DS121 Final Project - Sahir Doshi

Introduction

Music has changed significantly over the decades, seeing many shifts in instruments, energy, genre, and more. Today, the music is industry is the strongest its every been, with a plethora of artists representing all genders, races, and backgrounds.

Like many others, music played an instrumental part (pun intended) on my life from a young age. One of my earliest memories is tapping my fingers to the beat of a song around the age of 5. Throughout my life, I have explored music through a multitude of ways: electric guitar, piano, drums, singing, rapping, beat production, and djing. For me, music has been a form of expression and a way to escape reality, entering a place of pure imagination and no boundaries.

Ultimatly, I decided to pursue a career in STEM out of interest for the job security and higher employment rate in STEM fields, but I have always wondered, what would life have been like if I did pursue music?

Well, this project has provided me with the perfect opportunity to explore that, but from a statistical perspective.

For my final project, I have analyzed the **Top 50 songs on Spotify from 2019**, utilizing various data analysis techniques to deduce how a song's popularity is dictated by its properties. By the end of this deep dive, I hope you will walk away with a new understanding of how music works in this day age, specifically about the components of a song and how they impact a song's success. Personally, I hope to learn how to predict a song's popularity, so that one day if I embark on a career in music, I will have a statistical advantage on achieving success.

Data

As mentioned above, my dataset covers the Top 50 Songs on Spotify for the year of 2019:

- This dataset comes from Kaggle (https://www.kaggle.com/datasets/leonardopena/top50spotify2019?resource=download), and is actually a subset of another Kaggle dataset of the Top Songs on Spotify by year for the period of 2019 - 2019 (https://www.kaggle.com/datasets/leonardopena/top-spotify-songs-from-20102019-by-year).
- The actual data points come from a reputable website called "Organize Your Music" (http://organizeyourmusic.playlistmachinery.com/), which can take in various types of

Spotify playlists (which you can choose), and outputs information about the properties of each song in the playlist.

- One thing I could not find was the original playlist used. The closest source I could find was the *Top Hits of 2019* playlist (https://open.spotify.com/playlist/37i9dQZF1DWVRSukIED0e9?si=1e4c048e8e014360), which includes the Top 100 songs of 2019. My best guess is that the Top 50 dataset removed any songs which were ranked 51-100.

Specifically, the properties of a song are as follows:

- Track.Name (Name of song)
- Artist.Name (Artist of song)
- Genre (Genre)
- Beats.Per.Minute (Tempo)
- Energy (Higher the value, more energetic song is, and vice versa)
- Danceability (Higher the value, easier it is to dance to the song, and vice versa)
- Loudness..dB.. (Higher the value, louder the song, and vice versa)
- Liveness (Higher the value, more likely the song is a live recording, and vice versa)
- Valence (Higher the value, more positive the song is, and vice versa)
- Length (Duration of song in seconds)
- Acousticness (Higher the value, more acoustic song is, and vice versa)
- Speechiness (Higher the value, more spoken word the song contains, and vice versa)
- Popularity (Higher the value, more popular the song is, and vice versa)

The raw form of the dataset:

```
import pandas as pd

df = pd.read_csv("top50.csv", encoding = "ISO-8859-1")
df
```

	Unnamed: 0	Track.Name	Artist.Name	Genre	Beats.Per.Minute	Energy	D
0	34	The London (feat. J. Cole & Travis Scott)	Young Thug	atl hip hop	98	59	
1	27	Dance Monkey	Tones and I	australian pop	98	59	
2	36	Summer Days (feat. Macklemore & Patrick Stump ...	Martin Garrix	big room	114	72	
3	39	Sucker	Jonas Brothers	boy band	138	73	
4	46	One Thing Right	Marshmello	brostep	88	62	
5	48	Happier	Marshmello	brostep	100	79	
6	15	Money In The Grave (Drake ft. Rick Ross)	Drake	canadian hip hop	101	50	
7	19	Lalala	Y2K	canadian hip hop	130	39	
8	28	It's You	Ali Gatie	canadian hip hop	96	46	
9	1	Señorita	Shawn Mendes	canadian pop	117	55	
10	26	If I Can't Have You	Shawn Mendes	canadian pop	124	82	
11	9	Old Town Road - Remix	Lil Nas X	country rap	136	62	
12	22	Panini	Lil Nas X	country rap	154	59	
13	3	boyfriend (with Social House)	Ariana Grande	dance pop	190	80	
14	12	Loco Contigo (feat. J. Balvin & Tyga)	DJ Snake	dance pop	96	71	
15	16	No Guidance (feat. Drake)	Chris Brown	dance pop	93	45	
16	32	7 rings	Ariana Grande	dance pop	140	32	

Note: Pandas automatically adds an index column to a dataframe, however, the dataset has its own index column titled "Unnamed: 0"

Note: due to the nature of the .csv file's encoding, I needed to specify the type of encoding in my read_csv method so Python could successfully read in the .csv file

Methodology

My initial thought was to compute a multivariable linear regression on the dataset to create a formula to predict a song's popularity. As I stated in the *Introduction* section, as someone who has been involved in music for almost my whole life, I was interested in seeing which traits impacted a song's popularity so that one day I might be able to make a song that becomes popular (based on the calculation metrics of *Organize Your Music*).

