A Review of Pitchfork Music Reviews

December 11, 2019

1 Introduction

Pitchfork, the self-proclaimed "Most Trusted Voice in Music," wields great power and influence in the modern music community. The Chicago-based online magazine is one of the largest sites dedicated to publishing music reviews and ratings for both tracks and albums. *Pitchfork* focuses largely on independent and popular music, but occasionally releases retrospective reviews of classic albums and reissues of dated projects. The magazine has developed a readership so large that articles and reviews posted by *Pitchfork* can have a significant effect on the reception of an artist's new projects, both critically and in sales. A writer at *Pitchfork* can effectively launch or end an artist's career, or at least seriously impact their livelihood. With such a significant effect on commercial success, I am concerned about the fairness of these reviews. Are there biases inherent in such a subjective writing process?

Pitchfork has historically been criticized for its heavy focus on independent bands, favoring lesser-known, more "hipster" bands over groups or individuals that already have a following. They've since expanded the breadth of what they cover, but other biases are likely inherent in their reviews. Understanding the effects and biases of these reviews will encourage individual thought and discernment in the music community and could inspire change in the way critical review impacts listeners. Questions I set out to answer include:

1. Are some genres critiqued more harshly than others?
2. Are artists that already have a significant following less likely to fare well in their ratings?
3. Are albums more likely to receive high ratings if they are musically unconventional?

Some research has been done previously exploring the distributions of scores by author, autocorrelation of reviews, borderline Best New Music decisions, and independence of Best New Music awards. However, published research is minimal, and I have not seen any attempts to explore other facets of these reviews, such as musical features or artist popularity.

To answer my questions, I use a dataset provided by a Kaggle user containing information about over 18,000 *Pitchfork* reviews, as well as data scraped from the *Billboard 200* charts and from Spotify.

2 Collecting and Cleaning Data

2.1 The Kaggle Dataset

The Kaggle dataset, found here, is a SQL database comprised of data from over 18,000 *Pitchfork* reviews. Columns include `reviewid`, `title`, `artist`, `url`, `score`, `best_new_music`, `author`,

author_type, pub_date, pub_weekday, pub_day, pub_month, pub_year, year, genre, label, and content.

[2]:
```
import sqlite3
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from difflib import get_close_matches
import requests
from bs4 import BeautifulSoup
from dateutil import parser
import time
from datetime import date, datetime, timedelta
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials
import statsmodels.api as sm
plt.style.use("seaborn")
```

[2]:
```
# Query data into desired DataFrames.
con = sqlite3.connect('pitchfork_data.sqlite')
years = pd.read_sql('SELECT * FROM years', con)
reviews = pd.read_sql('SELECT * FROM reviews', con)
genres = pd.read_sql('SELECT * FROM genres', con)
con.close()
```

Some aspects of the data require cleaning. Below, reviews for albums that have multiple years listed are removed. This is indicative of an album reissue, which we do not wish to consider in our analysis. We also are not interested in the column listing the url of the reviews, so this column is removed. Some categorical variables are represented as integers, but are changed to strings. We drop any duplicates found in the dataframe and change the publication date of the reviews to be DateTime objects for ease of analysis.

[3]:
```
# Remove albums with multiple years listed.
num_years = years.groupby('reviewid').count().reset_index()
one_year = num_years.loc[num_years.year == 1, 'reviewid']
reviews = reviews.loc[reviews.reviewid.isin(one_year)]
genres = genres.loc[genres.reviewid.isin(one_year)]
years = years.loc[years.reviewid.isin(one_year)]

# Drop url column.
reviews.drop(columns='url',inplace=True)

# Change categorical values to strings.
reviews.pub_weekday = reviews.pub_weekday.astype(str)
reviews.pub_day = reviews.pub_day.astype(str)
reviews.pub_month = reviews.pub_month.astype(str)

# Drop duplicates.
reviews.drop_duplicates(inplace=True)
```

```
# Change publication date to DateTime.
reviews['pub_date'] = pd.to_datetime(reviews['pub_date'])
```

We now wish to identify any missing data using `pd.isna()`. This analysis reveals we only had missing entries in three categories, which we remedy below. Missing author types are imputed with the mode of the author types, which is the vague title, "Contributor." Any missing data regarding genre or record label is assigned to be "other."

