The Billboard Charts

November 23, 2016

Antonia Carlotta

What are the Billboard Charts?

Billboard was founded in 1936 and started releasing popularity charts in 1940.

Currently the Billboard Hot 100 Charts measure the popularity of songs and albums on a weekly basis.

Rankings are based on sales, streams, and airplay.

Project objective: Use Data Science tools to explore trends and relationships on the Billboard Charts, and create a model to predict how well a song may peak.

Understanding the problem

Data Collection

Scrape from the Billboard website, Spotify API, and Wikipedia to gather information about each song and artist on the chart.

Clean data and organize in a dataframe.

Data Exploration

Create graphs and visualizations to find trends and relationships on the charts.

Discern important features, look for patterns and seasonality.

Predictive Modeling

Determine whether this should be a regression or classification problem.

Create a predictive model for peak chart position based on the aforementioned data.

Data Collection

Billboard

Creating and Parsing the Charts

Created the links to the Billboard charts that would be scraped by formatting dates, appending them to the link appropriately, and requesting the information from the Billboard website.

Used BeautifulSoup to scrape every Hot 100 Chart from January 1, 2012 to November 19, 2016, totaling 255 charts, or 25,500 chart entries.

Parsed specific features from the charts including the song and artist names, the peak position, previous week's position, number of weeks on the chart, and the Spotify Track ID number, which I would then use to collect more features from Spotify's API.

Billboard

```
dateslist = pd.date_range('2012/01/07', periods=255, freq='W-SAT')
dateslist = dateslist.strftime('%Y-%m-%d')
alldateslist = [((date + ' ')*100).split() for date in dateslist]
alldates  = [i for x in alldateslist for i in x]

scrapeurls = []
for i in dateslist:
    scrapeurls.append('http://www.billboard.com/charts/hot-100/' + i)
allcharts = []
for i in scrapeurls:
    allcharts.append(requests.get(i))

parsedcharts = []
for ch in allcharts:
    parsedcharts.append(BeautifulSoup(ch.text, 'html.parser'))

allsongs = []
allpeaks = []
lastweeks = []
weeksoncharts = []
alltrackids = []

for chart in parsedcharts:
    songs = chart.findAll('h2', {'class' : 'chart-row__song'})
    peakpositions = chart.findAll('div', {'class' : 'chart-row__top-spot'})
    lastweek = chart.findAll('div', {'class' : 'chart-row__last-week'})
    weeksonchart = chart.findAll('div', {'class' : "chart-row__weeks-on-chart"})
    for song in songs:
        allsongs.append(song.text)
    for peak in peakpositions:
        allpeaks.append(peak('span', {'class':'chart-row__value'})[0].text)
    for last in lastweek:
        lastweeks.append(last.find('span', {'class':'chart-row__value'}).text)
    for weeks in weeksonchart:
        weeksoncharts.append(weeks.find('span', {'class':'chart-row__value'}).text)
    for i in range(100):
        try:
            trackid = chart.find("article", { "class" : "chart-row chart-row--"+str(i+1)\
                                              +" js-chart-row" })['data-spotifyid']
            alltrackids.append(trackid)
        except:
            alltrackids.append('None')
```

Collecting Data from Spotify

Used the Spotify Track ID collected from the Billboard Chart to get features from Spotify including song length, tempo, and key, as well as measures like “danceability”, “energy”, “acousticness”, and “valence”, among others.

Certain artists, such as Taylor Swift and Kanye West, do not allow their songs on Spotify, so that was taken into account in the code, and the word “None” was appended for their features..

```
import urllib2
import json

allfeatures = []

for id in alltrackids:
    try:
        req = urllib2.Request('https://api.spotify.com/v1/audio-features/'+id)
        req.add_header('Accept', 'application/json')
        req.add_header('Authorization', 'Bearer BQAOzHR1ot58DLIPl875tp1fsyyltQ
        resp = urllib2.urlopen(req)
        allfeatures.append(json.load(resp))
    except:
        allfeatures.append('None')
        pass

duration_ms = []
key = []
tempo = []

for i in range(25500):
    try:
        duration_ms.append(allfeatures[i]['duration_ms'])
    except:
        duration_ms.append('None')
    pass
for i in range(25500):
    try:
        tempo.append(allfeatures[i]['tempo'])
    except:
        tempo.append('None')
    pass
```

Scraping Wikipedia for Labels

An artist's status as independent versus being signed to a major label may have an effect on their potential success, as a label implies funding and publicity.

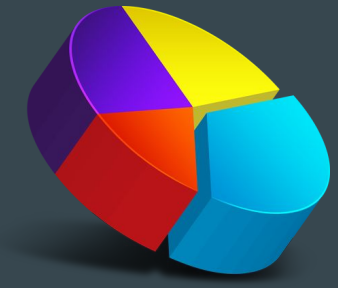
Found Wikipedia links to the biggest music labels with a list of artists signed to them.

Scraped each page and created a list of artists on the chart that were signed to one of the major labels.

