FinalAssigmentPDF

May 10, 2020

1 DSCI 235 Final Assignment

Data Source: https://www.kaggle.com/theoverman/the-spotify-hit-predictor-dataset

1.0.1 Project Description

For my final project in DSCI 235, I chose to analyze a dataset of over 40,000 songs since 1960 to try to answer questions about popular music, specifically how it can be distingushed from non-pop music, how well we can create a forumla based on these methods of separation, and how it has changed over time.

In this project, we'll attempt to answer the following questions:

1) Are hit songs different than non-hit songs?
2) What are the best predictors for what will become a hit song?
3) How well do these predictos separate hit songs from non-hit songs? What is the 'ideal' pop song according to our predictors?
4) How has popular music changed through the last 50 years?

Note: Information about the dataset used and a link to the data can be found at the bottom of the notebook.

[84]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

%autosave 60
```

```
Autosaving every 60 seconds
```

[210]:
```
sixties = pd.read_csv('https://raw.githubusercontent.com/ecreagar/DSCI235/
 ↪master/the-spotify-hit-predictor-dataset/dataset-of-60s.csv')
seventies = pd.read_csv('https://raw.githubusercontent.com/ecreagar/DSCI235/
 ↪master/the-spotify-hit-predictor-dataset/dataset-of-70s.csv')
eighties = pd.read_csv('https://raw.githubusercontent.com/ecreagar/DSCI235/
 ↪master/the-spotify-hit-predictor-dataset/dataset-of-80s.csv')
```

```
nineties = pd.read_csv('https://raw.githubusercontent.com/ecreagar/DSCI235/
 ↪master/the-spotify-hit-predictor-dataset/dataset-of-90s.csv')
thousands = pd.read_csv('https://raw.githubusercontent.com/ecreagar/DSCI235/
 ↪master/the-spotify-hit-predictor-dataset/dataset-of-00s.csv')
tens = pd.read_csv('https://raw.githubusercontent.com/ecreagar/DSCI235/master/
 ↪the-spotify-hit-predictor-dataset/dataset-of-10s.csv')

sixties["year"]=1960
seventies['year']=1970
eighties['year']=1980
nineties['year']=1990
thousands['year']=2000
tens['year']=2010

songs = pd.concat([sixties,seventies,eighties,nineties,thousands,tens],␣
 ↪ignore_index=True)
songs[["track","artist"]]
```

[210]:

	track	artist
0	Jealous Kind Of Fella	Garland Green
1	Initials B.B.	Serge Gainsbourg
2	Melody Twist	Lord Melody
3	Mi Bomba Sonó	Celia Cruz
4	Uravu Solla	P. Susheela
…	…	…
41101	Lotus Flowers	Yolta
41102	Calling My Spirit	Kodak Black
41103	Teenage Dream	Katy Perry
41104	Stormy Weather	Oscar Peterson
41105	Dust	Hans Zimmer

[41106 rows x 2 columns]

Seperate the data into two data frames - one for songs that reached the top 100 charts and one for those that didn't. In this project, we'll call those hits and flops (consistant with the naming system from the data provider on kaggle).

[201]:
```
hits = songs.loc[songs.target==1]
flops = songs.loc[songs.target==0]
```

1.1 Part 1: Are hit songs different than non-hit songs?

How can we seperate hit songs from non-hit songs? For this question, we'll be using a subset of the data - songs only from the 2000's and 2010's. We'll examine the data for patterns and find out what categories tell us the most about how hit songs are different than other songs. In this experiment, I will **not** be exploring whether or not we can *statistically* conclude differences between hits and flops through examination of t-tests or p-values. So I'll examine the data, but I'll also be

careful about the conclusions.

[202]:
```
recentTracks = pd.concat([thousands,tens])
recentHits = recentTracks.loc[recentTracks.target==1]
recentFlops = recentTracks.loc[recentTracks.target==0]
```

Create a dataframe containing the means and standard errors of each column, seperated by hits and flops.

[203]:
```
recentTrackData = pd.DataFrame(recentHits.mean(axis=0),columns=["Hits Mean"])
recentTrackData["Hits Standard Dev."] = recentHits.std(axis=0)
recentTrackData["Flops Mean"] = recentFlops.mean(axis = 0)
recentTrackData["Flop Standard Dev."] = recentFlops.std(axis=0)
recentTrackData
```

[203]:

	Hits Mean	Hits Standard Dev.	Flops Mean	\
danceability	0.636089	0.145602	0.476086	
energy	0.697796	0.166211	0.663324	
key	5.282967	3.592160	5.277262	
loudness	-5.780778	2.125671	-9.264301	
mode	0.675306	0.468299	0.615322	
speechiness	0.102863	0.103535	0.087757	
acousticness	0.155394	0.194868	0.276018	
instrumentalness	0.007397	0.059600	0.309429	
liveness	0.185154	0.144808	0.207711	
valence	0.522516	0.221928	0.401828	
tempo	121.973553	29.286459	122.025096	
duration_ms	229289.262592	44476.049286	264665.336756	
time_signature	3.974735	0.262007	3.870905	
chorus_hit	39.109636	17.656912	42.665666	
sections	10.170008	2.461165	11.171638	
target	1.000000	0.000000	0.000000	
year	2005.214344	4.995811	2005.214344	

	Flop Standard Dev.
danceability	0.197369
energy	0.294430
key	3.580127
loudness	6.551211
mode	0.486559
speechiness	0.087623
acousticness	0.361651
instrumentalness	0.379286
liveness	0.180757
valence	0.263109
tempo	30.715018
duration_ms	154625.947593
time_signature	0.480761

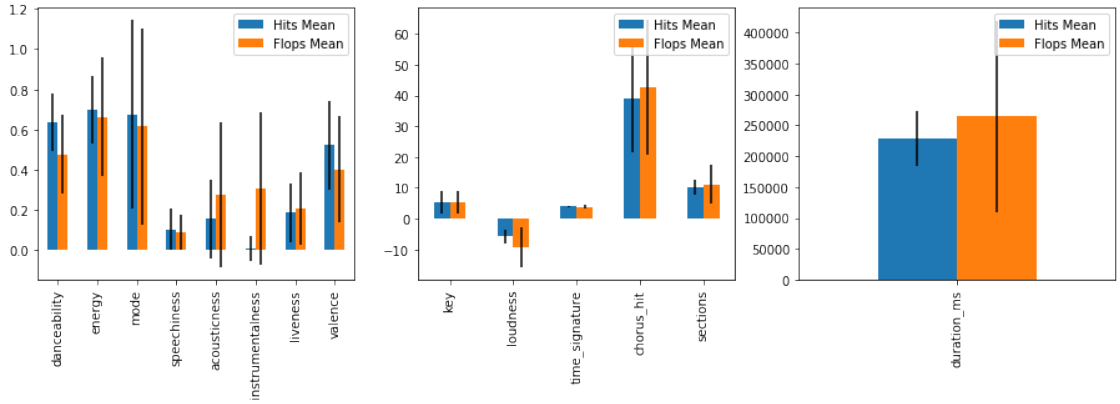
```
chorus_hit                21.762867
sections                   6.349744
target                     0.000000
year                       4.995811
```

Let's visualize this data and see if we can find any obvious differences. The plots are split up to account for the different scales of variables.

[204]:
```
fig, (ax1, ax2,ax3) = plt.subplots(1, 3)
recentTrackData[['Hits Mean', 'Flops Mean']].
 ↪drop(['key','loudness','tempo','duration_ms','time_signature',
                                                                 ␣
 ↪'chorus_hit','sections','target','year']).plot(kind='bar',␣
 ↪yerr=recentTrackData[['Hits Standard Dev.', 'Flop Standard Dev.']].
                                                                                ␣
 ↪          ␣
 ↪drop(['key','loudness','tempo','duration_ms','time_signature','chorus_hit','sections','targ
 ↪
                                                                                ␣
 ↪           values.T, alpha = 1,error_kw=dict(ecolor='k'),ax=ax1);
recentTrackData[['Hits Mean', 'Flops Mean']].
 ↪drop(['danceability','energy','mode','speechiness','acousticness','year',
                                                                 ␣
 ↪'instrumentalness','liveness','valence','duration_ms','tempo',
                                                   'target']).plot(kind='bar',␣
 ↪yerr=recentTrackData[['Hits Standard Dev.', 'Flop Standard Dev.']].
                                                                          ␣
 ↪drop(['danceability','energy','mode','speechiness','acousticness','year',
                                                                               ␣
 ↪'instrumentalness','liveness','valence','duration_ms','tempo','target']).
                                                                  values.T,␣
 ↪alpha = 1,error_kw=dict(ecolor='k'),ax=ax2);
recentTrackData[['Hits Mean', 'Flops Mean']].
 ↪drop(['danceability','energy','mode','speechiness','acousticness','year',
                                                                 ␣
 ↪'instrumentalness','liveness','valence','tempo',
                                                                 ␣
 ↪'target','key','loudness','tempo','time_signature',
                                                   'chorus_hit','sections']).
 ↪plot(kind='bar', yerr=recentTrackData[['Hits Standard Dev.', 'Flop Standard␣
 ↪Dev.']].
                                                                          ␣
 ↪drop(['danceability','energy','mode','speechiness','acousticness','year',
                                                                               ␣
 ↪'instrumentalness','liveness','valence','tempo','target','key','loudness','tempo','time_sig
                                                   'chorus_hit','sections']).
```

