Pitchfork Album Review Analysis

By: Jack Molesworth

Introduction

- Record labels are seeking new ways to improve ratings of their albums.
- If record labels know what factors influence, they will be able to make more informed decisions of what artists to sign and types of music to release.
- Using review data obtained from from Pitchfork.com, I aim to identify what these factors are.

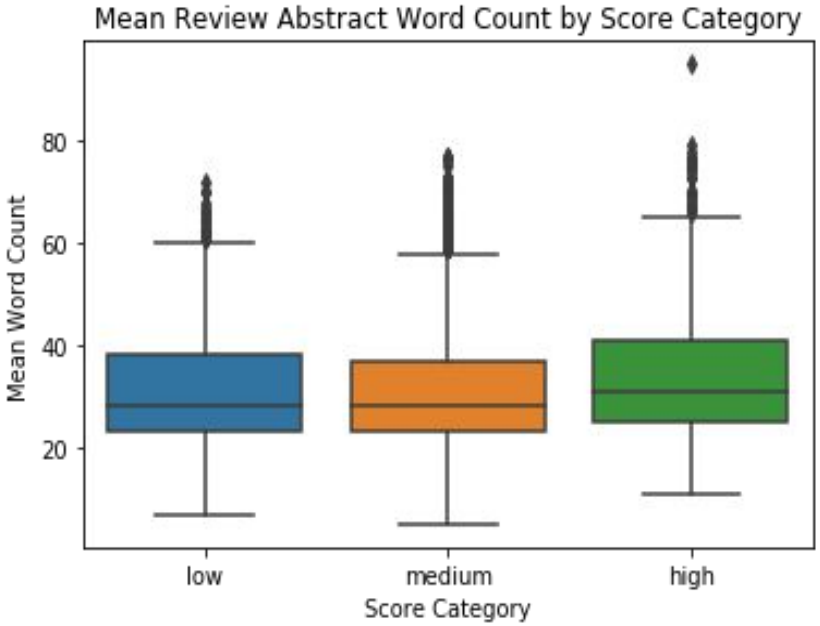
Data Wrangling

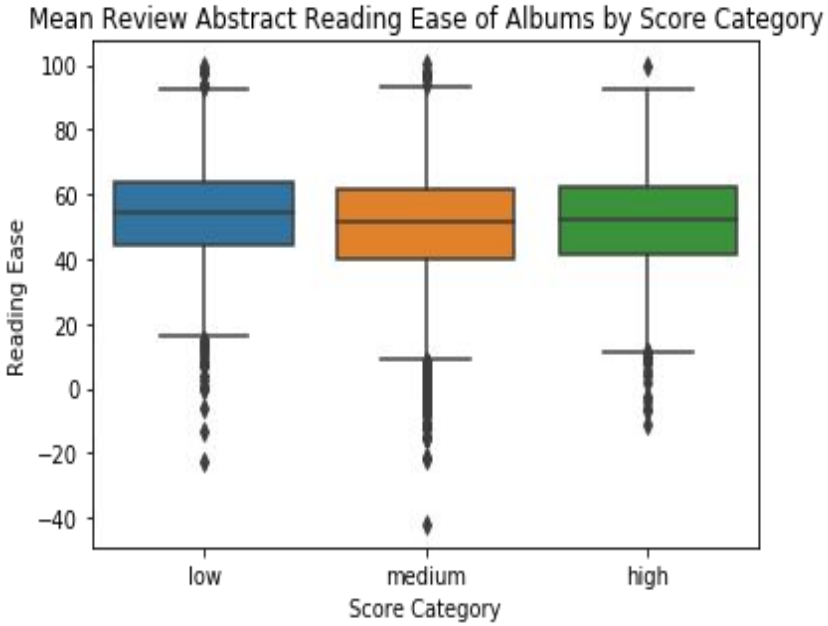
- To begin the data wrangling process, I created a function in Python which extracts variables from the “Albums” section of Pitchfork’s website using the packages BeautifulSoup and requests.
- These variables included artist, album name, reviewer, date of release, review abstract, and more.
- I then created another function which would repeat the previous function until album data from 2010 to present was retrieved and appended into a dataframe.
- This final dataframe featured over 10,000 reviews.

