Sapienza Università di Roma

Statistical learning

Predicting youtube videos' popularity - G11

Students

Riccardo Agabiti
2027220
Andrei Caraman
1664744
Filippo Mastrogiacomo
1824217
Roberta Padula
1808389
Simone Piperno
1792917

Professor

Prof. Pierpaolo Brutti

Accademic Year 2021/2022

1 Abstract

Given the popularity of the youtube content creator nowadays we decided to look into this subject for our project. The scope of the analysis carried is to predict video popularity and as a second objective find features that might help creators to produce better products (and as a consequence earn more). This report is so divided in multiple sections. In section 2 we describe the framework used, in sections 3 and 4 we cover how we got and prepared the dataset for the analysis, then in section 5 we show an exploratory data analysis, in section 6 we show metrics and results per each model we used and in section 7 we run methods to better interpret the results obtained, lastly in chapter 8 we give conclusions.

2 Main research aim and framework

We decided to cast the problem of predicting popularity to a multiclass one in order to deal with it. At first we extract data and create some more variables from scratch to better capture some features of the data, then we run different algorithms to find the best one for our problem and at last we try and explain why we got that results.

3 Data Extraction

After having chosen the scope of the analysis we decided to look for available repositories of youtube video data. The main one we found is the "Youtube-8M", first presented in Abu-El-Haija et al. [2016]. It's a repository containing data from over 8 milion videos from an old Google classification task, from it we basically sampled random video URLs on where to work. However these links were "crypted" and we used a tool given by the Google researchers to extract the real URL. We proceded mining data using 2 different approaches, the main one was simply using beautiful soup to extract such information from the html page; then we used the pytube library to scrape the highest quality from the available video downloads. This process was made for ~ 13000 different videos which we analyzed for the project. At the end the extracted features were:

- Title
- Duration
- Views
- Comments
- Likes
- Genre
- Subscribers
- Publication date
- Max quality

4 Data Wrangling

4.1 Handling titles

In order to plug the title's information in the different models we trained (through sklearn), we had to transform the text information into a scalar. The transformation we did is based on the intuition that given the fact that some words have multiple meanings the titles using those kind of words should be more likely to be found. We proceeded in the following way:

- The first step was to first lower all the characters of the string (so that we have no capital letters), remove the stopwords and then stem all the words of each title;
- The second step was to create a dictionary having as keys the single words and as values the frequency with which the word appeared;
- The third step was to compute the probability of the title as the sum of the probabilities of the single words we computed at the previous step;
- The fourth and final step was to normalize the such achieved feature of titles' probabilities using a min-max normalization.

4.2 Duration

For the duration feature we opted for taking only the information relative to the minutes, leaving out the information about seconds.

4.3 Genre

Having to deal once again with text data here the solution is way more immediate. The genres were indeed simply encoded into numeric labels (0,1,2....).

4.4 Popularity index

We decided to create a new index to evaluate the popularity of a video in relation to the amount of time passed since its publication. So at first we add a new feature based on *Publication Date*: *timedelta* which works like this:

$$timedelta = \frac{Publication\_date - min(Publication\_date)}{date\_of\_extraction - min(Publication\_date)}$$

It's a min-max normalization with numerator and denominator expressed as differences of seconds. Then we define the new score as

$$score = timedelta(\ln(Likes + 1) + \ln(Views + 1))$$

We placed "+1" inside the formula since the NA values were replaced with 0s for both *Likes* and *Views*. After this step the score was normalized between 0-1 using the min-max normalization.

This new score was then used to divide the data in classes through the following function:

$$class(i) = \begin{cases} 0 \text{ if score(i)} \in [0, 0.2] \\ 1 \text{ if score(i)} \in (0.2, 0.4] \\ 2 \text{ if score(i)} \in (0.4, 0.6] \\ 3 \text{ if score(i)} \in (0.6, 1] \end{cases}$$

where i refers to the i-th observation.