[10]:
```
# Fill missing author type with the mode, "Contributor."
reviews['author_type'].fillna(reviews['author_type'].mode()[0], inplace=True)

# Fill missing genres and labels with "other."
genres['genre'].fillna('other', inplace=True)
labels['label'].fillna('other', inplace=True)

# Add album year and genre using only the first genre tagged.
genres = genres.groupby('reviewid').agg(lambda x: x.tolist()[0])
reviews = reviews.merge(genres, on='reviewid')
reviews = reviews.merge(years, on='reviewid')
reviews['year'] = pd.to_datetime(reviews['year'], format='%Y')

# Get dummy variables for genres.
gens = pd.get_dummies(reviews.genre)
reviews = pd.concat([reviews,gens],axis=1)
```

While scraping the data, it is possible that some abnormal characters were not recorded correctly. We check for any reviews with album or artist names that empty strings and print out enough information about these projects to identify them through research. After looking up these reviews on *Pitchfork.com*, a description of the symbols used as artist or album names is assigned in place of the empty strings.

[5]:
```
# Make fixes according to what search revealed.
reviews.loc[3440,'artist'] = '(Three Cross Symbols)'
reviews.loc[3440,'title'] = '(Three Cross Symbols)'
reviews.loc[11311,'title'] = '(Cross Symbol)'
reviews.loc[3388,'artist'] = '(Sigma Symbol)'
```

Occasionally, we may expect to find spelling errors in the artist names, which would affect our analysis. We search for band names that are similar to one another using `difflib.get_close_matches()` with a cutoff value of 0.9. Printing off these potential spelling errors allows us to research the website once more to check if these are unique artist names or spelling errors.

[8]:
```
# Find potential typos in band names.
bands = reviews.artist.unique()
visited = []
print("Artist, [\'Potential Typos\']")
print()
for i in range(len(bands)):
    name = bands[i]
    matches = get_close_matches(name, bands,cutoff=.9)
    if len(matches) > 1 and name not in visited:
        # Add the name and its matches to visited.
        for x in matches:
            visited.append(x)
        print(name,matches[1:])
```

```
Artist, ['Potential Typos']

sammus ['samus']
thee oh sees ['the ohsees']
ladyhawke ['ladyhawk']
the kills ['the killers']
whitney ['whitey']
cough ['coughs']
the range ['the strange']
thao & the get down stay down ['thao with the get down stay down']
money ['monkey']
daughter ['daughters']
dj spinn ['dj spinna']
the mantles ['the antlers']
braids ['braid']
ranger ['rangers', 'oranger']
pearson sound ['prson sound']
mike jones ['mike bones']
sam amidon ['samamidon']
vessel ['vessels']
chromeo ['chrome']
the moles ['the mole']
indian ['indians']
vangelist ['vangelis', 'evangelista']
the stranger ['the strange']
the beatles ['the battles']
young moe ['young smoke', 'young money']
black twig pickers ['the black twig pickers']
buke and gase ['buke and gass']
the host ['the ghost']
canyons ['canyon']
figurines ['figurine']
panther ['panthers']
esser ['lesser']
the present ['the presets']
no neck blues band ['no-neck blues band']
emilana torrini ['emiliana torrini']
son, ambulance ['son ambulance']
thee silver mt. zion memorial orchestra & tra-la-la band ['the silver mt. zion
memorial orchestra & tra-la-la band', 'the silver mt. zion memorial orchestra &
tra-la-la band with choir']
```

```
the grates ['the graves']
the sadies ['the ladies']
```

Research revealed the data contained spelling errors for the following bands: Thee Oh Sees, Thao and the Get Down Stay Down, Pearson Sound, Sam Amidon, Vangelis, The Black Twig Pickers, Buke and Gase, No-Neck Blues Band, The Red Crayola, Emiliana Torrini, Son, Ambulance, and Thee Silver Mt. Zion Memorial Orchestra. The code below is used to uniformize the spelling of the artists.

[8]:
```python
# Standardize band names with spelling errors.
unique_artists = reviews.artist.unique()
names = ['thee oh sees','thao & the get down stay down','pearson sound','sam␣
 ↪amidon','vangelis','the black twig pickers','buke and gase','no-neck blues␣
 ↪band','emiliana torrini','son, ambulance','thee silver mt. zion memorial␣
 ↪orchestra']

for name in names:
    # Experimentation revealed cutoff of 0.8 captured all typos.
    matches = get_close_matches(name,unique_artists, cutoff=.8)
    for match in matches:
        # Evangelista was mistakenly captured and should not be changed.
        if match == 'evangelista':
            continue
        else:
            reviews.loc[reviews.artist==match,'artist'] = name
```

Part of our analysis will include trends in reception of the Best New Music award. This award was not given at the inception of the company, nor is it offered to albums that are part of Sunday Reviews. We wish to subset the data to only include albums that qualify to receive this award when performing this analysis. We remove all Sunday reviews and reviews before Best New Music began.

