# Statistics behind Spotify

a nonparametric approach to music

## Dataset and goals

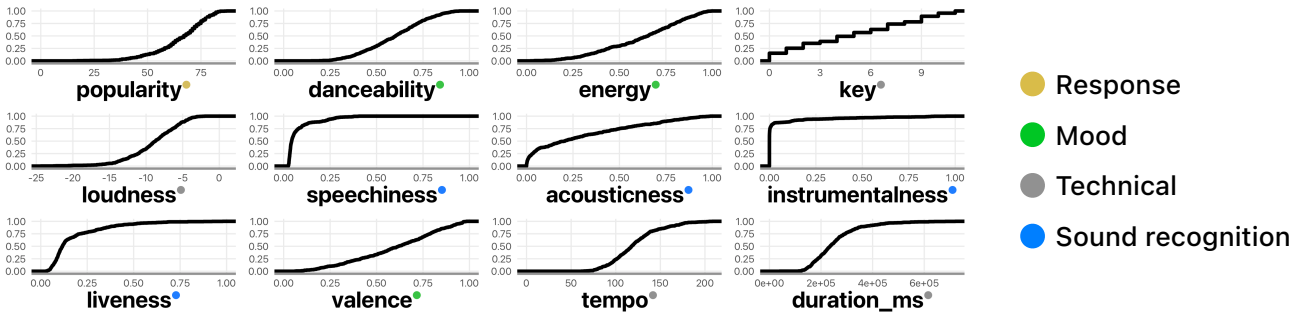
The initial dataset consisted of the top 500 tracks on Spotify according to a survey conducted in 2021 by the Rolling Stone magazine among the most important figures in the music industry. The aim of our project was to develop a model able to recognize what makes a song successful. Such a model would allow artists, labels, and Spotify itself to better adapt to consumer trends.

This approach is already well-developed in other media spaces. Netflix, for example, has an entire data science division working on statistical models that identify which TV shows and movies are worth investing in[1], allowing them to act both as a content host and as a content producer.

## Exploratory data analysis

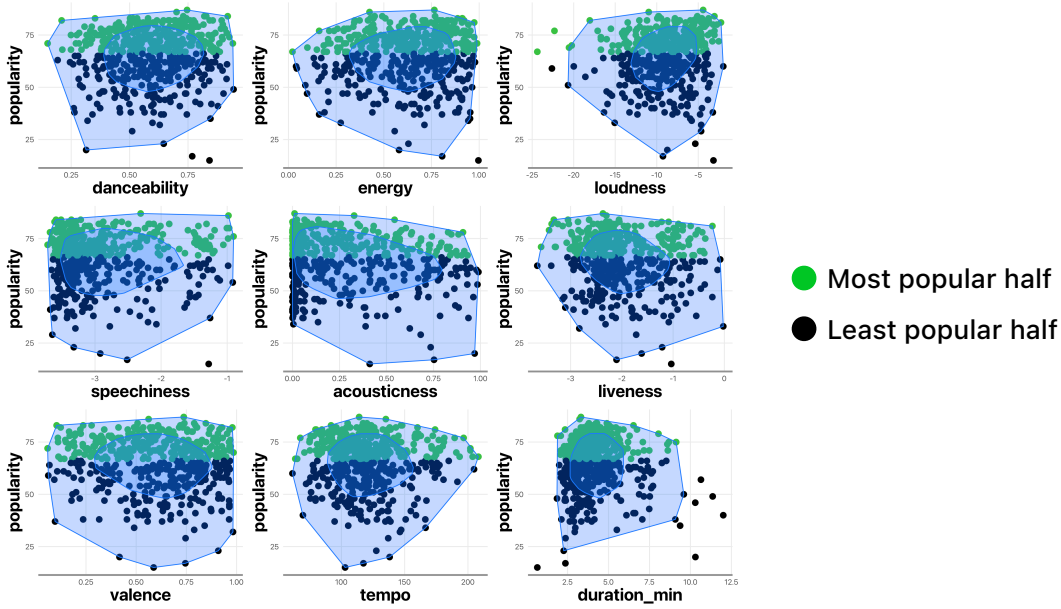
`s00_eda.R`

The dataset presents a mixture of technical features (i.e. with a well-defined measurement process), proprietary Spotify features that represent the mood of a song, and proprietary Spotify sound recognition features that represent the confidence of Spotify's models in identifying a certain sound.