After labeling the dataset the ratio of each class is the following:

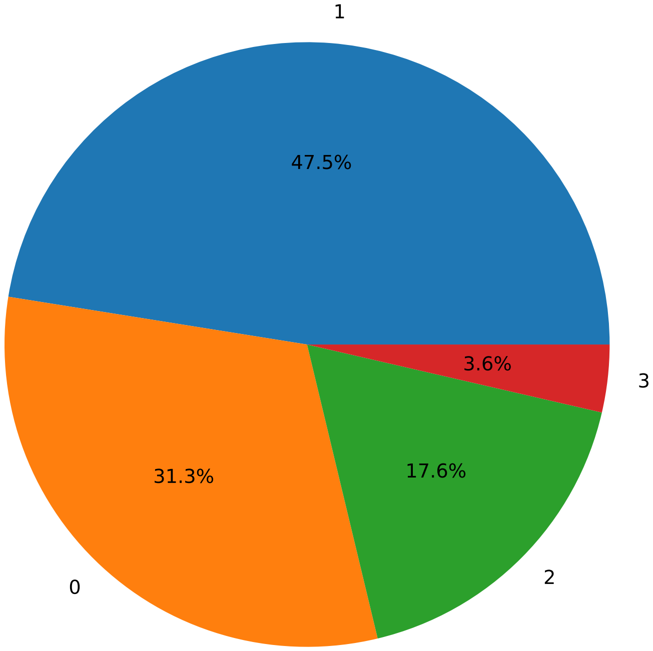


Figure 1: Distribution of the data.

4.5 Feature engineering

In order to achieve good results with our models we made use of a new feature and its inverse: *log(views)/log(commen* and *log(comments)/log(views)*, which are defined as follows

$$log(views)/log(comments) = \frac{ln(2 + views)}{ln(2 + comments)}$$

$$log(comments)/log(views) = \frac{ln(2 + comments)}{ln(2 + views)}$$

5 Data Exploration

We observed that most important features of the data overlapped and a separation is possible only using a scoring function. The number of likes achieved the best separation, maybe that is why youtubers insist so much on asking to "click the like button!". Data are represented on a semi-logarithmic scale (a data unit was added to each feature for this purpose) because of the strong variability between the successful youtubers and the averege Joe – those users constitued the bulk of the data while of course, the successful youtubers are the outliers. No significant relationship was observed between the data features and we have not deemed them of a graph.

