

PREDICTING ALBUM SCORES WITH LINEAR REGRESSION

Vinny Thompson

MOTIVATION

- Are critics **truly unbiased**, or are they predisposed to appreciate an album based on different factors?
- What information can we learn that might be insightful for a musician looking to release a critically successful album

Target Variable

Metascore: weighted average of critic reviews from metacritic.com



DATA ACQUISITION

Album Features:

Release Month

Release Day of Week

of Tracks

Avg. Track Length

Total Album Length

Artist's Monthly Spotify Listeners

% of Explicit Tracks

Acousticness

Danceability

Energy

Instrumentalness

Liveness

Loudness

Speechiness

Valence

Tempo

DATA ACQUISITION

Album Features:

Title Length
Release Month
Release Day of Week
of Tracks
Avg. Track Length
Total Album Length
% of Explicit Tracks
Acousticness
Danceability
Energy
Instrumentalness
Liveness
Loudness
Speechiness
Valence
Tempo
Artist's Monthly Spotify Listeners



Beautiful Soup



```
#Scrape data for each album: title, artist, critic score, user score, description, release date
for i in range(0,50):
    #Select page of results
    url_ext = f'&page={i}'
    url = base_url+url_ext
    response = requests.get(url, headers = user_agent)
    soup = BeautifulSoup(response.text, 'lxml')

    #Isolate table contents with album info
    tables = soup.findAll('table')
    table_rows = []
    for table in tables:
        table_rows.append(table.findAll('tr'))

    #Add info for each album
    for group in table_rows:
        for row in group:
            critic_reviews_links.append(row.find('a')['href'])
            titles.append(row.findAll('td')[1].find('h3').text)
            critic_scores.append(row.find('div').text)
            user_scores.append(row.findAll('td')[1].find('div', class_='user').text)
            artists.append(' '.join(row.findAll('td')[1].find('div').text.strip().split(' ')))
            descriptions.append(row.findAll('td')[1].find('p').text)
            if row.findAll('td')[1].find('span').text:
                dates_raw.append(row.findAll('td')[1].find('span').text)
            else:
                dates_raw.append('Empty')
```

DATA ACQUISITION

Album Features:

Title Length
Release Month
Release Day of Week
of Tracks

Avg. Track Length
Total Album Length
% of Explicit Tracks
Acousticness
Danceability
Energy
Instrumentalness
Liveness
Loudness
Speechiness
Valence
Tempo
Artist's Monthly Spotify Listeners



Beautiful Soup



Spotify



DATA ACQUISITION

Album Features:

Title Length
Release Month
Release Day of Week
of Tracks
Avg. Track Length
Total Album Length
% of Explicit Tracks
Acousticness
Danceability
Energy
Instrumentalness
Liveness
Loudness
Speechiness
Valence
Tempo
Artist's Monthly Spotify Listeners

Individual Track Danceability (10 Track Album)

[0.594, 0.447, 0.718, 0.743, 0.781, 0.63, 0.804, 0.326, 0.774, 0.606]

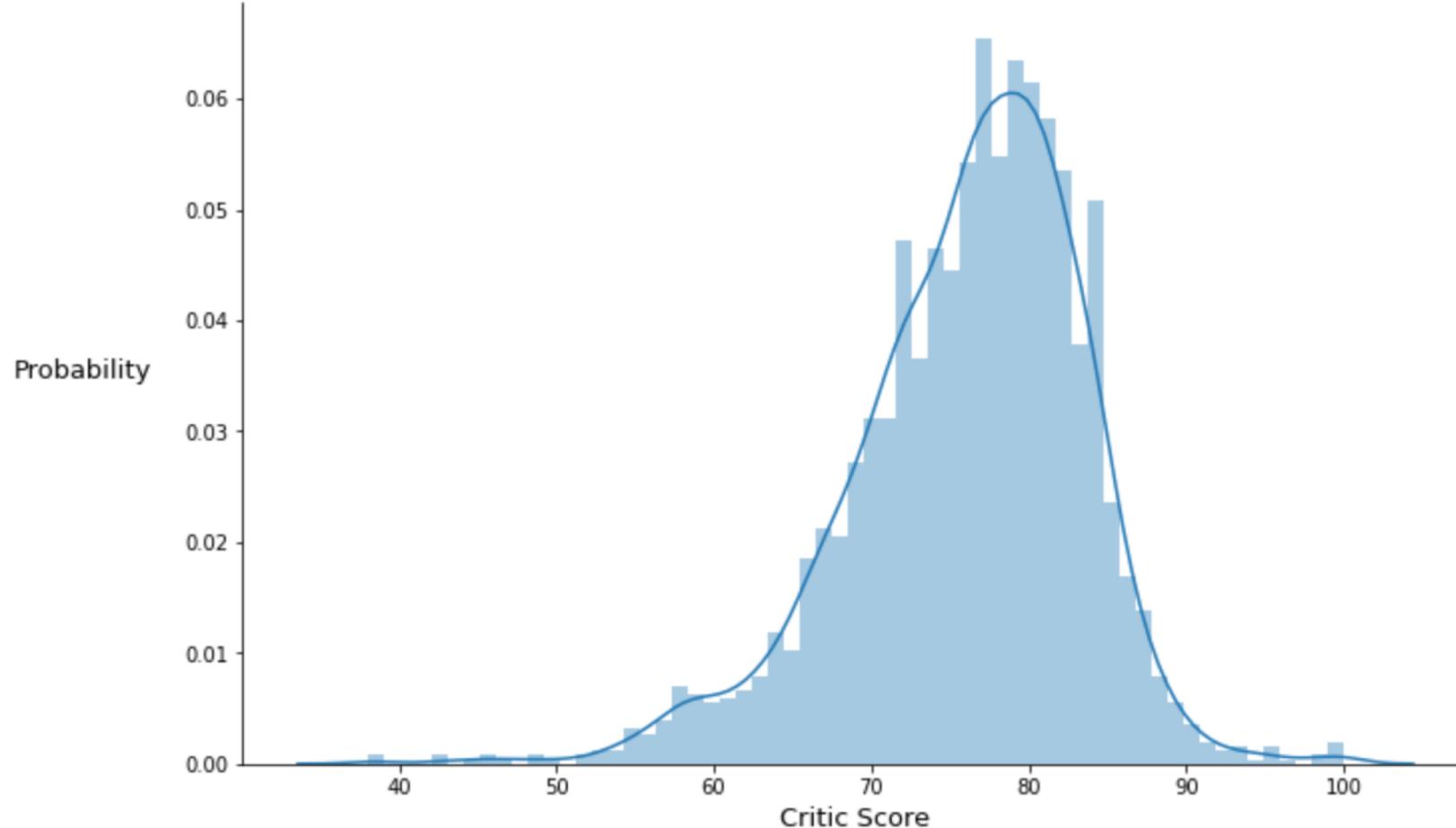
(Mean)