```
                                                                values.T,␣
↪alpha = 1,error_kw=dict(ecolor='k'),ax=ax3);

fig.subplots_adjust(left=1,right=3)
```



We notice a few things from the plots above:

- **There appear to be differences in the means of most categories**. However, since we are not investigating these differences statistically, we'll be careful about concluding that there really *are* differences.
- **The error margins for the hits are almost all smaller than the error margins for the flops.** What does this tell us? Hit music may not all be the same, but songs that become hits are more like each other than songs that don't become hits.
- **Some categories appear to have larger differences than others**. From this, we can say with confidence that hit songs are not completely different than non-hit songs, but they do appear to differ in some ways.

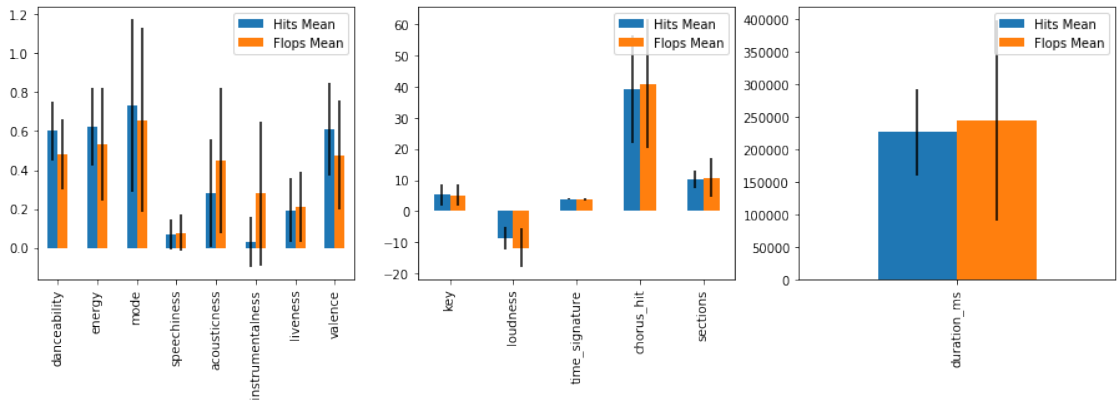
For interest's sake, what do these plots look like for data for every song since 1960?

[205]:
```
trackData = pd.DataFrame(hits.mean(axis=0),columns=["Hits Mean"])
trackData["Hits Standard Dev."] = hits.std(axis=0)
trackData["Flops Mean"] = flops.mean(axis = 0)
trackData["Flop Standard Dev."] = flops.std(axis=0)

fig, (ax1, ax2,ax3) = plt.subplots(1, 3)
trackData[['Hits Mean', 'Flops Mean']].
↪drop(['key','loudness','tempo','duration_ms','time_signature',
                                                              ␣
↪'chorus_hit','sections','target','year']).plot(kind='bar',␣
↪yerr=trackData[['Hits Standard Dev.', 'Flop Standard Dev.']].
```

