Explorations in ML Part II: Predicting Popularity

Github Repo: here

Data Source: here

What this is >>> Part II in a series of using fundamental ML techniques to understand data and make predictions. Unlike Part I, the data used here is of personal interest. As a lover of music, electronic music in particular, and as a wanna-be DJ and producer, I have been eager to practice ML techniques to understand better the music we stream. More practically, the dataset I stumbled across gave me the chance to build regressor estimators (instead of classifiers) and to practice working with a hybrid dataset of dense numerical and sparse text features.

Simply, the goal of this exercise was to (1) Learn more about a domain of great personal interest while at the same time (2) give me the chance to practice the prediction of a continuous target with hybrid features sufficiently different from the previous OKCupid exercise.

TL: DR >>

1. **What:** Gleaning insights on popularity and building a model that can predict song popularity
2. **Why:** To practice ML in a context relevant to my interests and showcase what I can do
3. **How:** We ended up using a ridge regression model that performs reasonably well (R^2 = 0.65).

Data Processing:

The source data came from 6 files with some interesting characteristics, as detailed below. While the Kaggle user uploaded files from both Beatport and Spotify, I chose to focus on the Spotify files for two reasons. First, Beatport is a niche streaming platform for electronic dance music, and I wanted to explore as diverse a set of genres as possible (the Spotify set did not entirely fulfill this wish, but more on that later). Second, the Spotify files were sufficiently large for my purposes. In a 'real world' scenario, there may have been some additional signal capture when combining the two sets with tools equipped to handle massive datasets, but the core techniques likely would have been the same.

The dataset features some typical fodder for ML students, but has some critical meta features worth calling out when interpreting the learnings:

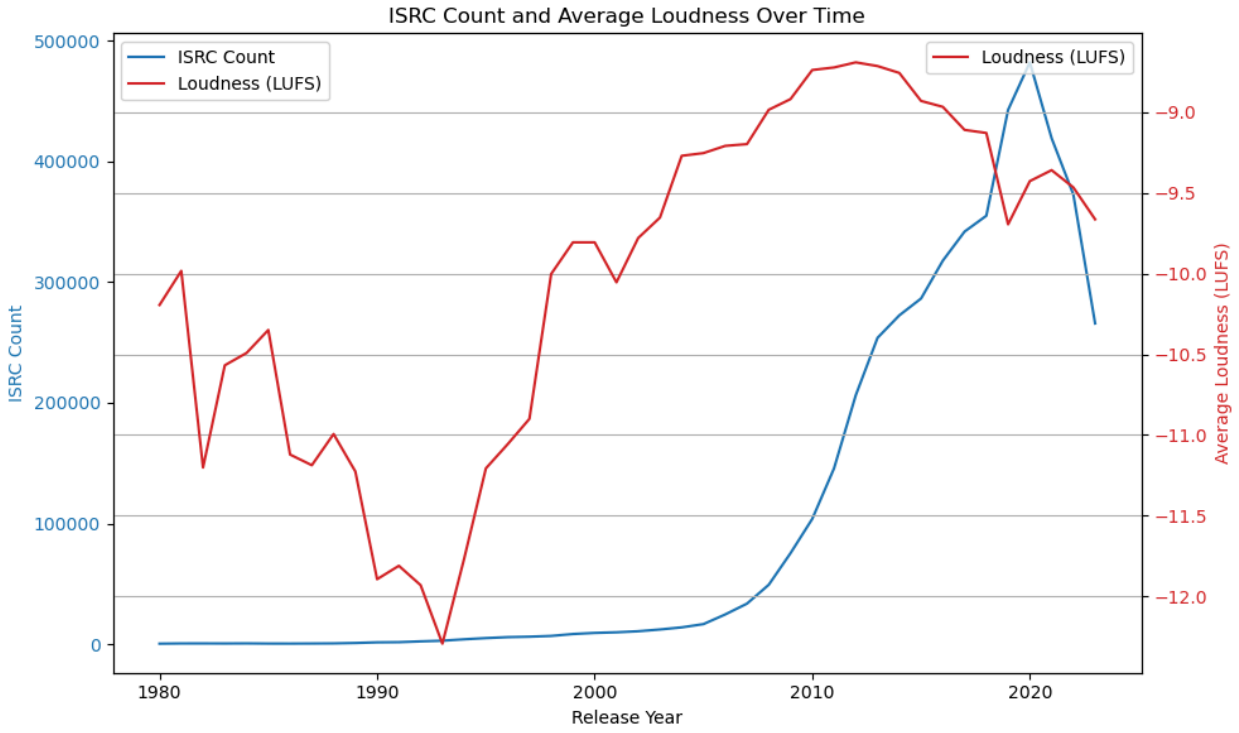
- **Training+Test data is aggregated to the ISRC level**: ISRCs represent unique recordings. Recordings can be on multiple *releases,* and releases can have multiple *tracks*. Therefore, ISRCs can be associated with various release_ids and track_ids
- **Audio Features live with the ISRC:** The juiciness of this data lies in its dense numerical features that attempt to objectively describe the recording, i.e., danceability, energy, tempo, etc. Intuitively, these features are unique to an ISRC as they would not differ across releases.
- **Text features live on the track and release:** Track name, artist(s) name(s), release name(s) (i.e., name of the album), and label name are, of course, unique to what is distributed (vs. recorded). While track names rarely differ, song titles usually remain the same–release can vary as an artist may release a track both as a single and as part of an album. To get these features at the ISRC