Plotting the empirical cumulative distribution functions shows a clear distinction between two classes of proprietary features. It can be observed that features representing the confidence of Spotify sound recognition models have markedly positive skewness, reaching a maximum for instrumentalness (skewness = 4.16), while features representing the perceived mood and tonality of the song seem to be more symmetric.

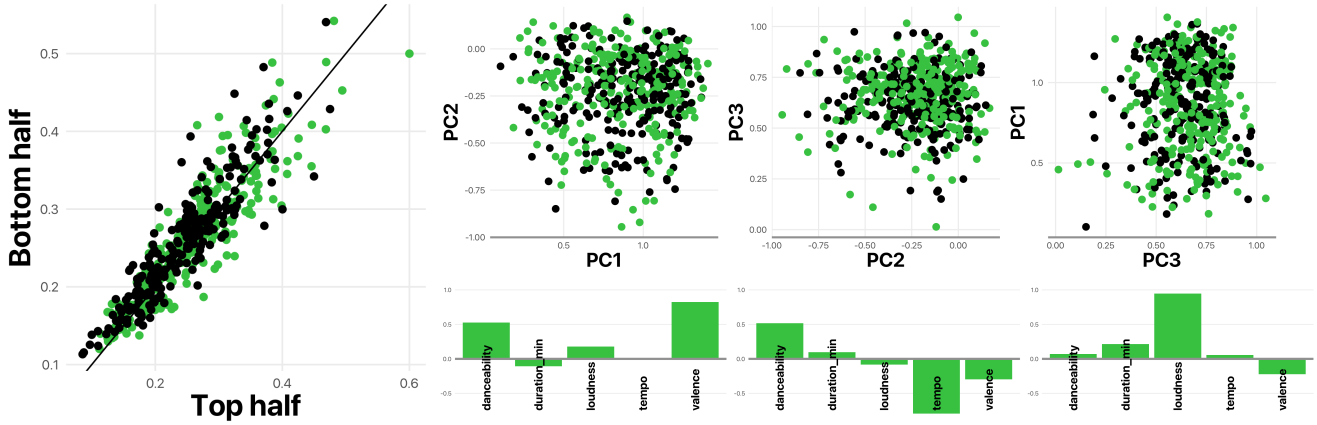
The scatterplots of continuous features against popularity show no discernible structure, except for a slight penalization in popularity to excessively long or excessively short songs.



# Principal components analysis

s01_pca.R

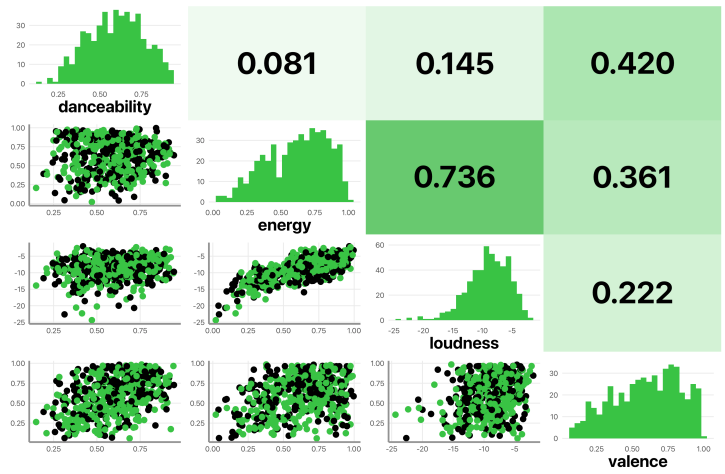
Before trying to predict popularity, we tried to solve an easier task by finding a way to separate the 50% most popular songs and the 50% least popular songs. An initial DD-plot of the distributions of continuous features doesn't highlight any distributional difference between the two classes of songs.

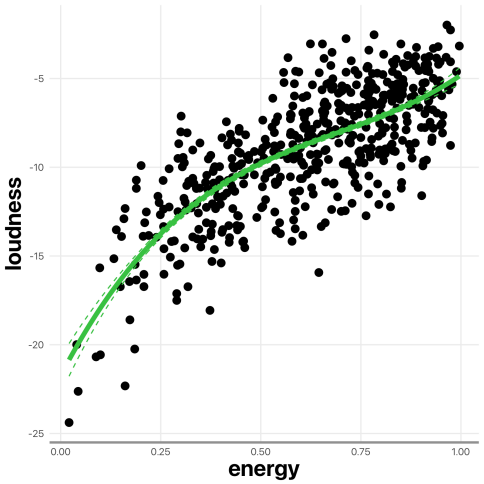



Perhaps unsurprisingly, we obtain similarly underwhelming results when we examine the two groups along the top three principal components.

