignettes ont tendance à plus attirer les likes pour les vidéos qui les emploient.