Predicting Pitchfork Album Review Scores

Introduction:

Many record labels may be seeking new ways to maximize critical acclaim of their albums. Before making any business decisions, it would be helpful to know what factors may contribute to an album's critical success. Knowing these factors would enable a record label to make more intelligent decisions when releasing an album, but what exactly are these factors? Using machine learning techniques with album review data retrieved from Pitchfork's website, I aim to answer this question. I will also build a machine learning model that can be used to automatically detect sentiment in music reviews.

Client Profile:

My primary client for this project will be record labels. If record labels can gain insight into what factors influence a record's success, they will be able to make more intelligent business decisions. Furthermore, predicting the sentiment of a review could be applied to social media to ascertain how well a new album has been received by the community and could help inform marketing and distribution decisions. Another potential client could be artists. If musical artists know what factors influence whether an album is received positively, they will be able to make adjustments to their music accordingly.

Section 1: Data Wrangling

Using Beautiful Soup, I scraped music reviews from the Pitchfork website. First gathering all the URLs, and then scraping each URL - generating returned a dataframe of various features such as artist, genre, album title, etc. A sample output from this function can be found in figure 1.1.

Out[3]:

	author	artist	title	genre	date	review_abstract	score
0	by: Nina Corcoran	Gulfer	Gulfer	Rock	October 15 2020	The Montreal band grows up on their third albu...	7.4
1	by: Sophie Kemp	Future Islands	As Long As You Are	Rock	October 14 2020	The Baltimore band goes back to the well for a...	6.1
2	by: Andy Beta	Moor Mother	Circuit City	Experimental	October 14 2020	In her debut theatrical work, the Philadelphia...	7.4

Figure 1.1: Sample output from "scrape_list_page" function

Using this function, I then scraped all album reviews dating back to 2010. The resulting dataframe featured 11,760 reviews.

After this dataframe was retrieved, I performed the following cleaning/modifying steps:

- Removed “by: “ from the author column using a regular expression.
- Dropped duplicates which reduced size of dataframe to 10,568
- Created “target” column that grouped by score. These score groups included
    - “low”: < 6.5
    - “medium”: >= 6.5 and < 8
    - “high”: > 8
- Changed “date” column to datetime and extracted the month and year to create “month” and “year” columns.
- Created binary columns:
    - EP
    - self-titled
    - sophomore
    - debut
    - high

Section 2: Exploratory Data Analysis

Figure 2.1 shows the frequency distribution of reviews by each target category. Reviews in the “medium” category (6.5-8/10) appear to be the most frequent and reviews in the “high” (8+/10) and “low” (<6.5/10) categories appear to be roughly equal in frequency.

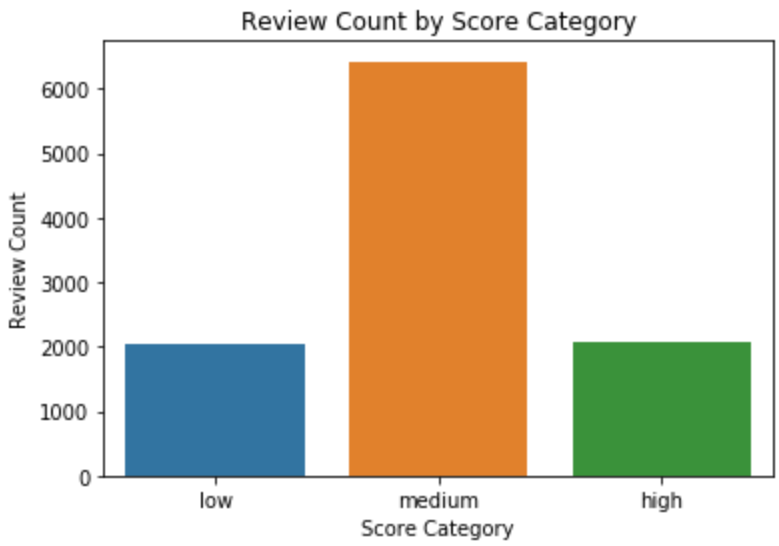


Figure 1.1: Review Count by Score Category Histogram

Figure 2.2 shows the comparison of mean word count between the mean word counts of albums in the “high” category and albums not in the “high” category. The average word count of review abstracts albums in the “high” category appears to be higher than those that are not. A one-way ANOVA test returned a p-value of < .05 which means that there is sufficient evidence to reject the null hypothesis that there is no significant difference in word count between score categories.