We also observe that some components are particularly correlated, as shown in the pairplot below.

Loudness, in particular, has a very tight link with energy. It was an expected outcome, as louder songs are perceived as more "energetic".



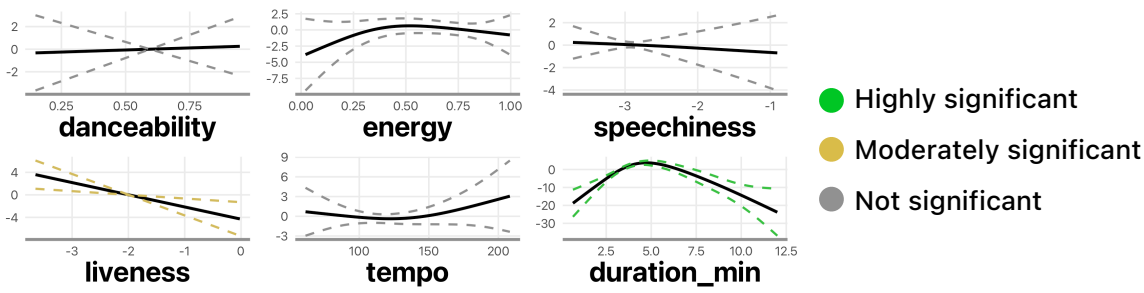




# Initial model

s02_gam.R

Looking at the bag plots previously generated, it's difficult to see any kind of relationship between sound features and popularity. This is confirmed by the poor performance of our initial model.

From the EDA, the only apparently relevant feature (duration) seems to have a non-linear link. GAMs can correctly model this dependence but, as shown in the plots below, struggle to identify any dependence of popularity from other variables.



Permutational tests (where partial tests were performed following Freedman & Lane's scheme[2]) identified liveness and duration as significant features, results in agreement with `mgcv`'s builtin variable selection method[3].

# Poor performance and new data

The predictive capability of our features is extremely low, as suspected from the initial analysis showing no distinction in distribution between the most and least popular songs. Thankfully, Spotify provides additional artist-specific information through their API.

We wanted to keep open the possibility of performing some kind of clustering among the artists, which would have possibly been slightly problematic on the original dataset (composed of 500 songs in total) so we chose to change dataset and try random sampling ~5000 songs from the Spotify API.

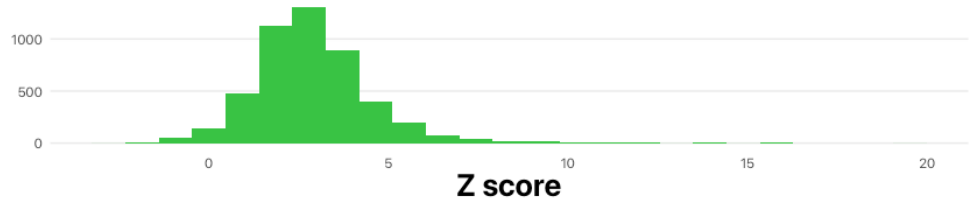
In order to download a random sample from a deterministic API, we implemented the following process:

1. Pick a random letter uniformly from the set of uppercase and lowercase letters, {A,...,Z,a,...,z}
2. Send the API a search query containing the random letter, Spotify then sends back a list of 1000 songs chosen by their search algorithm.
3. Pick a random result from the 1000 songs.

There are two main limitations to this method. The first limitation is that the search results are highly personalized. We did create a completely new Spotify account with no prior activity in order to "depersonalize" the results as much as possible. Unfortunately, some biases (e.g. geographical bias from the IP address) can not be avoided and, as a result, our dataset has a noticeable Italian bias.

The second limitation is more troublesome, and that is the 1000-results limit on Spotify API. Because their search algorithm gives an advantage to popular songs, and because it only shows the top 1000 results, the dataset will necessarily be biased in favor of more popular songs.