However, rather than just do one regression, I decided to analyze this dataset far deeper.

- 1) Calculate a multivariable regression using the Lasso technique to predict a song's popularity
- 2) Caluclate a multivariable regression using *all* variables and then compare regression models
- 3) Conduct Explanatory Data Analysis (EDA) on the dataset by looking at genre of songs to better understand which genres become more popular
- 4) Finding the "optimum" values for each of the variables in the Lasso-based regression to create the highest predicted popularity

After I conduct these 4 analyses, I hope to be more educatated in the anatomy of a song and its relation to success.

Analysis

Data Cleansing

```
# removing the "Unamed: 0" column, as it is a redundant column
df = df.drop(["Unnamed: 0"], axis = 1)
df.head(5)
```

	Track.Name	Artist.Name	Genre	Beats.Per.Minute	Energy	Danceability	Lo
0	The London (feat. J. Cole	Young Thug	atl hip	98	59	80	
1	[illegible]	Tones and I	[illegible]	98	59	82	

Basis Statistics of Dataset

```
# summary statistics on all numerical categories
df.describe()
```

	Beats.Per.Minute	Energy	Danceability	Loudness..dB..	Liveness	Vale
count	50.000000	50.000000	50.00000	50.000000	50.000000	50.00
mean	120.060000	64.060000	71.38000	-5.660000	14.660000	54.60
std	30.898392	14.231913	11.92988	2.056448	11.118306	22.33
min	85.000000	32.000000	29.00000	-11.000000	5.000000	10.00
25%	96.000000	55.250000	67.00000	-6.750000	8.000000	38.25
50%	104.500000	66.500000	73.50000	-6.000000	11.000000	55.50
75%	137.500000	74.750000	79.75000	-4.000000	15.750000	69.50
max	190.000000	88.000000	90.00000	-2.000000	58.000000	95.00

Analysis 1: Best Linear Regression Model

```
# setting up the regression variables, including adding dummies
# for categorical variables and adding a constant

import statsmodels.api as sm

Y = df["Popularity"]
X = df.drop(["Track.Name", "Artist.Name", "Popularity"], axis = 1)
X = pd.get_dummies(X)
X = sm.add_constant(X)
X.head()
```

Lasso Machine Learning Model:

The Lasso technique is a machine learning model that runs a form of "penalized regression". This regression tries to simulataneously accomplish 2 things:

- Minimize Residuals
- Minimize the Sum of Coefficients (to kick out less useful variables by setting them to 0)

In order to do this, Lasso takes in 1 parameter which aims to negotiate a tradeoff between the two tasks above. This parameter is called **alpha**, and stands for **how much weight we give to the sum of coefficients (which we are trying to minimize)**. For example:

- When alpha = 0, we are running a standard, simple linear regression
- When alpha = 1, we are weighting residuals and sum of coefficients equally

