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(' ')[1:
            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

→ Beautiful Soup → metacritic

Avg. Track Length
Total Album Length
% of Explicit Tracks
Acousticness
Danceability
Energy
Instrumentalness
Liveness
Loudness
Speechiness
Valence
Tempo

→ Spotipy → Spotify for Developers

Artist's Monthly Spotify Listeners

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

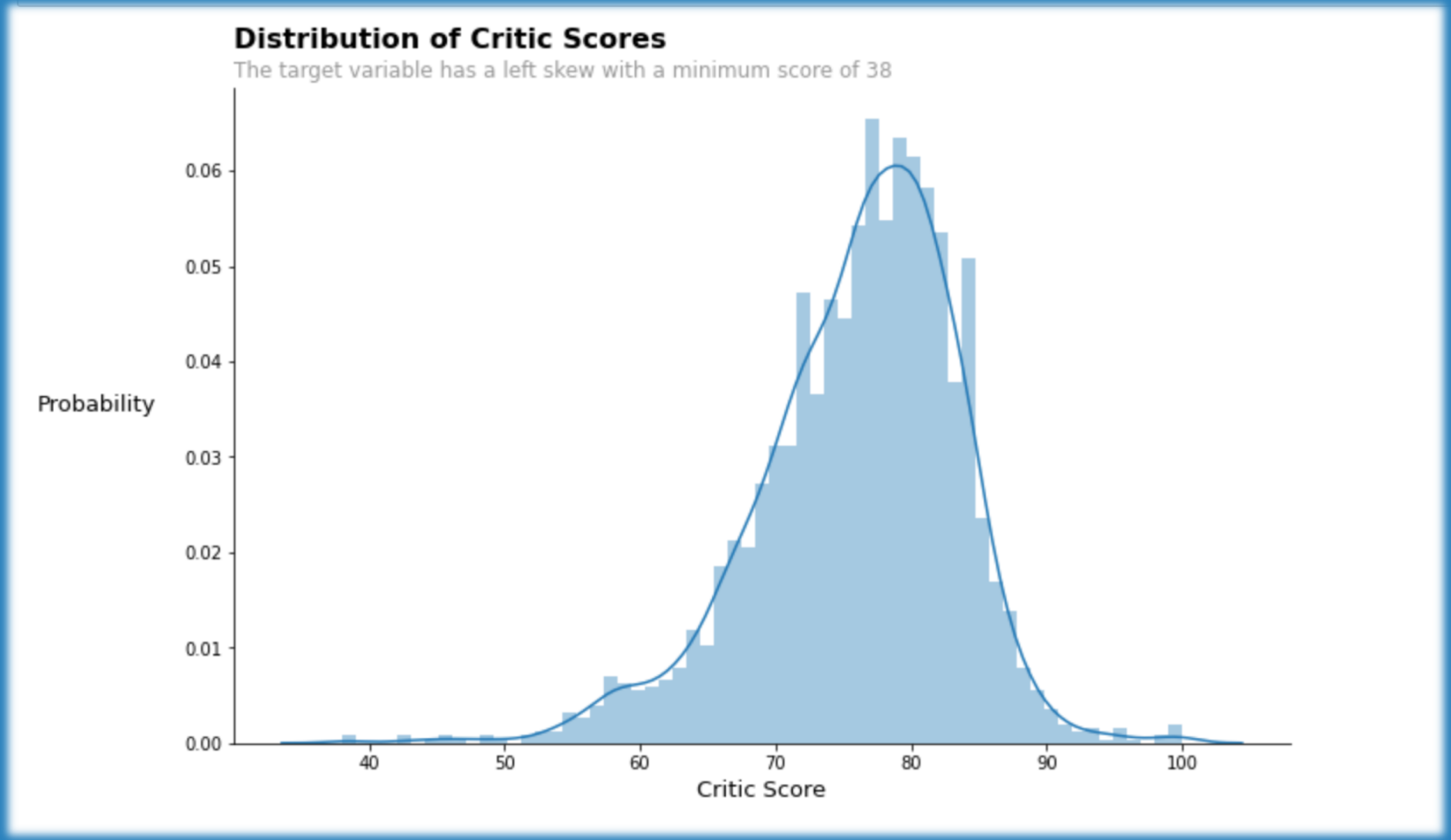
Individual Track Danceability (10 Track Album)