level, I aggregated all titles for an ISRC, along with the title or name associated with the most popular release.

- **Popularity (the chosen Target) lives on the release:** The more I thought about this, the more it made sense that popularity, regardless of how it is measured, would pertain to a particular release, such as a published single or album. Aggregating popularity across releases is possible, but it comes with context-specific challenges. In an ideal world, popularity (or revenue generated, total streams, total sales, etc) would be tied to a particular publishing agreement. An artist cares how well a release does generally, and a publisher (or streamer) cares about how well it does on (or how much they would need to pay) across their particular channels. But maybe you'll hire me to look at data like that....
    - As seen in the code, to get popularity on the ISRC level, I took the maximum popularity across all releases of a particular recording.

Interesting Tidbits

While the ultimate goal was to build a predictive model of song popularity and make some point predictions, I wanted to explore how our tastes have changed over time, particularly in terms of loudness. Let's start with volume. Novice producers never shut up about the loudness wars, myself included. Thankfully, the dataset contains an average LUFS value taken over a song's duration. Plotting average loudness and release volume across release date (as mentioned above, the release's most popular release date) yields the following:



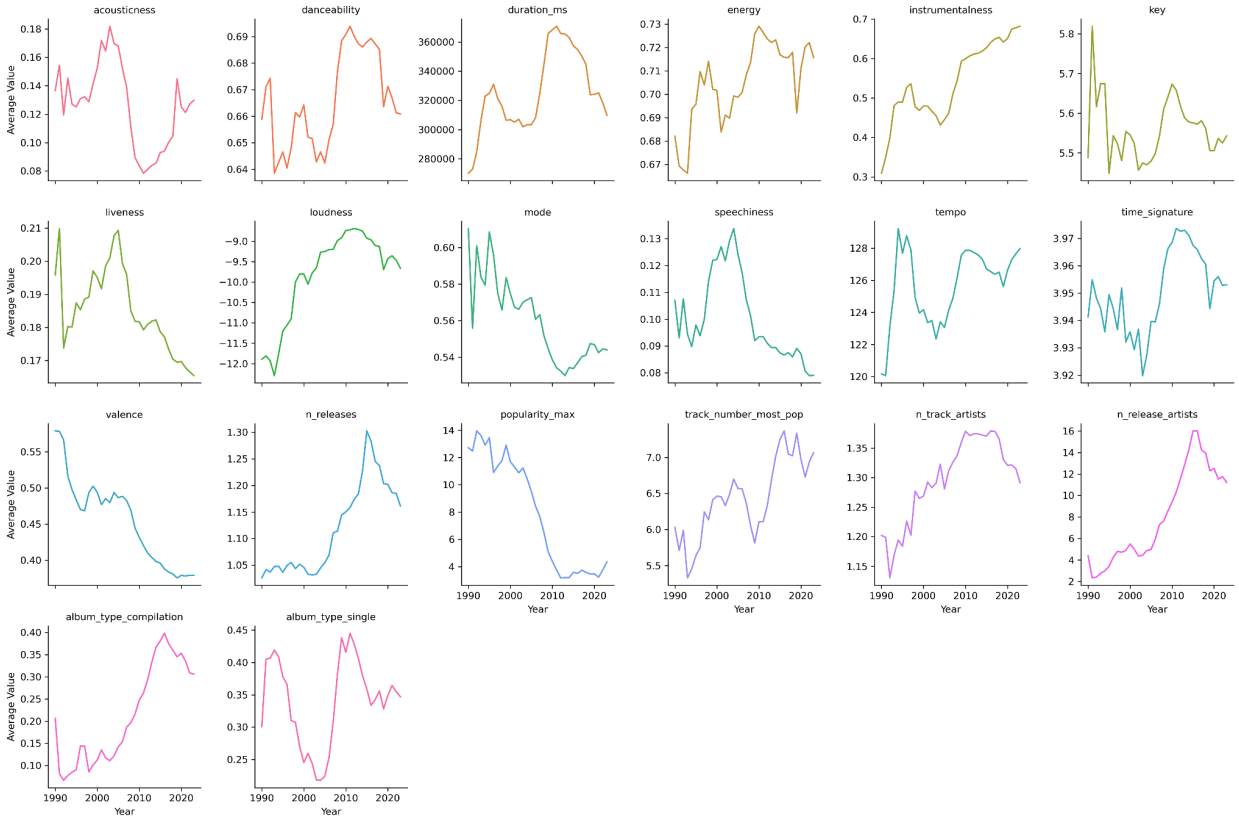
(Note: There are pre-1990 releases that I excluded from this chart as the n-sizes fall, making the average. LUFS measure messy)