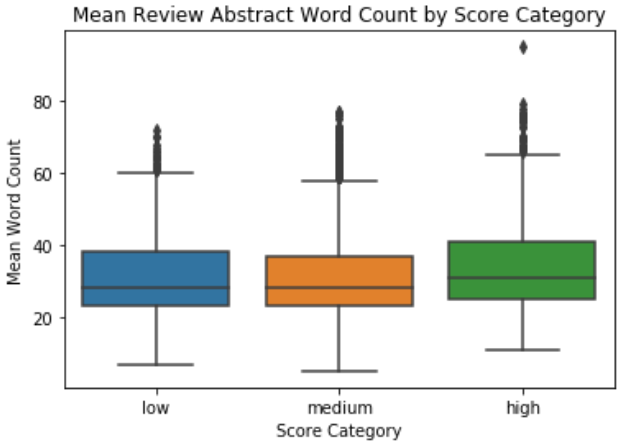


Figure 2.2: Mean Review Abstract Word Count by Score Category

Figure 2.3 compares the mean reading ease of high and not highly rated albums. These reading ease scores were obtained using the Flesch reading ease test. A higher number would indicate an easier to read album review abstract and vice versa. The average reading ease of both

categories appear to be roughly the same. A one-way ANOVA test returned a p-value of < .05 which means there is sufficient evidence to reject the null hypothesis that there is no significant difference in reading ease between score categories.

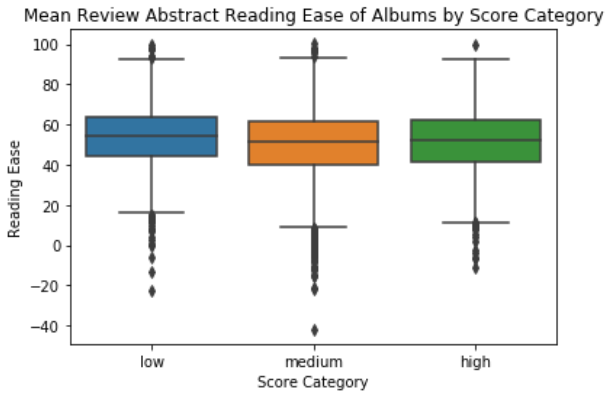


Figure 1.3: Mean Reading Ease of Review Abstract by Score Category

Figure 2.4 is a boxplot that compares scores for each reading grade level. These grade levels were obtained using the Flesh-Kincaid grade level test. The higher the grade level, the more advanced the text is. For grade levels 4-20, the mean score appears to be around 7.5. An interesting outlier that stands out is grade level 0, which has a mean score significantly greater than all other categories. Only two reviews were written at a 0 grade level; one having a score of 9.0 and the other having a score of 7.2. This would explain the significantly higher mean for 0 than all other grade levels.