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

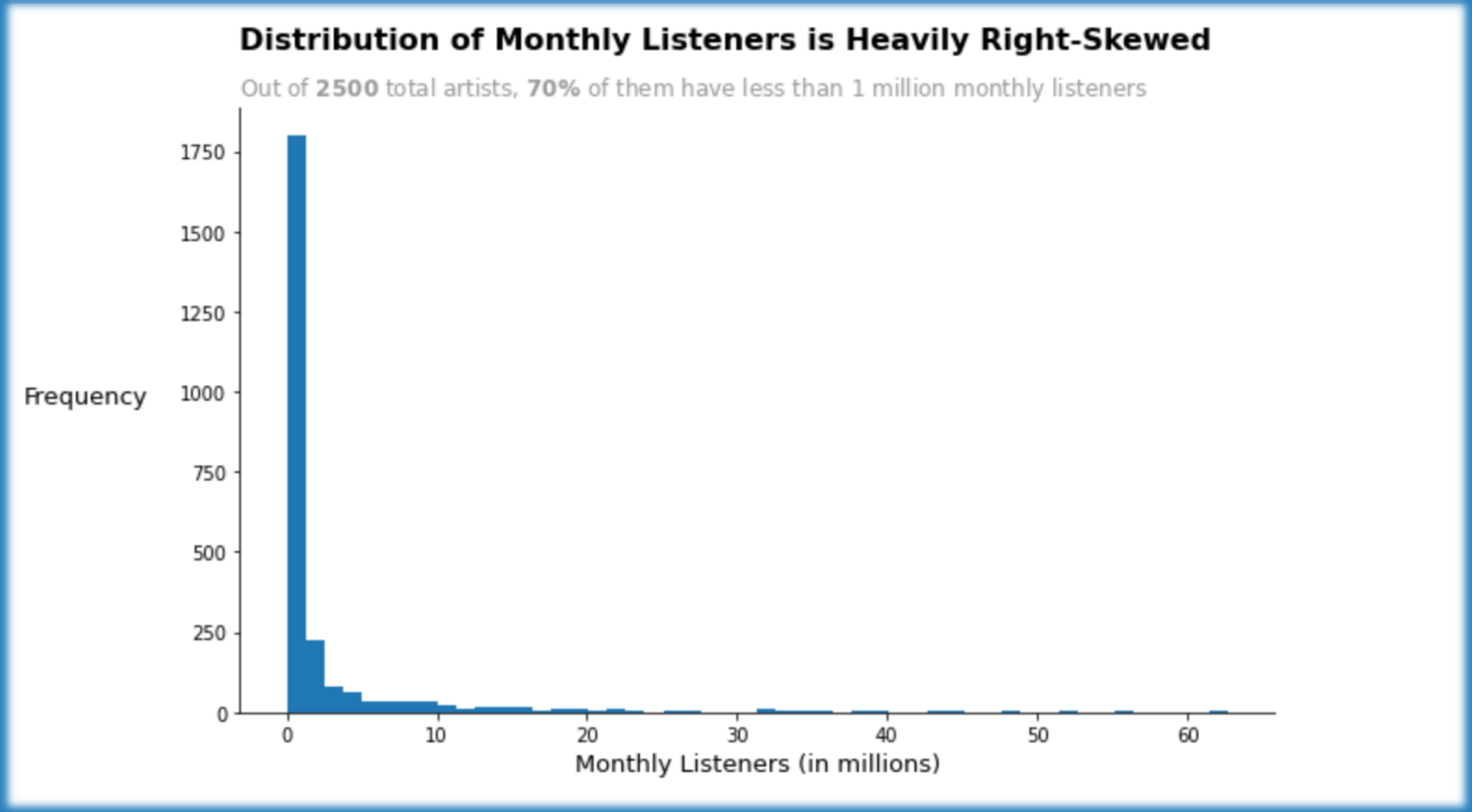
(Mean)

