Python Project #2 -- Data Analysis in Pandas

Ely Hahami
MA564 – Advanced Python
Dr. Laws

Exploring the Metadata

Name of Attribute (Type)	Description
Acousticness (Number)	A confidence measure from 0.0 (low acousticness) to 1.0 (high acousticness).
Danceability (number)	How suitable a track is for dancing (0.0=least danceable, 1.0=most danceable) based on tempo, rhythm stability, and beat strength.
Duration_ms (integer)	The duration of the track in milliseconds.
Energy (number)	Measure from 0.0 to 1.0 that represents a perceptual measure of intensity and activity.
Instrumentalness (number)	Predicts whether a track contains no vocals (ie. 1.0, = no vocal content)
Time_signature (integer)	An estimated time signature. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure). The time signature ranges from 3 to 7 indicating time signatures of "3/4", to "7/4".

Name of Attribute (Type)	Description
Key (integer)	Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C♯/D♭, 2 = D, etc. (no key = -1)
Liveness (number)	Higher liveness values represent an increased probability that the track was performed live.
Loudness (number)	The primary psychological correlate of physical strength of a track in decibels (-60 dB to 0dB).
Mode (integer)	Indicates the modality (major=1, minor=0)
Audio Valence (number)	Indicates how positive/happy/cheerful a song is
Speechiness (number)	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.
Tempo (number)	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.

Column Manipulation: Data Pre-processing and Normalization

After reading the data and importing necessary libraries, we observe that while there exist no null/NaN values, there are **duplicate songs** in the dataset (shown below), so we must clean the data. We do this via slides_df.dropduplicates(inplace = False).

```
slides_df = pd.read_csv('song_data.csv')
slides_df.nlargest(5, 'song_popularity')
```

	song_name	song_popularity	song_duration_ms	acousticness	danceability	energy	instrumentalness	key	liveness	loudness	audio_mode	speechiness
4299	Happier	100	214289	0.191	0.687	0.792	0.0	5	0.167	-2.749	1	0.0452
5593	Happier	100	214289	0.191	0.687	0.792	0.0	5	0.167	-2.749	1	0.0452
7568	Happier	100	214289	0.191	0.687	0.792	0.0	5	0.167	-2.749	1	0.0452
7636	Happier	100	214289	0.191	0.687	0.792	0.0	5	0.167	-2.749	1	0.0452
11665	Happier	100	214289	0.191	0.687	0.792	0.0	5	0.167	-2.749	1	0.0452

The mathematical impact that a larger valued attribute (such as song_duration_ms) has on each node in a neural network is going to be substantially greater than a smaller valued attribute (such as energy). Consequently, besides the columns that are categorical in nature (song_name, key, and time_signature), we iterate over each column and divide each column by its maximum value as a means to **normalize** each attribute so that its values are between 0 and 1. Since loudness values typically range between -60 and 0 db because they are on a logarithmic scale, we divide the loudness column by the absolute value of its max:

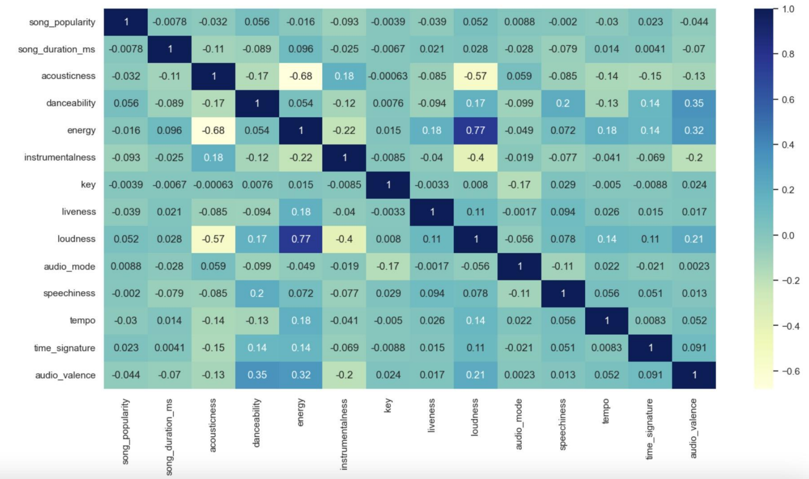
```
for column in slides_df.columns:
    if column != 'song_name' and column != 'key' and column!='time_signature' and column!='loudness':
        pd.to_numeric(column, errors='coerce') ## convert data values from stirng to numerical values (b/c math)
        slides_df[column] = slides_df[column]/(slides_df[column].max())#divide by max,thus normalizing data --> [0,1]
    elif column=='loudness':
        slides_df[column] = slides_df[column]/abs((slides_df[column].max()))#divide by abs value b/c col is log scale
```