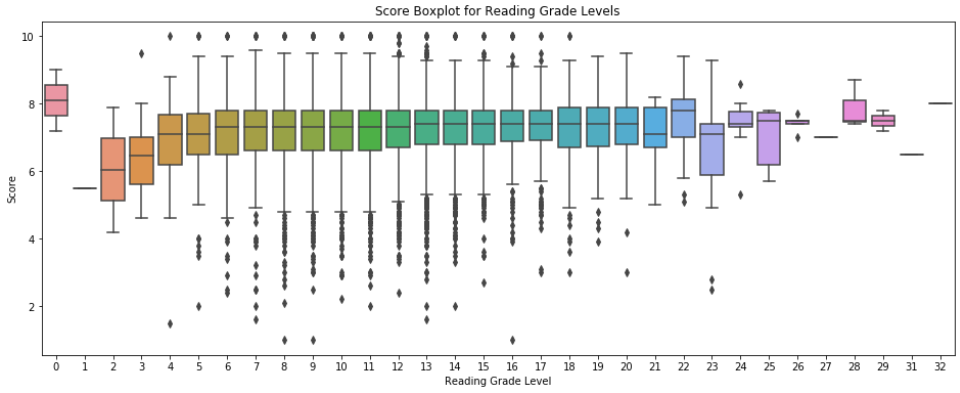


Figure 2.4: Boxplot of Scores by Reading Grade Level.

Figure 2.5 is a bar chart that compares average scores for remastered and non-remastered albums. The average score for remastered albums appears to be significantly higher than that of non-remastered albums, so I performed a chi-square test to check for significance. The test resulted in a p-value of 1.1717078204169938e-26, which means that there is sufficient evidence to reject the null hypothesis that there is not a significant difference in the mean scores of remastered and non-remastered albums.

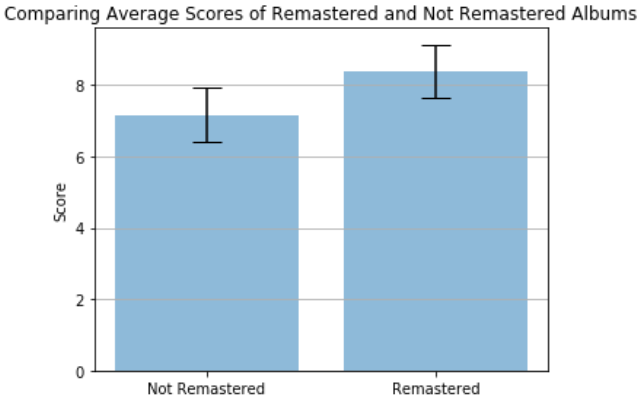


Figure 1.5: Average scores of remastered and non-remastered albums

Figures 2.6- 2.9 are word clouds of predictive words for a high rating for two of the most popular genres: rock and rap. The words in these data clouds were obtained using a Multinomial Naive Bayes classifier algorithm. For each word, the larger it is, the stronger it is as a predictor and vice versa.

One word that I found interesting in Figures 1.6 and 1.7 was ‘Sunday’, so I decided to investigate which reviews the word was used in. Through my investigation, I found that there were many reviews which began with “Each Sunday, Pitchfork takes an in-depth look at a significant album from the past”. The fact that Pitchfork considers these albums to be significant likely explains why review abstracts that contain “Sunday” tended to score more positively.

Figure 2.6: Predictors of High Rock Ratings:

Figure 2.7: Predictors of High Rap Ratings:

The predictors of low rock ratings word cloud presented several words I found interesting, but one that really caught my attention was “supergroup” . Supergroups are bands made up of artists who are each individually famous coming together to collaborate. These groups have a reputation of making music solely for commercial success and not a genuine desire to produce quality music. As a result, the music released by supergroups tends to not be received very well. Of the 15 rock reviews abstracts that contained the word “supergroup”, not a single one obtained a “high” review.

Figure 2.8: Predictors of Low Rock Ratings:

For the word cloud of low rap rating predictors, a word that caught my attention was “features”. In the rap community, there is a notion that rappers who rely heavily on features are trying to compensate for a lack of ability to produce high quality music on their own. This could explain why albums with review abstracts that contain the word “features” tend to not be rated positively.

Figure 2.9: Predictors of Low Rap Ratings:

Section 3: Preprocessing

Before modeling my data, The data needed to be preprocessed. I needed to scale the numeric variables such as “abstract reading ease” and “word count”. To do this, I created a scaler object and fitted it to these numeric variables.

	0	1
0	0.341412	-0.761603
1	0.847959	-0.143450
2	-1.621769	-0.231757
3	1.396926	-0.761603
4	1.249079	-1.291448

Figure 3.2: Output from scaling numeric variables

After scaling these variables, I renamed the columns “reading_ease_scaled” and “word_count_scaled” respectively, then appended them to the dummy variable dataframe.

