The Algorithm (at a glance)
We use *Alex's Quick and Dirty Classifier*, a homemade machine learning algorithm that computes the probability that a new data point is a member of a given set of previously seen *positive* data points; this is computed against a *negative* set for which no data points are available, which we define in terms of some kind of assumed/implicit prior distribution. Our algorithm is similar to a Gaussian Mixture Model, but heavily modified and repurposed for a classification task. We hope to use this algorithm to learn supervised classification given only unsupervised data (not an easy task!).

Predicting Hit TikTok Songs
In the case of predicting TikTok songs, we want to perform the supervised task of classifying songs as "hit" and "not a hit." The crux lies in the fact that we have a small set of data points for "hit" songs, and we have no data points at all for "not a hit" songs, and there are a lot of them, so it would be unreasonable for us to try to collect a representative sample of the set of all non-hit songs. That would amount to collecting enough data points to represent almost all the world's music!

In the context of our prediction algorithm, we see that our small set of data points for hit songs is our *positive* set, and the very large set of non-hit songs for which we have no data is the *negative* set. To address this lack of data, our algorithm assumes that any given song is, by default, not going to go viral. The set of songs that go viral is an extremely small subset of the songs on the internet, so it is safe to assume that the vast majority of songs will not go viral. Later, we can tweak certain parameters of our model to adjust the "strength" of this assumption (e.g. do we assume that the average song has a 1% chance of going viral? 10% chance? 50% chance?).

In order to predict if a song will be a hit, we see if a song has similar qualities to other songs that we know are hits. In order to do this, we have to define some sort of similarity metric between songs. To this end, we use 6 song features computed by the Spotify API: Acousticness, Danceability, Energy, Liveness, Loudness, and Tempo.

	Acousticness	Danceability	Energy	Liveness	Loudness	Tempo
Acousticness	1.000000	0.515098	0.415238	0.562000	-0.634072	0.717691
Danceability	0.515098	1.000000	0.893100	0.741420	-0.902684	0.945692
Energy	0.415238	0.893100	1.000000	0.860669	-0.715232	0.838972
Liveness	0.562000	0.741420	0.860669	1.000000	-0.691464	0.783959
Loudness	-0.634072	-0.902684	-0.715232	-0.691464	1.000000	-0.920295
Tempo	0.717691	0.945692	0.838972	0.783959	-0.920295	1.000000

Figure 1: Correlation Matrix for our Predictive Features

The Algorithm (implementation)
We center 30 Gaussian probability mass functions with height 1.0 on the data points corresponding to each of our 30 TikTok hit songs. To predict the probability of a new song being a hit, we take its n-dimensional Euclidean distance (n=6 because we have a 6-dimensional feature space) from each of the 30 points and plug those distances as plus-or-minus x into their corresponding Gaussian mass functions. We simply take the max() of these 30 probability mass function outputs to get our predicted hit probability. Essentially, this method guesses the probability that a song goes viral by checking how close it is to known hit songs in our dataset.

Preliminary Findings
Figure 2 provides a visualization of the single-feature predictor functions for each of our 6 features. These can sort of be seen as average “cross sections" of our overall model, which would normally be a 6-dimensional mass function whose dimensions represent acousticness, danceability, energy, liveness, loudness, and tempo.

What does this mean? These plots show the concentration of viral TikTok songs at different values of Acousticness, Danceability, Energy, Liveness, Loudness, and Tempo. Looking over these plots, we can make inferences about what features make a hit song! For example, many hit songs have Acousticness values between 0.0 and 0.15. We also see that Energy values 0.3 and 0.6 are good, and a Danceability value of 0.2 is definitely *not* good. Loudness values are relatively evenly distributed for hit songs, but lower values from -7 to -2 are generally a safe bet for a hit song. Most importantly, a hit song should be high in Liveness (almost all of them are skewed toward high Liveness values) and its tempo shouldn’t be in the range between 0 and 80. Interestingly, it seems that TikTok viewers generally just don’t like slow songs.

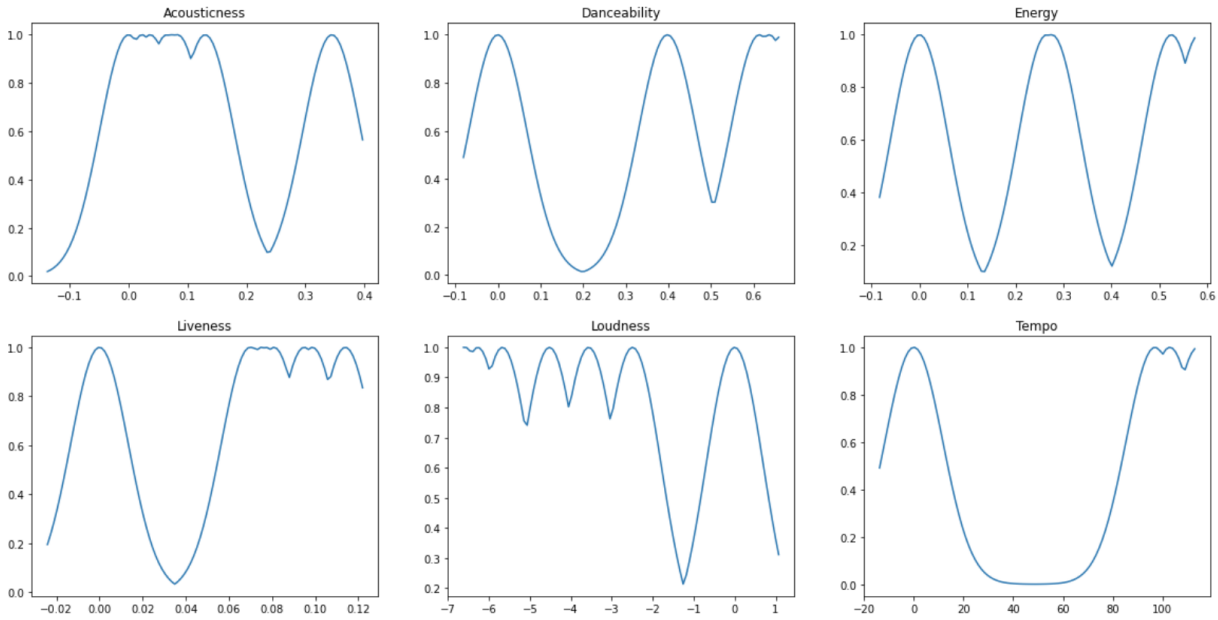

Figure 2: Single-Feature Predictor Functions for Acousticness, Danceability, Energy, Liveness, Loudness, and Tempo