Average Loudness had increased over time, particularly from the mid-nineties to around 2010-2012, a time period that many mark as the peak of the loudness wars (and peak recession pop). The early 2010s are also when streaming services emerged as the dominant way to consume music and implemented various normalization techniques that made pushing for loudness less relevant to many artists. As such, a decline in average loudness from that period makes some sense.

Some caveats: As you can see, the dataset is heavily skewed towards releases after 2010, which makes it hard to trust the pre-2010 values. More importantly, other controls would make this story more convincing, namely, a breakdown of loudness by genre (maybe I'll do a genre classification at some point). Music streaming tastes change over time, and some genres put a premium on volume more than others, so it's not entirely clear what's driving the variation.

At the same time, the pre-2010 release counts are not small, and the trends reveal themselves over many years, providing some evidence that the loudness wars are real.

How about the other numeric features over time?



(Note: Note, as before, I only include releases from 1990 onwards)

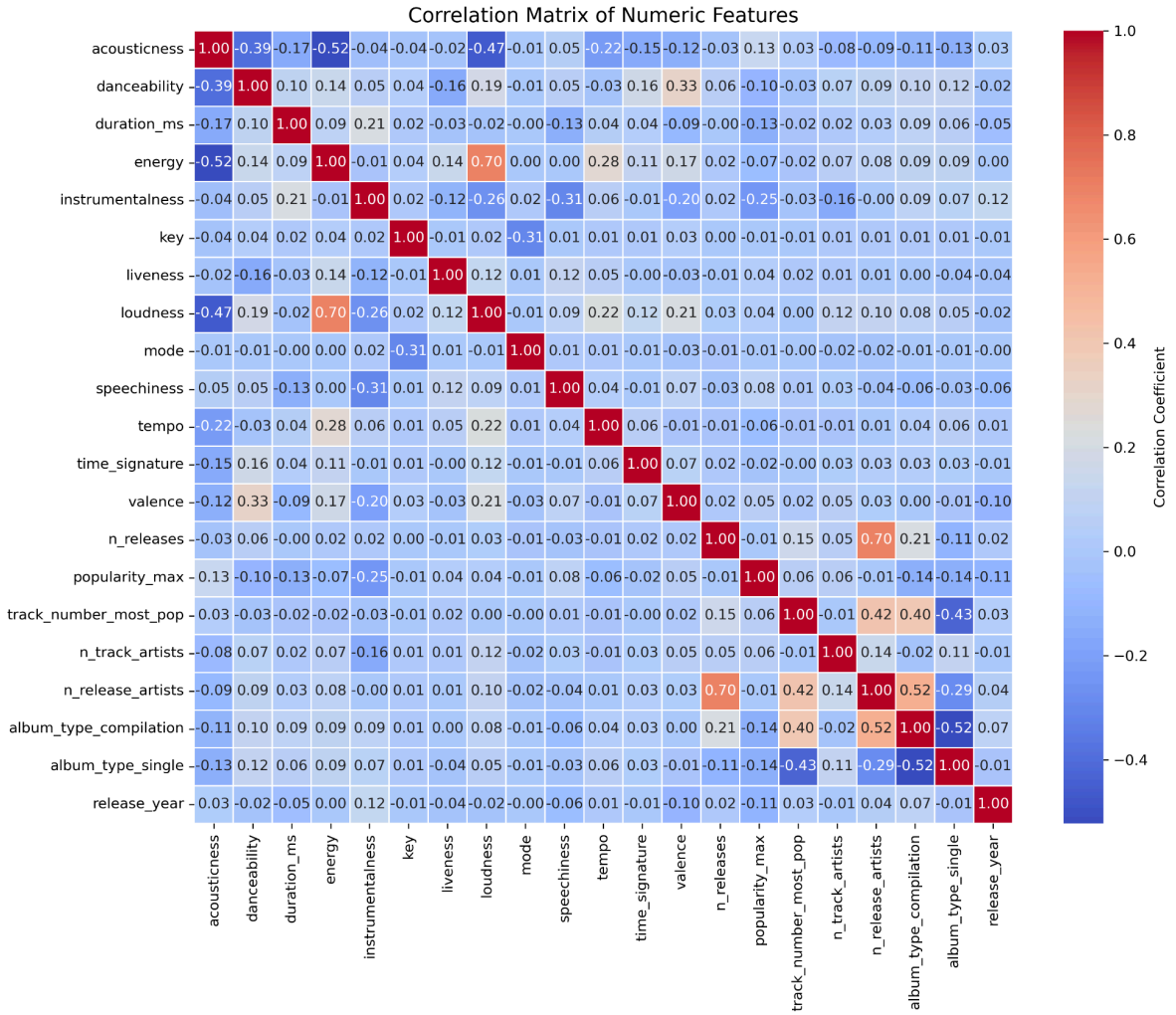
Some of the variance in these charts is due to uneven sample sizes, but a couple reveal interesting trends. The number of unique artists per track has steadily increased over time, lending some credence to the argument that music is increasingly written (not measured here) and recorded in teams.