```
                                                                                ␣
 ↪          ␣
 ↪drop(['key','loudness','tempo','duration_ms','time_signature','chorus_hit','sections','targ
 ↪
                                                                                ␣
 ↪           values.T, alpha = 1,error_kw=dict(ecolor='k'),ax=ax1);
trackData[['Hits Mean', 'Flops Mean']].
 ↪drop(['danceability','energy','mode','speechiness','acousticness',
                                                    ␣
 ↪'instrumentalness','liveness','valence','duration_ms','tempo',
                                                     'target','year']).
 ↪plot(kind='bar', yerr=trackData[['Hits Standard Dev.', 'Flop Standard Dev.
 ↪']].
                                                                    ␣
 ↪drop(['danceability','energy','mode','speechiness','acousticness','year',
                                                                         ␣
 ↪'instrumentalness','liveness','valence','duration_ms','tempo','target']).
                                                                      values.T,␣
 ↪alpha = 1,error_kw=dict(ecolor='k'),ax=ax2);
trackData[['Hits Mean', 'Flops Mean']].
 ↪drop(['danceability','energy','mode','speechiness','acousticness','year',
                                                    ␣
 ↪'instrumentalness','liveness','valence','tempo',
                                                    ␣
 ↪'target','key','loudness','tempo','time_signature',
                                                     'chorus_hit','sections']).
 ↪plot(kind='bar', yerr=trackData[['Hits Standard Dev.', 'Flop Standard Dev.
 ↪']].
                                                                    ␣
 ↪drop(['danceability','energy','mode','speechiness','acousticness','year',
                                                                         ␣
 ↪'instrumentalness','liveness','valence','tempo','target','key','loudness','tempo','time_sig
                                                     'chorus_hit','sections']).
                                                                      values.T,␣
 ↪alpha = 1,error_kw=dict(ecolor='k'),ax=ax3);

fig.subplots_adjust(left=1,right=3)
```



The plots are similar, but there is an obvious increase in variability as we would expect.

We conclude Part 1 in saying that yes, there are differences in songs that become hits and songs that don't, at least in the data examined between 2000 and 2015. These differences bay not be completely quantifiable without the use of more statistics and mathematics, but differences are evident from the plots and in the means and error margins. *** To continue our analysis, let's examine where the largest differences lie between hit songs and non hit songs.

1.2 Part 2: What are the biggest differences between hit songs and non-hit songs?

To examine this question, we'll stick with our data sets of songs only after 2000. This will introduce less variability and give us answers pertaining to the songs of this generation.

To start, we'll find the categories with the biggest percentage differences between their means, accounting for standard error. The formula we'll use for this is:

$$\frac{\left|\frac{mean(hit)-mean(flop)}{mean(hit)}\right|}{\frac{se(hit)}{mean(hit)}+\frac{se(flop)}{mean(flop)}}$$

This formula calculates the percentage change between the two means, then divides it by the standard error over mean of both terms, accounting for both error and scale of variable, giving favor to columns with larger difference between means and smaller standard error.

Create a new column in the dataset:

[211]:
```
recentTrackData["Percent Change"] = abs(((recentTrackData["Hits Mean"] -␣
↪recentTrackData["Flops Mean"]) / recentTrackData["Hits Mean"])/
↪((recentTrackData["Hits Standard Dev."]/recentTrackData["Hits␣
↪Mean"])+(recentTrackData["Flop Standard Dev."]/recentTrackData["Flops␣
↪Mean"]))).round(5)
recentTrackData.sort_values("Percent Change", ascending = False)[["Hits␣
↪Mean","Flops Mean","Percent Change"]].drop(["year","target"])
```

[211]:

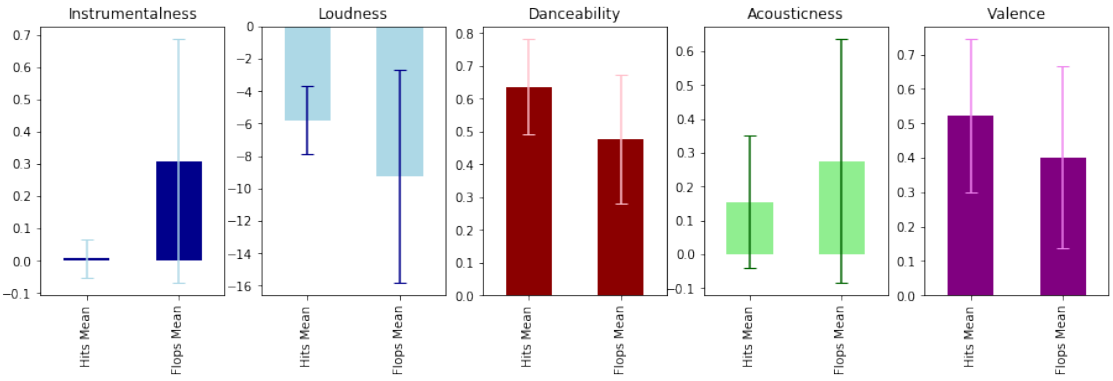
	Hits Mean	Flops Mean	Percent Change
instrumentalness	0.007397	0.309429	4.39845
loudness	-5.780778	-9.264301	0.56064
danceability	0.636089	0.476086	0.39092
acousticness	0.155394	0.276018	0.30272
valence	0.522516	0.401828	0.21396
duration_ms	229289.262592	264665.336756	0.19826
time_signature	3.974735	3.870905	0.13740
sections	10.170008	11.171638	0.12153
chorus_hit	39.109636	42.665666	0.09456
liveness	0.185154	0.207711	0.07373
speechiness	0.102863	0.087757	0.07325
energy	0.697796	0.663324	0.07243
mode	0.675306	0.615322	0.05985
tempo	121.973553	122.025096	0.00086
key	5.282967	5.277262	0.00079

Sort by Percent Change

Now, let's plot the top 5 in this category to visualize ther differences.

[208]:
```
fig, (ax1, ax2,ax3, ax4, ax5) = plt.subplots(1, 5)
recentTrackData.loc['danceability'][['Hits Mean','Flops Mean']].
 ↪plot(kind='bar',color='DarkRed', ax=ax3,capsize=5,
                                                                  title =␣
 ↪"Danceability", subplots = False,
                                                                  yerr =␣
 ↪recentTrackData.loc['danceability'][['Hits Standard Dev.', 'Flop Standard␣
 ↪Dev.']].
                                                                  values.
 ↪T,error_kw=dict(ecolor='Pink'));
recentTrackData.loc['instrumentalness'][['Hits Mean','Flops Mean']].
 ↪plot(kind='bar',color='DarkBlue', ax=ax1,capsize=5,
                                                                      ␣
 ↪title="Instrumentalness", subplots = False,
                                                                    yerr =␣
 ↪recentTrackData.loc['instrumentalness'][['Hits Standard Dev.', 'Flop␣
 ↪Standard Dev.']].
                                                                    values.
 ↪T,error_kw=dict(ecolor='LightBlue'));
recentTrackData.loc['acousticness'][['Hits Mean','Flops Mean']].
 ↪plot(kind='bar',color='LightGreen',ax=ax4,capsize=5,
                                                                ␣
 ↪title="Acousticness", subplots = False,
                                                                  yerr =␣
 ↪recentTrackData.loc['acousticness'][['Hits Standard Dev.', 'Flop Standard␣
 ↪Dev.']].
```