In order to figure out what the optimal alpha level is, we should test every possible alpha level (so every decimal value from 0 to 0.1). To do this, we are first going to create a model at each alpha level from 0 to 1 with 0.1 increases (e.g. Model 1 alpha = 0, Model 2 alpha = 0.1, Model 3 alpha = 0.2,...)

```
# implementing the Lasso technique to find the best predictor variables for the reg

from sklearn.linear_model import Lasso

a_lvl = 0

for x in range(11):
    model = Lasso(alpha = a_lvl).fit(X, Y)
    Xlasso = X.iloc[:,model.coef_ != 0]
    Xlasso = sm.add_constant(Xlasso)
    postlasso = sm.OLS(Y, Xlasso).fit()

    print("Alpha = {}".format(a_lvl))
    print("")
    print(postlasso.summary())
    print("")

    a_lvl = a_lvl + 0.1
```

```
Alpha = 0

                            OLS Regression Results
==============================================================================
Dep. Variable:             Popularity   R-squared:                       0.7
Model:                            OLS   Adj. R-squared:                  0.4
Method:                 Least Squares   F-statistic:                     2.1
Date:                Mon, 12 Dec 2022   Prob (F-statistic):             0.03
Time:                        19:53:40   Log-Likelihood:                -110.
No. Observations:                  50   AIC:                             280
Df Residuals:                      20   BIC:                             337
Df Model:                          29
```

```
Covariance Type:            nonrobust
==================================================================================
                              coef    std err          t      P>|t|      [0.0
----------------------------------------------------------------------------------
const                      91.1205      9.798      9.300      0.000      70.6
Beats.Per.Minute            0.0069      0.026      0.261      0.797      -0.0
Energy                      0.0470      0.076      0.615      0.546      -0.1
Danceability               -0.0304      0.064     -0.472      0.642      -0.1
Loudness..dB..             -0.4183      0.730     -0.573      0.573      -1.9
Liveness                    0.0694      0.057      1.215      0.239      -0.0
Valence.                   -0.0701      0.037     -1.901      0.072      -0.1
Length.                    -0.0393      0.027     -1.436      0.166      -0.0
Acousticness..             -0.0224      0.040     -0.561      0.581      -0.1
Speechiness.               -0.0543      0.096     -0.567      0.577      -0.2
Genre_atl hip hop           3.0220      3.785      0.798      0.434      -4.8
Genre_australian pop        1.3002      4.042      0.322      0.751      -7.1
Genre_big room              1.2407      4.252      0.292      0.773      -7.6
Genre_boy band             -1.6073      3.862     -0.416      0.682      -9.6
Genre_brostep               3.5644      2.985      1.194      0.246      -2.6
Genre_canadian hip hop      5.1258      2.296      2.232      0.037       0.3
Genre_canadian pop         -7.2562      2.742     -2.647      0.015     -12.9
Genre_country rap           3.3323      2.900      1.149      0.264      -2.7
Genre_dance pop             2.1618      1.497      1.444      0.164      -0.9
Genre_dfw rap               9.0452      2.729      3.315      0.003       3.3
Genre_edm                   1.1668      3.532      0.330      0.745      -6.2
Genre_electropop            9.0912      3.746      2.427      0.025       1.2
Genre_escape room           6.5011      3.909      1.663      0.112      -1.6
Genre_latin                 8.3240      2.237      3.721      0.001       3.6
Genre_panamanian pop       10.0083      3.974      2.519      0.020       1.7
Genre_pop                   2.3219      1.734      1.339      0.196      -1.2
Genre_pop house             4.8889      4.333      1.128      0.273      -4.1
Genre_r&b en espanol        6.7734      4.074      1.663      0.112      -1.7
Genre_reggaeton             8.8358      3.169      2.788      0.011       2.2
Genre_reggaeton flow        8.9797      4.187      2.145      0.044       0.2
Genre_trap music            4.3005      4.194      1.025      0.317      -4.4
==================================================================================
Omnibus:                        0.321   Durbin-Watson:                       2.6
Prob(Omnibus):                  0.852   Jarque-Bera (JB):                    0.4
Skew:                          -0.175   Prob(JB):                            0.8
Kurtosis:                       2.724   Cond. No.                         3.02e+
==================================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corre
```

^^ ignore this error above ^^

Based on the 11 models we created above, we see that from alpha = 0.5 to alpha = 1 there are no changes. To be exact with our alpha level to 2 decimal places, lets test over the range of alpha = 0.4 to alpha = 0.5, incrementing by 0.01 for each model, and seeing at which value does the model stop changing:

```
a_lvl = 0.4

for x in range(11):
```

```
model = Lasso(alpha = a_lvl).fit(X, Y)
Xlasso = X.iloc[:,model.coef_ != 0]
Xlasso = sm.add_constant(Xlasso)
postlasso = sm.OLS(Y, Xlasso).fit()

print("Alpha = {}".format(a_lvl))
print("")
print(postlasso.summary())
print("")

a_lvl = a_lvl + 0.01
```

```
Alpha = 0.4

                            OLS Regression Results
==============================================================================
Dep. Variable:             Popularity   R-squared:                       0.4
Model:                            OLS   Adj. R-squared:                  0.3
Method:                 Least Squares   F-statistic:                     3.6
Date:                Mon, 12 Dec 2022   Prob (F-statistic):            0.002
Time:                        19:53:47   Log-Likelihood:                -130.
No. Observations:                  50   AIC:                             281
Df Residuals:                      40   BIC:                             300
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
                        coef    std err          t      P>|t|      [0.025
------------------------------------------------------------------------------
const                88.3861      5.678     15.567      0.000      76.911
Beats.Per.Minute      0.0200      0.022      0.927      0.360      -0.024
Energy                0.0195      0.047      0.414      0.681      -0.076
Danceability          0.0045      0.046      0.098      0.923      -0.089
Liveness              0.0401      0.050      0.803      0.426      -0.061
Valence.             -0.0431      0.028     -1.540      0.131      -0.100
Length.              -0.0152      0.014     -1.055      0.298      -0.044
Acousticness..       -0.0017      0.030     -0.058      0.954      -0.062
Speechiness.          0.0392      0.060      0.659      0.514      -0.081
Genre_canadian pop  -12.1103      2.803     -4.321      0.000     -17.775
==============================================================================
Omnibus:                        1.255   Durbin-Watson:                   1.8
Prob(Omnibus):                  0.534   Jarque-Bera (JB):                1.0
Skew:                          -0.351   Prob(JB):                        0.5
Kurtosis:                       2.856   Cond. No.                     2.86e+
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corre
[2] The condition number is large, 2.86e+03. This might indicate that there
strong multicollinearity or other numerical problems.

Alpha = 0.41000000000000003

                            OLS Regression Results
==============================================================================
Dep. Variable:             Popularity   R-squared:                       0.4
Model:                            OLS   Adj. R-squared:                  0.3
Method:                 Least Squares   F-statistic:                     3.6
Date:                Mon, 12 Dec 2022   Prob (F-statistic):            0.002
Time:                        19:53:47   Log-Likelihood:                -130.
```

```
No. Observations:                  50   AIC:                             281
Df Residuals:                      40   BIC:                             300
Df Model:                           9
Covariance Type:            nonrobust
==========================================================================
                         coef    std err          t      P>|t|      [0.025
--------------------------------------------------------------------------
const                 88.3861      5.678     15.567      0.000      76.911
Beats Per Minute       0.0200      0.022      0.927      0.360      -0.024
```