Additionally, we see that valence, a measure of how 'positive' a track is (with one being the highest possible), has dropped significantly and consistently. Either music has gotten sadder, the pre-2000 averages are mainly due to lower sample sizes, music has always been sad, or the metric is fundamentally inaccurate. I'm not sure which story is most convincing, but the findings in either scenario are fascinating.

Danceability also stands out for rising significantly in the mid-2000s onward. EDM became part of the mainstream in the late 2000s, with most popular releases today being heavily influenced by house, techno, garage, and other dance genres.

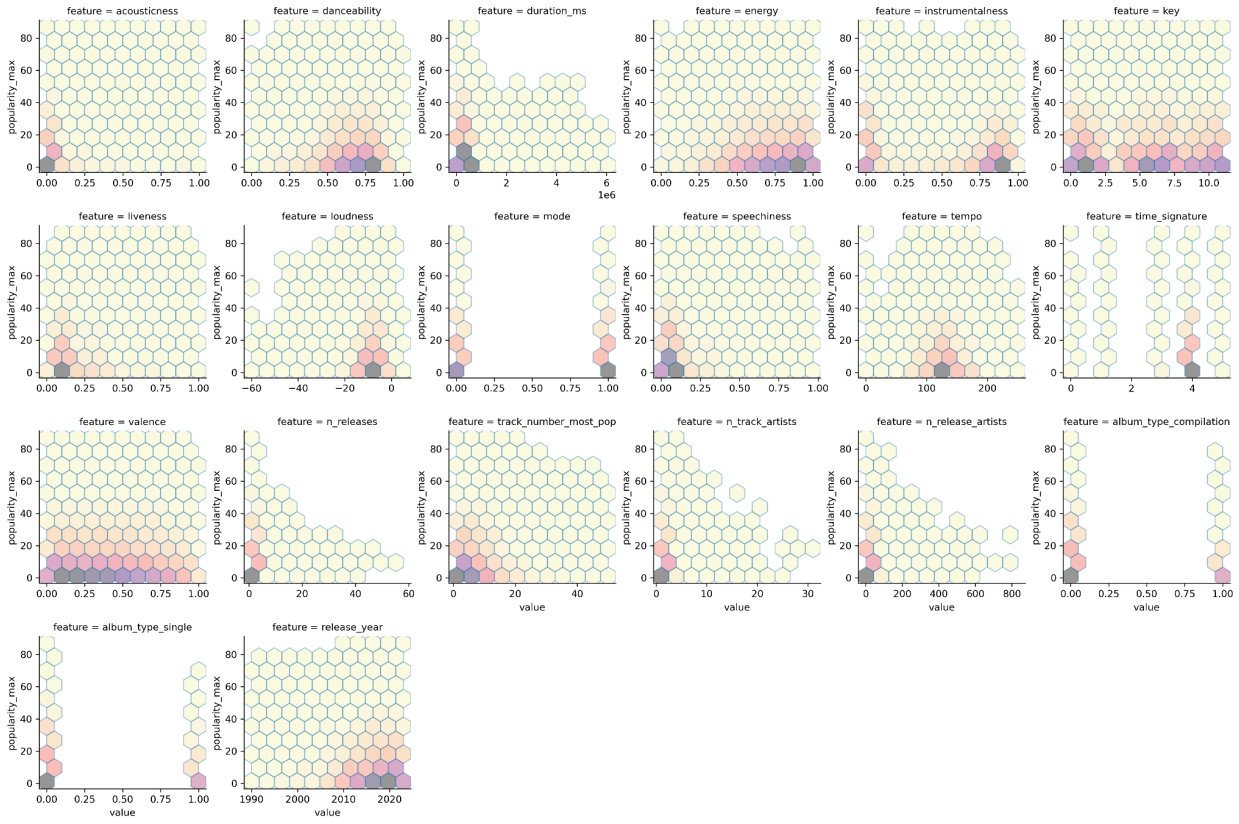
What about the ML?