Statistical Analysis: Correlation Matrix/Heatmap

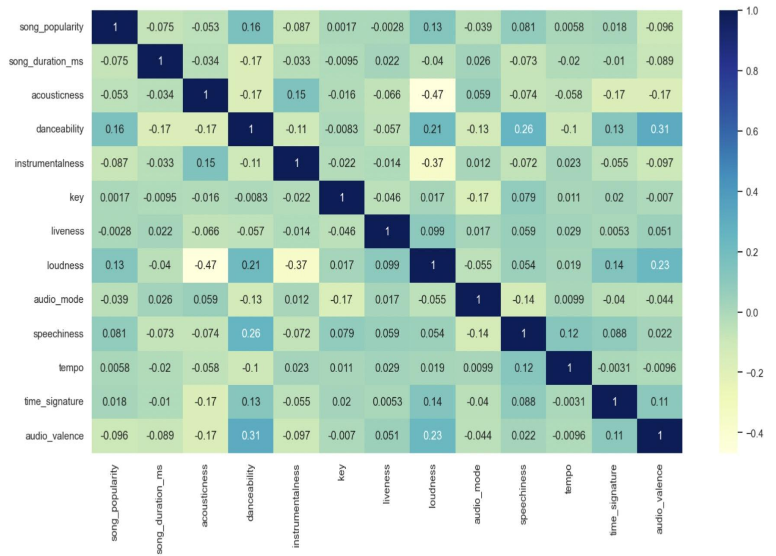
Via the seaborn library, we can simultaneously relay a correlation matrix and a heatmap using sn.heatmap(corr,annot=True, cmap="YlGnBu"), where 'annot = True' writes the data correlation value in each cell. Since the heatmap relays all small correlations, we arbitrarily set a popular song as having a song_popularity rating of 0.7, and create a filter than only includes these highly popular songs. Moreover, since energy and loudness are highly correlated (0.77) and energy and acousticness are highly anticorrelated (-0.68), and energy and audio_valence are relatively highly correlated (0.32), we choose to **drop the energy column** from the dataframe. Observe that for this new heatmap (right), the two highest correlations with song_popularity are **danceability** (0.16) and **loudness** (0.13). The other quantitative, continuous variables — that is, song_duration_ms, acousticness, instrumentalness, liveness, loudness, speechiness, and tempo — seem to comparatively not highly impact the popularity of a song.

```
plt.figure(figsize=(16, 8))
sn.set(style="whitegrid")
corr = slides_df.corr()
sn.heatmap(corr,annot=True, cmap="YlGnBu")
```

```
Out[187]: <AxesSubplot:>
```