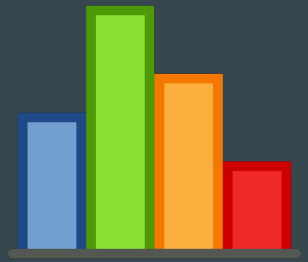
```
scrapelabelurls = ['http://store.warnermusic.com/all-artists',
                   'https://en.wikipedia.org/wiki/List_of_Universal_Music_Group_artists'
                   'https://en.wikipedia.org/wiki/List_of_Sony_Music_artists',
                   'https://en.wikipedia.org/wiki/List_of_Epic_Records_artists']
scrapelabels = []
for i in scrapelabelurls:
    scrapelabels.append(requests.get(i))

parsedlabels = []
for i in scrapelabels:
    parsedlabels.append(BeautifulSoup(i.text, 'html.parser'))

majorartists = []
for splitartist in uniquedata['splitartist']:
    if splitartist in strparsedlabels:
        majorartists.append(splitartist)

label = []
for i in uniquedata['splitartist']:
    if i in majorartists:
        label.append(1)
    else:
        label.append(0)
```

The DataFrame

alldates	currentrank	allsongs	allartists	allpeaks	lastweeks	weeksoncharts	duration_ms	key	tempo	...	danceab
2012-01-07	1	Sexy And I Know It	LMFAO	1	2.0	17	199480.0	7	1.0	...	0.115000
2012-01-07	2	We Found Love	Rihanna Featuring Calvin Harris	1	1.0	14	215760.0	1	1.0	...	0.022500
2012-01-07	3	The One That Got Away	Katy Perry	3	5.0	11	227333.0	1	0.0	...	0.000802
2012-01-07	4	It Will Rain	Bruno Mars	3	3.0	13	257848.0	2	1.0	...	0.359000
2012-01-07	6	Good Feeling	Flo Rida	3	4.0	13	248133.0	1	0.0	...	0.058800

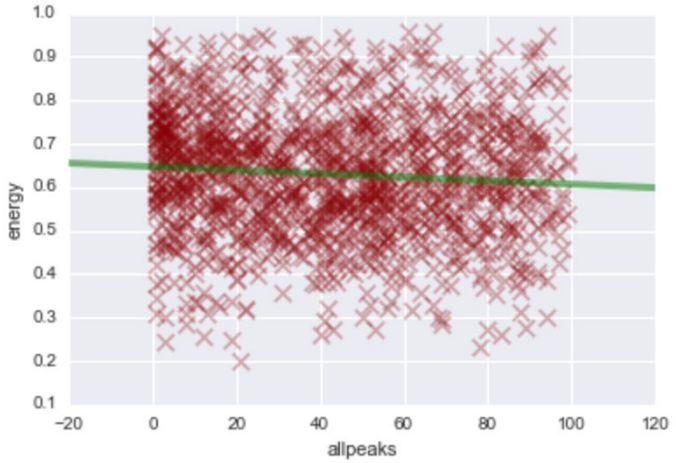


Data Exploration

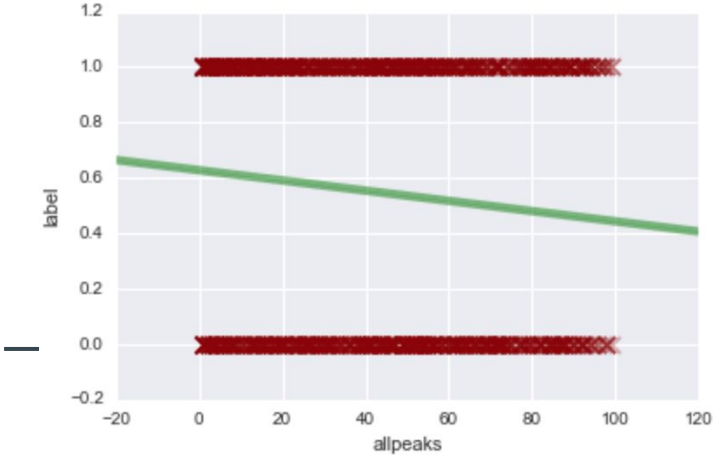


Graphing Features and Peaks

This first graph indicates songs with more “energy” (defined by Spotify as having “intensity and activity”) chart moderately higher than songs with lower energy.



This second graph demonstrates a fairly strong relationship to a high charting song when the artist is signed to a major label (Universal Music Group, Warner Music Group, or Sony Music.



Notable Findings

Finding 1	• 61.7% of artists with a song in the top 20 will at some point have another song on the charts.
Finding 2	• The charts are more country intense than one might suspect (More to be done...)
Finding 3	• There is a relationship between featured artists and chart placement.

Predictive Modeling

Regression or Classification?

Initially viewed this as a regression problem, trying to chart between 1 and 100. This was ineffective, however, as the model would never predict any song to reach the top or bottom of the chart, and instead placed every song between 30 and 65.

As a classification problem, I split the chart into three segments: 1-20, 21-60, and 61-100.

This range may be more meaningful to someone trying to determine which song to make a single, how much marketing a song needs, and what type of returns to expect.



```
chartsegment = []

for peak in uniquedata['allpeaks']:
    if peak <= 20:
        chartsegment.append(0)
    elif peak > 20 and peak < 61:
        chartsegment.append(1)
    else:
        chartsegment.append(2)
```



Random Forest Classifier

```
y = uniquedata['chartsegment']
X = uniquedata[['energy', 'danceability', 'valence', 'speechiness', 'tempo', 'label']]

from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=13)

from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(n_estimators=325, oob_score=True, min_samples_split=42)
clf.fit(X_train, y_train)
predictions = clf.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix

print classification_report(y_test, predictions)
print confusion_matrix(y_test, predictions)
```

Predictions and Results

Used a Random Forest Classifier.

The model still tends to predict toward the middle of the chart.

Still, this is better than the regression results as we can predict across the entire chart as opposed to just in one segment.

Important numbers here -- the ones that are correct -- are the top left, middle, and bottom right.

Better features, as well as data regarding songs that didn't chart at all, may help make these numbers better

	precision	recall	f1-score	support
0	0.36	0.47	0.41	133
1	0.49	0.47	0.48	179
2	0.37	0.26	0.31	126
avg / total	0.42	0.41	0.41	438

```
[[ 47  72  14]
 [ 42 122  15]
 [ 31  76  19]]
```

What Next?

Future objectives: Expand upon features for better predictive modeling. Explore time series for better trend analysis. Consider NLP and its significance.

THANK YOU