Let's show some more typical charts that all good data scientists need to consider. Starting with linear relationships, we don't see many strong correlations between numeric features.. The number of releases and the number of unique artists across those releases correlate strongly, likely because a song being on multiple releases increases the chances of it being on a compilation album instead of just a single.



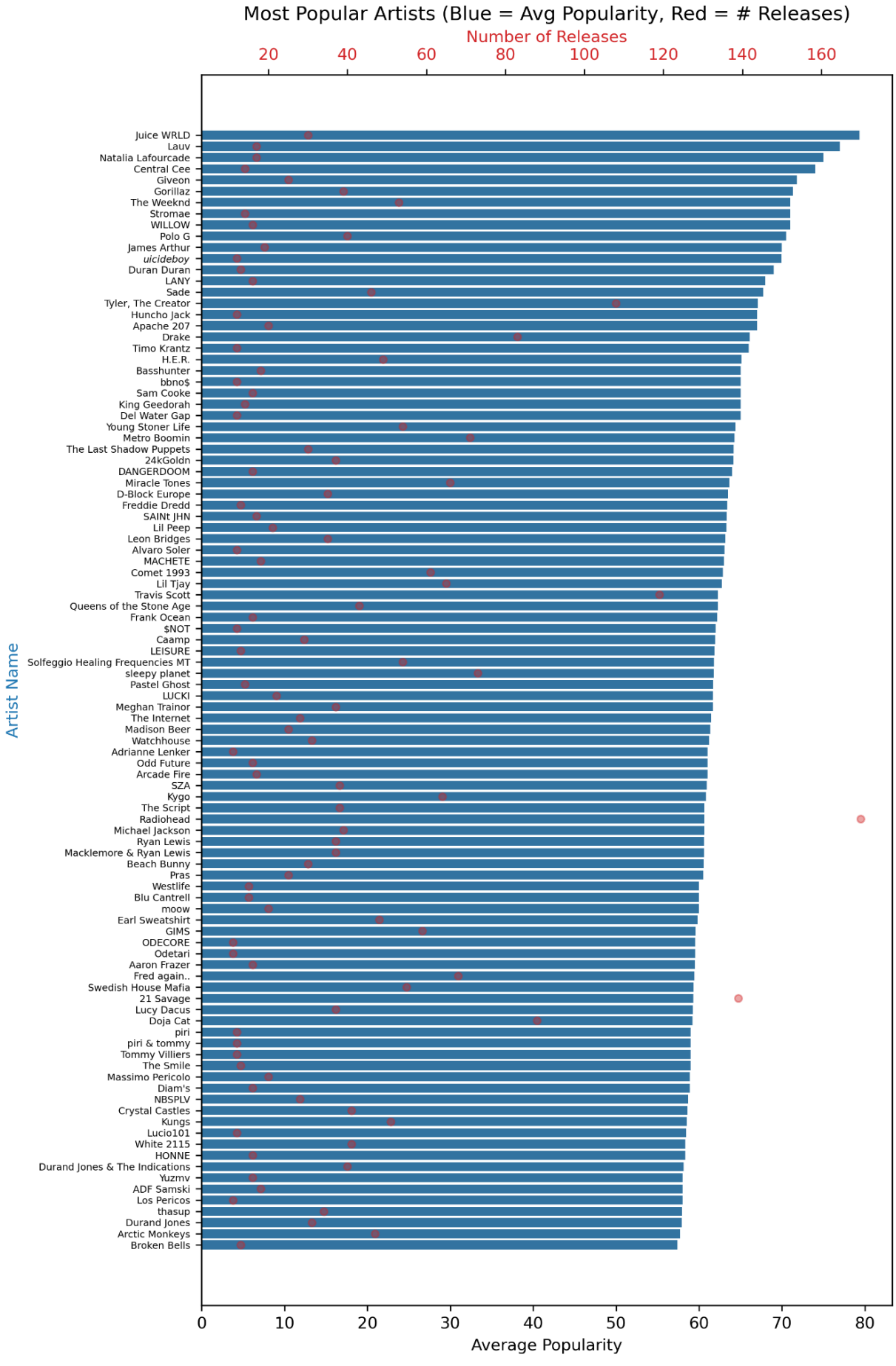
Additionally, we see a strong correlation between loudness and energy, which makes sense but is not so correlated as to raise any modelling concerns.