[6]:
```python
# Remove Sunday Reviews.
reviews = reviews[reviews.pub_weekday != '6']

# Start from the first time Best New Music was awarded.
first_award = min(reviews[reviews.best_new_music==1].pub_date)
reviews = reviews[reviews.pub_date >= first_award]
```

The Kaggle user who scraped and prepared this database has kindly posted links to his GitHub showing how he scraped the data. Having reviewed the script he used to scrape the data, I am confident it is reliable. This data comes straight from *Pitchfork*'s website and is a factual and holistic representation of their reviews.

I wish to understand the trends of *Pitchfork* reviews generally. While this data is sufficient for analyzing trends in critical reception of albums by genre and in the bestowal of the Best New Music award, it lacks information regarding artist popularity or musical features of the album.

I elected to scrape data from the *Billboard 200* charts and Spotify to gather this missing information.

2.2 Scraping *Billboard 200* Charts

The *Billboard 200* chart ranks the 200 most popular albums in the United States each week. Results are published each week and can be found on Billboard's website as far back as August of 1963. I chose to use the *Billboard 200* because it has been the most popular source for ranking the current most popular artists, albums, and songs for over 60 years and has had great influence and impact on popular music for decades. For that reason, I propose it is a valid source for determining music popularity.

The following code was used to scrape the artist name, album name, and publish date of every album to appear on the *Billboard 200* since August 17th, 1963. The functions `get_soup`, `every_week`, `get_artist`, and `get_album` can be viewed in the auxiliary code file.

```
[ ]: # Collect urls to the Billboard 200 charts for each week since inception.
urls = []
prefix = 'http://www.billboard.com/charts/billboard-200/'
for suffix in every_week(date(1963, 8, 17), date(2019, 11, 23),␣
 ↪timedelta(days=7)):
    link = prefix + str(suffix)
    urls.append(link)
```

```
[ ]: # Collect and store data from each page.
dfs = []
for url in urls:
    soup = get_soup(url)
    time.sleep(10) # Billboard requires a 10 second crawl delay.
    artist = get_artist(soup)
    album = get_album(soup)
    dates = [parser.parse(url.split('/')[5])]*len(artist)
    df = pd.DataFrame({'artist': artist.lower(), 'album': album.lower(),␣
 ↪'publish_date': dates})
    dfs.append(df)
```

```
[ ]: # Concatenate all dataframes together.
all_years = pd.concat(dfs).reset_index(drop=True)

# Remove repeated albums and change publication date to DateTime.
billboard200 = all_years.drop_duplicates(['album'], keep='first').
 ↪reset_index(drop=True)
billboard200['publish_date'] = pd.to_datetime(billboard200['publish_date'])
```

2.3 Collecting Features from Billboard and Spotify

The `billboard200` dataset is used to determine whether the artist being reviewed has previously appeared on the *Billboard 200*. This will be a metric of popularity of the artist and will be used to test the hypothesis that *Pithfork* writers favor unknown artists. This feature is added to the dataset as `billboard`.

Spotify provides a web API for retrieving information they provide about music projects. After retrieving a key from their site, the code in the auxiliary file was used to get musical data that Spotify measures on every track. For each album, the API is used to retrieve the following features:

popularity of the artist and album on spotify, number of tracks in the album, total duration of the album, average track duration, average track energy, average track loudness, average track valence (a measure of how happy a track sounds), and what proportion of the tracks are in a major key. The motivation behind these features is my belief that contrarian reviewers may reward musical risks, awarding higher scores to albums with features radically different from the norm.

With over 217 million users, Spotify is the most widely used service for listening to music worldwide. For that reason, I believe their statistics are reliable and representative of music trends worldwide.

3 Feature Engineering

After collecting features describing an album musically, I decided to define a metric to capture noncomformity to music norms. This new feature, "hipster meter," is initialized at 0 for every album. Then, if any of the following features are abnormal, the hipster meter is incremented by one: album duration, average track duration, average energy, average loudness, average valence, or proportion of tracks in a major key. Each feature is considered abnormal if it is more than 1.282 standard deviations above or below the mean. If this is the case, then that album is either in the highest 10% or lowest 10% of albums for that feature.