.642

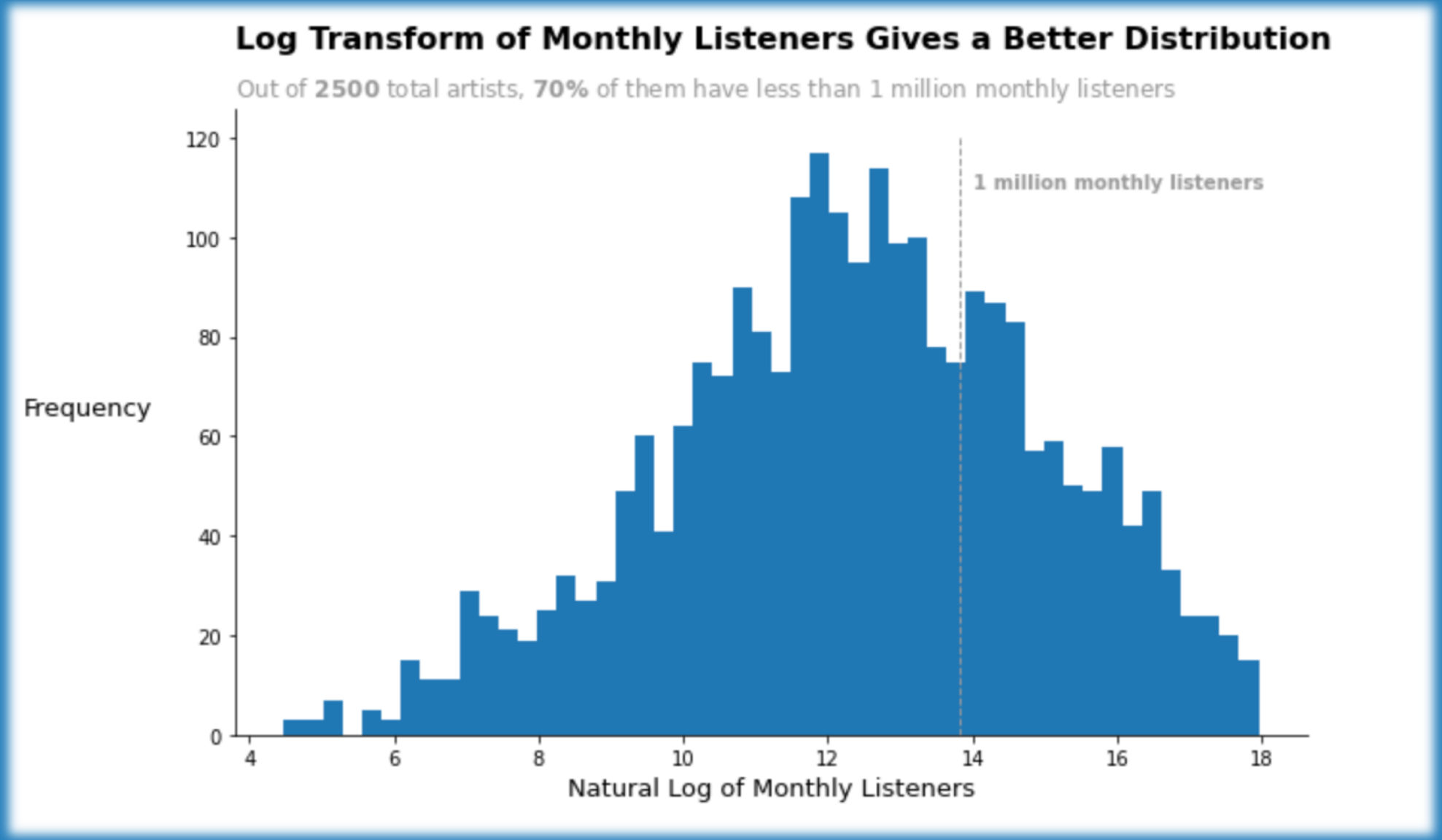
DATA PREPARATION / FEATURE SELECTION



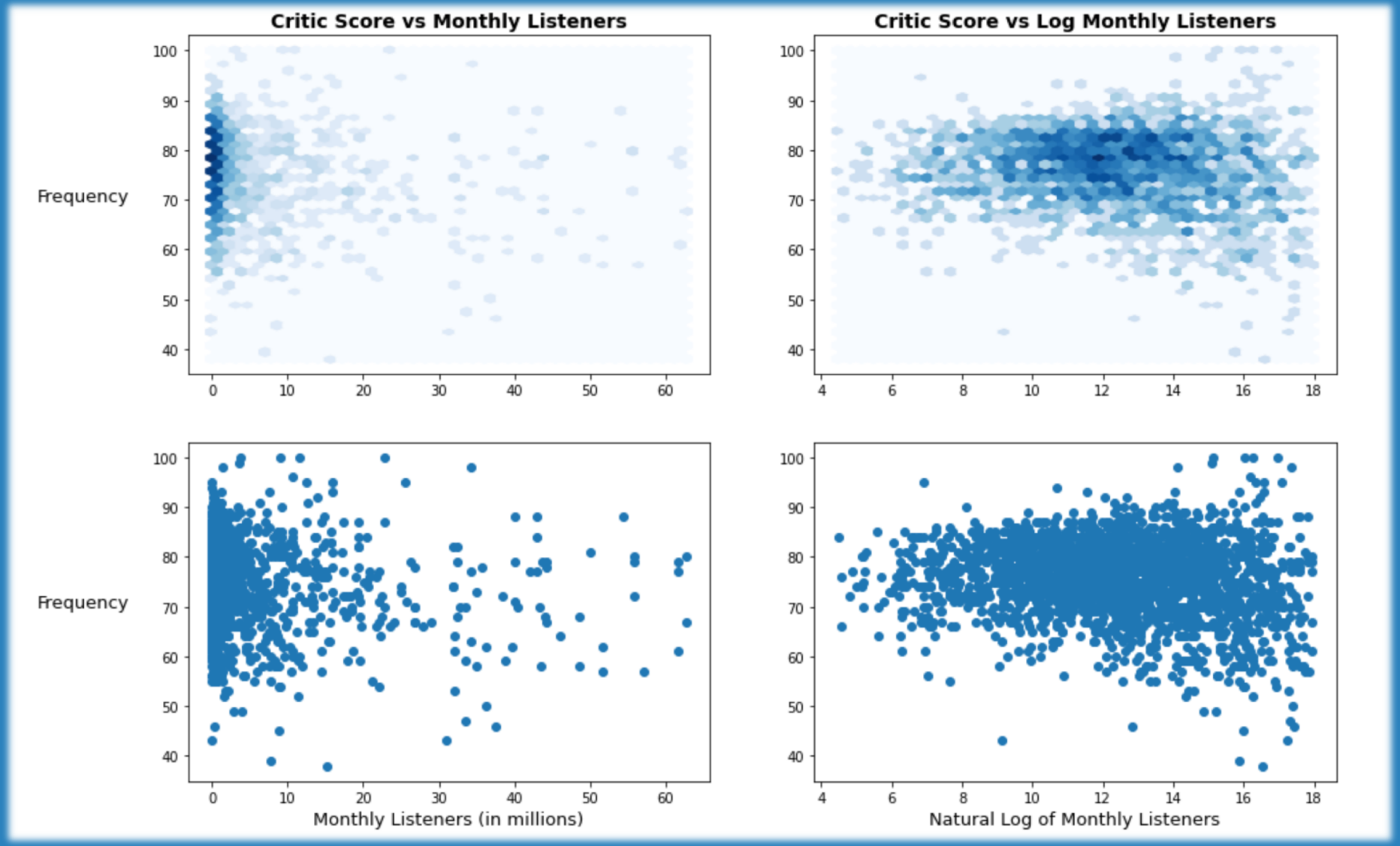
DATA PREPARATION / FEATURE SELECTION



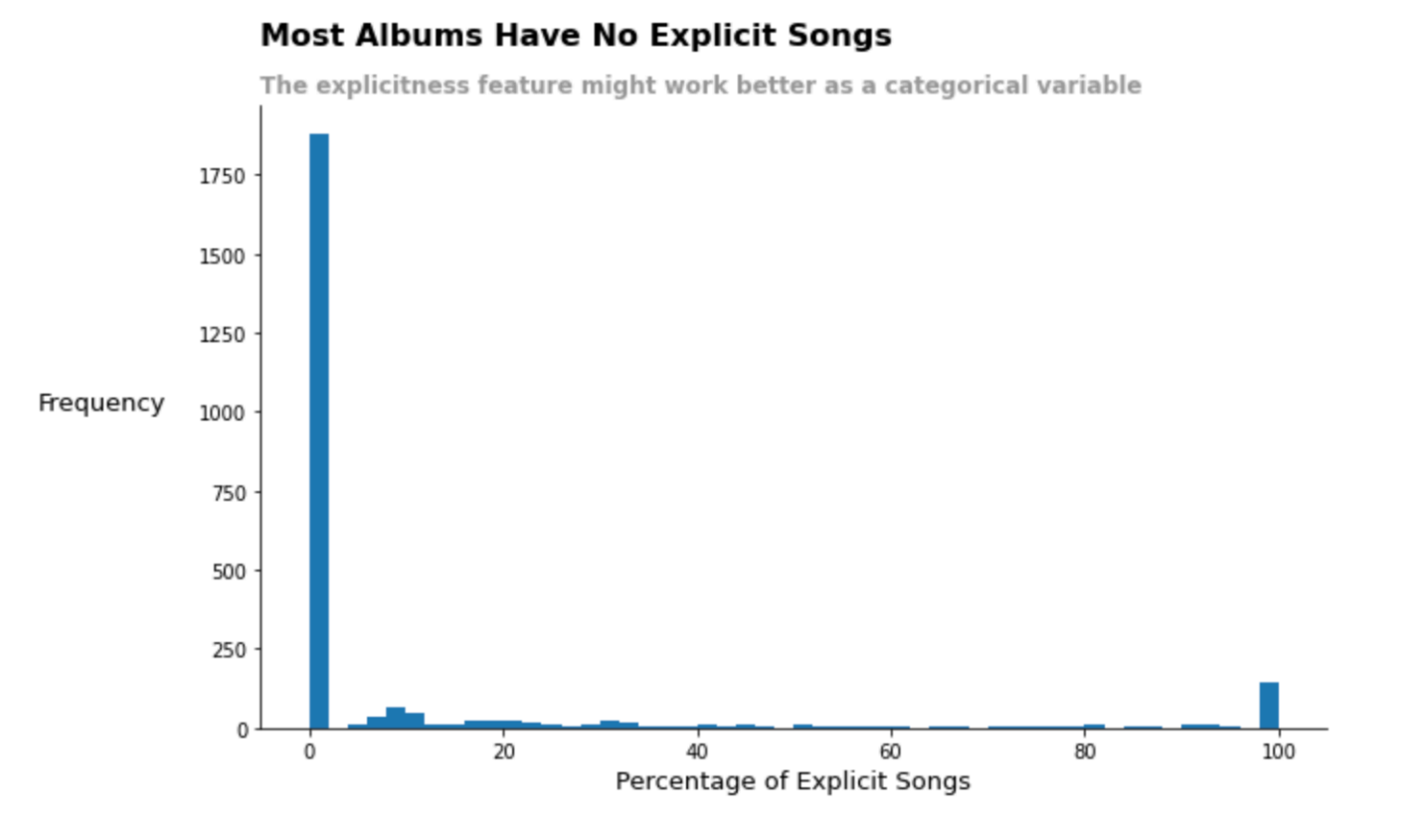