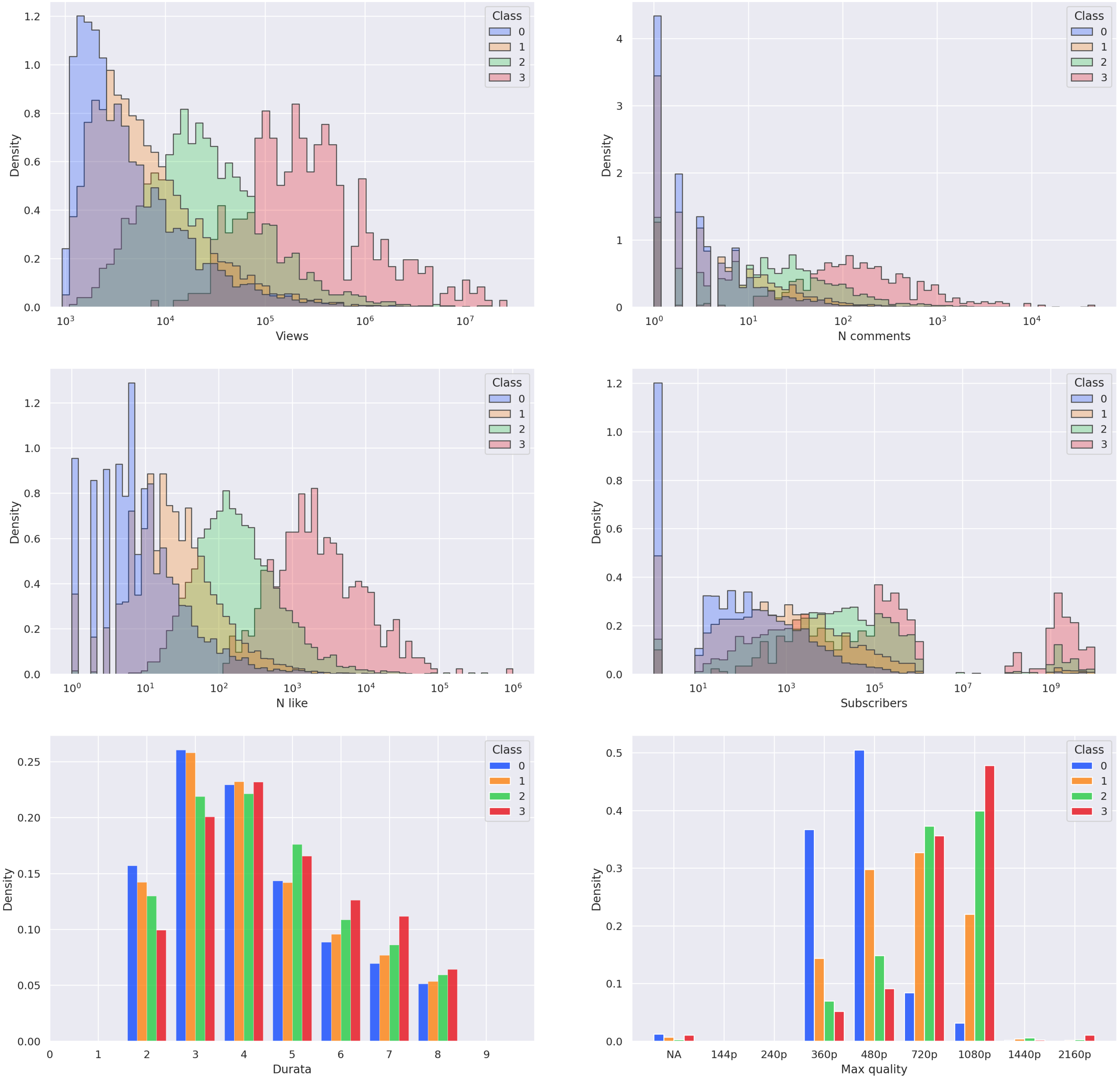


Figure 2: Histograms of the significant data features.

