Is *Pitchfork* Out of Pitch? Investigating Bias in *Pitchfork* Album Reviews

Ayush Pandit, Kimberly Batdorf, Colin Kalicki

Github Repo: https://github.com/apandit42/fork-you-pitchfork

ABSTRACT

The online music publication *Pitchfork* has gained infamy in recent years due to criticism from fans and artists alike who have accused the publication of being biased and unfair in its music reviews. *Pitchfork* however continues to highlight its reviews as being original, authoritative, and unbiased. Putting these claims to the test, we built a database of all published Pitchfork album reviews alongside audio metadata and song lyrics. Then applying machine learning techniques to investigate to what extent bias in *Pitchfork*'s review scores could be identified, we attempted to predict scores from audio and lyric metadata. We found that though there were some general trends visible in *Pitchfork*'s album reviews, overall the variability of album reviews, number of new artists reviewed with little available data, and subjectivity within review scores and our own audio metadata features suggest that *Pitchfork*'s album reviews are not broadly biased, but highly subjective.

INTRODUCTION

Over the years, the online music magazine *Pitchfork* has sparked repeated controversy over its critical reviews and reception of many artists and albums (Xie) (Shaer). Despite the criticisms Pitchfork has received over accusations of bias and favoritism in its review process, Pitchfork continues to claim to be "the most trusted voice in music" (Pitchfork). As musicians and fans ourselves, we began wondering whether *Pitchfork* retained relevance or validity in judging the quality of the modern music landscape, and whether *Pitchfork* itself had any clear biases in its album reviews.

Motivated by trying to identify any biases present in *Pitchfork*'s album reviews, we decided to collect all available *Pitchfork* album reviews and metadata for each album and its songs. We obtained audio metadata from Spotify and lyrics data from Genius, and created different sets of feature matrices with different combinations of the available data. We then planned to use these feature matrices into various different machine learning models to predict *Pitchfork* review scores. By observing differences in model performance based on available features, we hoped to identify any biases or preferences *Pitchfork* might place in certain albums, artists, or songs over others.

RELATED WORK

Previous related work includes the report "Using Twitter to Predict Chart Position for Songs" from the University of the West of England. This report utilizes machine learning models to predict how a track performs on charts given what Twitter users have to say about the song and artist on the social media platform (Tsiara E, Tjortjis C) Additional work includes the report "P4KxSpotify: A Dataset of Pitchfork Music Reviews and Spotify Musical Features" by the University of Colorado, Boulder. In this report, *Pitchfork* scores are directly compared to an album's Spotify characteristics and correlations between the data are outlined (Pinter A, Paul J, et al.). Similar to how the University of West England investigated the impact of public discourse on a track's performance, our investigation aims to analyze the impact of Spotify musical features and Genius lyrical data on an album's performance when being reviewed by

Pitchfork. Although our own exploratory data analysis aligns well with what was reported by the previous work mentioned above, we believe our combination of Spotify audio metadata with Genius lyrics as well as our application of machine learning techniques directly to predict *Pitchfork* scores are a novel addition to the existing work in this space, and have the potential to reveal deeper insights into potential biases or non-biases that *Pitchfork* holds overall.

DATASET*:* (Kimberly, Ayush, Colin)

We scraped *Pitchfork*'s website to build a dataset corresponding to each one of *Pitchfork*'s album reviews, grabbing the album name, artist name, year, and score associated with each review. Then, using Spotify's API, we searched for each album in Spotify's database in order to obtain information concerning an album's musical metadata and listening statistics. We initially found 23,524 album reviews through *Pitchfork*'s site and were able to find 16,442 of those albums on Spotify. We found that *Pitchfork* reviews many obscure albums that are unavailable on Spotify.

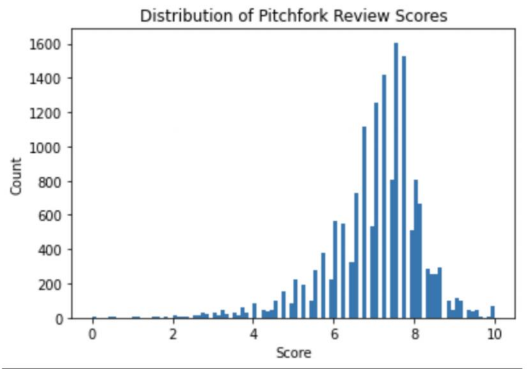


Figure 1

Given the number of instrumental and obscure albums, from the 16,442 albums found on Spotify, we were able to find lyric information through the Genius API for 9,996 albums.