.642
(album metric)

DATA PREPARATION / FEATURE SELECTION

Distribution of Critic Scores

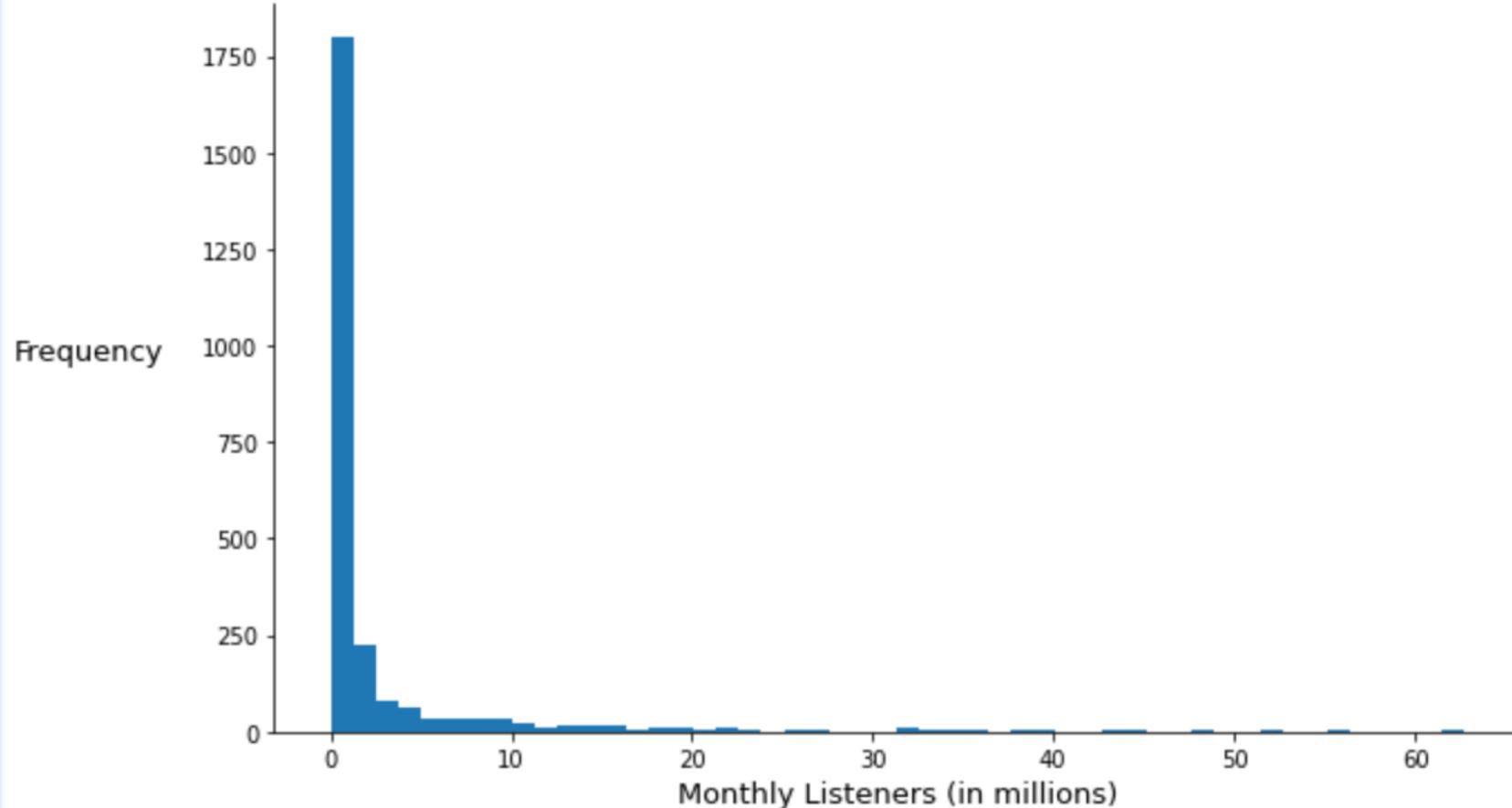
The target variable has a left skew with a minimum score of 38



DATA PREPARATION / FEATURE SELECTION

Distribution of Monthly Listeners is Heavily Right-Skewed

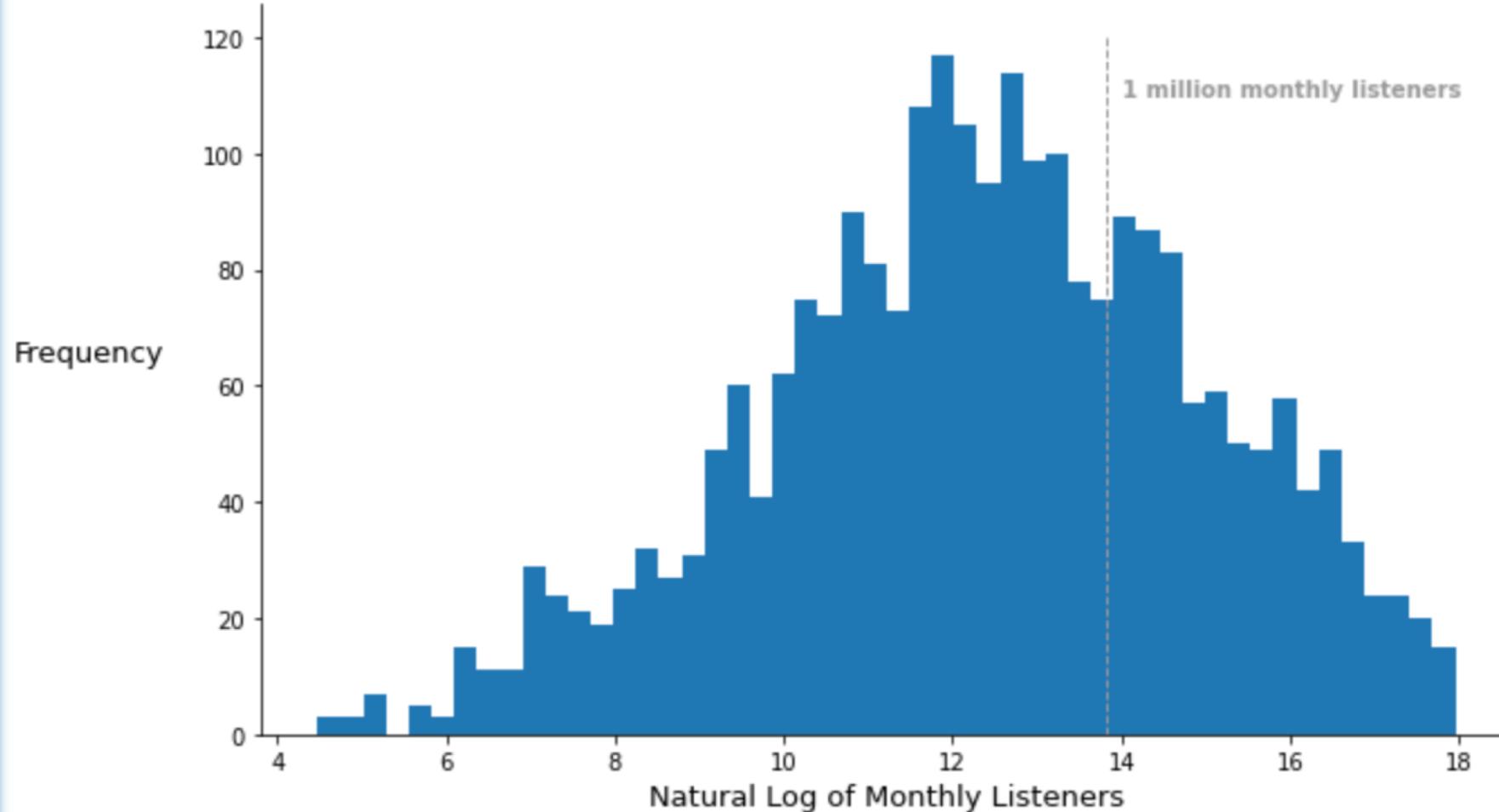
Out of 2500 total artists, 70% of them have less than 1 million monthly listeners