Data Wrangling

- Data wrangling involved the following tasks:
  - Removed duplicate reviews
  - Created “month” and “year” columns from the date column
  - Removed “By: “ from the “artist” column using a regular expression
  - Dropped NA rows
  - Creating a target column based on scores:
    - Low: < 6.5
    - Medium: >= 6.5 and < 8.0
    - High: >= 8.0
  - Generated binarized “self-titled”, “EP”, and “sophomore” columns based on text in the review abstracts

Exploratory Data Analysis





Exploratory Data analysis

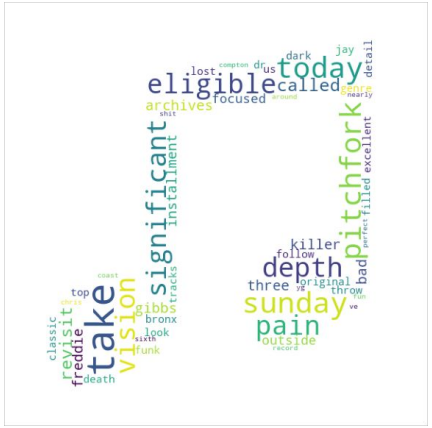
- Predictive features - Words
  - I examined the best and worst predictive words per genre

Genre	**Most Predictive Words**	**Least Predictive Words**
Rock	1994, eligible, archives	fi, rockers, trio
Electronic	become, rhythmic, mysterious	hard, guests, native
Rap	excellent, Kendrick, killer	French, Nikki, Chris
Pop/R&B	yet, minute, she	whose, heavy, often