Given the *Pitchfork*, Spotify, and Genius data, we created five different data matrices. The first, referred to as the "aggregated tracks dataset" throughout, includes the audio features and listening statistics from Spotify for each album. When analyzing audio features through the Spotify API though, it is only possible to do so on the level of a single track. For this reason, we applied various calculations such that we can aggregate the audio features for all tracks across an album (i.e. the mean tempo of songs across an album or the percentage of an album written in a certain time signature). In our next dataset, we retained information on the track level, as opposed to simply aggregating this information for each album. We found that 95% of albums have twenty tracks or less. For this reason, we created the "tracks dataset", including twenty-one columns for each audio feature for each album: one for each of the twenty longest tracks on the album and one to represent the average across all remaining tracks on the album, if such tracks exist. We later found though that these two initial datasets happened to be very sparse and included widely varying values. For example, artists had anywhere between 0 and 79 million Spotify followers. With this in mind, we chose to make scaled versions of these two datasets, transforming the data such that every value in the matrix is in between 0 and 100.

Lastly, we created a dataset including lyrical data obtained from the Genius API and combined it with the corresponding data from the scaled tracks dataset. Lyrical data consisted of features such as number of unique words, repetitiveness, sentiment, profanity usage, and more obtained through Natural Language Processing. This dataset served as a subset of the data, including only the 9,996 albums that had lyrical data available through the Genius API.

Throughout our investigation, we reserved 75% of each dataset for training and 25% for testing.

METHODS AND RESULTS

Data Set Selection and Classification (Kimberly)

Although *Pitchfork* assigns real numbers as scores for albums, we began our investigation by first exploring classification. To turn this into a classification problem, we assigned new scores to each album, such that they received a score of 0, 1, or 2 as opposed to any value in between 0.0 and 10.0. These new scores represent the quantile that each album belongs to based on its score: 0 if its score is in the bottom 33.33 percentile, 1 if in the middle 33.33 percentile and 2 if in the top 33.33 percentile. We applied a variety of classification models to each of our five datasets, and noted that the highest accuracy seen was 0.461, as achieved through the use of the Ridge classifier, when applied to the aggregated tracks dataset, as seen in figure 2.

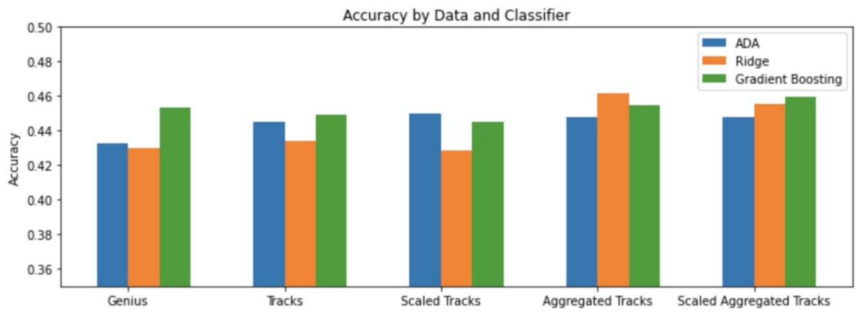


Figure 2

With these results in mind, we next aimed to optimize a classifier through feature analysis. Although when applied to the aggregated tracks dataset the Ridge classifier resulted in the highest accuracy, we noticed that the aggregated track dataset itself seemed to train much more slowly in comparison to the scaled version. For this reason, we chose to optimize the Gradient Boosting classifier when applied to the scaled aggregated tracks dataset. When initially applied before feature optimization, the Gradient Boosting classifier resulted in an accuracy of 0.455, which was quite close to the accuracy of the Ridge classifier on the aggregated track dataset (figure 2).