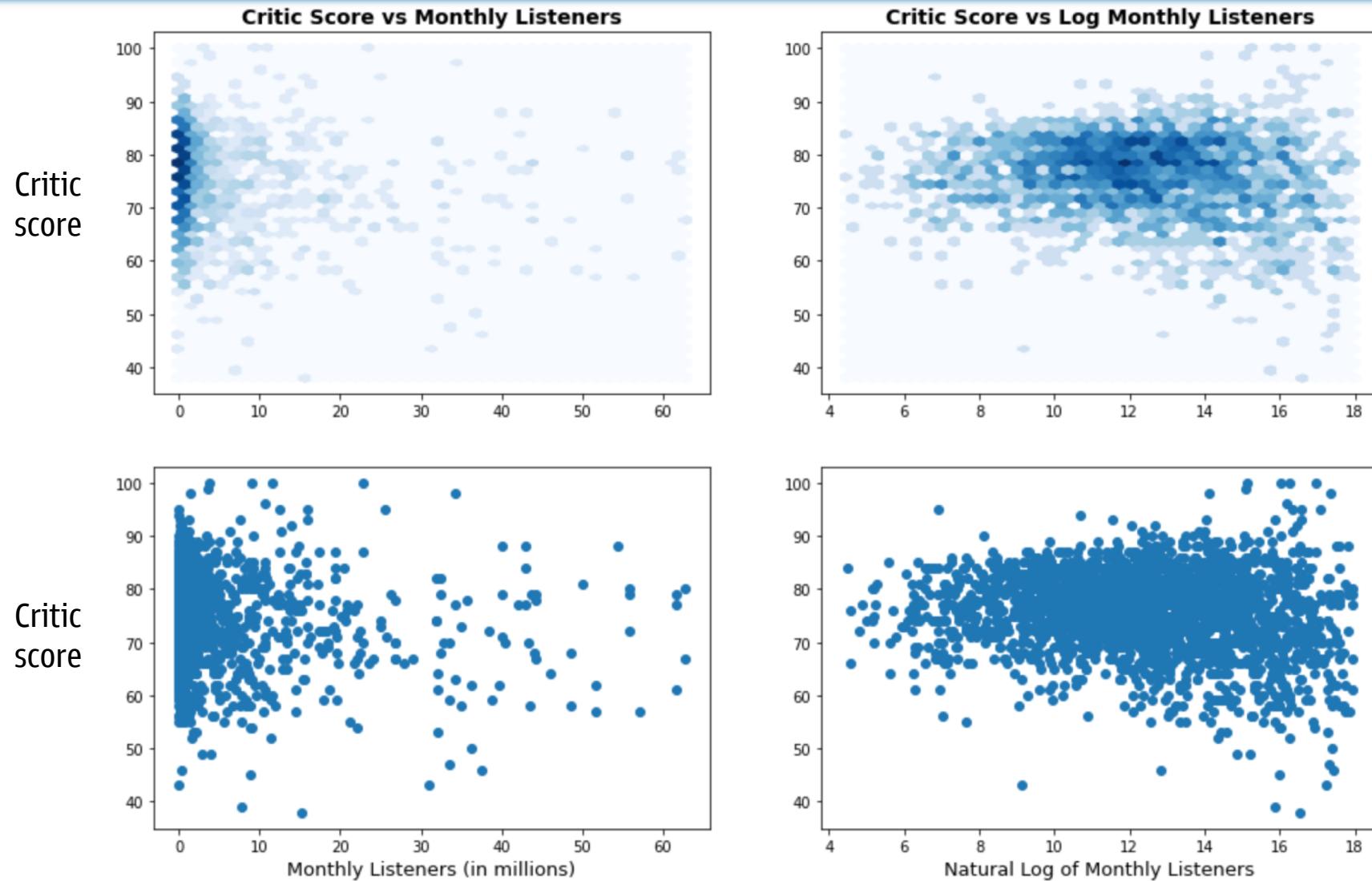
DATA PREPARATION / FEATURE SELECTION

Log Transform of Monthly Listeners Gives a Better Distribution

Out of 2500 total artists, 70% of them have less than 1 million monthly listeners