DATA PREPARATION / FEATURE SELECTION



DATA PREPARATION / FEATURE SELECTION



DATA PREPARATION / FEATURE SELECTION

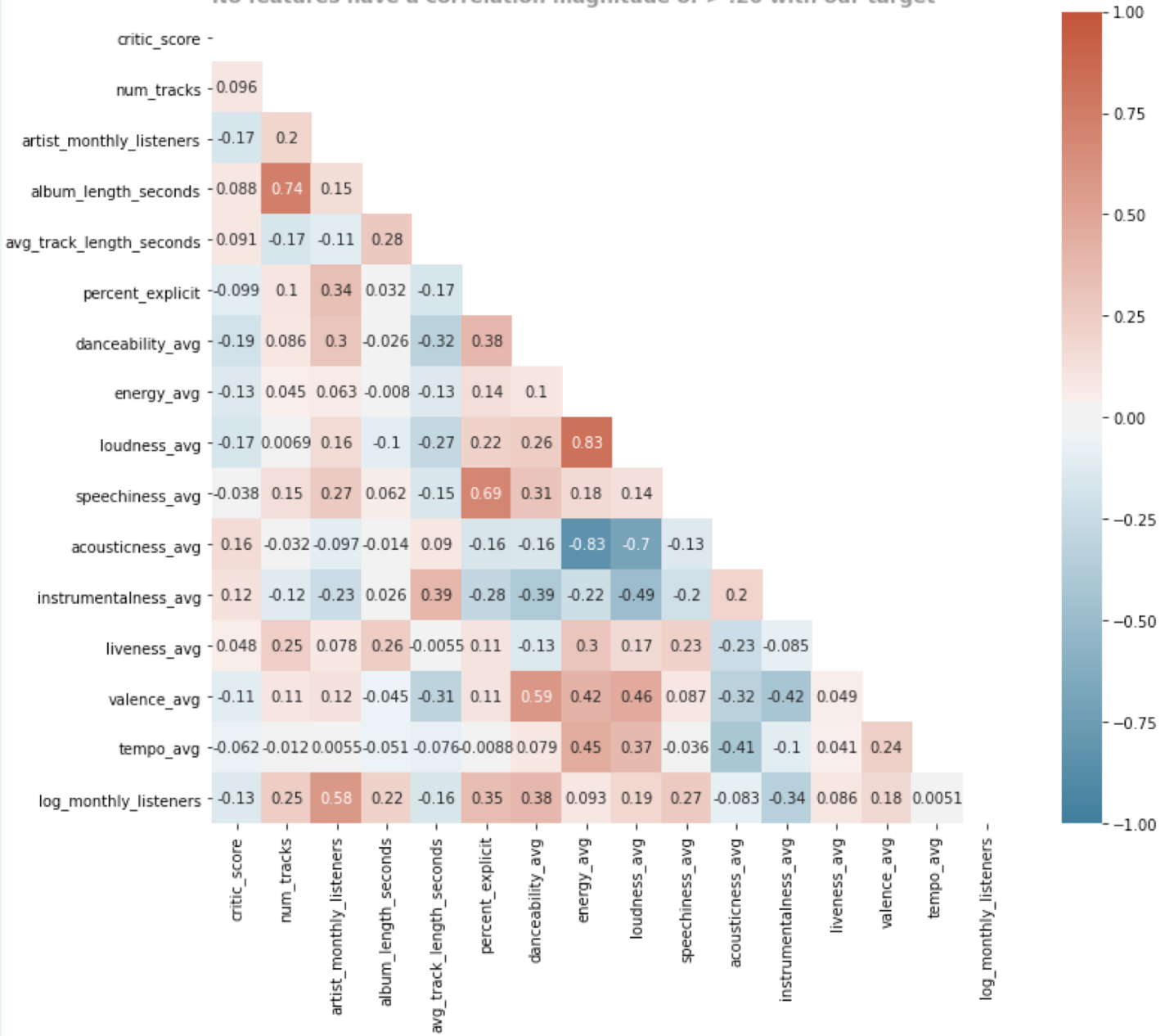


Three categories:

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

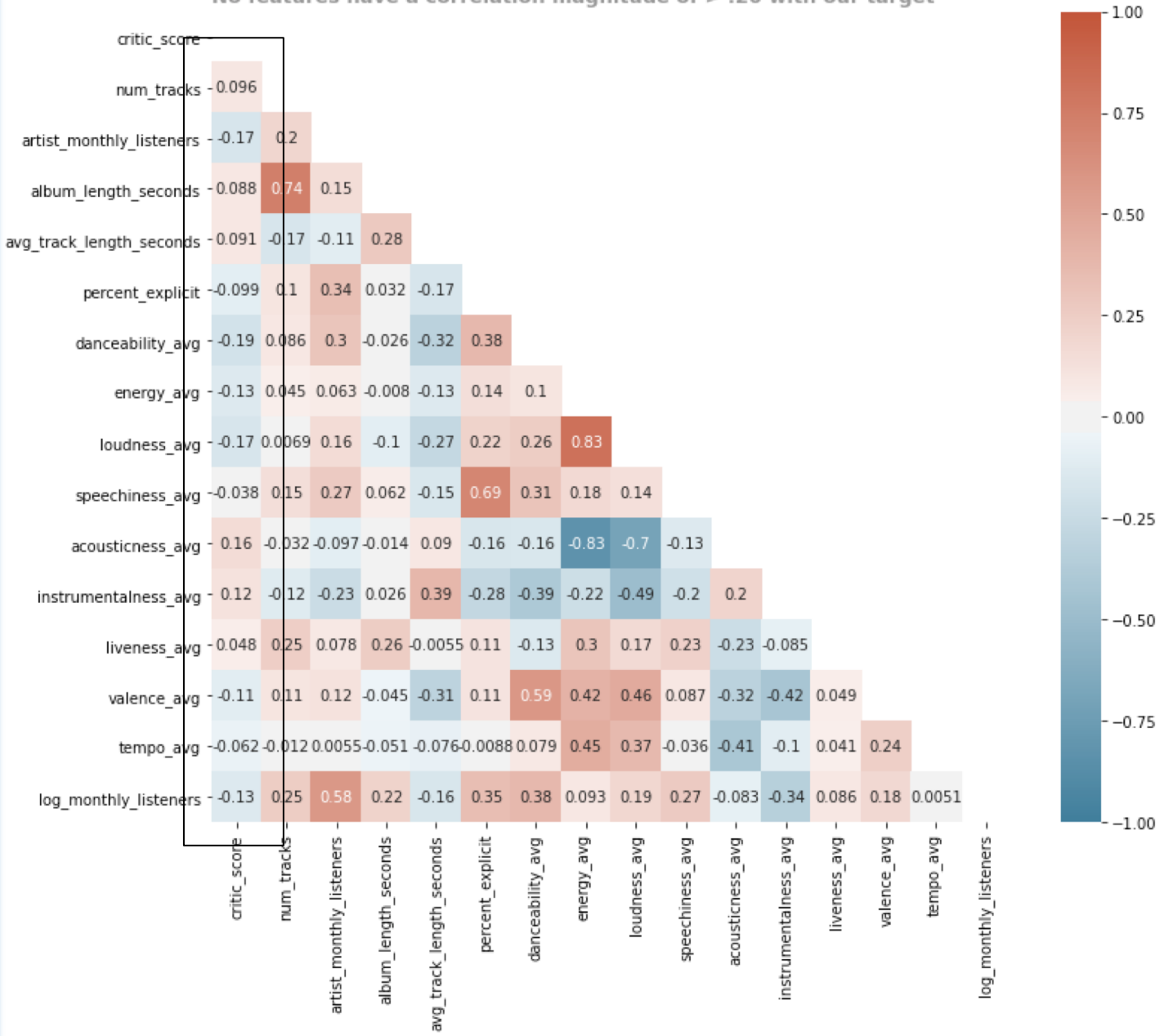
Feature Correlation Matrix

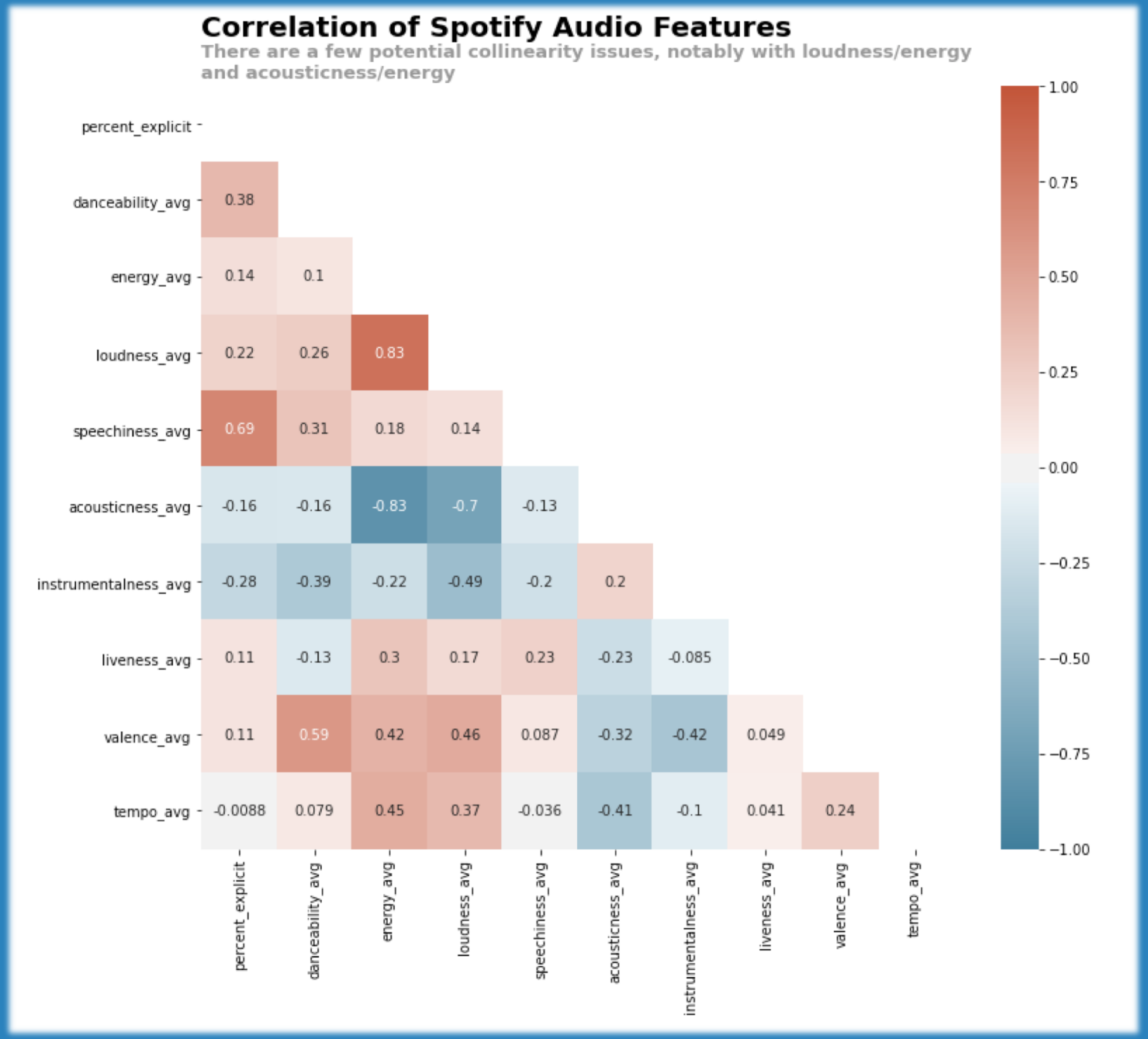
No features have a correlation magnitude of > .20 with our target