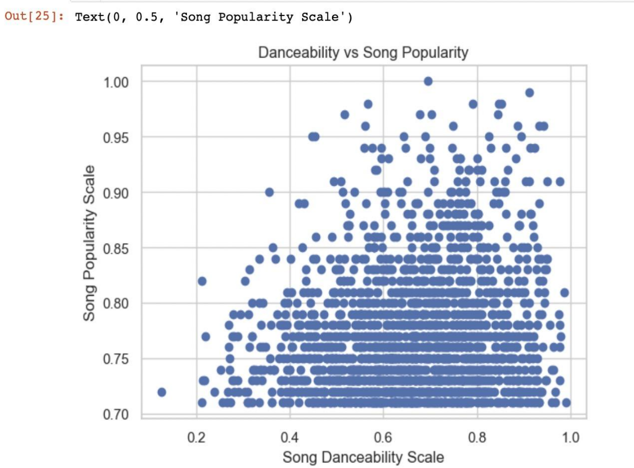


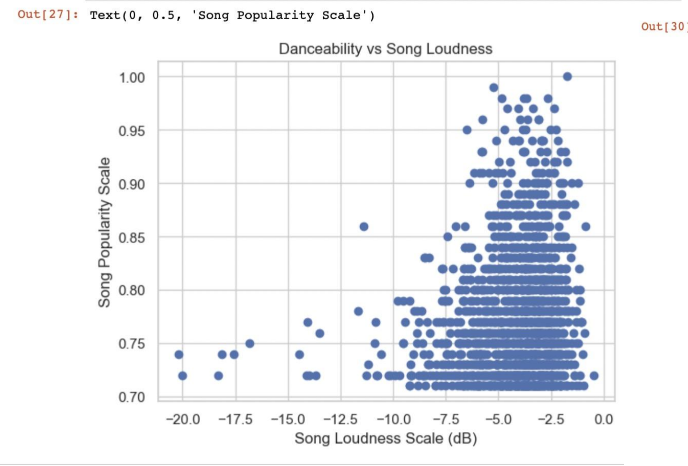

```
Out[267]: <AxesSubplot:>
```

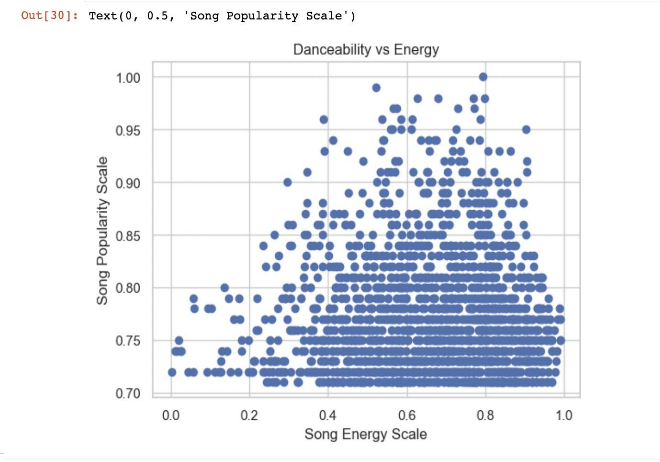


Graphing: Scatterplots for Important Continuous Variables

Via the Matplotlib library, we create a scatterplot for danceability (left) and loudness (middle-left) to further access their respective impacts on song popularity. As shown, songs with a higher danceability have somewhat of a more concentration of popular songs than songs with a lower danceability scale. Similarly, songs with a loudness closer to 0 (which is higher loudness, as loudness is measured on a logarithmic scale) have a higher concentration of popular songs.







Also, given that danceability and loudness are positively correlated with song popularity, and energy is highly correlated with danceability and loudness, it makes sense that, via the scatterplot (middle-right), as energy values increase, there seems to be higher concentrations of popular songs.

Column Manipulation: Binning Data/Pivot Tables

Despite all their merits, our scatterplots and correlation matrices fail to work for columns/attributes that are categorical in nature, such as the **key** of the track. Consequently, without a loss of generality, we **bin song_popularity** into low, mid-low, mid-high, and high categories corresponding to song popularities on the intervals [0,0.25), [0.25, 0.50), [0.50, 0.75), and [0.75, 1.00), respectively. We then create a pivot table to summarize where these categories fall for various keys. Since the dataset has a different number of songs in each key, we create representative **percentages** of low, mid-low, mid-high, and high songs for each key (ie. C, C#/D ♭ ... B ♭ , B).

	song_name			
song_popularity	Low	Mid-Low	Mid-High	High
key				
0	0.143022	0.321223	0.459631	0.062860
1	0.110414	0.327478	0.448557	0.097867
2	0.138670	0.335954	0.458899	0.052180
3	0.154734	0.321016	0.461894	0.050808
4	0.142066	0.336716	0.443727	0.062731
5	0.132060	0.330947	0.459825	0.060461
6	0.114504	0.311069	0.496183	0.067748
7	0.128779	0.339178	0.455865	0.056832
8	0.129895	0.340019	0.453677	0.059217
9	0.138298	0.335461	0.451773	0.055319
10	0.126316	0.317703	0.474641	0.070813
11	0.126945	0.289107	0.479115	0.080262