```
                                                                    values.
↪T,error_kw=dict(ecolor='DarkGreen'));
recentTrackData.loc['valence'][['Hits Mean','Flops Mean']].
↪plot(kind='bar',color='Purple',ax=ax5,capsize=5,
                                                                    ␣
↪title="Valence", subplots = False,
                                                                    yerr =␣
↪recentTrackData.loc['valence'][['Hits Standard Dev.', 'Flop Standard Dev.']].
                                                                    values.
↪T,error_kw=dict(ecolor='Violet'));
recentTrackData.loc['loudness'][['Hits Mean','Flops Mean']].
↪plot(kind='bar',color='LightBlue',ax=ax2,capsize=5,
                                                                    ␣
↪title="Loudness", subplots = False,
                                                                    yerr =␣
↪recentTrackData.loc['loudness'][['Hits Standard Dev.', 'Flop Standard Dev.
↪']].
                                                                    values.
↪T,error_kw=dict(ecolor='DarkBlue'));
fig.subplots_adjust(left=1,right=3)
```

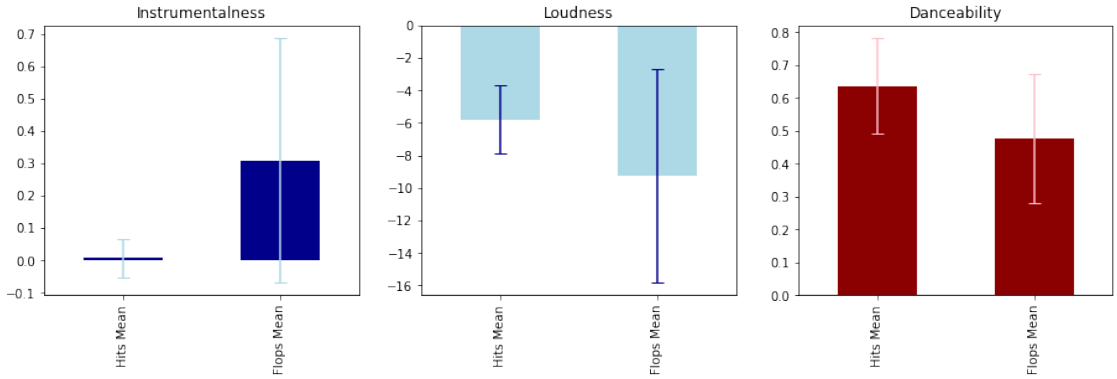


Above are the 5 categories that show the biggest difference between Hits and Flops. These categories all make sense; in fact, they might be 5 of the categories I would most expect to show differences. Based on the standard error margins, it appears that the three most seperable categories are **Intrumentalness**, **Loudness** and **Danceability**, plotted more visibly below.

[209]:
```
fig, (ax1, ax2,ax3) = plt.subplots(1, 3)
recentTrackData.loc['instrumentalness'][['Hits Mean','Flops Mean']].
↪plot(kind='bar',color='DarkBlue', ax=ax1,capsize=5,
                                                                    ␣
↪title="Instrumentalness", subplots = False,
```

```
                                                                    yerr =␣
↪recentTrackData.loc['instrumentalness'][['Hits Standard Dev.', 'Flop␣
↪Standard Dev.']].
                                                                    values.
↪T,error_kw=dict(ecolor='LightBlue'));
recentTrackData.loc['danceability'][['Hits Mean','Flops Mean']].
↪plot(kind='bar',color='DarkRed', ax=ax3,capsize=5,
                                                                title =␣
↪"Danceability", subplots = False,
                                                                yerr =␣
↪recentTrackData.loc['danceability'][['Hits Standard Dev.', 'Flop Standard␣
↪Dev.']].
                                                                values.
↪T,error_kw=dict(ecolor='Pink'));