Genre Word Clouds

Electronic

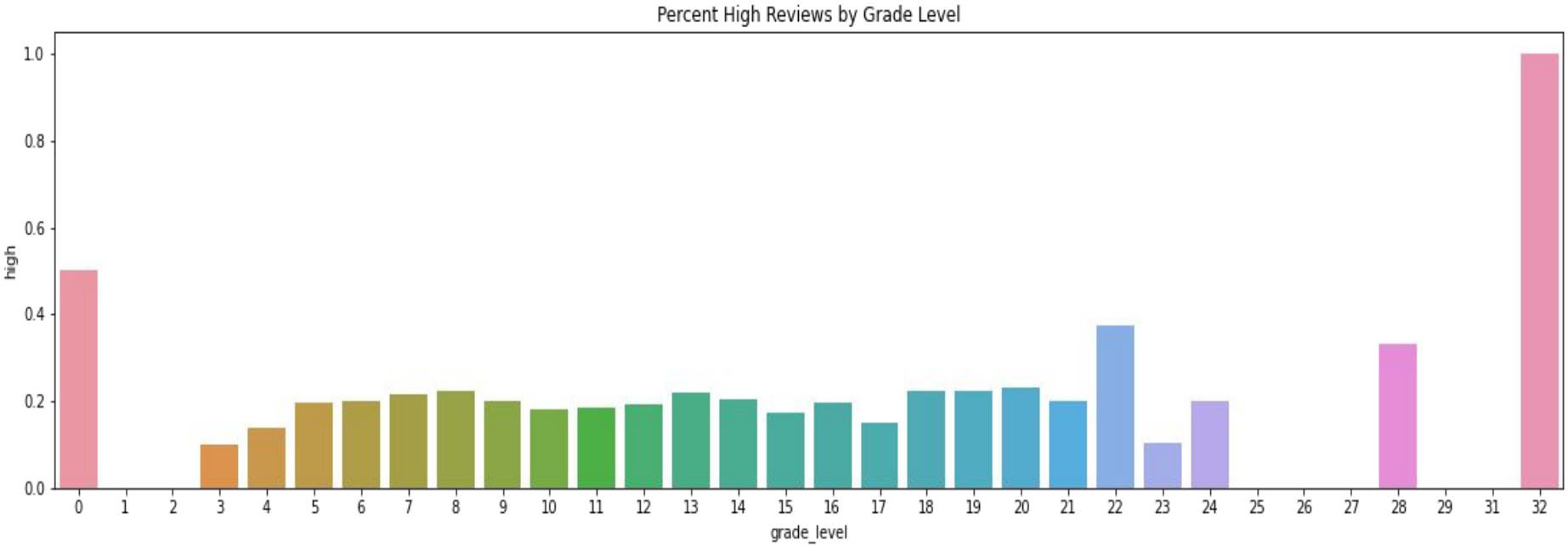
Rock



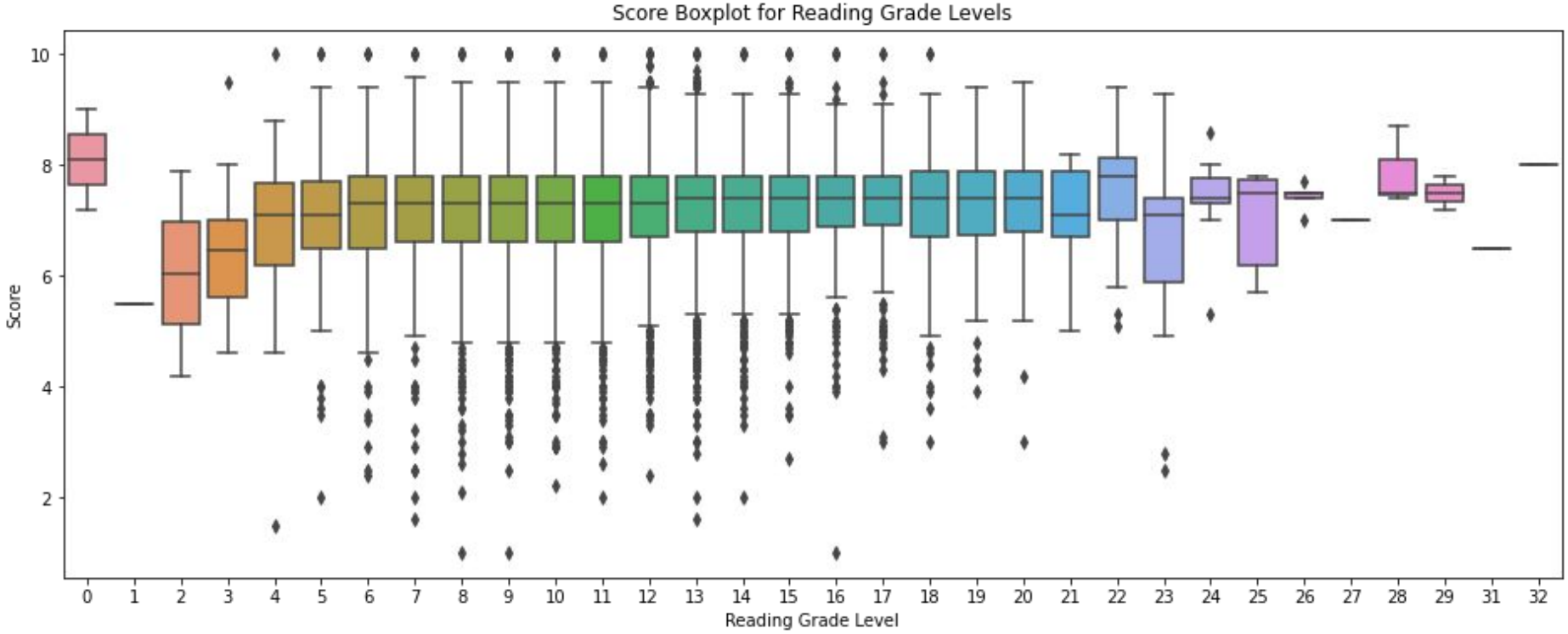
Rap

Pop/R&B

Percent High Reviews by Grade Level



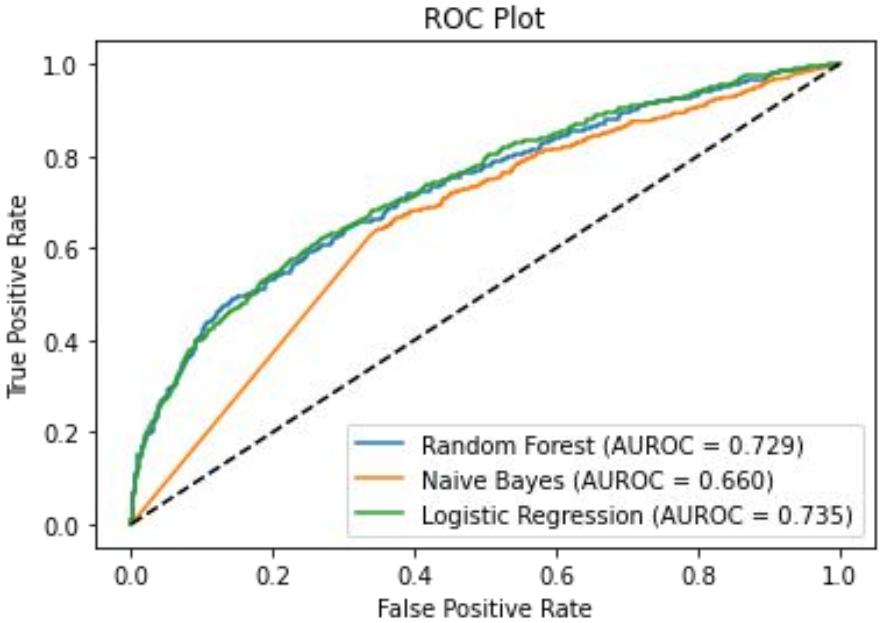
Average Score by Grade Level



Preprocessing and Modeling

- First I binarized the categorical variables and scaled the numeric variables.
- I then created a countvectorizer and grid searched the minimum document frequency with a Gaussian Naive Bayes model