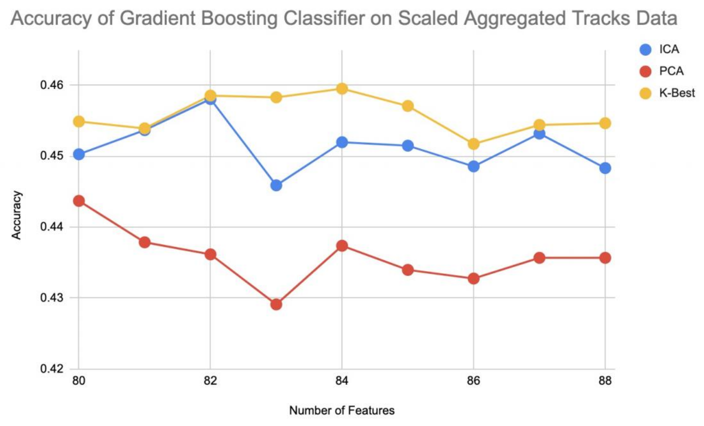


Figure 3

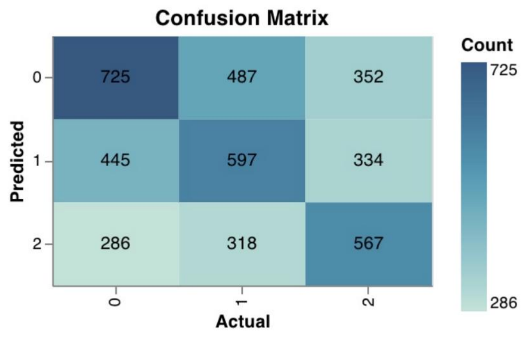
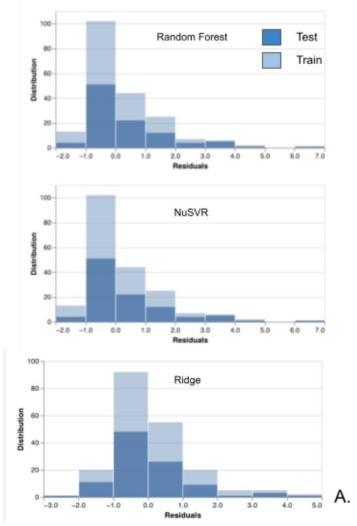
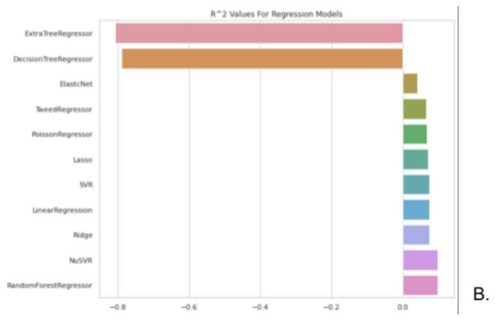


Figure 4
Confusion Matrix of the Gradient Boosting Algorithm with 84 features from the scaled tracks dataset, chosen from the K-Best Features algorithm

To reduce the dimensionality of our matrix and optimize our classifier, we then applied Independent Component Analysis, Principal Component Analysis, and the Select K Best algorithm on the scaled aggregated track dataset. We found that the Gradient Boosting classifier performed best, with an accuracy of 0.459 when the K-Best features algorithm was applied to select 84 features. With this use of the K-Best features algorithm, we found that the following features had been excluded: the standard deviation of the danceability of an album; the percentage of an album that is in key 2, key 4, and key 7; the percentage of an album that is in a major key; and the percentage of an album that is made up of an obscure album (not in the top 10 genres). While we saw an improvement, the accuracy of the Gradient Boosting classifier only increased by 0.004. With this in mind, we then shifted towards regression to investigate correlations between the *Pitchfork*, Spotify, and Genius data.

Regression (Colin)[1]

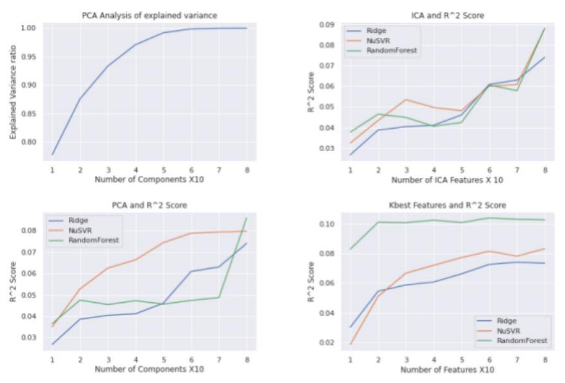




From A) we can see the residual error score distributions with these top performing models when comparing their training and test data. The graphs are all skewed to the left with most error for each model falling in between 0 and -1. The lack of a normal distribution corroborates what we see in B) with the poor performance of our linear regression models that have R^2 scores only slightly above randomly guessing (average of 0.07 if we take out the outliers that are the Decision Tree and Extra Tree models). Both A) and B) suggest the models with our data are not being accurately trained and they are doing only a little more than just guessing. However, in B) we can identify our three best performing models being Ridge, NuSVR, and Random Forest regression models. Their R^2 scores are shown explicitly in Table 1 along with other metrics and models.