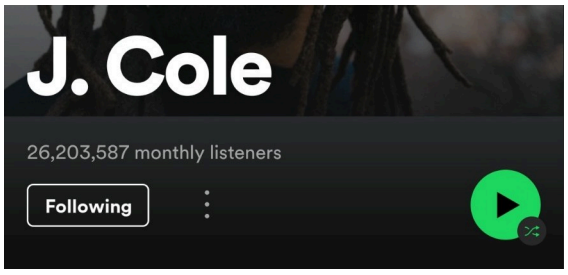
In fact, if we normalize song popularities artist-wise and obtain a Z-score, we notice that the songs in our dataset are on average almost 3 standard deviations more popular than other songs from the same artist.



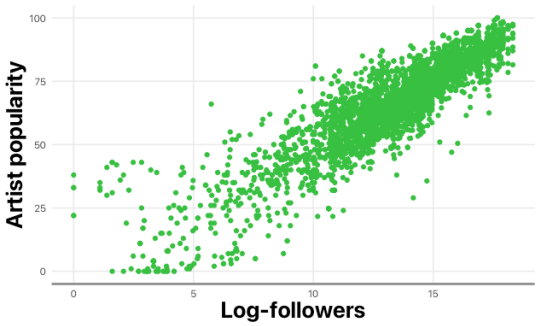

Moving back to the additional features, we collected:

- **Base popularity of the artist:** score in [0,100] representing a weighted mean of listeners.
- **Followers:** amount of people who "follow" the artist on Spotify.

Followers and listeners are different but highly correlated. Users who happen to listen to a song from an artist are automatically counted as listeners (and determine the popularity score), while followers are defined as users who consciously choose to press the "Follow" button on an artist's profile.




*"Following" allows Spotify users to receive notifications for new songs or albums.*

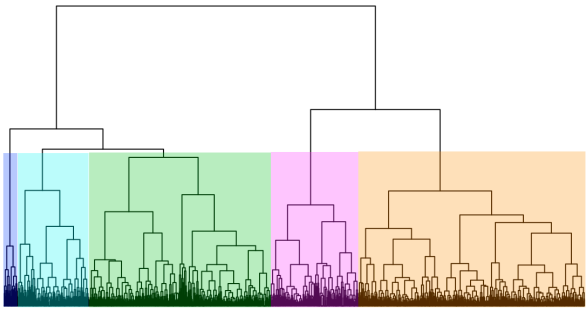


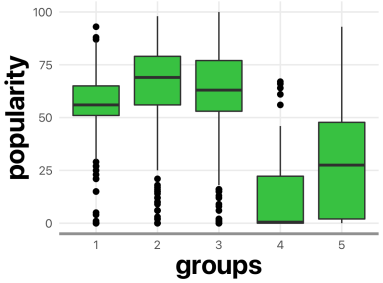
# Artist clustering by features

s03_cls.R

Our initial approach was to extract as much information as possible about the artist from the quantitative features of the songs. To do this, we computed for each artist in our training set the average quantitative features of their songs, using only the ones that are contained in our dataset (indeed, our data contains different songs for each artist). We used a hierarchical clustering algorithm to cluster the artists based on their average features. Songs will then be assigned to a specific cluster through a kNN model.

For the hierarchical clustering, we selected the cutoff at five groups despite the dendrogram suggests against this choice.



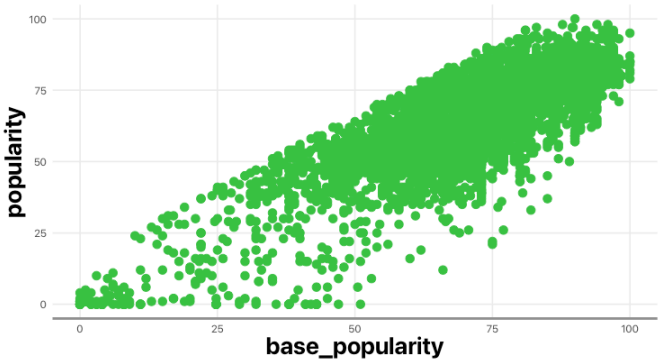



This is because the dendrogram does not have information about artist popularity (not included because it's our intended response variable) but permutational ANOVA tests confirmed that the fifth group has a significantly different mean for song popularity.