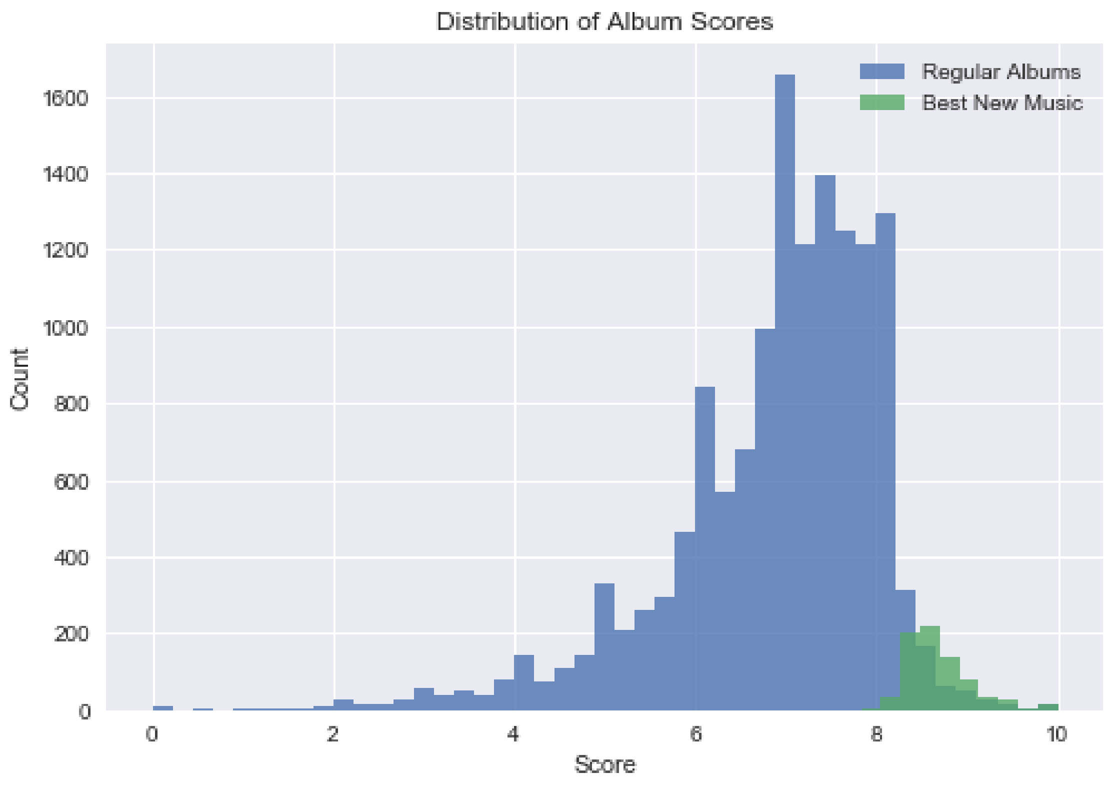
Additionally, the hipster meter is incremented by one if the genre is "experimental." This feature is engineered to answer my third question from the introduction about whether reviewers reward artist for being musically unconventional.

[13]:
```python
# Initialize hipster_meter.
reviews['hipster_meter'] = 0

# Increment hipster_meter.
hipster_cols = ['total_duration','avg_track_duration','avg_energy',
                'avg_loudness','avg_valence','major_proportion']
for c in hipster_cols:
    m, sd = reviews[c].mean(), reviews[c].std()
    reviews.loc[abs(reviews[c]-m) > 1.282*sd,'hipster_meter'] += 1
reviews.loc[reviews.genre=='experimental', 'hipster_meter'] += 1
```

4 Data Visualization and Analysis

If *Pitchfork* reviewed every album that was released, their scores might be expected to follow a normal distribution around a mean score of 5. Given that they have to pick and choose which albums to review, it isn't too surprising that the scores appear to follow a slightly skewed normal distribution with a mean of 6.97, as illustrated below. The albums that were awarded Best New Music are plotted in green and clearly have a much higher average score. The average of the Best New Music albums is 8.64 while the average of the others is 6.88. It is very uncommon to score above an 8 and doing so garners significant attention from the music community. Earning above a 9 is extremely rare and elevates an album to the "must-listen" category among music enthusiasts.