Figure 5

Utilizing the Scaled Aggregated Tracks dataset, we aimed to use Regression methods to analyze and attempt to predict the exact Pitchfork scores of albums (0.0 to 10.0). Given that our data shows no precise shape when plotted, it is best to use linear regression models. We identified 11 different models in order to first test the efficacy of these different regression models on our data set. We chose to use R^2 scores as our main metric of identifying model success. R^2 scores are a measurement of how closely the data fits to the desired regression line, with a range of -1 to 1, therefore they are a good metric of model success in predicting data. Figure 5 shows the initial efficacy of our 11 different models. We found that the three best functioning linear regression models were NuSVR, Ridge, and Random Forest; however, they each only had a R^2 score between 0.0744 to 0.100 which suggests that the models are not doing much better than randomly guessing, which is a R^2 score of 0. One area that we had identified as an issue with our dataset was the size and sparsity of it. Therefore, in order to try to improve our data we decided to run Component Analysis and K-Best Features algorithms on the Scaled Aggregated Tracks, as

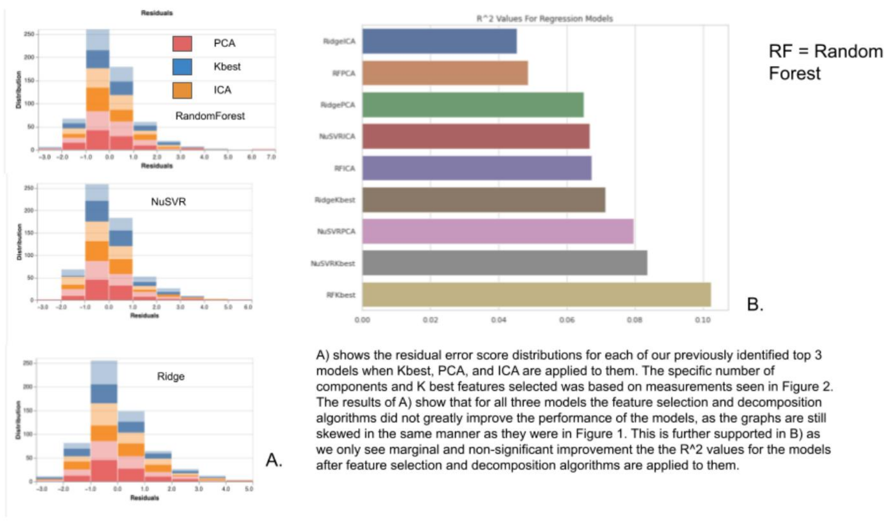


We wanted to identity the proper number of features to select for and number of components to decompose our data in order to maximize R^2 score while minimizing number of features. The three algorithms we identified to do this were Kbest, PCA, and ICA which were run across all models from 10 - 90 features being selected for, at increments of 10. We can see for general explained variance based on number of components, it takes about 50 components to explain 99% of our data variance. For each PCA and ICA we can see that most models (besides NuSVR under PCA) continued to increase to near max number of components in our dataset. We found that around 75 - 80 features for PCA and ICA was ideal to maximize the R^2 score. Below this would not effectively maximize the R^2 score and anything below this would not decompose enough features. Lastly, Kbest reached equilibrium with Random forest at 20 features, however for Ridge and NuSVR there was R^2 score increase until about 60 features. Therefore, we determined that 60 features was the most effective condition to select for when applying Kbest that both maximized R^2 scores of our models while minimizing our data by the greatest amount.

Figure 6

[1] For a better view of the Regression data look here.