# Artist-based models

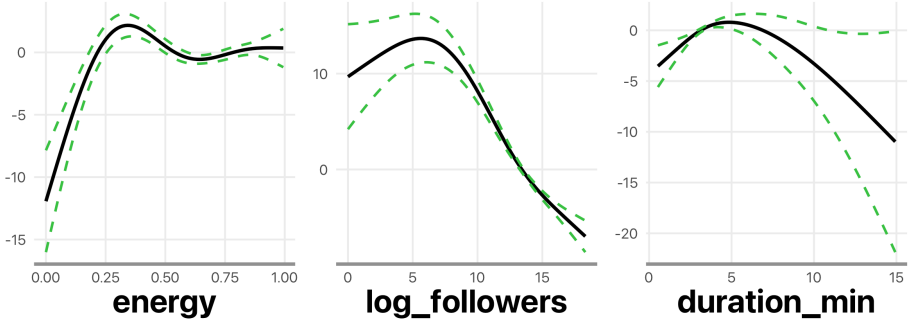
s04_gam.R

Including this new information, we now develop a model which considers the belonging of the artist to a specific group, but also the duration of a song, the number of artists featured in the song, the number of their followers and their popularity. The performance of the following models will be extremely dependent on the presence of the "artist popularity" feature. In fact, it disappointingly turns out that the single best predictor of the success of a song is the popularity of the artist.




Since the relation between the other features and popularity is nonlinear, we opted for a GAM model.

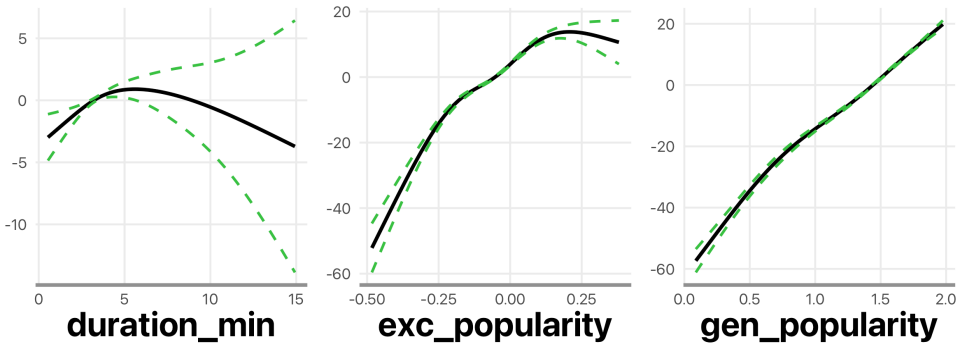
**popularity** ~ **artist popularity** + s(**energy**) + s(**log followers**) + s(**duration**) + **groups** + **featuring**



It's reassuring to see that, at least for the duration (our most promising feature), the resulting spline is virtually identical. That's despite having changed dataset (from the Rolling Stone's top 500 songs to a random sample) and having included new features.

We performed some light variable transformation on artist popularity and log-followers. As shown in a previous section, these two variables are highly correlated, so we choose to decouple them by considering their mean as one feature, and their difference as another feature.

The first transformed feature (the mean) can be interpreted as a more general measure of artist popularity, while the second transformed feature (the difference) can be interpreted as giving some information on the fan base of the artist (e.g. if an artist has more followers than we would expect from their popularity, it might mean that they're predominantly listened to by their most loyal fans and lack widespread appeal). Using these new features we obtain a similar model.




As before, we run permutational partial test for covariate significance. The p-values point towards the relationship being highly significant. To test the accuracy of the model, we not only looked at the $R^2$ (roughly equal to 70%), but we also tested this model on 500 randomly sampled songs. To quantify the performance, we used a mean absolute error loss function: in both the train and test set, the MAE is between 7 and 8 points, meaning that on average our model misses the true popularity by roughly 7 points in absolute value. Considering that the popularity feature is an integer ranging from 0 to 100, we are reasonably satisfied with this result.

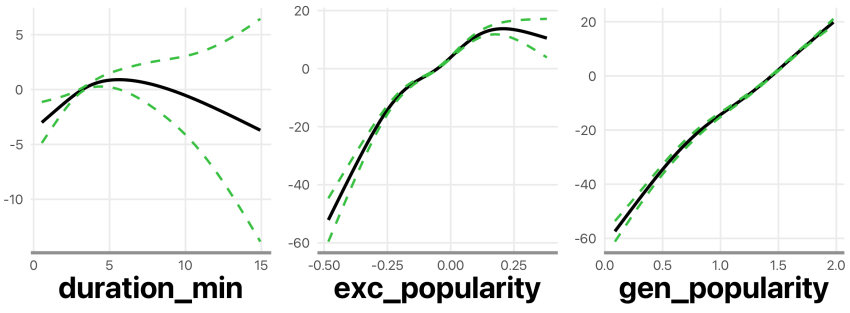
We call this approach **artist-based**, because the results depend heavily on the artist base popularity and the artist's cluster. This approach can be useful from the point of view of Spotify (they can simply focus their investment on specific artists) but presents some problems from the point of view of the single artist looking to improve the performance of their songs.