recentTrackData.loc['loudness'][['Hits Mean','Flops Mean']].
↪plot(kind='bar',color='LightBlue',ax=ax2,capsize=5,
                                                                    ␣
↪title="Loudness", subplots = False,
                                                                yerr =␣
↪recentTrackData.loc['loudness'][['Hits Standard Dev.', 'Flop Standard Dev.
↪']].
                                                                values.
↪T,error_kw=dict(ecolor='DarkBlue'));
fig.subplots_adjust(left=1,right=3)
```



We can see here that these categories all show large differences in their means.

1.2.1 So, what kind of interesting things can we say about the differences hit songs vs. non hit songs?

- First, we can say that hit songs appear to be **more instrumental** than non-hit songs. Although there is a large error spread of non-hit songs, we can notice that the mean value

for hit songs is close to 0, with a small error margin. This tells us that although non-hit songs range from intrumental to non-instrumental, hit songs are almost never instrumental. From the variable descriptions: "The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content". So, almost all hit songs contain vocal content, while non-hit songs have a larger range from vocal content to no vocal content.

- Another conclusion we can make from this data set is that hit songs appear to be **louder** than non-hit songs on average. They have a mean of about 3.5 dB higher, and have a standard error of about 4.5 dB less. This may help to explain why pop music is sometime described as headache-inducing. While non-hit songs range from quieter to louder, hit songs have a smaller range around their mean, which is louder. *Note: dB levels are negative because when dealing with sound recording, 0 dB usually refers to the loudest level before distortion begins. More reading can be done here*
- Finally, we can observe that hit songs appear to have a higher average of **danceability** in our data set. From the data description: "Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity". The data for hits and non-hits appear to have similar standard error ranges, but the mean of hit songs is higher by about .15. This is a percentage value, so in this data set, danceability was about 15% higher for hit songs than non-hit songs on average.

We conclude Part 2 with the answers above - **the most obvious seperators in this dataset between hits and non-hit songs are instrumentalness, volume, and danceability**. If we wanted to develop predictors that would best help us decide whether a song would become a hit, these categories would be a good place to start, and if we were looking to write a hit song we would likely want to make sure we were withing the standard error ranges of these categories as well.

1.3 Part 3: Does our "formula" work? Which songs best fit our pop music formula?

Now that we've learned more about the different categories, let's see how well we can distinguish hits vs. flops by using the means and standard deviations of the hits as "ranges" for what makes a hit.

Create columns in the DataFrame for the "ranges" of hits using the formula:

$$(mean - stdev, mean + stdev)$$

[154]:
```
recentTrackData["Range Low"] = recentTrackData["Hits Mean"] -␣
 ↪recentTrackData["Hits Standard Dev."]
recentTrackData["Range High"] = recentTrackData["Hits Mean"] +␣
 ↪recentTrackData["Hits Standard Dev."]
recentTrackData[["Range Low","Hits Mean","Range High"]].drop(["year","target"])
```

[154]:

	Range Low	Hits Mean	Range High
danceability	0.490488	0.636089	0.781691
energy	0.531586	0.697796	0.864007
key	1.690806	5.282967	8.875127
loudness	-7.906449	-5.780778	-3.655108

mode	0.207007	0.675306	1.143604
speechiness	-0.000671	0.102863	0.206398
acousticness	-0.039474	0.155394	0.350262
instrumentalness	-0.052203	0.007397	0.066998
liveness	0.040345	0.185154	0.329962
valence	0.300588	0.522516	0.744444
tempo	92.687093	121.973553	151.260012
duration_ms	184813.213306	229289.262592	273765.311877
time_signature	3.712728	3.974735	4.236742
chorus_hit	21.452724	39.109636	56.766547
sections	7.708843	10.170008	12.631173

Now, using the 5 categories from Part 2 that we found to show the biggest seperation between hits and flops, find all songs from the 2000's and after that fit inside ranges.

[198]:
```
inSpecificRanges = recentTracks.loc[(recentTracks.danceability >␣
↪recentTrackData.loc['danceability']["Range Low"]) & (recentTracks.
↪danceability < recentTrackData.loc['danceability']["Range High"]) &
(recentTracks.loudness > recentTrackData.loc['loudness']["Range Low"]) &␣
↪(recentTracks.loudness < recentTrackData.loc['loudness']["Range High"]) &
(recentTracks.acousticness > recentTrackData.loc['acousticness']["Range Low"])␣
↪& (recentTracks.acousticness < recentTrackData.loc['acousticness']["Range␣
↪High"]) &
(recentTracks.instrumentalness > recentTrackData.loc['instrumentalness']["Range␣
↪Low"]) & (recentTracks.instrumentalness < recentTrackData.
↪loc['instrumentalness']["Range High"]) &
(recentTracks.valence > recentTrackData.loc['valence']["Range Low"]) &␣
↪(recentTracks.valence < recentTrackData.loc['valence']["Range High"])]
inSpecificRanges[["track","artist"]]
```

[198]:

	track	artist
1	On The Hotline	Pretty Ricky
11	Amarillo Sky	Jason Aldean
16	Daddy Won't Sell The Farm	Montgomery Gentry
17	Move Along	The All-American Rejects
34	Bed	J. Holiday
…	…	…
6366	Metal Machine	U.D.O.
6371	Whiskey In My Water	Tyler Farr
6390	Tear In My Heart	twenty one pilots
6392	Untouchable	YoungBoy Never Broke Again
6395	Teenage Dream	Katy Perry

[2250 rows x 2 columns]

What percent of the songs in these ranges are hits, and what percent are not?

[156]:
```
print("Hits:",len(inSpecificRanges[inSpecificRanges.target ==␣
↪1]),round((len(inSpecificRanges[inSpecificRanges.target == 1])/
↪len(inSpecificRanges)),3)*100,"%")
print("Flops:",len(inSpecificRanges[inSpecificRanges.target ==␣
↪0]),round((len(inSpecificRanges[inSpecificRanges.target == 0])/
↪len(inSpecificRanges)),3)*100,"%")
```

```
Hits: 1751 77.8 %
Flops: 499 22.2 %
```

So, we've seperated hits from flops from all songs after the 2000's with 77.8% accuracy. What percentage of the hits from this time period did we extract using this method?

[157]:
```
print(len(inSpecificRanges[inSpecificRanges.target == 1]),"out␣
↪of",len(recentHits),",",round((len(inSpecificRanges[inSpecificRanges.target␣
↪== 1])/len(recentHits)),3))
```

```
1751 out of 6135 , 0.285
```

Using this range method, we successfully selected about 28.5% of the hits. This may seem low, but this means that 28.5% of all songs that reached the top 100 charts since 2000 fall into the ranges we've created from 5 categories of music. Looking at it this way, this seems like a fairly good way to pick out songs.

If we run this code on all of the songs since the 1960's, is it still such a good estimation?

[193]:
```
allInSpecificRanges = songs.loc[(songs.danceability > recentTrackData.
↪loc['danceability']["Range Low"]) & (songs.danceability < recentTrackData.
↪loc['danceability']["Range High"]) &
(songs.loudness > recentTrackData.loc['loudness']["Range Low"]) & (songs.
↪loudness < recentTrackData.loc['loudness']["Range High"]) &
(songs.acousticness > recentTrackData.loc['acousticness']["Range Low"]) &␣
↪(songs.acousticness < recentTrackData.loc['acousticness']["Range High"]) &
(songs.instrumentalness > recentTrackData.loc['instrumentalness']["Range Low"])␣
↪& (songs.instrumentalness < recentTrackData.loc['instrumentalness']["Range␣
↪High"]) &
(songs.valence > recentTrackData.loc['valence']["Range Low"]) & (songs.valence␣
↪< recentTrackData.loc['valence']["Range High"])]
```

[195]:
```
print("Extracted Hits:",len(allInSpecificRanges[allInSpecificRanges.target ==␣
↪1]),round((len(allInSpecificRanges[allInSpecificRanges.target == 1])/
↪len(allInSpecificRanges)),3),"%")
print("Extracted Flops:",len(allInSpecificRanges[allInSpecificRanges.target ==␣
↪0]),round((len(allInSpecificRanges[allInSpecificRanges.target == 0])/
↪len(allInSpecificRanges)),3),"%")
```

```
Extracted Hits: 2665 0.787 %
Extracted Flops: 722 0.213 %
```

By percentages, our model actually picks out hits better when we run it on all of the data. But, when we check how many of the hits we extracted from the total number of hits:

[160]:
```
print(len(allInSpecificRanges[allInSpecificRanges.target == 1]),"out␣
↪of",len(hits),",",round((len(allInSpecificRanges[allInSpecificRanges.target␣
↪== 1])/len(hits)),3))
```