seen inFigure 6. Our goal when applying these algorithms was to attempt to find the number of components and K-Best features that could best minimize the data while possibly enhancing performance, or at least give us an insight into features that are more important in our data set than others. We also wanted to see if decomposing our data using ICA would reveal underlying independent weights. Applying ICA, PCA, and K-Best features algorithms to each of our previously identified top 3 models we tested performance, finding the maximized R^2 scores with the minimal amount of components or features to reach that max steady state. We were able to identify that the ideal number of components for ICA and PCA was between 75-80 which is only slightly smaller than our dataset (90 features) while for K best features it was around 60 features that reached maxed R^2 scores steadily for each of the three models. Figure 6 shows how our models (Ridge, NuSVR, and Random tree) performed under these conditions (selecting for the 60 best features and decomposing our models down to 77 features). As figure 2 shows there were only small improvements to our models R^2 scores. After searching for the 60 best features and using a halving search to identify the optimal hyperparameters for our model (n_estimators = 47, max_depth = 6, max_features = auto, min_samples_split = 15, and min_samples_leaf = 13), we were able to achieve a maxR^2score of 0.11 with our ensemble Random Forest method. Furthermore, our three models' residual error distributions hardly changed after applying feature selection and decomposition. Nevertheless, we were only able to improve theR^2score of the Random Forest model by 0.01. This suggests that these models are still only marginally better than randomly guessing where to assign scores via the data. This also shows that, though we can remove up to 30 features from our data and achieve practically the same score, the features that remain do not have strong explanatory power from our data to our labels. Random Forest being the best performing model also makes sense as we are able to hyper tune the model in ways we are unable to do so for other non-ensemble linear regression models. Next, we wanted to see if applying deep learning methods to our data could help us figure out these hidden parameters that attached our features to their score.



A) shows the residual error score distributions for each of our previously identified top 3 models when Kbest, PCA, and ICA are applied to them. The specific number of components and K best features selected was based on measurements seen in Figure 2. The results of A) show that for all three models the feature selection and decomposition algorithms did not greatly improve the performance of the models, as the graphs are still skewed in the same manner as they were in Figure 1. This is further supported in B) as we only see marginal and non-significant improvement the the R^2 values for the models after feature selection and decomposition algorithms are applied to them.

Figure 7

Deep Learning (Ayush)

Having found that both standard classification models as well as regression models were insufficient for achieving high accuracy with the data, we decided to implement custom deep learning models using the

PyTorch framework. Although ultimately we would like to develop an algorithm capable of predicting real number scores, given the challenges that we faced with implementing regression models and the low performance our classification models had, we focused on initially building neural network classifiers to predict whether an album would be rated below average, average, or above average by *Pitchfork*.

Given the low variability of the columns in the aggregated tracks matrix we had designed earlier, we reasoned that a neural network might be able to improve performance over the standard supervised learning algorithms we had implemented by learning more complex interactions between features in the scaled aggregated tracks dataset that the previously attempted methods were not able to capture.

When implementing deep learning, we selected Cross Entropy Loss as our loss function because it works well for multi-class classification tasks. We next chose Stochastic Gradient Descent as our optimizer function because of its robustness for convergence despite the high dimensional and sparse nature of our dataset.