Feature Correlation Matrix

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





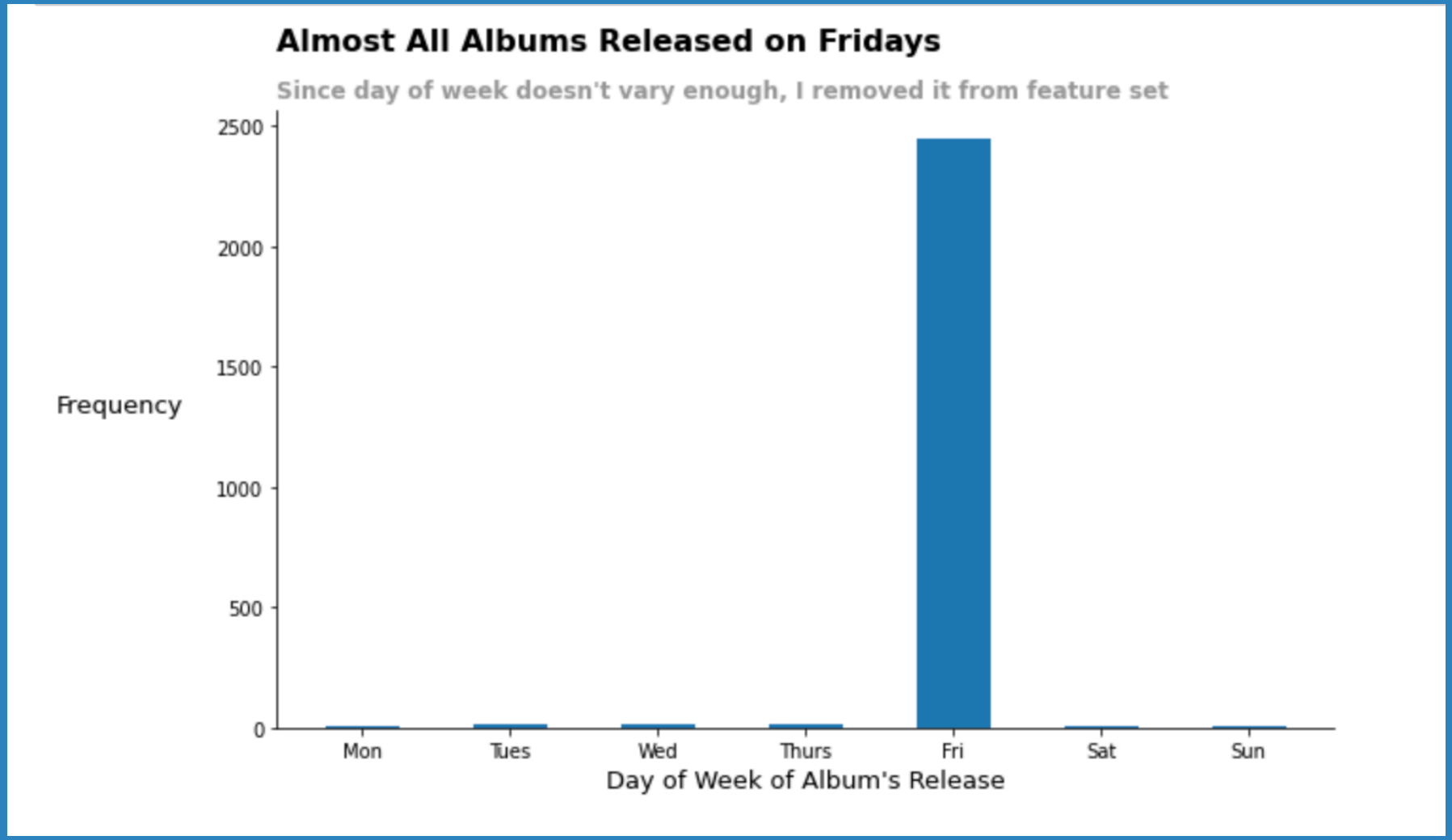
Potential multicollinearity issues:

- 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

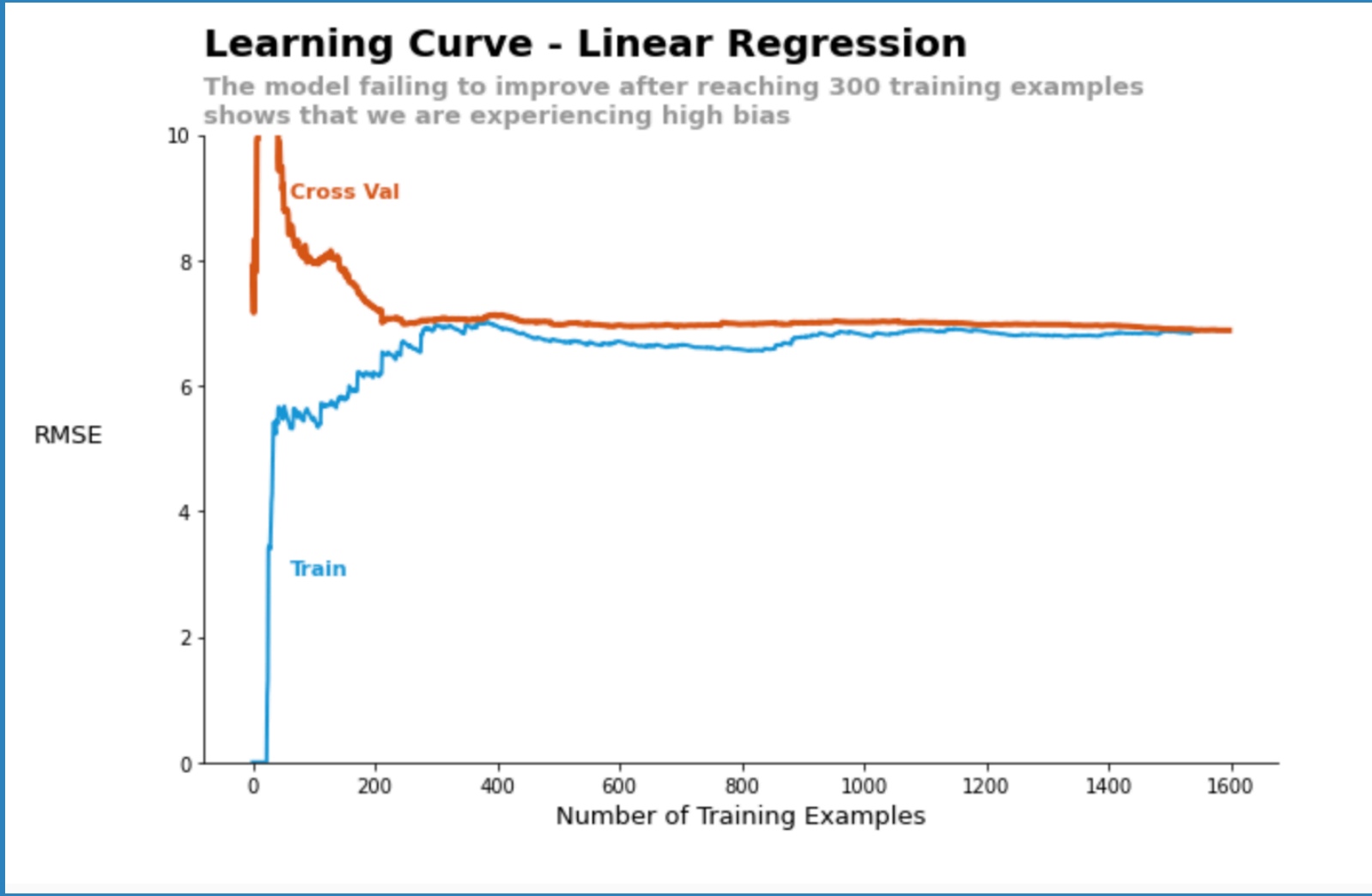
The good:
-No P values above .80 and statsmodels would have warned if there were any strong multicollinearities in the model, so feature selection successfully eliminated that obstacle

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

```
def lin_reg_scores(x, y):
    lin_reg = LinearRegression()
    lin_reg_fit = lin_reg.fit(x, y)
    print('Training set R^2:', lin_reg_fit.score(x, y))
    print('Cross Val R^2:', cross_val_score(lin_reg, x, y, cv=5))

lin_reg_scores(X, y)
```

```
Training set R^2: 0.10798623425505316
Cross Val R^2: [0.06088725 0.08710115 0.08157277 0.08573311 0.09141628]
```

MODEL VALIDATION / ITERATION

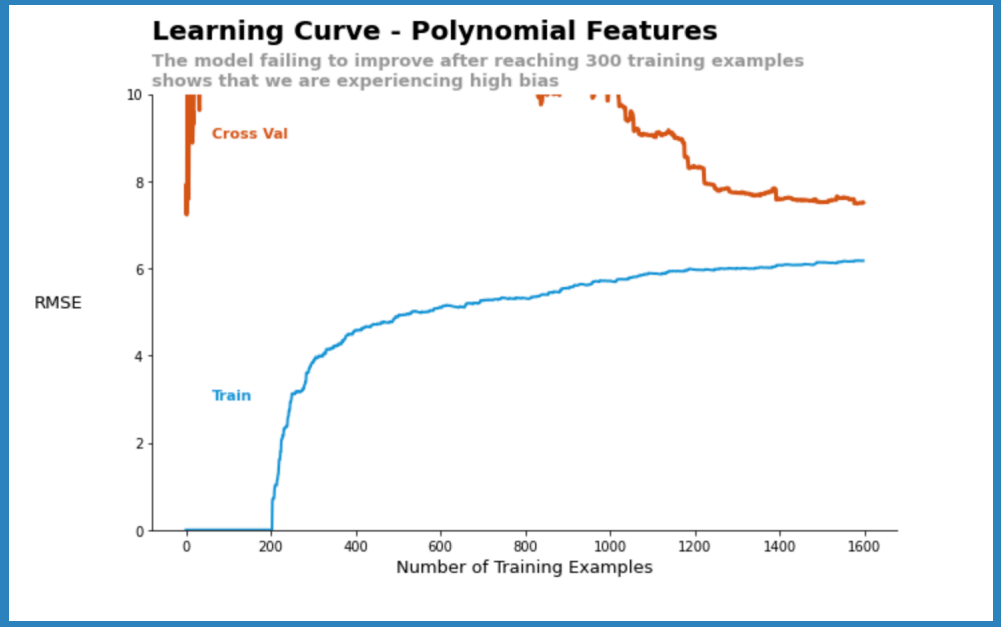


MODEL VALIDATION / ITERATION

In [291]:
```
def lin_reg_scores(x, y):
    lin_reg = LinearRegression()
    lin_reg_fit = lin_reg.fit(x, y)
    print('Training set R^2:', lin_reg_fit.score(x, y))
    print('Cross Val R^2:', cross_val_score(lin_reg, x, y, cv=5))

lin_reg_scores(X_poly, y)
```

```
Training set R^2: 0.25243348821944334
Cross Val R^2: [-0.14758094 -0.04306689 -0.11583694 -0.24737058 -0.05747164]
```