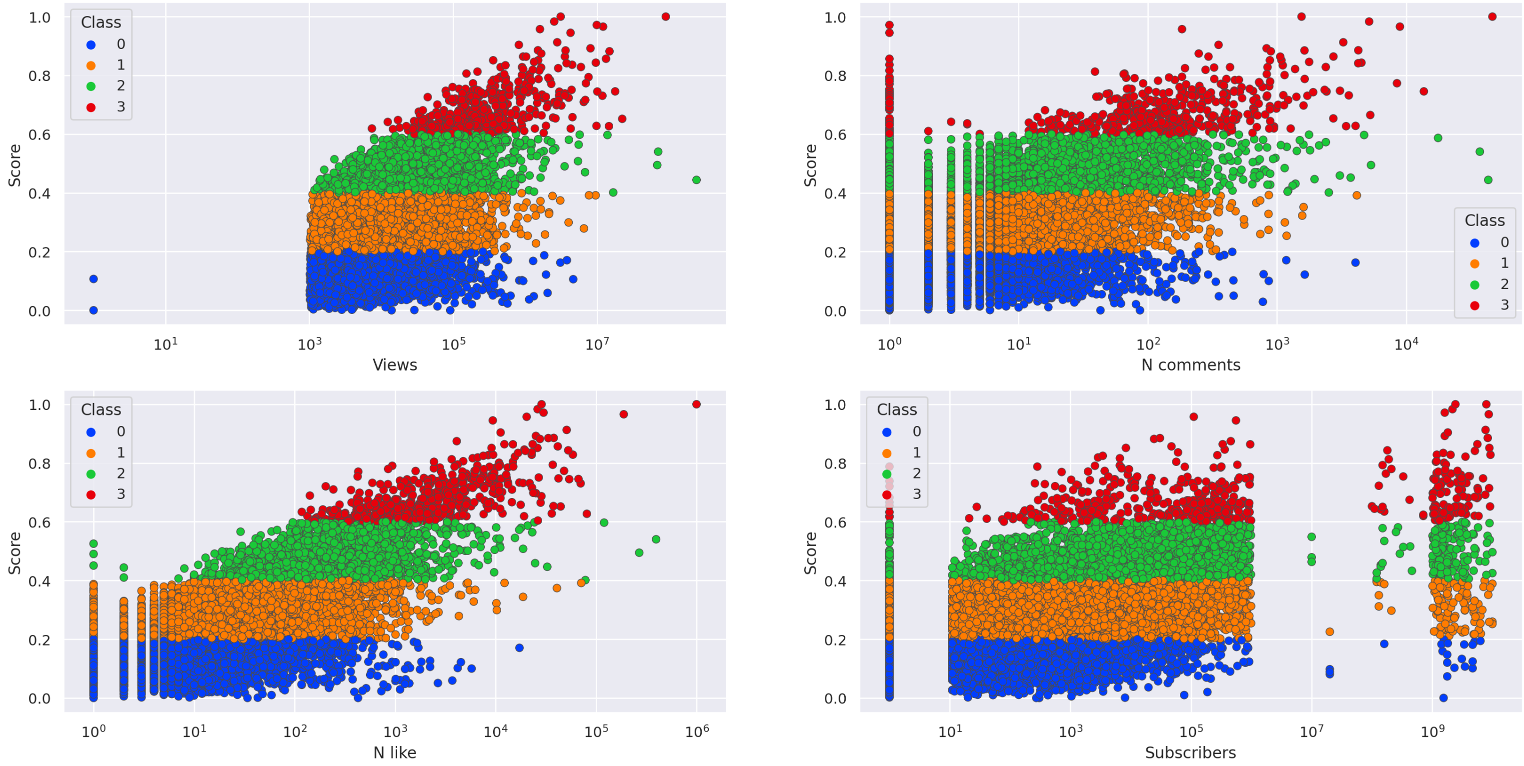
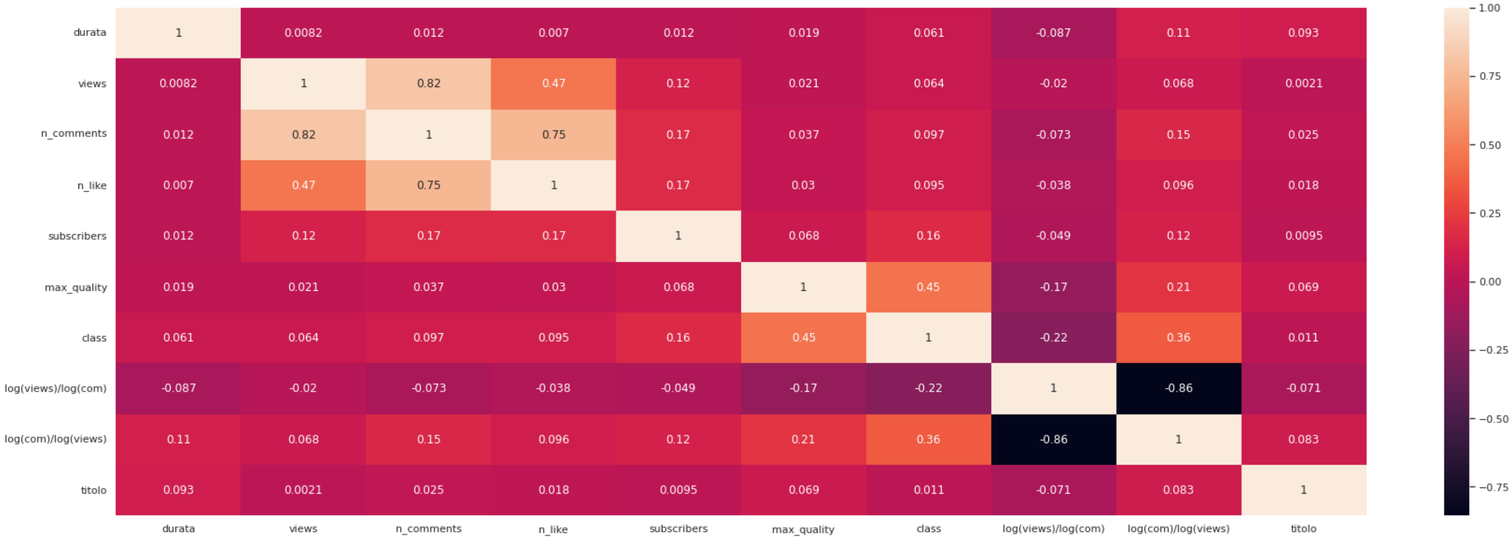