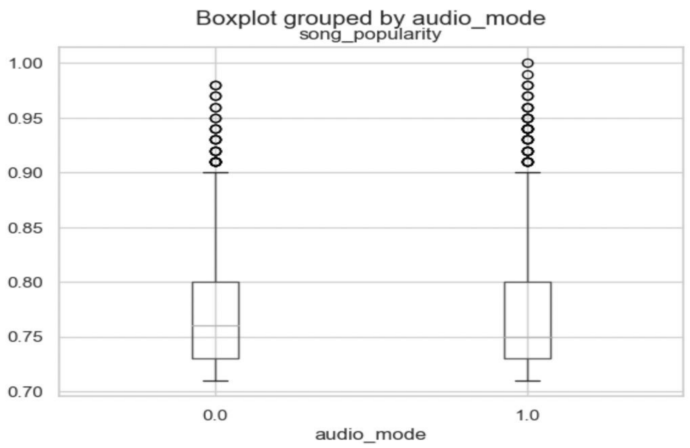
				song_name	High+Mid-High	Low+Mid-Low	Diff
song_popularity	Low	Mid-Low	Mid-High	High			
key							
0	0.143022	0.321223	0.459631	0.062860	0.522491	0.464245	0.058247
1	0.110414	0.327478	0.448557	0.097867	0.546424	0.437892	0.108532
2	0.138670	0.335954	0.458899	0.052180	0.511079	0.474625	0.036455
3	0.154734	0.321016	0.461894	0.050808	0.512702	0.475751	0.036952
4	0.142066	0.336716	0.443727	0.062731	0.506458	0.478782	0.027675
5	0.132060	0.330947	0.459825	0.060461	0.520286	0.463007	0.057279
6	0.114504	0.311069	0.496183	0.067748	0.563931	0.425573	0.138359
7	0.128779	0.339178	0.455865	0.056832	0.512696	0.467956	0.044740
8	0.129895	0.340019	0.453677	0.059217	0.512894	0.469914	0.042980
9	0.138298	0.335461	0.451773	0.055319	0.507092	0.473759	0.033333
10	0.126316	0.317703	0.474641	0.070813	0.545455	0.444019	0.101435
11	0.126945	0.289107	0.479115	0.080262	0.559378	0.416052	0.143325

Since a pivot-table is simply a separate data frame, we add columns 'High+Mid-High, 'Low+Mid-Low' that essentially sum the percentage of songs in that key that have song_populart(ies) of [0.5, 1.0) and [0.0, 0.5], respectively. We then make a column entitled 'Diff' that subtracts the 'Low+Mid-Low' column from the 'High+Mid-High' column. A higher value in the 'Diff' column relays a key that has a higher percentage of well-liked songs. For instance, key 11 (key=B, diff ≈ 0.143) and key 6 (key=F, diff ≈ 0.138) seem to have large proportions of popular songs, while keys such as key 4 (key = E, diff ≈ 0.028) and key 9 (key= G#/A ♭ , diff ≈ 0.033) do not have a large proportion of popular songs. Since there seems to be lots of variability in the 'diff' column among various keys, the pivot table **suggests that the key of the song plays a somewhat relevant role for song_popularity.**

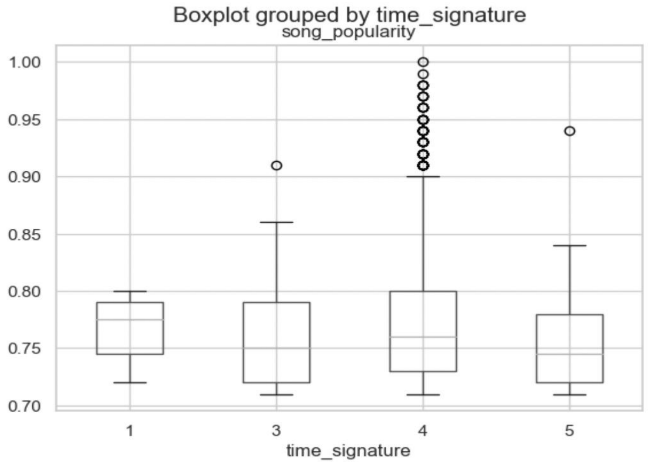
Graphing: Ruling out Categorical Variables Via Boxplots/ANOVA