What about potential non-linear relationships?
To get a more profound sense of the relationship between popularity and a song's measures, I used hexplots to show density (scatter plots were too messy to be useful and took too long to compute). Nothing noteworthy jumps out. More recent songs tend to be more popular, which isn't surprising, considering Spotify didn't exist until 2008, so there would be some bias toward releases after that time. Additionally, louder songs are both more common, and the most popular songs tend to be above -20 LUFS (which isn't a particularly high threshold or particularly loud). Take a look:



(Note: The charts above only include songs released on or after 1990 and which have a popularity greater than zero in an effort to make density differences easier to see.)

OK, last tidbit before some modeling. Don't you want to see the most popular artists? The chart below reveals some interesting quirks about this particular dataset. The user who compiled the data files does not go into depth as to how they chose the songs or whether the sample was random. You may be surprised to know that artists including Beyonce, Lady Gaga, or Taylor Swift do not appear in the dataset at all despite being among the most popular artists of the last decade. In the 'real world' I would want to dive deeper into the selection methodology. However, it is still a sizable and rich dataset and this is all about implementing the fundamentals. Here are the most popular artists with 10+ releases:


Most Popular Artists (Blue = Avg Popularity, Red = # Releases)
Number of Releases
20
40
60
80
100
120
140
160
Artist Name
Juice WRLD
Lauv
Natalia Lafourcade
Central Cee
Giveon
Gorillaz
The Weeknd
Stromae
WILLOW
Polo G
James Arthur
uicideboy
Duran Duran
LANY
Sade
Tyler, The Creator
Huncho Jack
Apache 207
Drake
Timo Krantz
H.E.R.
Basshunter
bbno$
Sam Cooke
King Geedorah
Del Water Gap
Young Stoner Life
Metro Boomin
The Last Shadow Puppets
24kGoldn
DANGERDOOM
Miracle Tones
D-Block Europe
Freddie Dredd
SAINt JHN
Lil Peep
Leon Bridges
Alvaro Soler
MACHETE
Comet 1993
Lil Tjay
Travis Scott
Queens of the Stone Age
Frank Ocean
$NOT
Caamp
LEISURE
Solfeggio Healing Frequencies MT
sleepy planet
Pastel Ghost
LUCKI
Meghan Trainor
The Internet
Madison Beer
Watchhouse
Adrianne Lenker
Odd Future
Arcade Fire
SZA
Kygo
The Script
Radiohead
Michael Jackson
Ryan Lewis
Macklemore & Ryan Lewis
Beach Bunny
Pras
Westlife
Blu Cantrell
moow
Earl Sweatshirt
GIMS
ODECORE
Odetari
Aaron Frazer
Fred again..
Swedish House Mafia
21 Savage
Lucy Dacus
Doja Cat
piri
piri & tommy
Tommy Villiers
The Smile
Massimo Pericolo
Diam's
NBSPLV
Crystal Castles
Kungs
Lucio101
White 2115
HONNE
Durand Jones & The Indications
Yuzmv
ADF Samski
Los Pericos
thasup
Durand Jones
Arctic Monkeys
Broken Bells
0
10
20
30
40
50
60
70
80
Average Popularity

As you can see, there are many names an average Spotify user would expect to see–The Weeknd, SZA, Radiohead, Drake, etc–in addition to many names that are arguably less familiar. This chart alone does not tell us much about how artists were selected, given that it features artists associated with a diversity of genres. However, some post-modeling exploration suggested that it may be biased towards dance music and that many of the songs from the mainstream artists in this dataset are remixes.