[82]:
```python
best = reviews[reviews.best_new_music == 1]
not_best = reviews[reviews.best_new_music == 0]
plt.hist(not_best.score.values, bins=45, label='Regular Albums', alpha=.8)
plt.hist(best.score.values, bins=30, label='Best New Music', alpha=.8)
plt.xlabel("Score")
plt.ylabel("Count")
plt.title("Distribution of Album Scores")
plt.legend()
plt.show()
```



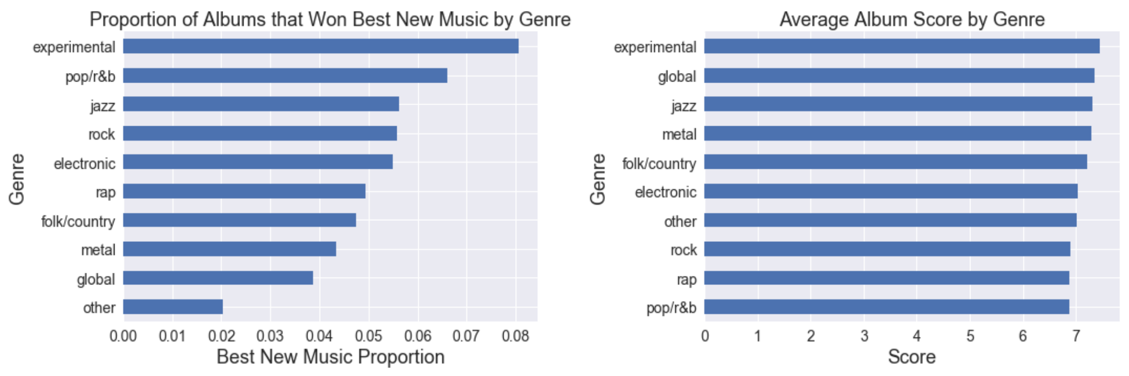
4.1 Are all genres treated equally?

Pitchfork started as an Indie Music review company and quickly spread to cover a wide variety of music genres. Again, because of my assumptions of the contrarian nature of their writers, I expected to see much higher scores given to genres such as "experimental" or "global" on average. These genres are not often popular in America and I expect writers to want to shed more light on the unpopular projects that impacted them by giving those projects higher scores or awards.

As I plotted average score by genre, this prediction held true, with the highest scores given to experimental, global, and jazz music, and the lowest given to pop/r&b, rap, and rock. These last three genres are the most popular in music today.

The trend didn't follow so precisely for the Best New Music award. While the most awards per album were given in the experimental genre, the next highest genre was pop/r&b. Despite this, I still contend that this is compelling evidence for contrarian bias.

[144]:
```
plt.figure(figsize=(15,5))
plt.subplot(121)
plt.tick_params('both',labelsize=14)
reviews.groupby('genre').agg('mean').best_new_music.sort_values().plot.barh()
plt.xlabel("Best New Music Proportion", fontsize=18)
plt.ylabel("Genre", fontsize=18)
plt.title("Proportion of Albums that Won Best New Music by Genre", fontsize=18)
plt.subplot(122)
plt.tick_params('both',labelsize=14)
reviews.groupby('genre').agg('mean').score.sort_values().plot.barh()
plt.xlabel("Score", fontsize=18)
plt.ylabel("Genre", fontsize=18)
plt.title("Average Album Score by Genre", fontsize=18)
plt.tight_layout()
plt.show()
```



4.2 Are Popular Artists Critiqued More Harshly?

Because of *Pitchfork*'s contrarian nature, I anticipated that they may favor lesser-known artists over popular artists when it came to their Best New Music award and scoring system, but I did not find that to be the case. In fact, a much higher percentage of the albums that received Best New Music had previously been on the Billboard 200 than the other albums. Similarly, if an artist had previously appeared on the Billboard 200, they were much more likely to receive Best New Music awards.

Below, the percentage of artists that received the Best New Music award are calculated for the group of artists that had previously appeared on the *Billboard 200* and for those that had not. Only 4.44% of the "less popular" artists received this award, while 7.83% of the "popular" artists receieved the award.