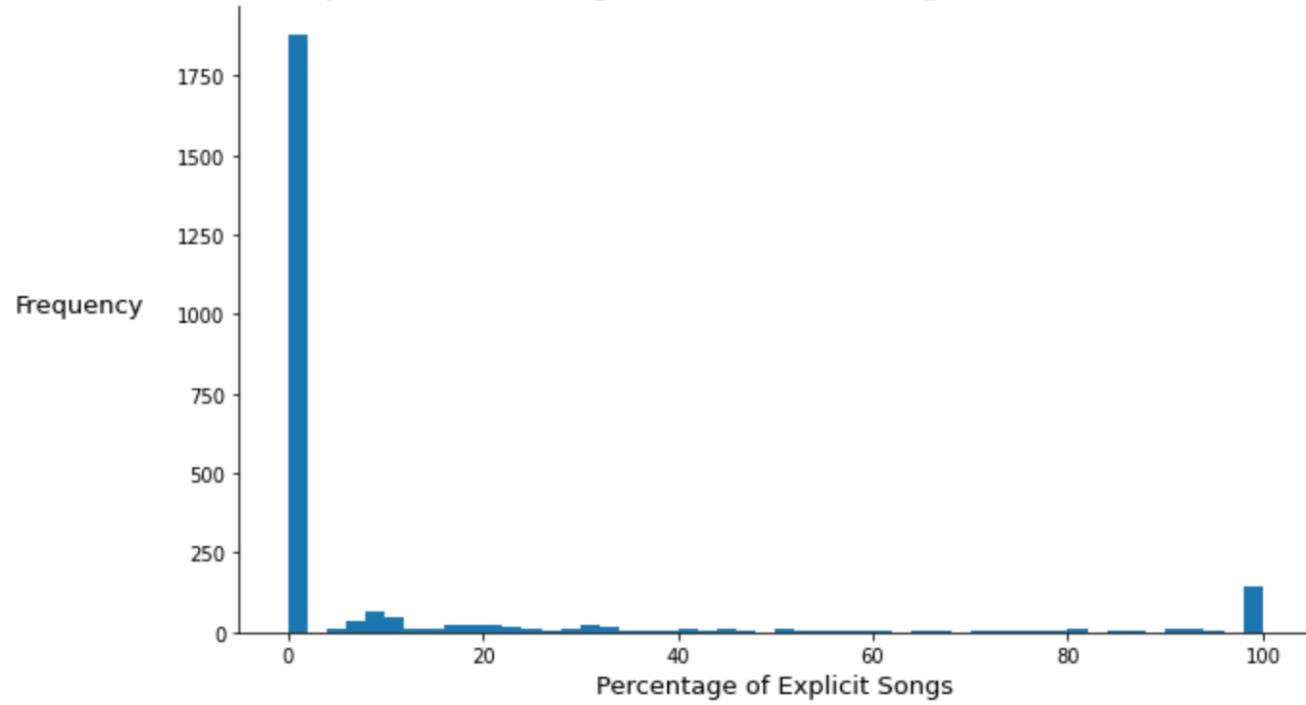
DATA PREPARATION / FEATURE SELECTION



DATA PREPARATION / FEATURE SELECTION

Most Albums Have No Explicit Songs

The explicitness feature might work better as a categorical variable

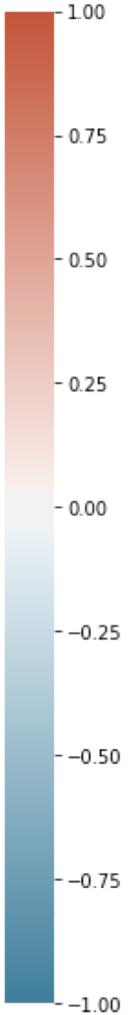
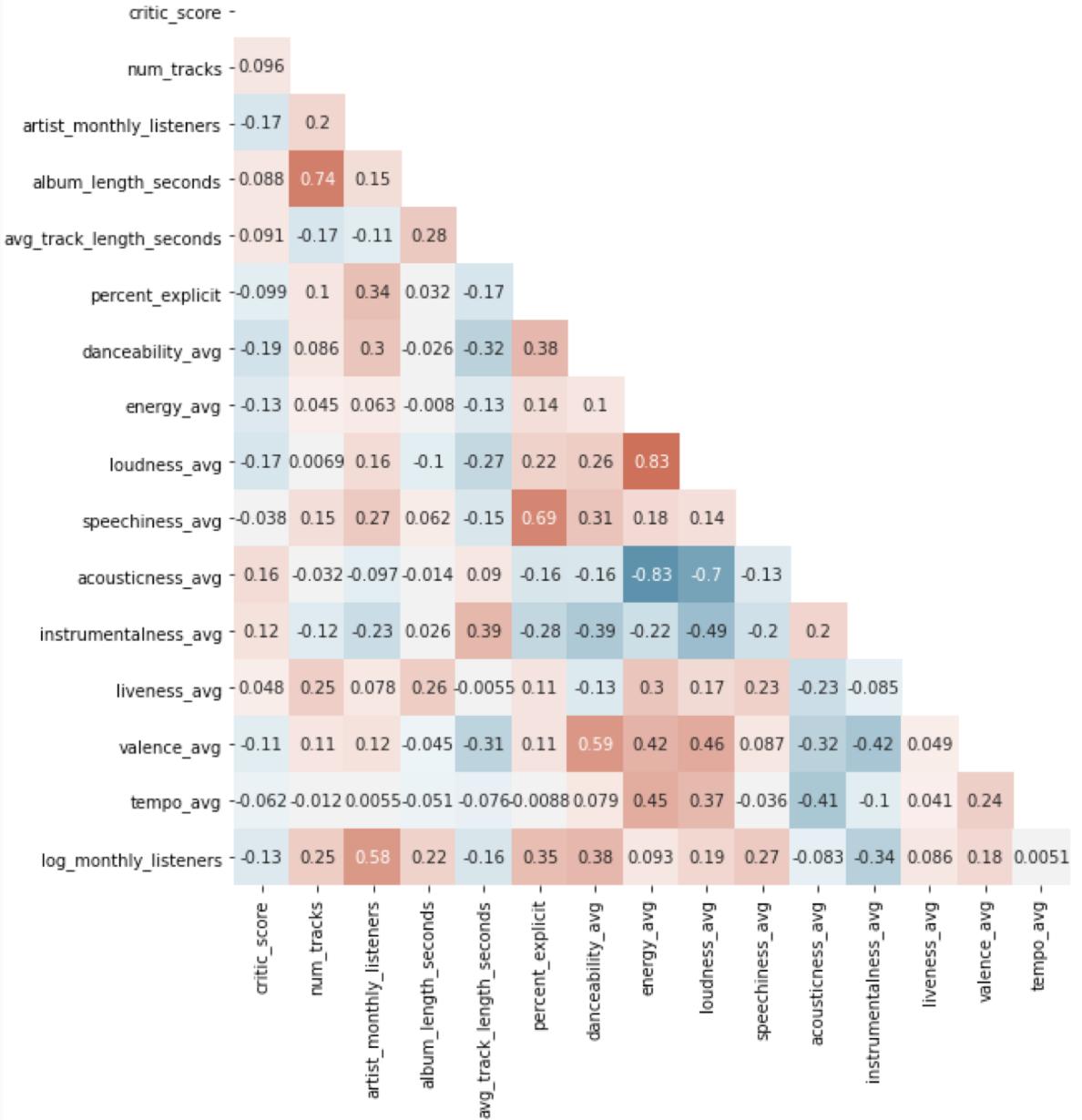


Three categories:

