LKH'S DATASET PROPOSAL

https://www.kaggle.com/ektanegi/spotifydata-19212020

Dataset Size: 169909 rows & 19 columns

IMPORTANT FEATURES:

1. Acousticness (0-1 float, not having electrical amplification eg. guitar over electric guitar, piano over keyboard)
2. danceability (0-1 float, How likely one can dance to the song)
3. energy (0-1 float, whatever keeps the listener engaged and listening -> abit too subjective, might drop)
4. explicit (0 or 1 int, contains explicit language)
5. instrumentalness (0-1 float, Predicts whether a track contains no vocals 0 = vocals, 1 = no vocals)
6. key (0-11 int, All keys on octave encoded as values ranging from 0 to 11, starting on C as 0, C# as 1 and so on -> needs musical background to determine what it really means)
7. liveness (0-1 float, Detects the presence of an audience in the recording, 0= not live , 1= live)
8. loudness (-60 to 3 float, -60 = soft, 3 = loud)
9. mode (0-1 float, 0= minor (sad), 1=major(happy))
10. release_date (datetime)
11. speechiness (0-1 float, >0.66 = made of spoken words (audio book), 0.33-0.66 = mix of music and speech, <0.33 = no speech)
12. tempo (0-244 float, speed of music, higher = faster)
13. valence (0-1 float, 0= sad/negative, 1=happy/positive)

IMPORTANT FEATURES:

14. Artist (str, name of artist -> able to feature engineer into length of name)
15. duration_ms (float, Duration in miliseconds)
16. id (dropping this as it is useless)
17. name (str, of song)
18. popularity (0-200 int, 0 = no ranking)
19. release_date (datetime)
20. year (1921-2020)

SMALL SAMPLE (TAIL 5)

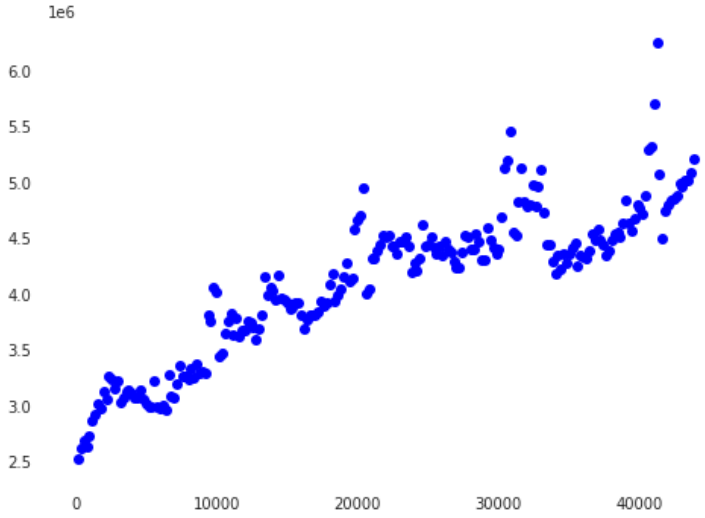
```
spotify_top_songs.tail() #small sample of data
```

artists	danceability	duration_ms	energy	explicit	id	instrumentalness	key	liveness	loudness	mode	name	popularity	release_date	speechiness	tempo	valence	year
['DripReport', 'Tyga']	0.875	163800	0.443	1	4KppkfIX7I3vJQk7urOJaS	0.000032	1	0.0891	-7.461	1	Skechers (feat. Tyga) - Remix	75	2020-05-15	0.1430	100.012	0.306	2020
['Leon Bridges', 'Terrace Martin']	0.719	167468	0.385	0	1ehhGITvjtHo2e4xJFB0SZ	0.031300	8	0.1110	-10.907	1	Sweeter (feat. Terrace Martin)	64	2020-06-08	0.0403	128.000	0.270	2020
['Kygo', 'Oh Wonder']	0.514	180700	0.539	0	52eycxprLhK3lPcRLbQiVk	0.002330	7	0.1080	-9.332	1	How Would I Know	70	2020-05-29	0.1050	123.700	0.153	2020
['Cash Cash', 'Andy Grammer']	0.646	167308	0.761	0	3wYOGJYD31sLRmBgCvWxa4	0.000000	1	0.2220	-2.557	1	I Found You	70	2020-02-28	0.0385	129.916	0.472	2020
['Ingrid Andress']	0.512	214787	0.428	0	60RFlt48hm0l4Fu0JoccOl	0.000000	0	0.1050	-7.387	1	More Hearts Than Mine	65	2020-03-27	0.0271	80.588	0.366	2020

SUPERVISED LEARNING

- First suggestion: Regression with simple linear regression model.
- Problem statement: What is the minimum no. Of streams to hit top 200s
- Goal: whether a song can hit top 200
- Label: Streams, 2 features: streams and rank .. the rest im not very sure if can add, see how

```
plt.plot(spotify_top200[spotify_top200['Rank']==200]["Streams"], linestyle='none', marker='o', color='b')
plt.show()
```



SUPERVISED LEARNING

- Second suggestion: classification
- Problem statement: Whether a song is popular or not, based off rank and streams.
- Goal: Which features are important for popularity and if a song will be popular enough to hit the top 200 given x no. Of features.

- Maybe im able to use my first suggestion to get the popularity range first and use that as my label. Otherwise.... use minimum of top 200 (lazy method)?
- Label: > predicted popularity = popular, < pp = not popular.
- Features: Everything else

CHALLENGES

- * probably need to encode certain features further to make the model simpler eg. Speechiness from 0-0.33, 0.33-0.66 and 0.66 - 1 to 0, 1 & 2.
- Feature selection and engineering:
- * some of the features might not be too relevant (eg. ID of track)
- * Need to engineer some features like artists into no. of artist per track and maybe length of artist name?
- * feature aggregation might be useful to calculate overall performance of song but will not be in line with label of top 200 song per year. maybe top 200 songs overall? Eg. Song + stream -> Number of times song is played.
- * need to reorder my ranking data from 1-200-0 where 0 is last.
- Test out lazy classifier for ensembles

HOW

- Using Linear regression, Random forest and ensemble methods like bootstrapping (variance) & XGBoost + **hyperopt** (bias) to improve scores.

Measure with Confusion matrix, AUC and k fold validation (prevent overfitting of random forest although it is resistant). Able to use Neural networks, KNN & TPOT (autoML) to see which give better results and perhaps give more options to client/ self.

Eg. If NN gives 20% more accuracy, perhaps wewould want to consider following the machine regardless of whether we understand the reasonings why the algorithmn works.
Evaluate models with the test (& validation) set and finally select the best model.

Get the best tuning parameters with Grid search and retrain all models.

Sub-goals:

- is there a seasonal trend? Eg. Some years Rap / Live/ Upbeat / depressing / fast music are more popular.

Are they perhaps linked to any special event eg. Valentines/Christmas etc.

- is there a formula to hit the top 100 songs? Eg. Short duration, many artists, album has multiple songs.

- Do old songs come back year after year? All I want for christmas is you – Mariah Carey?

- if so, do we really need to continuously make new songs? Re-hashing a song? Re-mixes/Covers/live versions etc.

- maybe I will give my second suggestion a try? Other wise I can try to predict top 200 with just the min. Streams in the entire of 1921-2020 dataset and use that as my label.