So, after we further narrow down our potential alpha level, we find that, from alpha = 0.43, our model never changes. Therefore, we know that the optimal alpha level is 0.43.

Multivariate Regression General Formula (based on Lasso, alpha = 0.43):

Predicted Song Popularity = a + (b * *Beats.Per.Minute*) + (c * *Energy*) + (d * *Danceability*) + (e * *Liveness*) + (f * *Speechiness*) + (g * *Valence*) + (h * *Length*) + (i * *Acousticness*)

With Coefficients Based On This Dataset:

Predicted Song Popularity = 88.6489 + (0.0114 * *Beats.Per.Minute*) + (0.0271 * *Energy*) + (0.0138 * *Danceability*) + (0.0598 * *Liveness*) + (0.0856 * *Speechiness*) + (-0.0716 * *Valence*) + (-0.0154 * *Length*) + (-0.0079 * *Acousticness*)

Analysis 2: Linear Regression for *all* variables and dummies

```
# regression but with all variables (adding the song's genre as a predictor variabl

fullmodel = sm.OLS(Y, X).fit()
fullmodel.summary()
```

OLS Regression Results

Dep. Variable:	Popularity	**R-squared:**	0.757
Model:	OLS	**Adj. R-squared:**	0.405
Method:	Least Squares	**F-statistic:**	2.149
Date:	Mon, 12 Dec 2022	**Prob (F-statistic):**	0.0396
Time:	19:53:56	**Log-Likelihood:**	-110.18
No. Observations:	50	**AIC:**	280.4
Df Residuals:	20	**BIC:**	337.7
Df Model:	29		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	91.1205	9.798	9.300	0.000	70.682	111.559
Beats.Per.Minute	0.0069	0.026	0.261	0.797	-0.048	0.062
Energy	0.0470	0.076	0.615	0.546	-0.112	0.206
Danceability	-0.0304	0.064	-0.472	0.642	-0.165	0.104
Loudness..dB..	-0.4183	0.730	-0.573	0.573	-1.941	1.104
Liveness	0.0694	0.057	1.215	0.239	-0.050	0.189
Valence.	-0.0701	0.037	-1.901	0.072	-0.147	0.007
Length.	-0.0393	0.027	-1.436	0.166	-0.096	0.018
Acousticness..	-0.0224	0.040	-0.561	0.581	-0.106	0.061
Speechiness.	-0.0543	0.096	-0.567	0.577	-0.254	0.145
Genre_atl hip hop	3.0220	3.785	0.798	0.434	-4.874	10.918
Genre_australian pop	1.3002	4.042	0.322	0.751	-7.132	9.732
Genre_big room	1.2407	4.252	0.292	0.773	-7.628	10.109
Genre_boy band	-1.6073	3.862	-0.416	0.682	-9.663	6.448
Genre_brostep	3.5644	2.985	1.194	0.246	-2.663	9.791
Genre_canadian hip hop	5.1258	2.296	2.232	0.037	0.336	9.915
Genre_canadian pop	-7.2562	2.742	-2.647	0.015	-12.975	-1.537
Genre_country rap	3.3323	2.900	1.149	0.264	-2.717	9.381
Genre_dance pop	2.1618	1.497	1.444	0.164	-0.961	5.284
Genre_dfw rap	9.0452	2.729	3.315	0.003	3.353	14.737
Genre_edm	1.1668	3.532	0.330	0.745	-6.201	8.534
Genre_electropop	9.0912	3.746	2.427	0.025	1.278	16.905