Percent Explicit	Avg Critic Score
0%	76.38
1-99%	75.57
100%	73.49

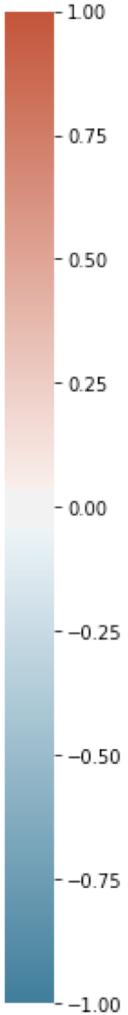
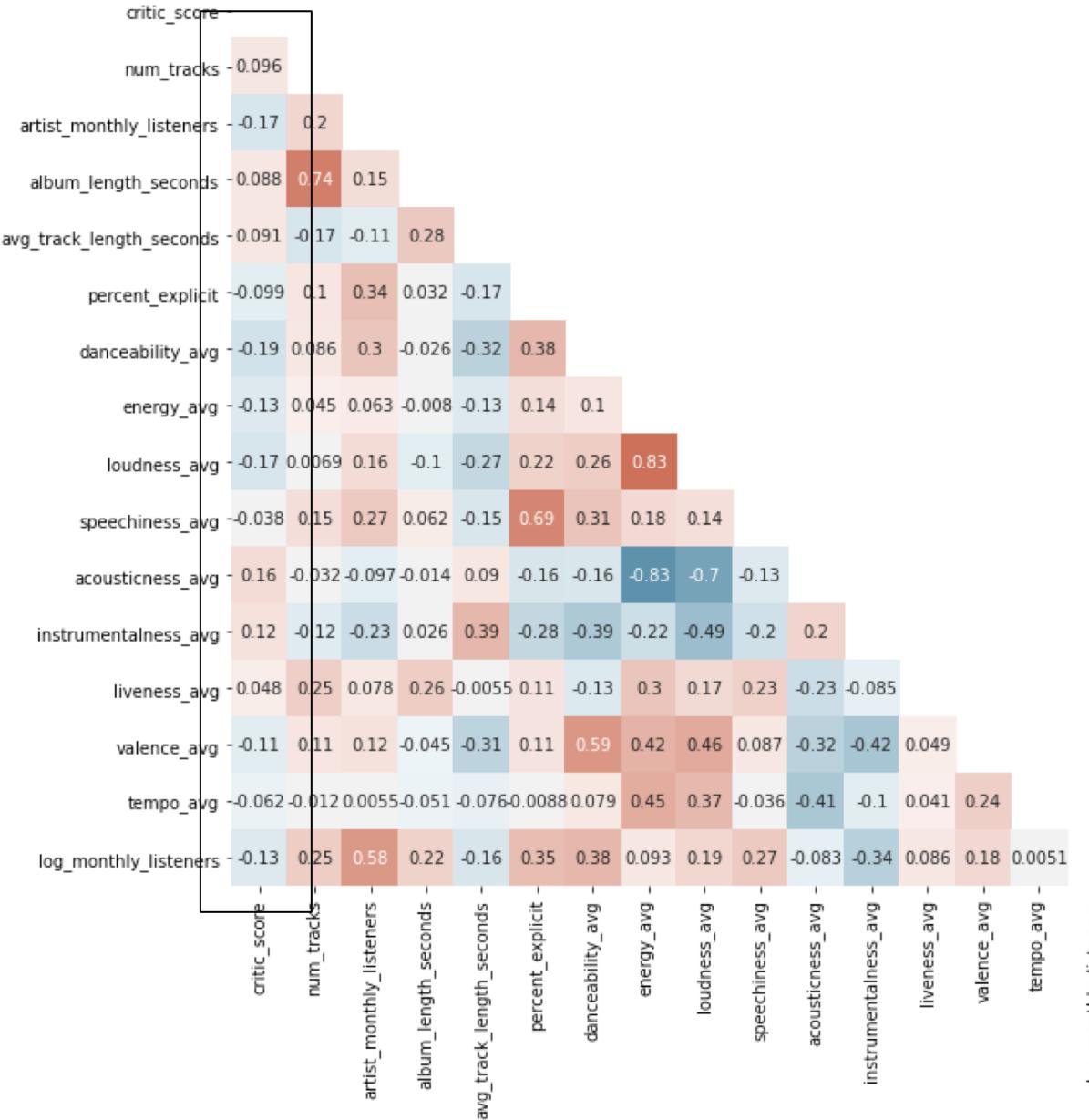
Feature Correlation Matrix