Figure 3: Scatterplots of the significant data features vs the scoring function.



	durata	views	n_comments	n_like	subscribers	max_quality	class	log(views)/log(com)	log(com)/log(views)	titolo
durata	1	0.0082	0.012	0.007	0.012	0.019	0.061	-0.087	0.11	0.093
views	0.0082	1	0.82	0.47	0.12	0.021	0.064	-0.02	0.068	0.0021
n_comments	0.012	0.82	1	0.75	0.17	0.037	0.097	-0.073	0.15	0.025
n_like	0.007	0.47	0.75	1	0.17	0.03	0.095	-0.038	0.096	0.018
subscribers	0.012	0.12	0.17	0.17	1	0.068	0.16	-0.049	0.12	0.0095
max_quality	0.019	0.021	0.037	0.03	0.068	1	0.45	-0.17	0.21	0.069
class	0.061	0.064	0.097	0.095	0.16	0.45	1	-0.22	0.36	0.011
log(views)/log(com)	-0.087	-0.02	-0.073	-0.038	-0.049	-0.17	-0.22	1	-0.86	-0.071
log(com)/log(views)	0.11	0.068	0.15	0.096	0.12	0.21	0.36	-0.86	1	0.083
titolo	0.093	0.0021	0.025	0.018	0.0095	0.069	0.011	-0.071	0.083	1

Figure 4: Correlation matrix of the significant data features vs the scoring function.

6 Models used and summary of their performances

Our main focus here is on the multiclass classification of the popularity of the video. The final features taken into consideration are: *Title, Duration, Views, Comments, Likes, Genre, Subscribers, Max quality, log(views)/log(comments) and log(comments)/log(views).*

For this task, the support vector classifier (SVC) was applied as a starting point. Then, we tried to model the problem using Random Forest, Logistic Regression, K-nearest neighbors, a Decision Tree classifier, XGBoost, One vs Rest Classifier (ORC) and One vs One Classifier (OOC) based on XGBoost then Gradient Boosting and ORC, OOC based on Gradient Boosting were analyzed.

In order to compare the performance of different models, since the dataset is very imbalanced for the classes, the F_1 *score* score was used as a comparison tool and its defined as follows:

$$F_1 = 2\,\frac{precision * recall}{precision + recall}$$

where

$$precision = \frac{TP}{TP + FP},$$

and

$$recall = \frac{TP}{TP + FN}.$$

	Accuracy	F_1
SVC	0.57	0.42
Random Forest	0.68	0.64
Logistic Regression	0.55	0.36
K-nearest neighbors	0.55	0.44
Decision Tree Classifier	0.59	0.54
XGBoost	0.68	0.64
ORC(XGB)	0.67	0.63
OOC(XGB)	0.67	0.62
Gradient Boosting	0.69	0.66
ORC(GB)	0.69	0.65
OOC(GB)	0.70	0.67

Table 1: Summary table of the peformance of the models.