Predicting Popularity

The charts above primarily showcase the numerical features of recordings. However, we cannot reasonably separate a song's popularity from the brand of a particular artist and their label. As such, I combined the dense numerical data with a subset of artist names, release titles, and label names to produce a hybrid dataframe. The complete numeric feature set is listed below, along with the text feature headers:

RangeIndex: 4687104 entries, 0 to 4687103
Data columns (total 23 columns):

#	Column	Dtype
0	track_artist_names	object
1	track_title_most_pop	object
2	label_name_clean	object
3	acousticness	float32
4	danceability	float32
5	duration_ms	float32
6	energy	float32
7	instrumentalness	float32
8	key	float32
9	liveness	float32
10	loudness	float32
11	mode	float32
12	speechiness	float32
13	tempo	float32
14	time_signature	float32
15	valence	float32
16	n_releases	float32
17	track_number_most_pop	float32
18	n_track_artists	float32
19	n_release_artists	float32
20	album_type_compilation	float32
21	album_type_single	float32
22	release_year	float32

text_features = ['track_artist_names', 'track_title_most_pop', 'label_name_clean']

The high-level process was to start with randomized searches to choose (1)the optimal estimator, (2)the best way to limit dimensionality, and (3) a grid search to find the best solver and regularization size.

What did we end up with?
I focused only on estimators that handle sparse text with relative ease (avoiding specific ensemble methods such as random forest that would have taken forever to run locally). Due to the size of the dataset and my computer's computational limitations, tuning and fitting took a long time, even with small subsets of the training split. In the end, I conducted randomized searches on four different estimators: RidgeRegressor, ElasticNetRegressor, SGDRegressor, and LinearSVR. Ridge performed the best.

Implementing SVD on top of dimensionality control during vectorization and hashing resulted in a drastic drop in test R^2, so I ultimately chose not to use it. I then used a grid search to tune the solver type and alpha for the ridge model, which was then fit on the entire test set. The result was a Ridge model with a test set R^2 of 0.648 and a modest alpha of 1.589. The numeric coefficient estimates are listed below:

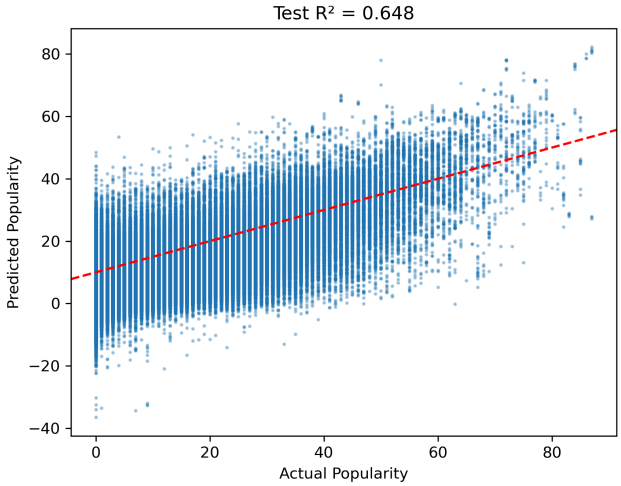
feature	coef
num__release_year	29.61355
num__n_release_artists	22.22173
num__n_track_artists	12.75986
num__duration_ms	-10.58712
num__loudness	4.12209
num__album_type_single	-3.25927
num__album_type_compilation	-3.02831
num__track_number_most_pop	1.78474
num__instrumentalness	-1.10847
num__acousticness	0.81273
num__speechiness	-0.63006
num__energy	-0.59739
num__n_releases	0.42792
num__tempo	-0.26923
num__danceability	-0.24573
num__time_signature	0.23899
num__valence	0.19106
num__liveness	-0.08492
num__mode	-0.02942

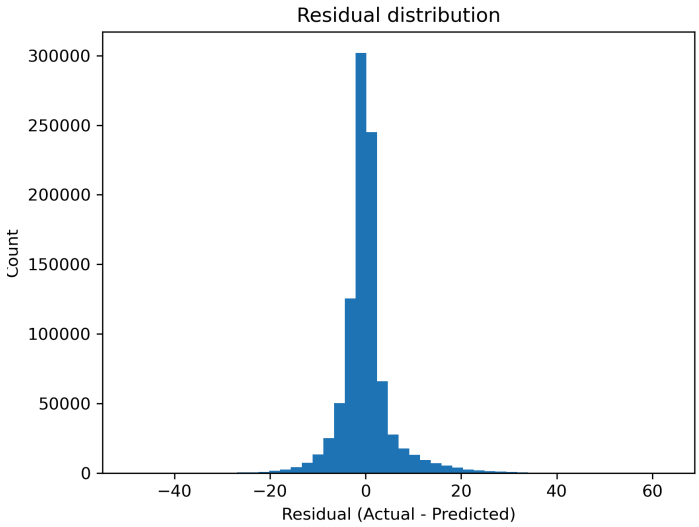
While the text features overall dominate the model due to their sheer number, the numerical features were, on average, many times more impactful on the model. Unsurprisingly, a release's title did not, on

average, influence popularity, whereas artist and label names did, most likely because the two strongly correlate with artist.

group	sum_per_feature	mean	count
num	92.012598	4.60063	20
artist_name	212426.2055	1.620683	131072
label_name	182448.4216	1.391971	131072
title	5957.731111	0.595773	10000

The model's accuracy does not appear to be biased across any sub-range of popularity, with normally distributed residuals.