```
2665 out of 20553 , 0.13
```

We can see that we did not pull as high of a percentage of the hits out of the full data set using this model.

To find which songs are closest to the pop song "formula" we've created, we'll narrow the ranges by using a new model:

$$(mean - \frac{stdev}{4}, mean + \frac{stdev}{4})$$

The remaining songs will be the ones closest to the means of our top 5 categories, or the ones that "best fit" the formula we've created.

[166]:
```
recentTrackData["Smaller Range Low"] = recentTrackData["Hits Mean"] -␣
↪(recentTrackData["Hits Standard Dev."]/4)
recentTrackData["Smaller Range High"] = recentTrackData["Hits Mean"] +␣
↪(recentTrackData["Hits Standard Dev."]/4)
recentTrackData[["Smaller Range Low","Hits Mean","Smaller Range High"]].
↪drop(["year","target"])
```

[166]:

	Smaller Range Low	Hits Mean	Smaller Range High
danceability	0.599689	0.636089	0.672490
energy	0.656244	0.697796	0.739349
key	4.384927	5.282967	6.181007
loudness	-6.312196	-5.780778	-5.249361
mode	0.558231	0.675306	0.792380
speechiness	0.076980	0.102863	0.128747
acousticness	0.106677	0.155394	0.204111
instrumentalness	-0.007503	0.007397	0.022298
liveness	0.148951	0.185154	0.221356
valence	0.467034	0.522516	0.577998
tempo	114.651938	121.973553	129.295167
duration_ms	218170.250270	229289.262592	240408.274913
time_signature	3.909233	3.974735	4.040237
chorus_hit	34.695408	39.109636	43.523864
sections	9.554717	10.170008	10.785299

[191]:
```
inSmallerRanges = recentTracks.loc[(recentTracks.danceability > recentTrackData.
↪loc['danceability']["Smaller Range Low"]) & (recentTracks.danceability <␣
↪recentTrackData.loc['danceability']["Smaller Range High"]) &
```

```
(recentTracks.loudness > recentTrackData.loc['loudness']["Smaller Range Low"])␣
↪& (recentTracks.loudness < recentTrackData.loc['loudness']["Smaller Range␣
↪High"]) &
(recentTracks.acousticness > recentTrackData.loc['acousticness']["Smaller Range␣
↪Low"]) & (recentTracks.acousticness < recentTrackData.
↪loc['acousticness']["Smaller Range High"]) &
(recentTracks.instrumentalness > recentTrackData.
↪loc['instrumentalness']["Smaller Range Low"]) & (recentTracks.
↪instrumentalness < recentTrackData.loc['instrumentalness']["Smaller Range␣
↪High"]) &
(recentTracks.valence > recentTrackData.loc['valence']["Smaller Range Low"]) &␣
↪(recentTracks.valence < recentTrackData.loc['valence']["Smaller Range␣
↪High"])]
inSmallerRanges[["track","artist"]]
```

[191]:

	track	artist
2659	Don't Say No, Just Say Yes	Avant
5121	Kick Push	Lupe Fiasco
2879	X	Chris Brown
4118	Wake Up	Fetty Wap
4797	Gone, Gone, Gone	Phillip Phillips

The 5 songs above are the ones that best fit our pop song model from the 2000's and after. All of them ended up making the top charts. If we run this model on all of the data:

[190]:
```
allInSmallerRanges = songs.loc[(songs.danceability > recentTrackData.
↪loc['danceability']["Smaller Range Low"]) & (songs.danceability <␣
↪recentTrackData.loc['danceability']["Smaller Range High"]) &
(songs.loudness > recentTrackData.loc['loudness']["Smaller Range Low"]) &␣
↪(songs.loudness < recentTrackData.loc['loudness']["Smaller Range High"]) &
(songs.acousticness > recentTrackData.loc['acousticness']["Smaller Range Low"])␣
↪& (songs.acousticness < recentTrackData.loc['acousticness']["Smaller Range␣
↪High"]) &
(songs.instrumentalness > recentTrackData.loc['instrumentalness']["Smaller␣
↪Range Low"]) & (songs.instrumentalness < recentTrackData.
↪loc['instrumentalness']["Smaller Range High"]) &
(songs.valence > recentTrackData.loc['valence']["Smaller Range Low"]) & (songs.
↪valence < recentTrackData.loc['valence']["Smaller Range High"])]
allInSmallerRanges[["track","artist"]]
```

[190]:

year	track	artist
1960	Make Believe	Wind
1960	Meu Sonho de Amor	Leno e Lilian
1970	Rolene	Moon Martin
1970	Just Too Many People	Melissa Manchester
1970	Tried To Love	Peter Frampton

1980	¿Hasta Cuando?	Beatriz Adriana
1990	Go Away	Lorrie Morgan
1990	Breakfast At Tiffany's	Deep Blue Something
2000	Don't Say No, Just Say Yes	Avant
2000	Kick Push	Lupe Fiasco
2010	X	Chris Brown
2010	Wake Up	Fetty Wap
2010	Gone, Gone, Gone	Phillip Phillips

We end up adding quite a few songs, even though this is "extrapolating" in a way, since we created this pop song model from only songs after 2000. Note that using all songs, we pulled two songs that did not end up making the top 100 lists (the 0's in the **target** column).

In conclusion of part 3, by creating intervals based on the mean and standard deviations of categories, we were able to find songs that were closest to our "formula" for a pop song. Using the top 5 categories found in part 2, we were able to separate hit songs from flop songs in our data set of songs after 2000 with near 80 percent accuracy while selecting almost 30 percent of the total hits. By narrowing these intervals, we found the few songs that best fit our pop song formula from after 2000, as well as from all years in the data set. The 5 songs from after 2000 are shown below. All reached the top 100 list.

[168]:
```
inSmallerRanges[["track","artist"]]
```

[168]:

	track	artist
2659	Don't Say No, Just Say Yes	Avant
5121	Kick Push	Lupe Fiasco
2879	X	Chris Brown
4118	Wake Up	Fetty Wap
4797	Gone, Gone, Gone	Phillip Phillips

1.4 Part 4: How has popular music changed from 1960 to now?

To explore how popular music has changed the most, we will use a similar process to how we separated hits from flops. First, we'll visualize all of the data, then find the ones with the largest change from 1960 to now. For this data set, we'll be using only the hit songs, since we really only want to explore how pop songs have changed and not music as a whole (and to minimize variation among year). First, we'll calculate means and standard deviations for every category for every decade, and put them into a dataframe.

[171]:
```
songs = songs.set_index(["year"])
```

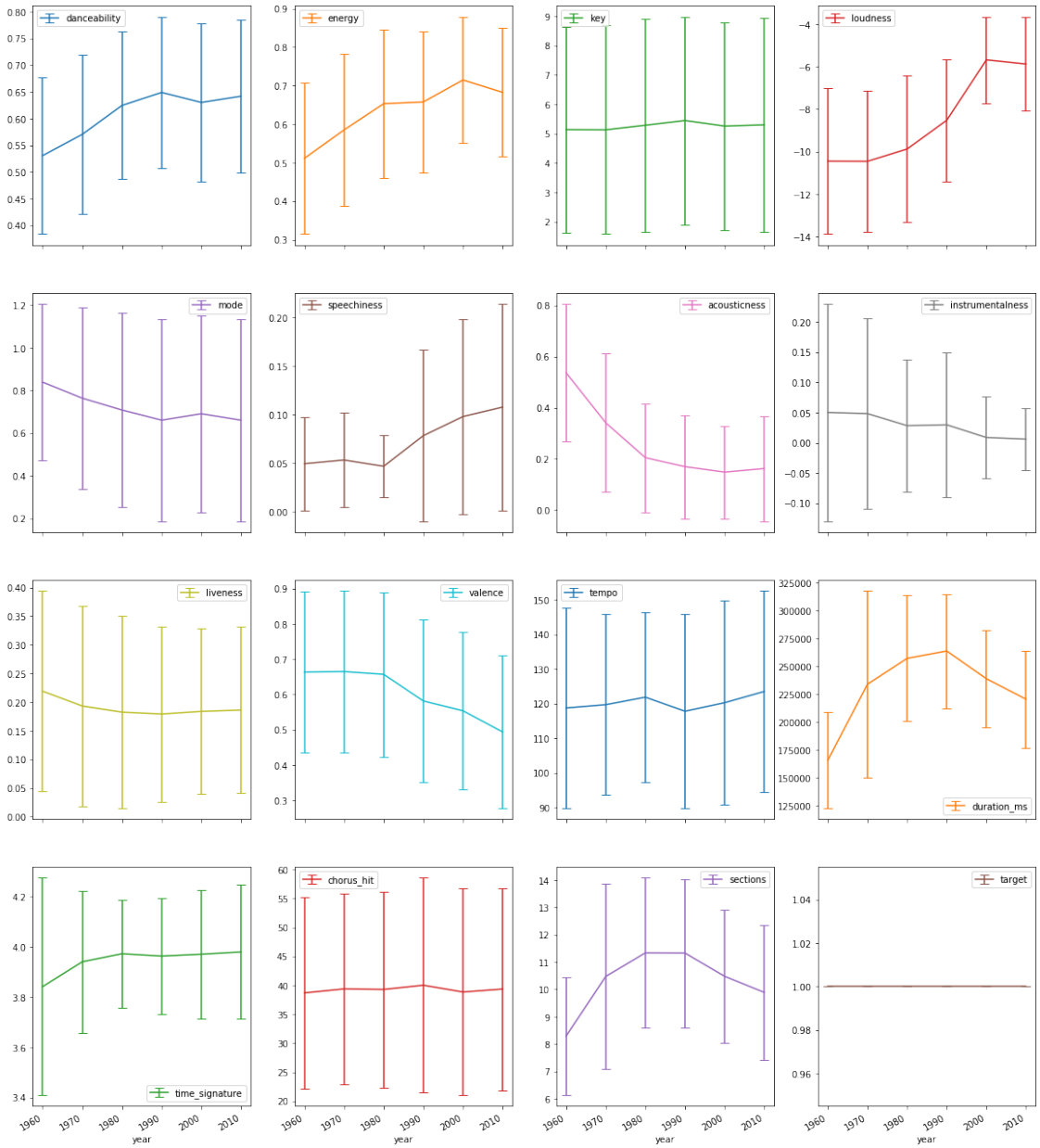
[189]:
```
songsData = pd.DataFrame()
means = songs.loc[songs.target==1].mean(level=0)
means["type"]="mean"
means = means.reset_index()
std = songs.loc[songs.target==1].std(level=0)
std = std.reset_index()
```