# Random effects

`s05_gamm.R`

A similar but different approach to correct for the effect of the artist uses random-effects GAMs. Conveniently, the `mgcv` library can easily include random effects using the same syntax used for smooth terms.

The usual approach would have us consider a random effect for each artist, but it can't really be applied to this dataset, as many artists have only one songs in the dataset. To solve this problem we chose to consider a random effect for each group of artists (as identified in the previous section on clustering).



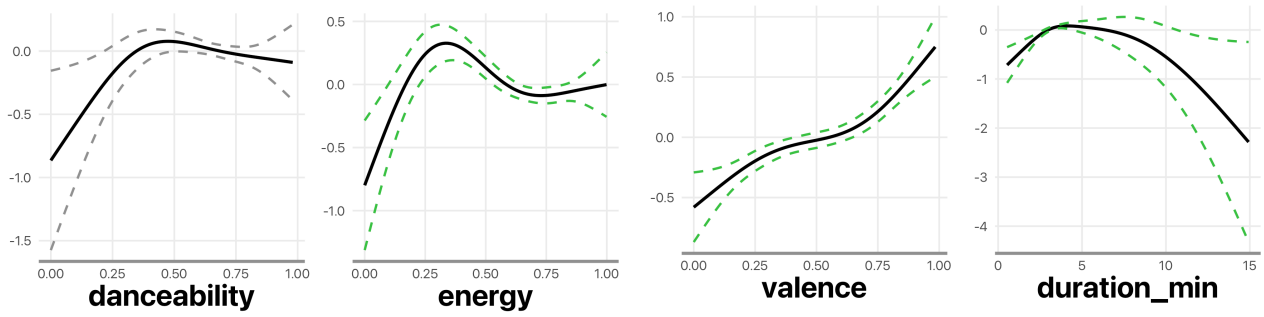

The results are (expectedly) identical to the previous results.

# Difference in differences

s06_did.R

In order to try and answer the research question from the point of view of a single artist trying to improve their success on Spotify, we try to look at a "normalized popularity" of a song.

What we are doing is basically a difference in differences approach, for each song we send an API request to get other songs from that same artist. This way we can look at the popularity of the song in our dataset not in absolute terms, but in relation to the (almost) entire discography of the artist. As already reported in a previous section, the peculiarities of Spotify's API mean that our dataset is very skewed towards more popular songs.




Despite an abysmal $R^2$ of 4%, we can see some significant effect of duration (as usual) and, somewhat surprisingly, valence. Where "valence", in Spotify's API, is a measure of "happiness" conveyed by a song.

While it may be intuitively obvious that high-valence songs are more popular, and low-valence (i.e. "sad") songs have a more niche audience, this relationship could not be identified before the differencing step.

# Closing remarks

It is very difficult to predict the popularity of a song from a handful of quantitative features. From a certain point of view, it wasn't entirely unexpected, as we were aware of the absence of some very important features (e.g. song lyrics).

Unfortunately, the results of our analysis brought little hope for the possibility of predicting song popularities from quantitative features. Our main models relied extensively on features not related to the song itself, but to the artist and their starting popularity.

Looking at the normalized popularity yielded some interesting, albeit maybe not surprising, insight in the relationship between valence and popularity. However, the very poor amount of explained variance leads us to believe that we are either missing some very important features, or popularity has a high degree of randomness.

# References

[1] : *Netflix Technology Blog*, Supporting content decision makers with machine learning, **url:** https://netflixtechblog.com/supporting-content-decision-makers-with-machine-learning-995b7b76006f

[2] : *Freedman D., Lane D.*, A Nonstochastic Interpretation of Reported Significance Levels (1983), Journal of Business & Economic Statistics, 1:4, 292-298, **DOI:** 10.1080/07350015.1983.10509354

[3] : *Marra G., Wood S.*, Practical variable selection for generalized additive models (2011), Computational Statistics & Data Analysis, 55(7):2372-2387, **DOI:** 10.1016/j.csda.2011.02.004