- After finding the optimal minimum document frequency, I created a dataframe of binarized text features and appended it to the original dataframe.
- I grid searched three separate models to find the optimal parameters:
    - Gaussian Naive Bayes
    - Random Forest
    - Logistic Regression

Model Performance



word_count_scaled	0.027590
reading_ease_scaled	0.022034
reissued	0.014364
box	0.011277
sunday	0.009195
eligible	0.006788
set	0.006684
remastered	0.006442
depth	0.006315
reissue	0.006309

Top Ten Important Features

Thresholding

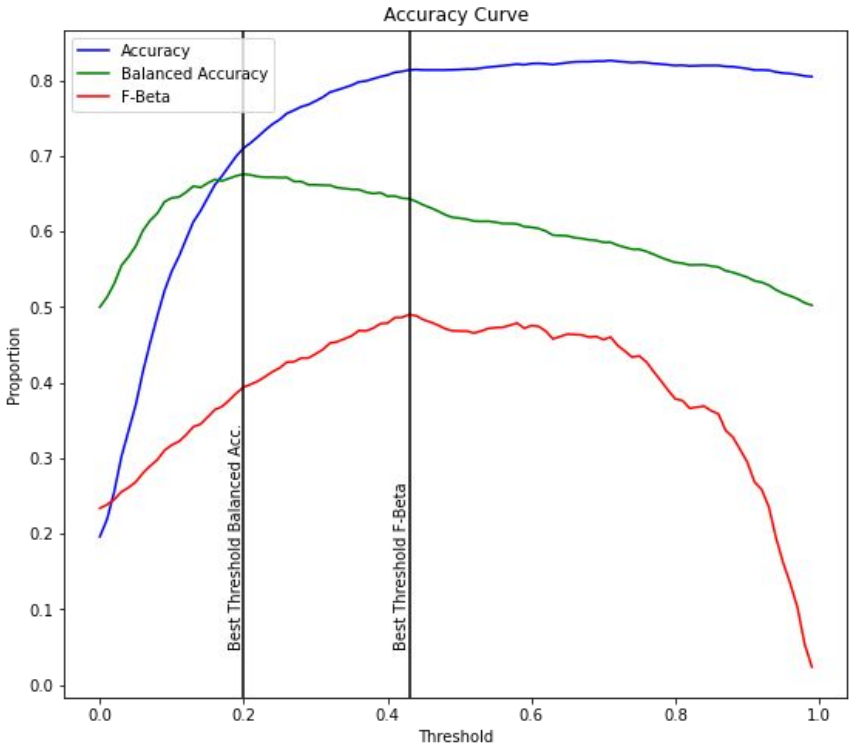
Business Case #1:

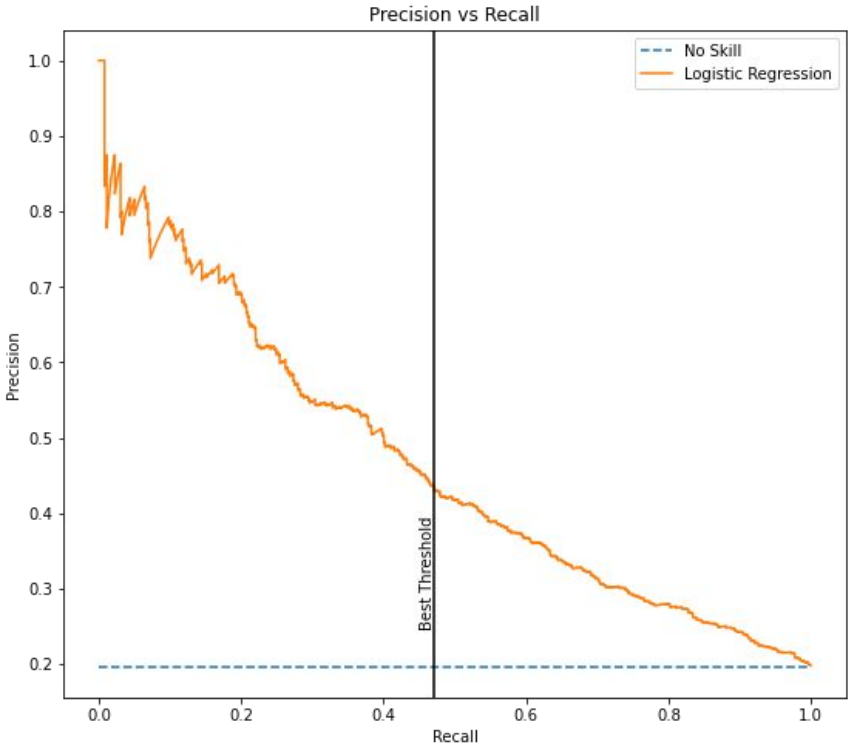
A record label wants to know how well an album was received generally by social media where the sentiment is unlabelled. They could use our model to predict sentiment across thousands of tweets mentioning the album, and generate a score for how generally positive the sentiment was. In this case,it would 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 Results





Thresholding Classification Reports

F-Beta

	precision	recall	f1-score	support
0	0.85	0.93	0.89	2549
1	0.54	0.34	0.42	622
accuracy			0.81	3171
macro avg	0.70	0.63	0.65	3171
weighted avg	0.79	0.81	0.80	3171

Balanced Accuracy

	precision	recall	f1-score	support
0	0.89	0.74	0.81	2549
1	0.37	0.60	0.46	622
accuracy			0.72	3171
macro avg	0.63	0.67	0.63	3171
weighted avg	0.78	0.72	0.74	3171

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.

Ideas to Improve Model in Future

- Remove reviews with repeated phrases such as “Each Sunday, Pitchfork takes an in-depth look”
- Extract data from similar music review websites such as RollingStone and append it to data, then retrain the model.
- Research and apply different NLP techniques to see if they improve performance further.
- Use ensemble methods.
- Try TF-IDF and Hash vectorizers.

Thank You!

Github: https://github.com/jsmolesworth96