How does the model square up to our intuition?

An R^2 in the mid-60s is decent. The coefficients do not reveal anything groundbreaking: we prefer shorter, louder songs on average for particular artists. What happens when we use it to predict the popularity of a specific song, real or imagined?

As a massive Charli XCX fan, I decided to use her song ‘Sympathy is a Knife’ as a mini post-test run. Before doing this, I wanted to make sure the song wasn’t in the dataset, and to see which of her songs were. The results surprised me.

To start, none of her feature albums were in the data set, and the majority of the tracks featuring her were remixes. The only singles of hers were songs she recorded with other big names, such as ‘I Love It,’ an Icona Pop song she helped write and record, and ‘Hot In It,’ a song she made with the EDM legend Tiesto. None of the tracks from Brat featured in the playlist.

After checking the songs, I created a test dataframe and filled in the feature values as accurately as possible. I could look up or measure the values for tempo, key, and loudness. Some of the others, such as energy or danceability, I had to input subjectively:

```
charli_song = pd.DataFrame([{
    "track_artist_names": "Charli XCX",
    "track_title_most_pop": "Sympathy is a Knife",
    "label_name_clean": "Atlantic Records",
    "acousticness": .2,
    "danceability": 0.75,
    "duration_ms": 151020,       #~2:31
    "energy": 0.85,
    "instrumentalness": 0.2,
    "key": 3.0,
    "liveness": 0.0,
    "loudness": -6.8, #Integrated LUFS reading
    "mode": 0.8,
    "speechiness": 0.55,
    "tempo": 132.0,
    "time_signature": 4,
    "valence": 0.6,
    "n_releases": 2,
    "track_number_most_pop": 3,
    "n_track_artists": 1,
    "n_release_artists": 1,
    "album_type_compilation": 0,
    "album_type_single": 0,
    "release_year": 2024,
}])
```

Along with ‘Sympathy is a Knife,’ I made two more made-up test songs and then used the model to predict their popularity scores. The two imaginary tracks had contrasting characteristics to see how they led to different predictions. Charli’s predicted score surprised me:

track_artist_names	track_title_most_pop	predicted_popularity
Charli XCX	Sympathy is a Knife	**9.627953**
Phoebe Bridgers	Late Night Memory	**3.133056**
DJ Solaris	Neon Pulse	**5.374889**

‘Sympathy is a Knife’ has been streamed over 125 million times on Spotify alone and is part of her Grammy award-winning album ‘Brat.’ More so, the majority of Charli XCX's songs in the dataset have a score above 10. The prediction of 9.628 seems a bit low. The dataset’s bias towards dance tracks, as well as the artist-name hashing, likely contribute to this low-ball prediction. I get the sense that she is being ‘drowned out’ by other artists who are more prominently featured.

With that said, ‘Sympathy is a Knife’ is a loud, quick, danceable song with high energy. It did get a higher prediction than the fake down-tempo Phoebe Bridgers folk song, as well as a super loud, energetic fake song from a DJ who does not exist (well, some guy on Spotify with 128 followers has the same, to be accurate).

Some Concluding Thoughts

(1) We have some evidence that the loudness wars peaked in the mid-2000s and entered a new era in the streaming age, where producers appear less concerned with average loudness.
(2) Artist brand matters, but so do the song attributes themselves. Numerical proxies for brand could be highly useful for improving predictability and taming dimensionality.

What would I do differently if I re-did this?

(1) More efficient code structure: Less redundancy and cleaner variable creation
(2) Longer searches: More importantly, I would spend more time searching for estimators. I got annoyed early on by how long it was taking to conduct randomized searches, even with a significantly smaller sample of 100,000. Gauging performance, all else equal, on even smaller subsets could have yielded a more predictive model in the end (though with some risk of choosing an estimator that naturally does better on smaller, not necessarily larger, datasets).

As with any ML exploration, we are only looking at one layer, so to speak. How cool would it be to have a breakdown by genre? To have other metrics on how users interacted with these recordings? To isolate the musical journeys of artists that started small and became mega-famous?

Maybe you’ll hire me, and I’ll be able to help answer these questions and more.