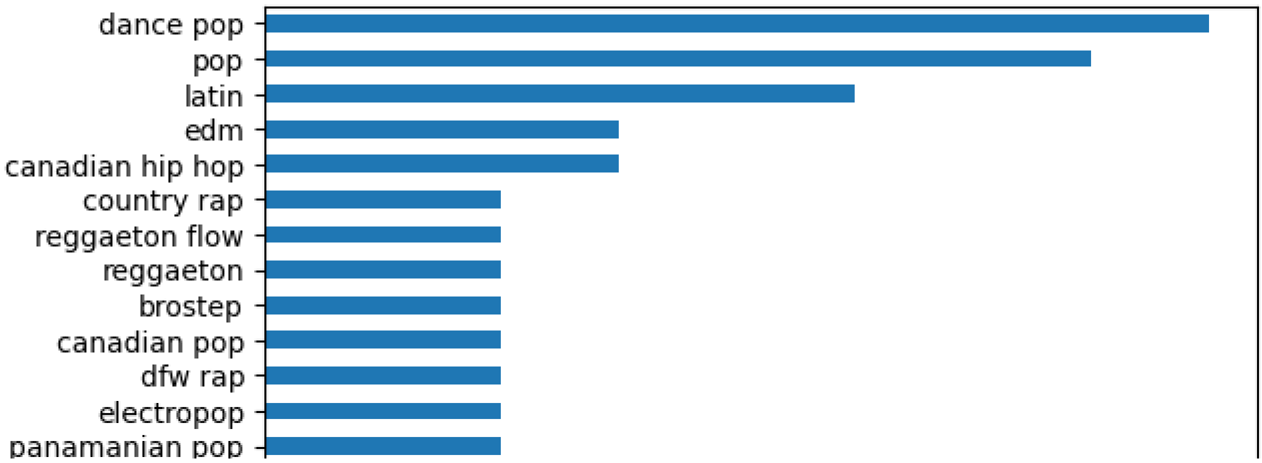
Analysis 3: EDA of *Genre* Data

Genre_panamanian pop	10.0083	3.974	2.519	0.020	1.720	18.297

```
# The following graph displays the count of each genre in the original dataset

df.Genre.value_counts().sort_values().plot(kind = 'barh')
```

<AxesSubplot: >



Regrouping Genres which are actually *Subgenres* into their *Parent Genres*

Subgenre (loose definition, interpolated from Merriam-Webster):

- is a group of music that has small changes in its traits, however, they are part of a larger "parent" genre
- ex: Hip-Hop/Rap is a very broad genre with many different interpretations from location of origin to instruments used. A rap song from Atlanta is often very different from a rap song from San Francisco.

Genre Regrouping-Process Based On:

- Personal Knowledge
  - ex: the 1 *Escape Room* song was *Truth Hurts* by the artist Lizzo, however, it is generally acknowledged that this song falls under the category of Hip-Hop/Rap
- Common Knowledge
  - e.g. every subgenre that had "pop" in it was changed to *pop* in general
  - note: the *panamanian pop* song was put in the parent genre *Spanish/Latin* since this is more accurate

```
# creating a new dataframe to alter the data points' genres
df2 = df

# regrouping genres (edm)
for i in df2.index:
    if df2.at[i, "Genre"] == "brostep":
        df2.at[i, "Genre"] = "edm"
    elif df2.at[i, "Genre"] == "big room":
        df2.at[i, "Genre"] = "edm"
    elif df2.at[i, "Genre"] == "pop house":
        df2.at[i, "Genre"] = "edm"

# regrouping genres (rap)
for i in df2.index:
    if df2.at[i, "Genre"] == "atl hip hop":
        df2.at[i, "Genre"] = "rap"
```

```
    elif df2.at[i, "Genre"] == "canadian hip hop":
        df2.at[i, "Genre"] = "rap"
    elif df2.at[i, "Genre"] == "country rap":
        df2.at[i, "Genre"] = "rap"
    elif df2.at[i, "Genre"] == "escape room":
        df2.at[i, "Genre"] = "rap"
    elif df2.at[i, "Genre"] == "trap music":
        df2.at[i, "Genre"] = "rap"
    elif df2.at[i, "Genre"] == "dfw rap":
        df2.at[i, "Genre"] = "rap"

# regrouping genres (pop)
for i in df2.index:
    if df2.at[i, "Genre"] == "australian pop":
        df2.at[i, "Genre"] = "pop"
    elif df2.at[i, "Genre"] == "boy band":
        df2.at[i, "Genre"] = "pop"
    elif df2.at[i, "Genre"] == "canadian pop":
        df2.at[i, "Genre"] = "pop"
    elif df2.at[i, "Genre"] == "dance pop":
        df2.at[i, "Genre"] = "pop"
    elif df2.at[i, "Genre"] == "electropop":
        df2.at[i, "Genre"] = "pop"

# regrouping genres (spanish/latin)
for i in df2.index:
    if df2.at[i, "Genre"] == "latin":
        df2.at[i, "Genre"] = "spanish/latin"
    elif df2.at[i, "Genre"] == "panamanian pop":
        df2.at[i, "Genre"] = "spanish/latin"
    elif df2.at[i, "Genre"] == "r&b en espanol":
        df2.at[i, "Genre"] = "spanish/latin"
    elif df2.at[i, "Genre"] == "reggaeton":
        df2.at[i, "Genre"] = "spanish/latin"
    elif df2.at[i, "Genre"] == "reggaeton flow":
        df2.at[i, "Genre"] = "spanish/latin"

df2.Genre.value_counts().sort_values().plot(kind = 'barh')
```

```
<AxesSubplot: >
```


```
# regression with parent genres

Y2 = df2["Popularity"]
X2 = df2.drop(["Popularity", "Track.Name", "Artist.Name"], axis = 1)
X2 = pd.get_dummies(X2)
X2 = sm.add_constant(X2)

model_parent_genres = sm.OLS(Y2, X2).fit()
model_parent_genres.summary()
```