Next, I needed to vectorize the text of the review abstract column and extract the most important features. In making this text vectorizer, I gridsearched a range of document frequencies between 5 and 30 and found that the optimal frequency was 29. I also set the stop words parameter to English so that words such as “the” and “so” would not be included as well. After successfully extracting these text features, I appended them to the previous dataframe. This final dataframe would feature 10,568 rows and 2,522 columns.

After this dataframe was created, I needed to split it into train/test sets to be run through machine learning algorithms. The target variable I used for the y_train and y_test dataframes was “high”. Once the data had been split into train/test sets, I noticed there was an imbalance in albums that scored high and albums that did not, with high albums outnumbering not-high albums roughly 5-to-1. To address this, I created a SMOTE (Synthetic Minority Oversampling Technique) object which artificially generated new high album data so that the number of high and low albums would be even, with both categories having 5,932 instances.

Section 4: Modeling

For the modeling process, I grid-searched three different classification models with my train/test sets: Gaussian Naive Bayes, Random Forest, and Logistic Regression. The performance results were as follows:

Classifier	**ROC-AUC**	**Best Parameters**
Random Forest	0.698	● n_estimators: 100 ● max_features: sqrt ● criterion: entropy ● max_depth: 50 ● min_samples_split: 6 ● min_samples_leaf: 1
Gaussian Naive Bayes	0.592	● priors: None ● var_smoothing: 1e-09
Logistic Regression	0.711	● penalty: l2 ● C: 5, 10 ● random_state: 0 ● max_iter: 1000

Figure 4.1: Model Performance Table

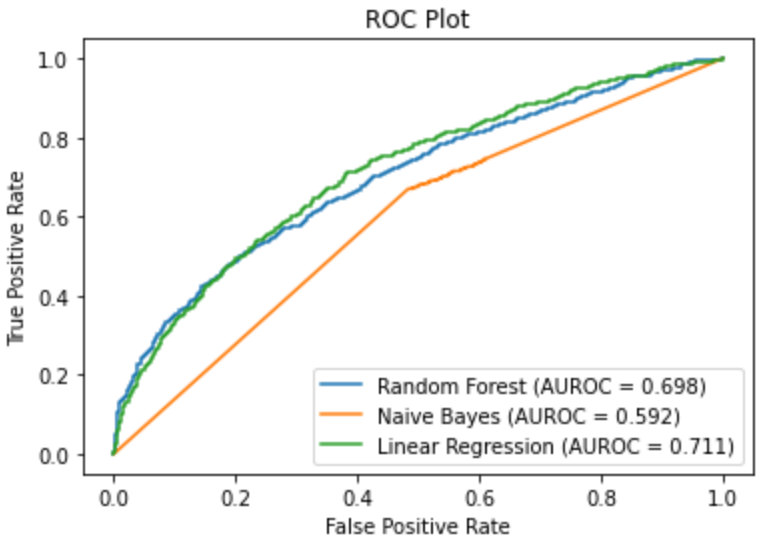


Figure 4.2: ROC Plot

As shown by the model performance table and ROC curve, the Logistic Regression model performed the best based on the AUC score. For analysis purposes, it is also important to know what specific features had the greatest influence in predicting whether an album scored high or low. Random forests allow users to search for what features were the strongest predictors with a function called “feature_importances_”. Executing this function on my random forest classifier yielded the following top ten results:

album	0.016586
year_2017	0.016423
music	0.013827
year_2016	0.012048
word_count_scaled	0.011394
reading_ease_scaled	0.011100
latest	0.010809
record	0.010118
new	0.009916
pop	0.009784