```
std['type']='std'

songsData = pd.concat([means,std])
songsData = songsData.set_index(["year"])
```

Now, lets' visualize all of the data and check which categories look like they have the biggest changes.

[174]:
```
songsData[songsData.type=="mean"].plot(layout =␣
↪(4,4),subplots=True,figsize=(20,25), yerr=songsData[songsData.
↪type=="std"],capsize=5);
```



In the above plot, some categories stand out as showing obvious differences, like speechiness, acousticness, and loudness. Below, we'll calculate the percent change for each category from the 1960's to the 2010's to find which categories changed the most according to the data.

Aside: We can make some interesting observations from the plots above, even among the categories which do not show the most change. For instance: - There is a shrinking variation in **time signature** and a lightly growing mean as we move from left to right. We can conclude from this that there were more songs in the 1960's with time signatures other than $\frac{4}{4}$, and in particular, pop songs were more likely to be in $\frac{3}{4}$ time. - The mean of **mode** is shrinking slightly from left to right, while the variation appears to be growing or at least moving similarly. This tells us that *hit songs throughout time are more likely to be in minor keys than they were in 1960.* - The **duration** of songs has undergone quite a bit of change through the years. this is one of the only categories that shows change from left to right, then reverts back closer to the original value, showing that *the average length of songs got longer towards the 1980's, then shorter again towards the 2000's.* Unsurprisingly, the **sections** plot looks very similar.

What categories had the largest sum percent change?

[186]:
```
abs(songsData.loc[songsData["type"]=="mean"].drop(columns=["type"]).
↪pct_change(periods=5).loc[2010]).sort_values(ascending=False)
```

[186]:
```
speechiness          1.180313
instrumentalness     0.879447
acousticness         0.697884
loudness             0.437707
energy               0.334959
duration_ms          0.330880
valence              0.255660
mode                 0.212289
danceability         0.209300
sections             0.192980
liveness             0.149957
tempo                0.040271
time_signature       0.035878
key                  0.031813
chorus_hit           0.017158
target               0.000000
Name: 2010, dtype: float64
```

[182]:
```
abs(songsData.loc[songsData["type"]=="mean"].drop(columns=["type"]).
↪pct_change().sum(axis=0)).sort_values(ascending=False)
```

[182]:
```
instrumentalness     1.420308
speechiness          0.981778
acousticness         0.967364
```

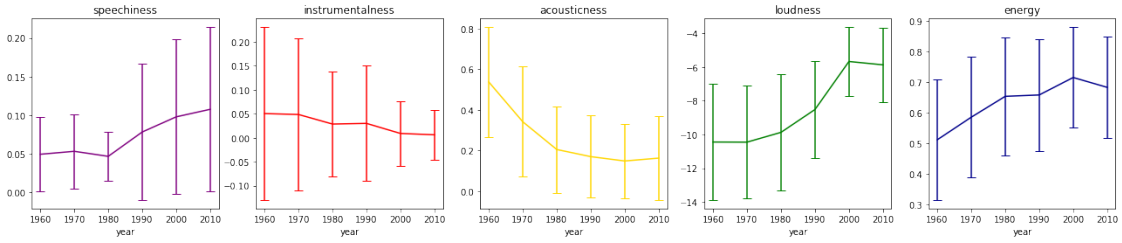
```
loudness            0.490788
duration_ms         0.365600
energy              0.309337
valence             0.280534
mode                0.227237
sections            0.213911
danceability        0.198280
liveness            0.152755
tempo               0.040818
time_signature      0.035615
key                 0.032875
chorus_hit          0.017868
target              0.000000
dtype: float64
```

Plot the top 5 categories that showed the largest total percent change below.

[187]:
```
fig, (ax1, ax2,ax3, ax4,ax5) = plt.subplots(nrows=1, ncols=5)