OLS Regression Results

Dep. Variable:	Popularity	**R-squared:**	0.469
Model:	OLS	**Adj. R-squared:**	0.296

Analysis 4: Most Optimal Values of a Song's Traits (based on the Lasso-deduced variables)

Df Residuals:	37	**BIC:**	310.3

Goal: Optimize the Lasso-based Regression

Method:

- For positive-coefficient variables, find the max values of each variable from the original dataset
- For negative-coefficient variables, find the min values of each variable from the original dataset
- Plug in max and min values into regression to find highest possible predicted song popularity

```
# final Lasso regression
model = Lasso(alpha = 0.43).fit(X, Y)
Xlasso = X.iloc[:,model.coef_ != 0]
Xlasso = sm.add_constant(Xlasso)
postlasso = sm.OLS(Y, Xlasso).fit()

print("Alpha = {}".format(0.43))
print("")
print(postlasso.summary())
print("")
```

```
Alpha = 0.43

                            OLS Regression Results
==============================================================================
Dep. Variable:             Popularity   R-squared:                       0.196
Model:                            OLS   Adj. R-squared:                  0.039
Method:                 Least Squares   F-statistic:                     1.252
Date:                Mon, 12 Dec 2022   Prob (F-statistic):              0.295
Time:                        19:54:21   Log-Likelihood:                -140.09
No. Observations:                  50   AIC:                             298.2
Df Residuals:                      41   BIC:                             315.4
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                       coef    std err          t      P>|t|      [0.025
------------------------------------------------------------------------------
const               88.6489      6.792     13.052      0.000      74.933      1
Beats.Per.Minute     0.0114      0.026      0.444      0.660      -0.040
Energy               0.0271      0.056      0.480      0.634      -0.087
Danceability         0.0138      0.055      0.251      0.803      -0.097
Liveness             0.0598      0.060      1.004      0.321      -0.060
Valence.            -0.0716      0.033     -2.200      0.034      -0.137
Length.             -0.0154      0.017     -0.895      0.376      -0.050
```

```
Acousticness..      -0.0079      0.036     -0.220      0.827      -0.080
Speechiness.         0.0856      0.070      1.222      0.229      -0.056
==============================================================================
Omnibus:                       15.302   Durbin-Watson:                   1.555
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               20.703
Skew:                          -1.029   Prob(JB):                     3.19e-05
Kurtosis:                       5.388   Cond. No.                     2.86e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correct
[2] The condition number is large, 2.86e+03. This might indicate that there ar
strong multicollinearity or other numerical problems.
```

```
# coefficients of each variable
postlasso.params
```

```
const               88.648897
Beats.Per.Minute     0.011375
Energy               0.027052
Danceability         0.013845
Liveness             0.059784
Valence.            -0.071589
Length.             -0.015410
Acousticness..      -0.007867
Speechiness.         0.085556
dtype: float64
```

```
# explicitly stating positive-coefficient and negative-coefficient variables for cl
print("(+)-coefficient variables = Beats Per Minute, Energy, Danceability, Liveness
print("(-)-coefficient variables = Valence, Length, Acousticness")
```

```
(+)-coefficient variables = Beats Per Minute, Energy, Danceability, Liveness,
(-)-coefficient variables = Valence, Length, Acousticness
```

```
# max values of each positive-coefficient variable
max_bpm = df["Beats.Per.Minute"].max()
max_nrg = df["Energy"].max()
max_dnc = df["Danceability"].max()
max_liv = df["Liveness"].max()
max_spe = df["Speechiness."].max()

# min values of each negative-coefficient variable
min_val = df["Valence."].min()
min_len = df["Length."].min()
min_aco = df["Acousticness.."].min()

# explicitly stating min & max values for clarity
print("Max BPM = {}".format(max_bpm))
print("Max Energy = {}".format(max_nrg))
print("Max Danceability = {}".format(max_dnc))
print("Max Liveness = {}".format(max_liv))
```

```
print("Max Speechiness = {}".format(max_spe))
print("Min Valence = {}".format(min_val))
print("Min Length = {}".format(min_len))
print("Min Acousticness = {}".format(min_aco))
```

```
Max BPM = 190
Max Energy = 88
Max Danceability = 90
Max Liveness = 58
Max Speechiness = 46
Min Valence = 10
Min Length = 115
Min Acousticness = 1
```