No features have a correlation magnitude of > .20 with our target



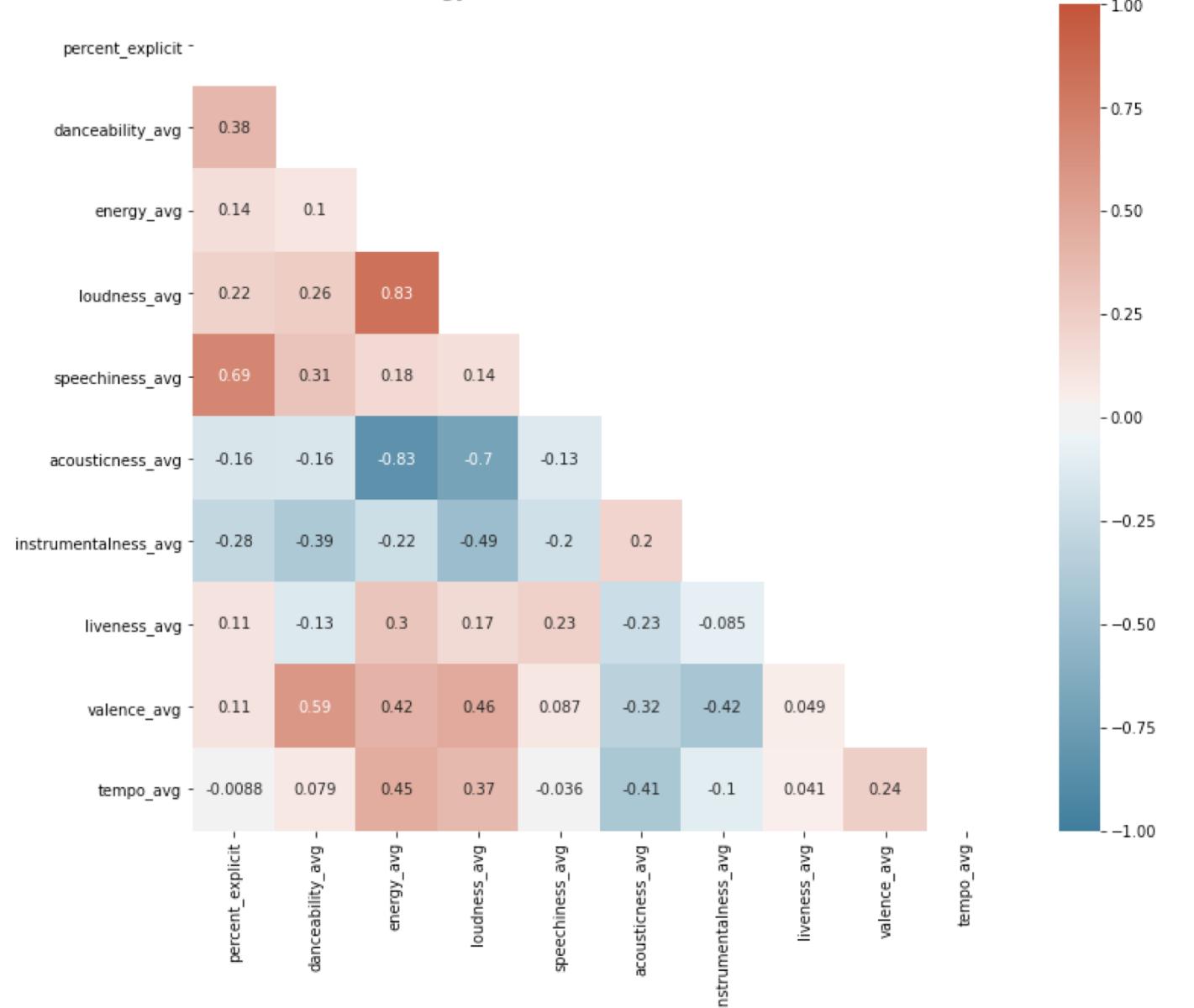
Feature Correlation Matrix

No features have a correlation magnitude of > .20 with our target



Correlation of Spotify Audio Features

There are a few potential collinearity issues, notably with loudness/energy and acousticness/energy



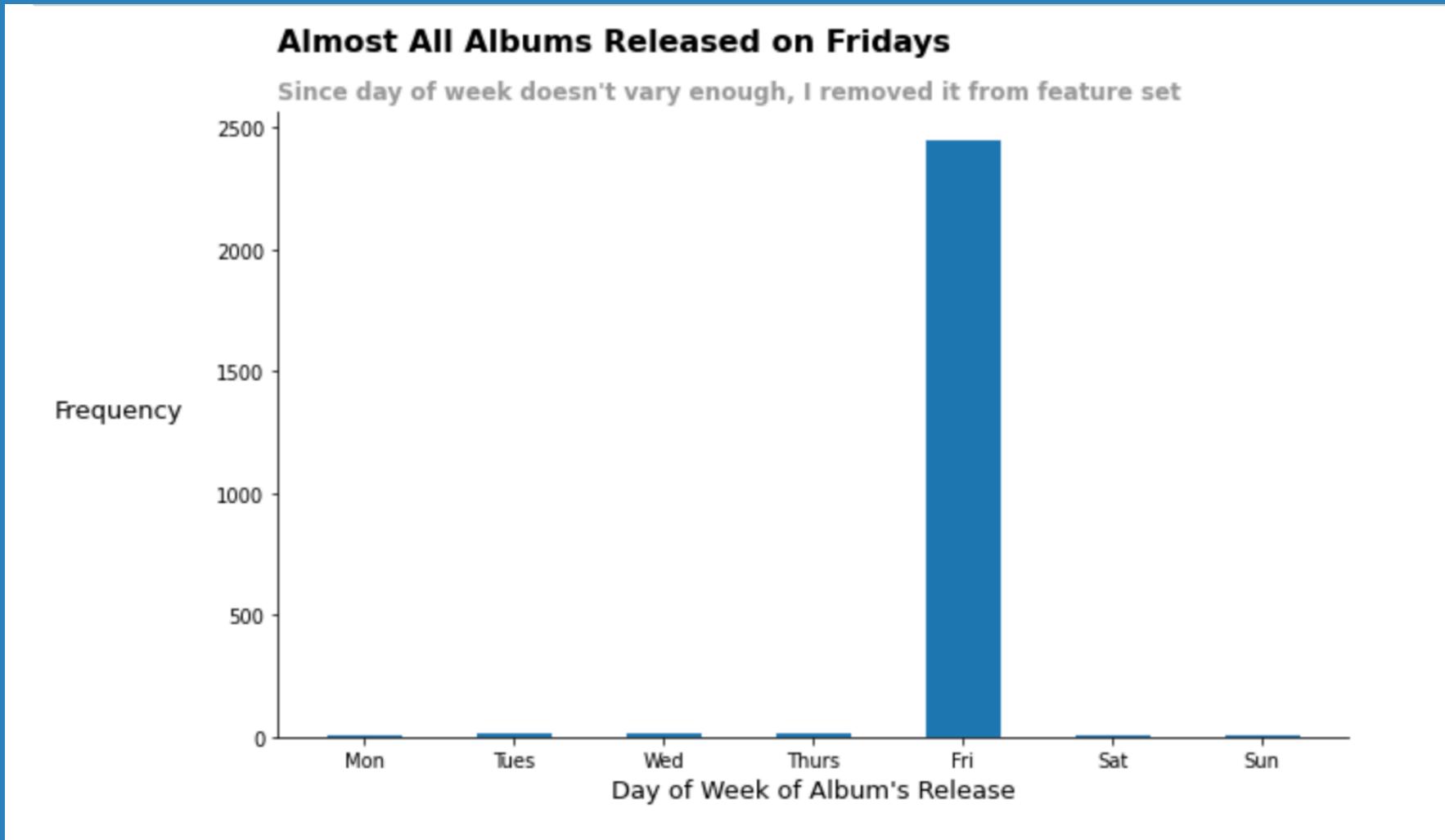
Potential multicollinearity issues:

- Loudness and Energy
- Acousticness and Energy
- Acousticness and Loudness
- Valence and Danceability

Ended up dropping:

- Energy
- Valence
- Tempo

DATA PREPARATION / FEATURE SELECTION



MODEL VALIDATION / ITERATION

The good:

- No P values above .80 and statsmodels would have warned if there were any strong multicollinearities in the model, so feature selection successfully eliminated that obstacle

The bad:

- Poor fit, with an R^2 of .108. Not a promising start, but not surprising since no one feature seemed to have a strong linear correlation with critic score