7 Methods and Interpretability

The final two models that were chosen to be further analyzed are Gradient Boosting ad Random Forest Classifier, given their very high results for both F_1 and accuracy. For the moment however we limited ourself to predicting the popularity class for these videos without really looking into the model features and how they interact to get the results. To do so, and gain a necessary insight for everyone using the youtube platform as a full time job, we need to look at some variable importance procedures for these 2 algorithms. For both methods we will use:

- the ”Feature permutation”, which calculates the mean decrease in accuracy for the model when we shuffle each variable in the tree structure, a higher decrease means that the variable is more important for the good functioning of the model.
- the ”SHAP”(Shapeley addictive explanation method), which is supported only for the Random Forest. It was first introduced in Lundberg and Lee [2017] and allows to understand which features the model

considers as 'most important'. It's done through Game Theory's Shapley values[1].
Now consider an input $\bar{x}$ and a simplified local input $\bar{x'}$ described respectively from the models $f(\cdot)$ and $g(\bar{x'}) = \phi_0 + \sum_{i=1}^{N} \phi_i x'_i$, where ϕ_0 is the null output of the model (average output) and ϕ_i represents the attribution (feature effect). $g(\cdot)$ is called 'Additive feature attribution' and we require few properties for it:

1. Local accuracy: If $\bar{x} \sim \bar{x'} \Rightarrow f(\bar{x}) \sim g(\bar{x'})$;
2. Missingness: If $x'_i = 0 \Rightarrow \phi_i = 0$;
3. Consistency: If the feature contribution changes, the feature effect cannot change in the opposite direction.

These properties are all statisfied if we take the ϕ_i's to be Shapley values.
Computationally speaking the SHAP introduces a problem, we can see how fast the things scale: for 4 features we'll have 64 possible coalitions, while with 32 features we get 17.1 billion possible coalitions, which makes everything computationally very heavy. That's why Lundberg and Lee introduced the Shapley kernel.
Now consider the feature vector $\bar{x}$ composed by binary values, where the 1 means we take the feature at that index and 0 we don't (e.g. $\bar{x} = [1, 1, 0, 1]^T$ means we take the first, the second and the fourth feature). Since in ML models we can't directly omit a feature we'll replace it with some values $b_i \in B$, where with B we represent the background set, which usually is a portion of the training set.
At this point we want to compute the mean value for each coalition (e.g. for $\bar{x} = [1, 1, b_i, 1]^T$ we want to compute $\mathbb{E}[y_{1,2,i,4} \ \forall i \in B]$, where y represents the output of the model). Once we have a number of samples computed in this way we can formulate this as a weighted linear regression with each feature assigned a coefficient c_i. If we assign weights according to the Shapley kernel:

$$w_{C_i} = \frac{\#\text{ features} - 1}{\#\text{coalitions of size } |C_i| \ \#\text{features included in } |C_i| \ \#\text{features excluded from } |C_i|}$$

we'll have that the coefficients c_i represent the Shapley values, hence $c_i = \phi_i$.

7.1 Random Forest

Applying these techniques we get the following results:

[1] The Shapley value of a member in a coalition represents the marginal contribution of that member in the coalition and is computed as

$$\phi_i = \frac{1}{\#\text{ members}} \sum_{\forall C \text{ s.t.} i \notin C} \frac{\text{Marginal contribution of i to C}}{\#\text{ coalitions of size C}}$$