Regression Formula:

Predicted Song Popularity =

a + (b * *Beats.Per.Minute*) + (c * *Energy*) + (d * *Danceability*) + (e * *Liveness*) + (f * *Speechiness*) + (g * *Valence*) + (h * *Length*) + (i * *Acousticness*)

which in this case =

88.6489 + (0.0114 * *Beats.Per.Minute*) + (0.0271 * *Energy*) + (0.0138 * *Danceability*) + (0.0598 * *Liveness*) + (0.0856 * *Speechiness*) + (-0.0716 * *Valence*) + (-0.0154 * *Length*) + (-0.0079 * *Acousticness*)

Therefore:

```
# computing + printing predicted song popularity based on max & min values
max_pred_song_pop = 88.6489 + (0.0114 * max_bpm) + (0.0271 * max_nrg) + (0.0138 * m
print("The Predicted Song Popularity of a Song with the Most Optimal Values for the
```

```
The Predicted Song Popularity of a Song with the Most Optimal Values for the r
```

A more accurate representation of the optimal predicted popularity of a song would be based on the average value of each variable:

```
# avg value of each variable
avg_bpm = df["Beats.Per.Minute"].mean()
avg_nrg = df["Energy"].mean()
avg_dnc = df["Danceability"].mean()
avg_liv = df["Liveness"].mean()
avg_spe = df["Speechiness."].mean()
avg_val = df["Valence."].mean()
avg_len = df["Length."].mean()
avg_aco = df["Acousticness.."].mean()

# computing + printing predicted song popularity based on avg values
avg_pred_song_pop = 88.6489 + (0.0114 * avg_bpm) + (0.0271 * avg_nrg) + (0.0138 * a
print("The Predicted Song Popularity of a Song with the Average Values for the rele
```

```
The Predicted Song Popularity of a Song with the Average Values for the releva
```

Results

Analysis 1

Based on the Lasso-regression approach, the multivariate regression is

- *Predicted Song Popularity* = a + (b * *Beats.Per.Minute*) + (c * *Energy*) + (d * *Danceability*) + (e * *Liveness*) + (f * *Speechiness*) + (g * *Valence*) + (h * *Length*) + (i * *Acousticness*)

and specific to this dataset, with, coefficients, the model is

- *Predicted Song Popularity* = 88.6489 + (0.0114 * *Beats.Per.Minute*) + (0.0271 * *Energy*) + (0.0138 * *Danceability*) + (0.0598 * *Liveness*) + (0.0856 * *Speechiness*) + (-0.0716 * *Valence*) + (-0.0154 * *Length*) + (-0.0079 * *Acousticness*)

Key takeaways from this analysis:

- The optimal alpha level is 0.43. This was discovered after creating a model at every alpha level from 0 to 1 with 0.1 increments, followed by creating a model at every alpha level from 0.4 to 0.5 with 0.01 increments. After producing these 22 models, we saw that after alpha = 0.43, there were no changes to the model.
- Generally speaking, each variable's coefficient has a similar value, meaning that all song properties influence its predicted popularity by a roughly equal amount.
- According to this model, the genre of a song **does not** matter when considering it's popularity. This is a very interesting takeaway, since initially I assumed that a genre's song would influence its popularity. Normally, there are certain genres that tend to be more popular than others, for example there is an entire genre of music called "Pop", but we have also seen a rise in Spanish/Latin music and Hip-Hop/Rap in recent years. This is further explored in Analysis 3 below.
- The R^2 = 0.196 and the Adjusted R^2 = 0.039. These values are low, however, since we used a Lasso regression, we know that the model utilizes the optimal predictor variables. To improve these low values, it might make sense to increase the sample size to 100 or more, since 50 is only just enough to create a decent model.

Analysis 2

This regression is almost-identical to the one in Analysis 1, except for it accounts for the genre of a song.

- The R^2 = 0.757 and the Adjusted R^2 = 0.405, but the increase in values is expected since we have many more predictor variables (all the genres). Models with more predictor variables will *always* have a higher R^2, since more variables allows the model to have a more accurate prediction, but the more-variable model becomes too specific to the dataset it is built on.

- Many of the binomial, genre variables have very low p-values, suggesting that they are not significant. This is further backed by the observation that not one single genre has enough data points for it to be considered as a viable predictor variable. You need ideally 30 data points for a categorical variable to be considered. The small sample sizes for each genre means that the model will be heavily skewed since it is based on variables with too few data points (the samples cannot be representative of the population).

Analysis 3

After regrouping the subgenres into their parent genres, we see that pop songs appear the most in the Top 50. This is followed by Spanish/Latin songs, which at first surpised me, since I thought Hip-Hop/Rap songs (the 3rd most prominent category) would be more popular. However, I realized that this is based on my opinion, since I listen to more Hip-Hop/Rap compared to Spanish/Latin music. Lastly, EDM was the least prominent category, but there are many pop songs that are heavily EDM-inspired, so the line is very blurred when it comes to classifying a song's genre.