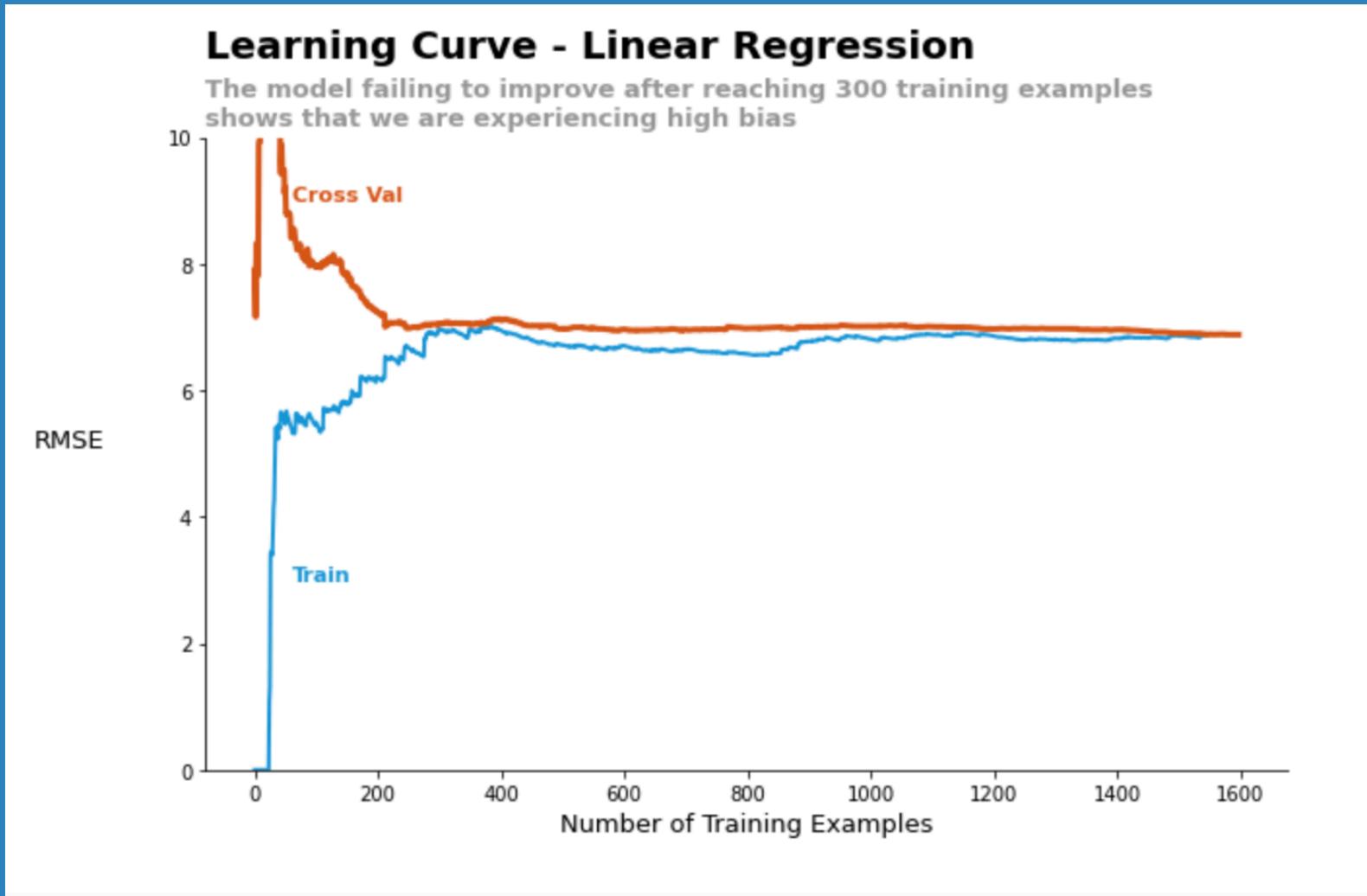
0.108

Train Set R^2

0.0813

Cross Val Mean R^2 (5 folds)