Figure 4.2: Top ten most predictive features

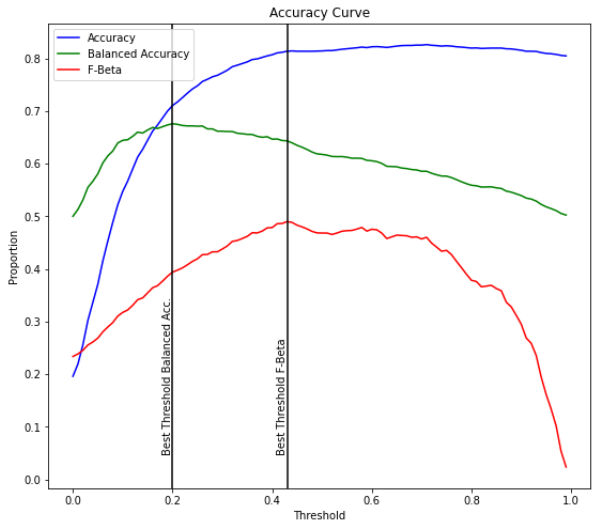
Section 5: Thresholding

There are several different business cases in which it would be better to threshold with certain metrics over others.

Business Case #1: A record label wants to know how well an album was received generally by social media where the sentiment is unlabelled. They could use our model to predict sentiment across thousands of tweets mentioning the album, and generate a score for how generally positive the sentiment was. In this case, it would make more sense to threshold for balanced accuracy, to make sure that this score was as representative as possible of the reality of the general reaction to the album - weighting both high and not high reviews as equally important.

Business Case #2: A record label is looking to discover customers that are very happy with the album, that might be super fans they could offer incentives to to promote the artist's music. In this case, since the label would need to incur an up front cost for each potential influencer, they would want to be sure that they only chose people they could be relatively certain actually liked the album. In this case, it would make sense to use an F-Beta score which favors precision, to ensure that the sample of predicted fans is pure. I chose a beta of .5, but this could be adjusted depending upon the cost flexibility of the client and number of influencers they desired.

Thresholding using both of these metrics yielded the following results:



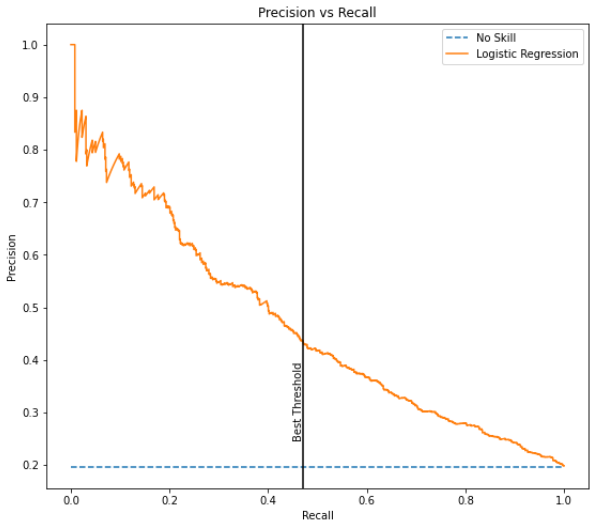


Figure 5.1 (left) Optimal thresholds for balanced accuracy and f-beta.

Figure 5.2 (right): Best threshold on precision vs recall curve.

Section 6: Conclusion

Examining various features can tell you a lot about what factors contribute to a higher rating. For example, of the genres with a minimum of 100 reviews, jazz had the highest average rating of 7.88 while ElectronicRock had the lowest average rating of 6.66. Adding text vectorized features allows for significantly more in depth analysis. When we examine the predictive strength of all features, including those that were obtained using a text vectorizer, we find that “word_count_scaled”, “reading_ease_scaled”, “reissued”, and “remastered” were among the strongest in predicting whether an album obtained a high score.

In using the different variations of this model (one optimized for precision, the other for balanced accuracy), the client could perform various tasks such as monitoring the perception of an album via tweets or choose influencers to promote the album.

Section 7: Future work

- Extract data from music review sources other than Pitchfork and see if similar patterns emerge.
- Conduct a time series analysis to gain insight on how perceptions of different genres have changed over time.
- Investigate whether groups/bands perform better than solo artists or vice versa.
- Find out whether certain reviewers are more likely to give higher scores than others.