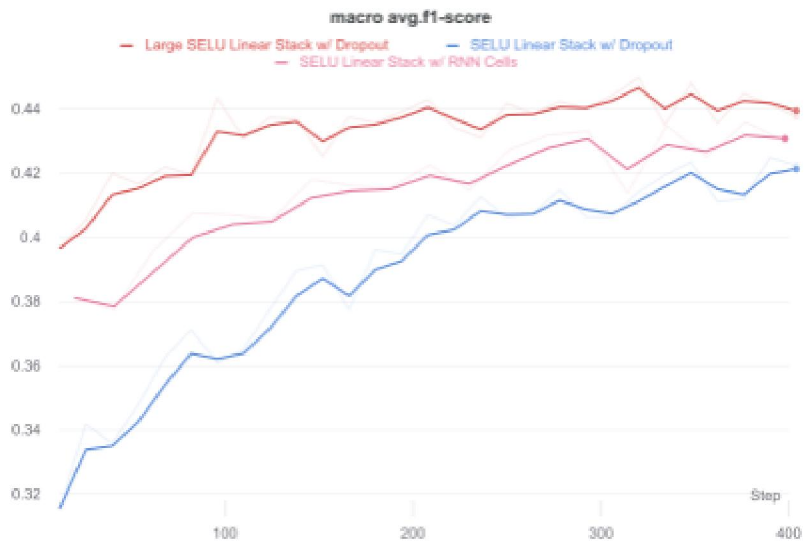


Figure 8: F1 Scores for Top 3 Neural Network Classifiers for Pitchfork Score

Finally, we then implemented several different architectures for our neural networks, and selected the best three and their average F1 scores to be shown in Figure 8. Our third best neural network (in blue in the figure) consisted of 3 linear layers applied with the SELU activation function, which performed more robustly across trials than the standard ReLU for our dataset. In addition to the 3 core linear layers, we also implemented a dropout layer in between each linear layer to improve model robustness and performance, added a 1-dimensional batch normalization layer to regularize inputs through the model, and had a 3 neuron output layer that was converted into class label predictions through a softmax function. Our second best performing neural network had a similar structure to the SELU Linear Stack, except with two additional layers that were recurrent neural network (RNN) cells. Since our dataset featured inputs that were often correlated with each other due to the dates that each review was released, and since we saw that loss values could become unstable across batches, we reasoned having RNN cells could help stabilize our network, and potentially provide additional time-series learning. Finally, our best performing neural network was also very similar to the SELU Linear Stack, but instead featured 6 linear layers activated with the SELU function, and with 2 rounds of batch normalization. This was implemented to help reduce the risk of exploding gradients and overfitting, since batch normalization helps act as a regularization term on the network.

Although we tried many additional models to those listed above, we found that the vast majority of the attempted architectures performed worse than the standard machine learning algorithms. Furthermore, even with the above neural networks, we were only able to match or slightly outperform (~1% accuracy), the best classifier algorithms we'd tested previously.

CONCLUSION & NEXT STEPS

Next steps to improve our data would include finding better methods in quantifying the available data. There were issues with determining best ways to scale or quantify features such as genres or keys, especially in relation to other features and their overall score. Since these scores are seemingly subjective we could take further steps in improving our data sets by adding features about individuals' thoughts about the music. This could take the form of including social media data or specific music reviewer data as they may be able to influence thought across the music review industry and general public opinion at large. Furthermore, although we did implement several deep learning models and try multiple neural network architectures, additional feature engineering around our dataset geared towards optimizing our feature representations for deep learning could help significantly. This could also include additional unstructured data for more complex learning tasks that aren't suitable for classic machine learning algorithms, such as building a network for natural language processing that could integrate direct lyric data into making predictions on what *Pitchfork*'s reception of a song or album might be.

Aside from the subjective nature of rating music, our models may have reached a peak in accuracy and scores due to a lack of available data. We found that many albums did not contain any genre information on Spotify, were missing lyrical data, or were simply not available on Spotify. We predict that these discrepancies between the *Pitchfork*, Spotify, and Genius datasets may have limited the predictive power of our models.

Despite the claims that *Pitchfork* reviews are rigged, we find that audio data, Spotify listening statistics, and lyrical data are not sufficient enough to predict *Pitchfork* scores or otherwise explain any sorts of biases that may exist in *Pitchfork* reviews. We find that *Pitchfork* reviews albums across many different spectrums: popular and obscure genres and artists, long and short albums, instrumental and lyrical, and so much more. We have shown that through classification, regression, and even deep learning neural network models, our predictor is consistently only able to perform slightly better than if it were to simply guess the mean score. We can see this with classification and neural network performances peaking around 0.46 accuracy rates and regression models peaking near 0.1 R^2score. This suggests that *Pitchfork* reviews do not necessarily have any sort of bias towards certain genres, audio features, artists, labels, etc. but instead that they are simply subjective critic reviews.

CITATIONS

Pinter, Anthony T, et al. “P4KxSpotify: A Dataset of Pitchfork Music Reviews and Spotify Musical Features.” *Proceedings of the Fourteenth International AAAI Conference on Web and Social Media*.

Pitchfork. “Pitchfork.” *Pitchfork*, Pitchfork, https://pitchfork.com/. Accessed 2 June 2021.

Shaer, Mathew. “Die Pitchfork Die.” *Slate*, Slate, 28 November 2006, https://slate.com/culture/2006/11/the-indie-music-site-that-everyone-loves-to-hate.html. Accessed 2 June 2021.

Tsiara, Eleana, and Christos Tjortjis. “Using Twitter to Predict Chart Position for Songs.” *Artificial Intelligence Applications and Innovations: 16th IFIP WG 12.5 International Conference, AIAI 2020, Neos Marmaras, Greece, June 5–7, 2020, Proceedings, Part I* vol. 583 62–72. 6 May. 2020, doi:10.1007/978-3-030-49161-1_6

Xie, Teresa. “Why BROCKHAMPTON Hates Pitchfork.” *34th Street*, The Daily Pennsylvanian Inc, 12 September 2019, https://www.34st.com/article/2019/09/brockhampton-pitchfork-dispute-kevin-abstract-reviews. Accessed 2 June 2021.