MODEL VALIDATION / ITERATION



MODEL VALIDATION / ITERATION

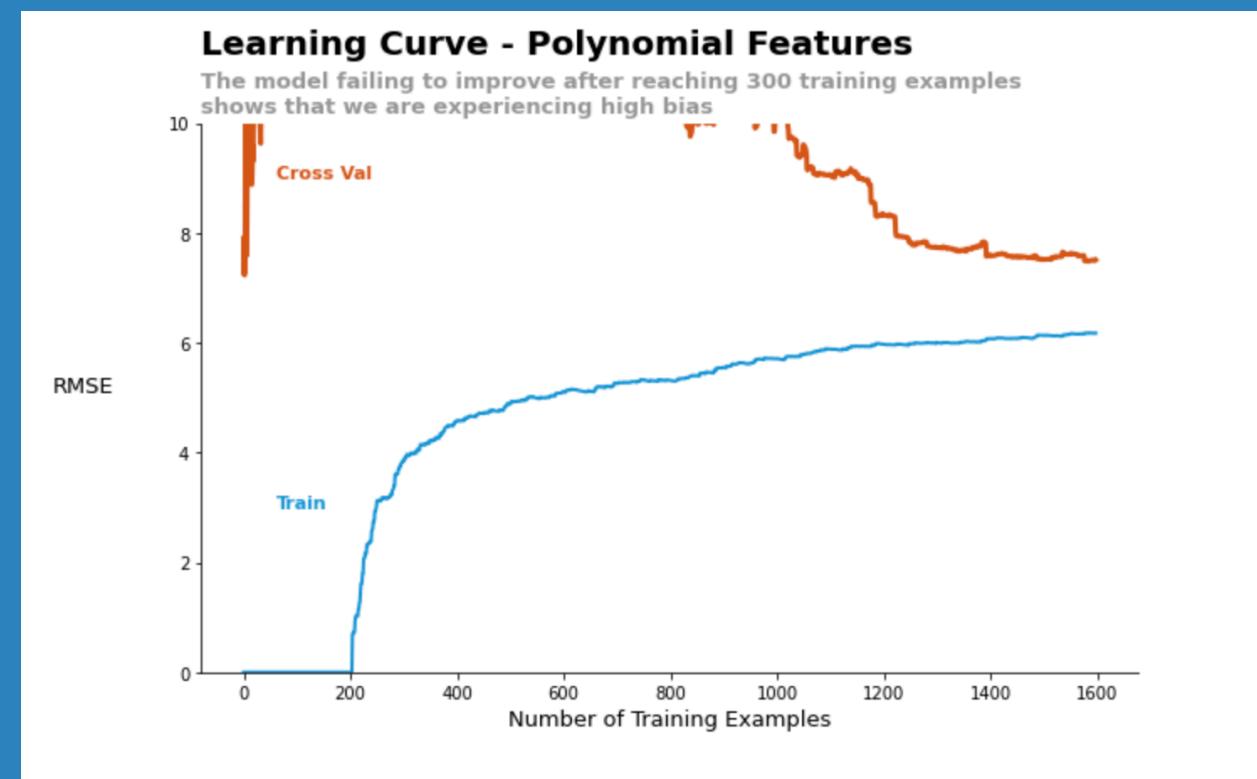
0.253

Train Set R²

0.057

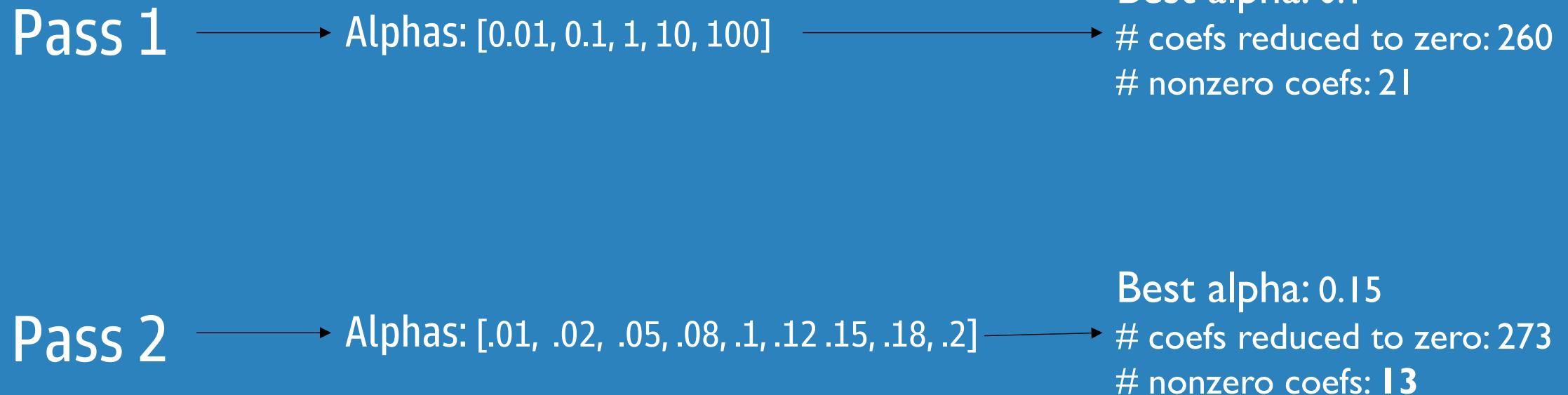
Cross Val Mean R² (5 folds)

Polynomial features
+
Interaction features
=
Overfitting



MODEL VALIDATION / ITERATION

Lasso CV:



0.15 chosen for hyperparameter

MODEL VALIDATION / ITERATION

Better than baseline model, but still underfitting the data

Baseline

0.108

Train Set R²

0.081

Cross Val Set R²

Lasso w/ Poly Features

0.120

Train Set R²

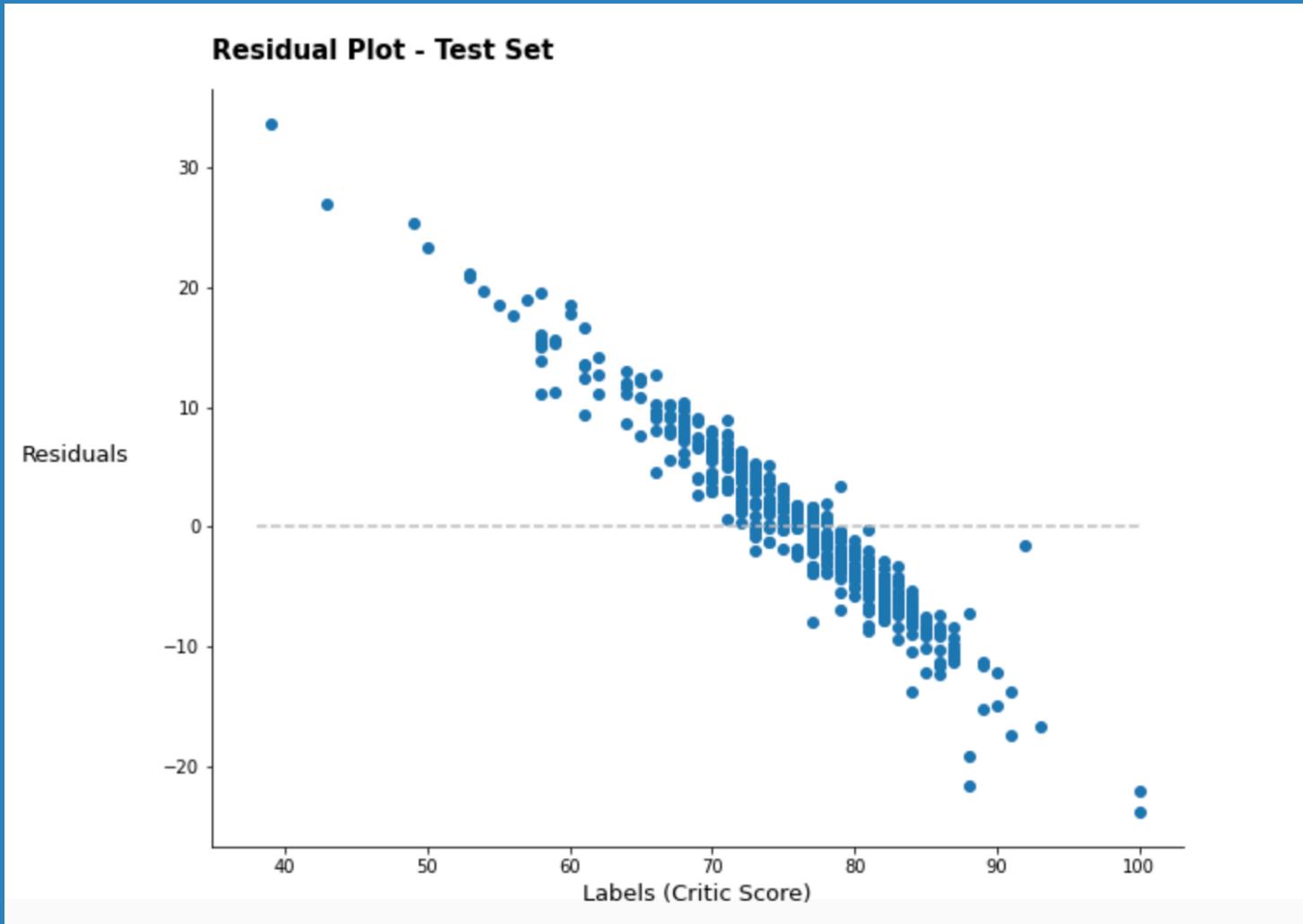
0.117

Cross Val Set R²

0.092

Test Set R²

MODEL VALIDATION / ITERATION



The further an album's score was from the mean, the worse the prediction was

	Mean	St. Dev
Predictions	76.04	1.99
Actual	75.89	7.78

CONCLUSION / NEXT STEPS

-Unable to prove any obvious critic biases



Next Steps

- Scrape additional sources for more predictive features, or features that might have important interaction with current features
 - Eg. Record label, genre
- Consider alternatives to using the strategy of averaging track features to get an album metric
 - Eg. Are audio features related to the first track on an album important to how it is received
- Try additional feature transformations to bring out linear relationships
- Possible that linear regression is just a poor model for this problem. Perhaps more success with a more powerful algorithm (SVM, neural network, ensemble learning)