Out of curiosity, I created a model with the 4 parent genres as binomial predictor variables, and once again they had p-values = ~0, most likely suggesting that the same issue that arised in Analysis 2's regresion is happening here as well.

Analysis 4

Analysis 4 - optimizing my regression model - was my original goal when I discovered this dataset. This is how I did it:

- Using the regression from Analysis 1, I calculated the highest possible predicted song popularity to be 99.3528. To do this, I used the highest values from the dataset for positive-coefficient variables, and the lowest values from the dataset for negative-coefficient variables
- While the above prediction of 99.3528 is the highest possible song popularity, it is quite unlikely that a song would have the properties that created this prediction. For example, a song with 190 beats per minute, the max beats per minute value, is very unlikely (since this is an extremely high value). Therefore, I used the *average* value of each property, which is more representative of the sample size (and by extension the population), and found the predicted song popularity to be ~87.5. This is the same value as the average popularity based on .describe() of the entire dataset (seen in the "Data Cleansing" subsection of "Methodology"), indicating that 1) our model is accurate and 2) genre doesn't really matter when predicting song popularity.

Conclusion

In conclusion, predicting a song's popularity is a difficult task to accomplish when analyzing a dataset with a small sample size. Regardless, after cleaning the data, the best possible multivariate regressino formula, based on the Lasso technique, is:

Predicted Song Popularity = 88.6489 + (0.0114 * *Beats.Per.Minute*) + (0.0271 * *Energy*) + (0.0138 * *Danceability*) + (0.0598 * *Liveness*) + (0.0856 * *Speechiness*) + (-0.0716 * *Valence*) + (-0.0154 * *Length*) + (-0.0079 * *Acousticness*)

It appears that a song's genre is not a fundamental factor when determining its success. Rather, the numerically-measurable properties, such as danceability, energy, and valence are more important. Notably, the R^2 value of this model is 0.196, which suggests we should either refine our model and/or increase the sample size.

In trying to add more variables, two attempts at including genre in the formula yielded different results:

1) In the first try, we attempt to include every genre in the model, but this produces a regression which is highly specific to this dataset, something we want to avoid.

2) In the second try, we attempt to re-group each genre by their "parent genre". In this situation, we see that different genres slighly affect the overall song popularity, but the p-value for each genre is so small Python classifies it at 0, meaning they are not significant in the model.

Since adding genre as a predictor variable did not work, I am almost certain that the best way to improve our model would be to increase our sample size. As mentioned above in *Analysis 1*, perhaps we should analyze the top 100 songs instead of the top 50, since the larger dataset should be more representative of the population.

If we assume that the above model *is* accurate, we can use it to predict the highest possible song popularity. We find this value to be 99.3528, based on using the dataset's max values for positive-coefficient variables, and the dataset's min values for negative-coefficient variables. To remphasize, this is all in theory, but given that this song's predicted popularity is almost 100, we can see that the model is working properly.

For me personally, I now have a much better understanding of the song properties I should consider if I make music in the future. For example, a song which is too happy can actually reduce a song's popularity (evidenced by the negative coefficient for this variable in the model). By constast, a more speech-heavy song tends to become more catchy (evidenced by this variable's coefficient being the large coefficient of all variables), so if I were to make a song I would place special emphasis on the lyrics and how they flow with the rest of the track. Finally, we concluded that there are 4 big genres - Pop, Spanish/Latin, Pop/Rap, and EDM - that people usually like. Fortunate for me, I have already dabelled in all of these genres, so my analysis only reaffirmed that I was on the right path.

Despite the informative analysis one can conduct on a song, music, like any art, is a pure, un-calculable thing. Almost all truly successfuly artists, no matter the discipline, produced their iconic work from a place of emotion rather than numbers. People want to hear the emotions from a song, because this creates a beautiful connection between the audience and the artist, often on a personal level. The emotions that drive the direction and production of a song, and the emotions that very same song evokes in the ears of listeners, is a truly subjective experience. Maybe, despite how sophisticated data analysis and machine learning becomes, we should leave art in the hands of the human, instead of in the hands of the computer.

References

- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html
- https://www.merriam-webster.com/dictionary/subgenre
- https://www.kaggle.com/datasets/leonardopena/top50spotify2019
- https://www.kaggle.com/leonardopena/top-spotify-songs-from-20102019-by-year
- http://organizeyourmusic.playlistmachinery.com/
- https://www.geeksforgeeks.org/python-programming-language/?ref=shm
- https://stackoverflow.com/questions/23330654/update-a-dataframe-in-pandas-while-iterating-row-by-row
- https://stackoverflow.com/questions/19699367/for-line-in-results-in-unicodedecodeerror-utf-8-codec-cant-decode-byte
- Content from QST BA 222, the *Intro to Programming* course in Questrom
- Technical discussions with a peer about optimizing a regression