[14]:
```
bill = reviews[reviews.billboard == 1]
not_bill = reviews[reviews.billboard == 0]
print("Percentage of artists that received Best New Music:\n")
bill_percent = 100*bill.best_new_music.value_counts().values/bill.
 ↪best_new_music.count()
```

```
reg_percent = 100*not_bill.best_new_music.value_counts().values/not_bill.
 ↪best_new_music.count()
print("Artists that had previously appeared on Billboard 200: {:.2f}%".
 ↪format(bill_percent[1]))
print("Artists that had not appeared on Billboard 200: {:.2f}%".
 ↪format(reg_percent[1]))
```

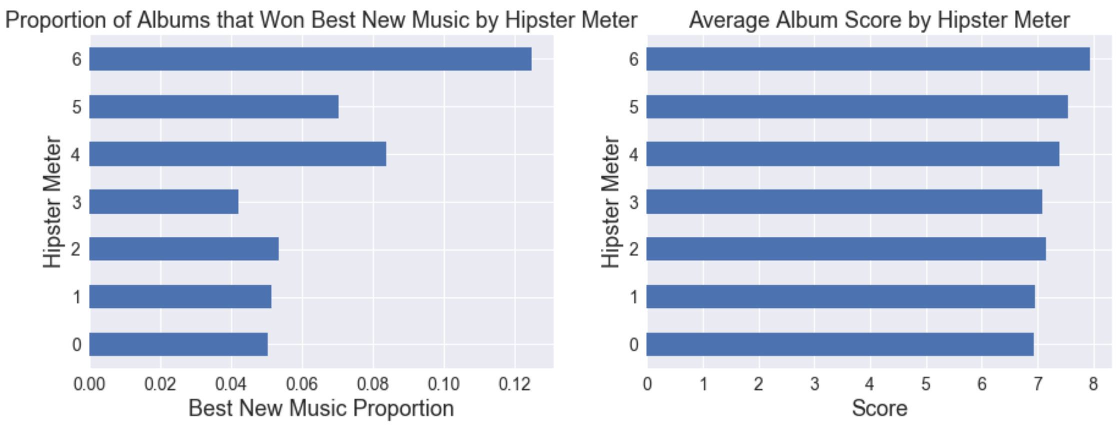
```
Percentage of artists that received Best New Music:

Artists that had previously appeared on Billboard 200: 7.83%
Artists that had not appeared on Billboard 200: 4.44%
```

4.3 Does unconventional music score better?

I wanted to test the validity of my intuition that unconventional, risk-taking albums were more likely to be rewarded by reviewers than safe, standard albums. I plotted the average score and Best New Music awards per album for each level of how different an album's sound was, as measured by its hipster meter.

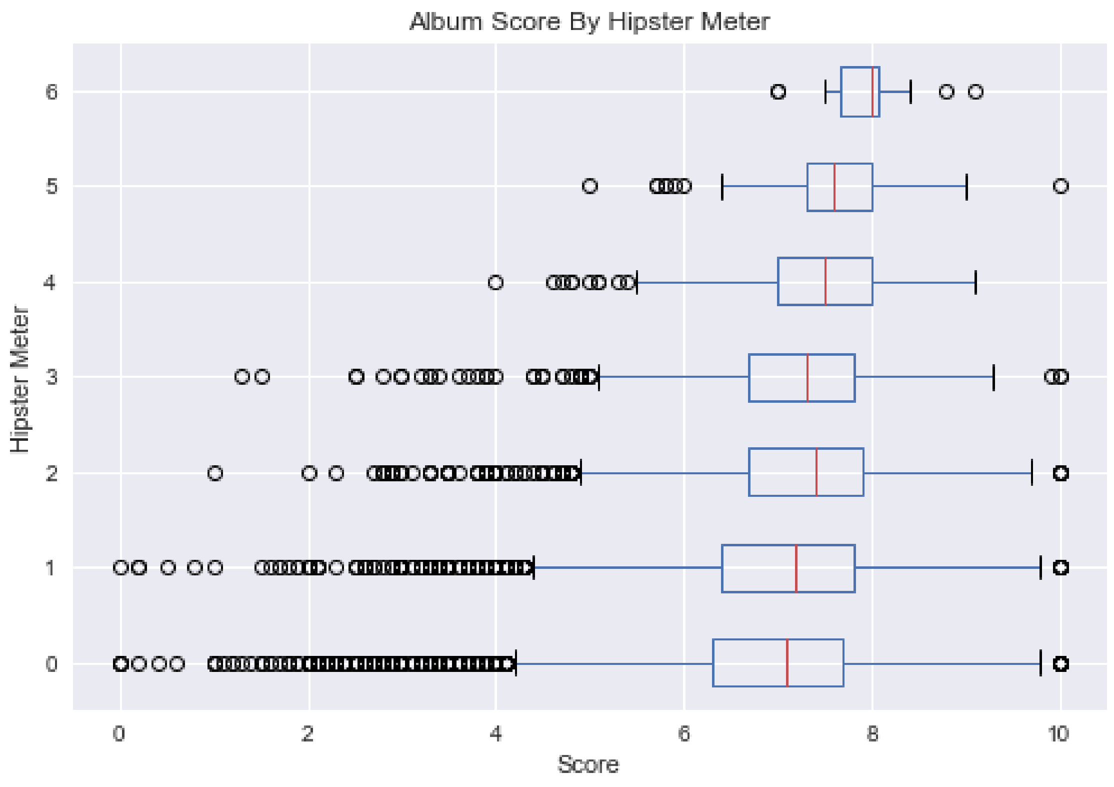
As shown below, the score increased consistently as hipster meter increased. Those albums with a full hipster meter of 6 scored a full point higher on average than those albums with a hipster meter of 0. Additionally, a much higher proportion of albums with a high hipster meter were awarded Best New Music compared to those albums with a low hipster meter. This is consistent with my expectation of contrarian bias.