Polynomial features & Interaction features: Overfitting



MODEL VALIDATION / ITERATION

Lasso CV: 0.15 chosen for hyperparameter

```
In [300]: lasso_CV_score(X_poly, y.values.reshape(-1, ), alphas = [.1, 1, 10, 100], coefs=True)

          Best alpha:  0.1
          # of coefs reduced to 0:  260
          # of nonzero coefs:  21
```

```
In [303]: alphas = [0.01, 0.02, 0.04, 0.08, 0.1, .15, .18, .2, .25, .3, .4]
          lasso_CV_score(X_poly, y.values.reshape(-1, ), alphas = alphas, coefs=True)

          Best alpha:  0.15
          # of coefs reduced to 0:  273
          # of nonzero coefs:  13
```

MODEL VALIDATION / ITERATION

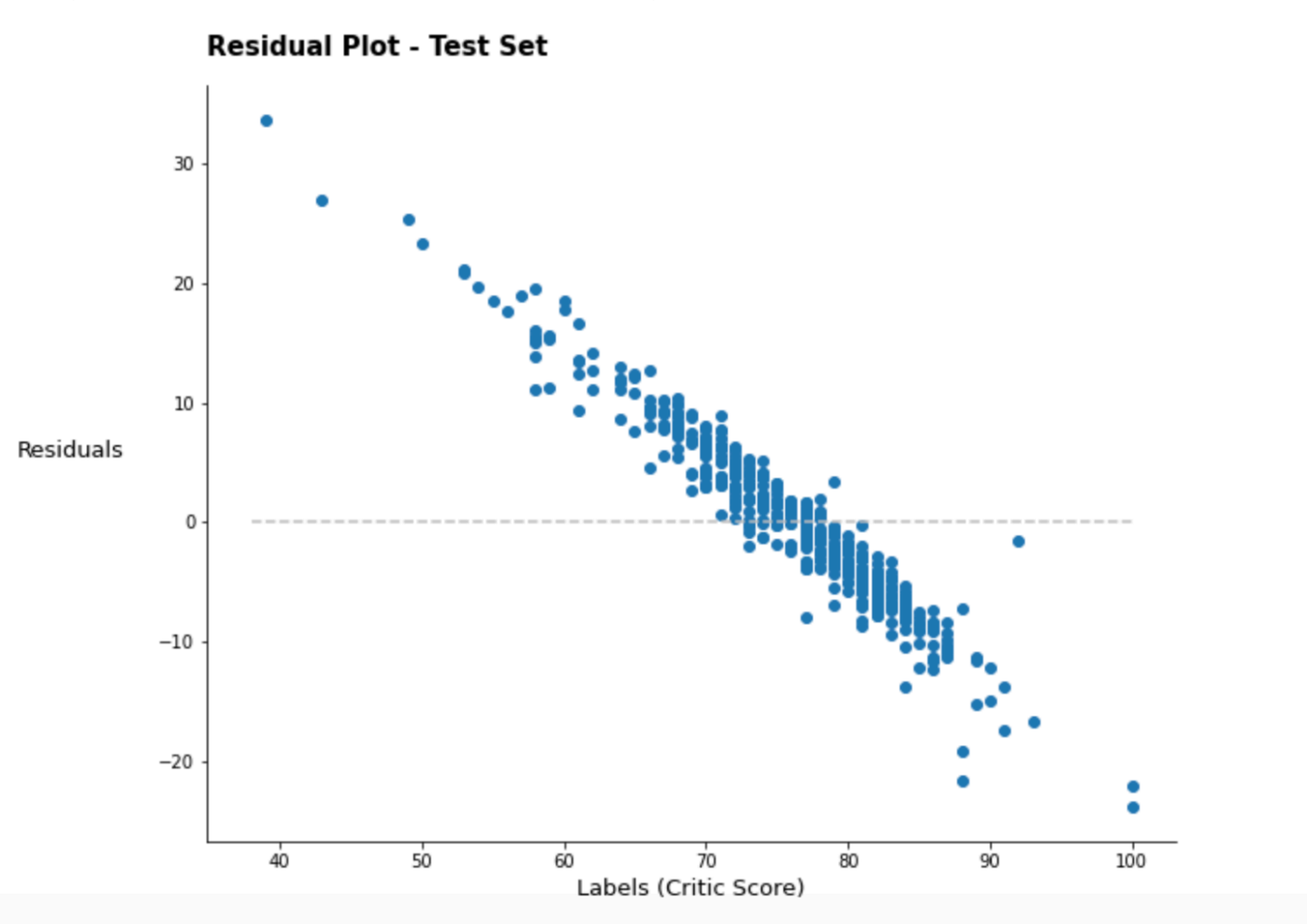
In [306]:
```
lasso = Lasso(alpha=0.2)
lasso.fit(x_train, y_train)

# score fit model on validation data
val_score = lasso.score(x_val, y_val)
# score train data
train_score = lasso.score(x_train, y_train)


print('Training set R^2:', train_score)
print('Cross Val R^2:', val_score)
```

```
Training set R^2: 0.11860450285120405
Cross Val R^2: 0.11722220606462475
```

```
test_score = lasso.score(X_test, y_test)
test_score
```

```
0.09192206473238718
```

MODEL VALIDATION / ITERATION



CONCLUSION / 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)