songsData[songsData.type=="mean"]["instrumentalness"].plot(yerr=songsData.
 ↪loc[songsData.type ==␣
 ↪'std']["instrumentalness"],capsize=5,ax=ax2,title="instrumentalness",color="r");
 ↪
songsData[songsData.type=="mean"]["acousticness"].plot(yerr=songsData.
 ↪loc[songsData.type ==␣
 ↪'std']["acousticness"],capsize=5,ax=ax3,title='acousticness',color="gold");
songsData[songsData.type=="mean"]["loudness"].plot(yerr=songsData.loc[songsData.
 ↪type == 'std']["loudness"],capsize=5,ax=ax4,title='loudness',color="green");
songsData[songsData.type=="mean"]["energy"].plot(yerr=songsData.loc[songsData.
 ↪type == 'std']["energy"],capsize=5,ax=ax5,title='energy',color="darkblue");
songsData[songsData.type=="mean"]["speechiness"].plot(yerr=songsData.
 ↪loc[songsData.type ==␣
 ↪'std']["speechiness"],capsize=5,ax=ax1,title='speechiness',color="purple");

fig.subplots_adjust(right=3)
```



These 5 categories tell the best story about how pop music has changed over the last 50 years. To conclude part 4, we'll analyze each of the changes:

- **Speechiness** is the category that has changed the most over the last 50 years. From the data discription, "Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value." As we can see in the plot, over time not only did the mean grow, but so did the deviation. This shows that more songs contained more spoken words over time. The growth of this category is likely due to rap music growing in the top charts. As we can see, the largest jump happens in the 1990's, which is when rap music first began to imerge as a popular genre [*source*]
- **Instrumentalness** is a category that "Predicts whether a track contains no vocals." Instrumentalness was never very high in the dataset, but from its starting point at around 5% of the songs, it has shrunk to almost 0 over the years with a much smaller deviation.
- **Acousticness** is another category between 1 and 0, songs closer to 1 being more acoustic and songs closer to 0 being less acoustic. Our data shows that acousticness has decreased significantly from the 60's to the 10's. This tells us that pop songs today have much less of an acoustic presence, which would be expected due to the emergence of better music technology and accessability of electronic instruments.
- **Loudness** is the category that showed the most difference between hit songs and non-hit songs. It also shows up in the top 5 for total percentage difference over the last 50 years. This shows that the loudness in pop music is not something that has always been true, but that the volume of pop music has actually risen over time. This phenomena is very well documented, and descibed in the Wikipedia Article The Loudness War
- **Energy**. Energy is an interesting category, because it initially shows an increase, but then a decrease from the 2000's to the 2010's. Nevertheless, it rounds off the top 5 largest percentage change categories.

1.4.1 Conclusion

We've learned some interesting things from analyzing this dataset of 41,000 hit and non-hit songs. In particular, we've leared that Hit songs are different than non-hit songs on average; specifically, hit songs are louder, less instrumental, and more danceable on average than non-hit songs. Using these categories, we can separate hit songs from non hit songs with about 80% accuracy while selecting almost 30% of the total hit songs, showing that our "model" does a fairly good job at selecting which songs end up as hits and which don't. Using this model and shrinking the confidence interval, we can select the songs that best fit the model we've created, shown in the conclusion of part 3. Finally, we examined how music has changed from the 1960's to the 2010's. We found categories that have changed the most over time, and they included Speechiness, Instrumentalness, Acousticness, Loudness, and Energy.

Data: Spotify hit predictor: https://www.kaggle.com/theoverman/the-spotify-hit-predictor-dataset

1.4.2 Data Category Description:

---

track - The Name of the track.

artist - The Name of the first artist listed for the track.

uri - The resource identifier for the track.

danceability - Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.

energy - Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.

key - The estimated overall key of the track. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C?/D?, 2 = D, and so on. If no key was detected, the value is -1.

loudness - The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typical range between -60 and 0 db.

mode - Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.

speechiness - Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.

acousticness - A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.

instrumentalness - Predicts whether a track contains no vocals. “Ooh” and “aah” sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly “vocal”. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.

liveness - Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live. valenceA measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).

tempo - The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration. duration_msThe duration of the track in milliseconds.

time_signature - An estimated overall time signature of a track. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure).

chorus_hit - This the the author’s best estimate of when the chorus would start for the track. Its the timestamp of the start of the third section of the track (in milliseconds). This feature was extracted from the data recieved by the API call for Audio Analysis of that particular track.

sections - The number of sections the particular track has. This feature was extracted from the data recieved by the API call for Audio Analysis of that particular track.

target - The target variable for the track. It can be either '0' or '1'. '1' implies that this song has featured in the weekly list (Issued by Billboards) of Hot-100 tracks in that decade at least once and is therefore a 'hit'. '0' Implies that the track is a 'flop'. The author's condition of a track being 'flop' is as follows:

```
- The track must not appear in the 'hit' list of that decade.
- The track's artist must not appear in the 'hit' list of that decade.
- The track must belong to a genre that could be considered non-mainstream and / or avant-g
- The track's genre must not have a song in the 'hit' list.
- The genre list for the particular decades are as follows:
- The track must have 'US' as one of its markets.
```