[149]:
```
plt.figure(figsize=(15,5))
plt.subplot(121)
plt.tick_params('both',labelsize=14)
reviews.groupby('hipster_meter').mean().best_new_music.plot.barh()
plt.xlabel("Best New Music Proportion", fontsize=18)
plt.ylabel("Hipster Meter", fontsize=18)
plt.title("Proportion of Albums that Won Best New Music by Hipster Meter",␣
 ↪fontsize=18)
plt.subplot(122)
plt.tick_params('both',labelsize=14)
reviews.groupby('hipster_meter').mean().score.plot.barh()
plt.xlabel("Score", fontsize=18)
plt.ylabel("Hipster Meter", fontsize=18)
plt.title("Average Album Score by Hipster Meter", fontsize=18)
plt.show()
```



This box plot of album score by hipster meter again illustrates the increasing average score, and further demonstrates the reduced variance of the scores as hipster meter increases. This reduction in variance is likely to due to the reduced size of the groups as the hipster meter increases.

[34]:
```
reviews.boxplot(by='hipster_meter',column='score', vert=False)
plt.xlabel("Score")
plt.ylabel("Hipster Meter")
plt.title("Album Score By Hipster Meter")
plt.suptitle("")
plt.show()
```



To further illustrate the effects of these "hipster features" on album score, I fit an ordinary least squares linear regression model with these features and a column of ones as independent variables and the album score as the dependent response variable. From this model I was able to get the following coefficients and p-values for the features.

Significance of Hipster Features

Feature	Coefficient	P-value
Album Duration	6.8e-05	0.000
Track Duration	0.0001	0.033
Energy	0.3795	0.000
Loudness	-0.0436	0.000
Valence	-0.3222	0.000
Major/Minor Key	0.0574	0.322
Constant	6.1582	0.000

Using a significance level of 5%, all features are significant except the proportion of songs on an album that are in major versus minor keys. The coefficients indicate that the measured energy and valence of an album likely have the greatest effect on the score given.

5 Conclusion

As predicted, there is evidence of contrarian biases in *Pitchfork* music reviews. As reflected in the distribution of scores and Best New Music awards issued, writers tend to favor certain genres such as experimental and global over others. They also tend to favor new and different sounds over safer, more familiar musical expression. Albums that have musical traits that are drastically different from the norm are more likely to receive higher scores and more likely to receive the Best New Music award.

While many musical traits did show significant effect on determining score with a regression model, it is worth noting that the number of songs in an album that are in major versus minor keys did not have a significant effect on album score. Furthermore, there is no evidence that an artist's popularity has a negative effect on the score they received. The opposite effect appears to be true: more popular artists are more likely to receive Best New Music.

I'm interested in understanding the effect of these reviews on the success of an artist going forward. I would like to further explore the immediate impact of these reviews on album sales and streams. In the future, I hope to use machine learning techniques to try to predict an album score from the features discussed in this report. I would also like to employ unsupervised algorithms to try to cluster albums together in different ways in an attempt to find interesting patterns for organizing music in ways other than by genre.