We can visualize the other categorically-natured columns through boxplots. For audio_mode, there seems to be no impact of a song in major (audio_mode = 1) or minor (audio_mode = 0) on song_popularity, as the comparative boxplot (left) shows quite similar Q1s, medians, Q3s, etc. for songs in major and songs in minor. The conclusion for time_signature is slightly more complicated, as the comparative boxplot (right) is quite deceiving. As shown, there seems to be somewhat important differences in Q1s, medians, Q3s, etc. across various time signatures. However, we run an Analysis of Variance (ANOVA) statistical test to to analyze if there is a true difference between population means for the four different time_signatures. After checking the conditions to run an ANOVA test we import the f_oneway method from the scipy.stats library and perform an ANOVA test with H_0 (null): $\mu_1 = \mu_2 = \mu_3 = \mu4$ (for the four different time_sigs!) and H_1 (alternate): Not all song_popularities means for different time_signatures are equal. As shown below, since our p-value (0.23) is greater than our significance level, $\alpha = 0.05$, we fail to reject the null hypothesis for the ANOVA test. **In total, we rule out audio_mode and time_signature** as highly important attributes that determine a songs popularity.

```
In [226]: print(slides_df.boxplot(column = 'song_popularity', by='audio_mode'))
          AxesSubplot(0.1,0.15;0.8x0.75)
```




```
In [232]: print(slides_df.boxplot(column = 'song_popularity', by='time_signature'))
          AxesSubplot(0.1,0.15;0.8x0.75)
```




```
Out[71]: F_onewayResult(statistic=1.4290461889115313, pvalue=0.2331205770724331 7)
```

Conclusions – Ultimately, What Determines Song Popularity?

After exploring the metadata, pre-processing the data, and analyzing/visualizing correlation heatmaps, scatterplots, pivot tables, and boxplots, we ultimately posit that **danceability, loudness, and key** are the **three main factors** that determine a songs popularity. Via the data, popular songs tend to have high danceability ratings, tend to be loud, and songs in certain keys have a higher proportion/concentration of very popular songs. These results, to some extent, make intuitive sense.

Danceability refers to the ease/extent to which people can dance or physically move to a song, which is largely influenced by the song's rhythm and tempo. A song with a ‘catchy’ beat and rhythm that makes people want to move their bodies is more likely to become popular, as it can create a sense of **fun** and **pure enjoyment**. Furthermore, danceability is often associated with genres of music that are popular in **clubs** and **parties**, such as hip-hop, pop, and electronic dance music (EDM), which tend to have a **broad appeal** and can attract a large audience.

Additionally, songs that are louder can feel more **exciting** and **engaging**, which can create a sense of **intensity** and **emotion** in listeners. This can be effective in genres of music that are designed to be played in large arenas or stadiums, such as rock or metal, where the power and intensity of the music can enhance the overall experience for the audience.

Lastly, the key of a song can also play a significant role in its **emotional resonance,** as different keys can convey different **moods** and **feelings**.

Sources (APA)

Song popularity dataset. (n.d.). Retrieved May 9, 2023, from https://www.kaggle.com/datasets/yasserh/song-popularity-dataset2

De-Yu, C. (2023, January 5). Anova test, with python. Medium. https://towardsdatascience.com/anova-test-with-python-cfbf4013328b

Pandas documentation—Pandas 2. 0. 1 documentation. (n.d.). Retrieved May 9, 2023, from https://pandas.pydata.org/docs/

Action, C. (2019, May 1). Pandas: To_numeric for multiple columns [Forum post]. Stack Overflow. https://stackoverflow.com/q/36814100

Waskom, M. (2021). Seaborn: Statistical data visualization. Journal of Open Source Software, 6(60), 3021. https://doi.org/10.21105/joss.03021

https://www.kaggle.com/datasets/yasserh/so
ng-popularity-dataset2