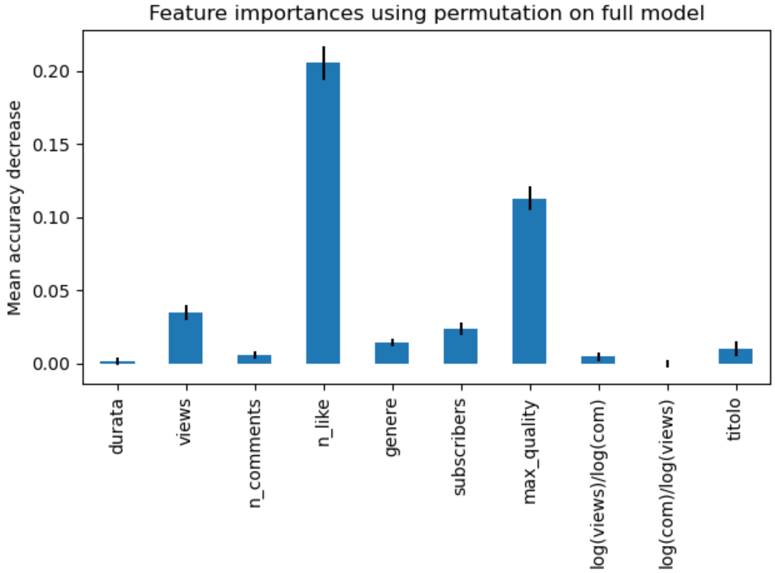


Figure 5: Feature permutation results for Random Forest.

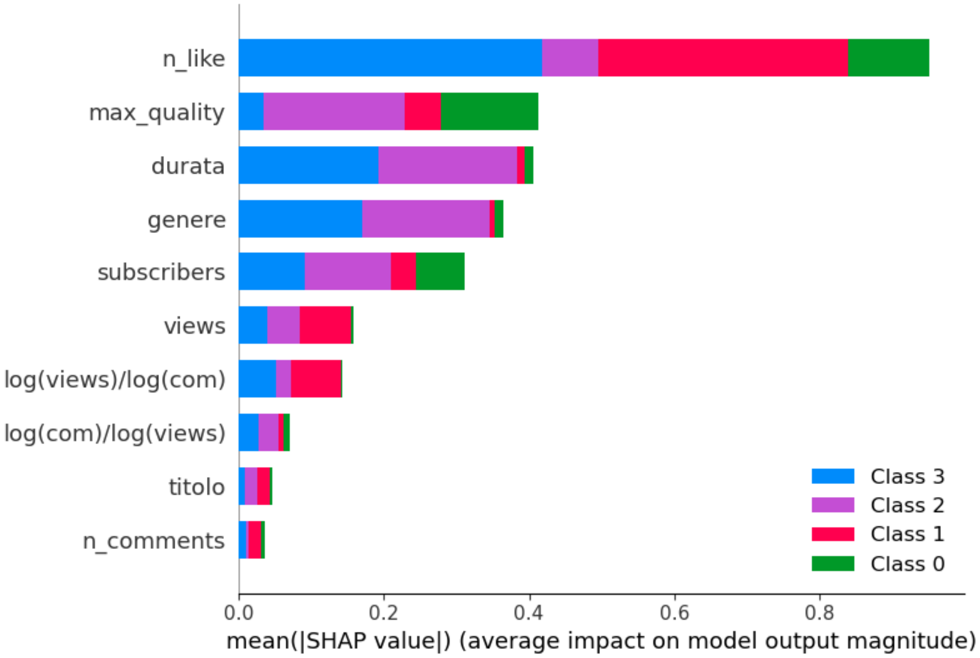


Figure 6: SHAP values for Random Forest.

The graphics above refer to the Random Forest model, the feature permutation assures that almost only *max quality* and *n_likes* have any reamarkable decrease in the accuracy if shuffled, meaning they are supposedly the most important. However the shap results show something very different: not only that *durata* and *genere* are useful to the model, but that for very popular videos are actually tiebreakers. Still also shap recovers that *n_likes* is the most important variable and *max quality* follows as the second most important, it's also important to note that the quality of a video is not so used to predict a video as very popular (this is probably due to the fact that most if not all popular videos have at least Full HD quality).

7.2 Gradient Boosting

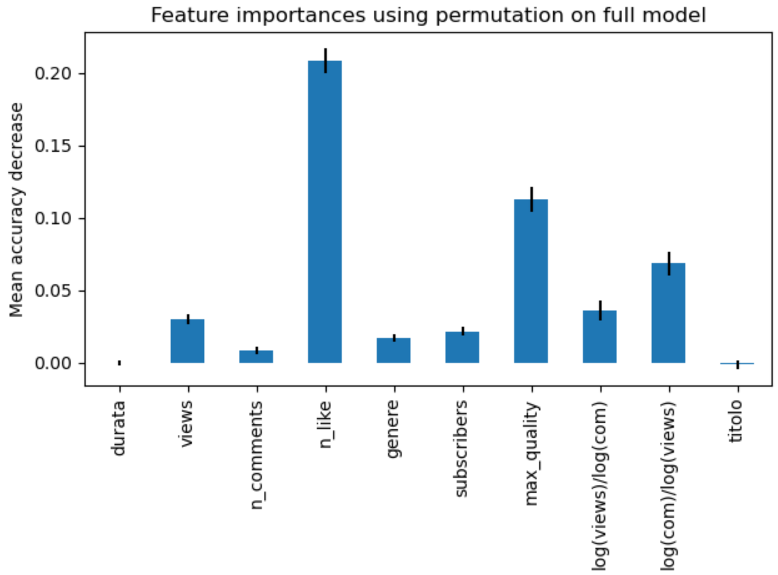


Figure 7: Feature permutation results for Gradient Boosting.

On the other hand the results for Gradient Boosting show that the engineered features defined beforehand are very important for GB, while they did not decrease accuracy as much in the Random Forest. Also here *n_likes* comes out on top as the most important feature, and by approximately the same amount as per the RF.

8 Conclusions

To conclude the project we wanted to first express the insight gained and then adress some issues. As far as insights go it's clear that a youtuber seeking popularity might wish to produce content that increase the number of interactions such as likes, since they allow the video to be brought up more often in correlated videos for other users. This in conjuntion with a trending genre such as gaming, sport or entertainment and a high quality camera to record should ensure a profitable career. With regards to issue the main one is related to performances, not being able to surpass the 0.7 accuracy threshold was somewhat disappointing, nontheless the interpretation methods revealed that many features extracted were not as much important as we thought they were. For example the number of comments rated very low on all 3 interpretation plots, meaning that the number isn't necessarily an index of a popular video. We also think that a bigger dataset might have had an effect to improve the metrices, however given the little time, the computational capacity, and the long procedure to prepare and process the dataset we stuck with 13000 videos.

For further development of the project it could be interesting to add as a feature also the thumbnail of the videos, and make a deeper analysis of the titles since some youtubers[2] claim they have a high impact on the final result of the video.

[2] https://www.youtube.com/watch?v=S2xHZPH5Sng

References

S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. https://arxiv.org/pdf/1609.08675.pdf, 2016.

Y. Li1, K. Eng1, and L. Zhang. Youtube videos prediction: Will this video be popular? https://www.semanticscholar.org/paper/YouTube-Videos-Prediction%3A-Will-this-video-be-Li-Eng/4e8ed9fca7320e70909c235bced8ecde614b7909, 2019.

S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. https://arxiv.org/pdf/1705.07874.pdf, 2017.

M. U. Nisa, D. Mahmood, G. Ahmed, S. Khan, M. A. Mohammed, and R. Damaševiˇcius. Optimizing prediction of youtube video popularity using xgboost. https://mdpi-res.com/d_attachment/electronics/electronics-10-02962/article_deploy/electronics-10-02962-v2.pdf?version=1638278882, 2021.

C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. https://www.nature.com/articles/s42256-019-0048-x, 2019.

G. G. E. Scott M. Lundberg and S.-I. Lee. Consistent individualized feature attribution for tree ensembles. https://arxiv.org/abs/1802.03888, 2018.