Judah Tanninen

https://www.kaggle.com/datasets/nelgiriyewithana/top-spotify-songs-2023/data

---

In [25]:
```
# shbang
# Judah Tanninen
# Description: Take home test, using spotify most streamed data.

# Imports
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split # Does the splitting for me
from sklearn.metrics import r2_score
```

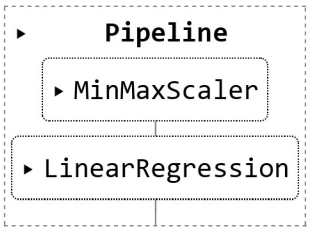
In [26]:
```
# Open the csv into a dataframe
df = pd.read_csv('spotify-2023.csv', encoding='ISO-8859-1') # Goofy encoding, looke
# Determine what columns we are gonna use (determined through heat map and personal
columns=['in_spotify_playlists', 'in_spotify_charts', 'in_apple_charts', 'in_deezer

target_feature='streams' # Total streams (ever)
# Above feature being total streams ever makes the score a lot worse. We could do a

# Do some initial data cleaning, before we can use the df
# Remove commas from any number fields
df.replace(',','', regex=True, inplace=True) # Remove commas, they appear in some c

df.fillna(value=0, inplace=True) # Replace na and nan values with 0s, may not be ne

# One of the streams is a goofy long string, need to remove that row
df = df[pd.to_numeric(df[target_feature], errors='coerce').notnull()]
# Finished data cleaning

xs = df[columns] # Get the columns we want
ys = df[target_feature] # Get all the targets
# Split them up nicely using the train_test_split
x_train, x_test, y_train, y_test = train_test_split(xs, ys, test_size=0.3)
# Printing all the splits
print(x_train)
print(x_test)
print(y_train)
print(y_test)
```

```
     in_spotify_playlists  in_spotify_charts  in_apple_charts  \
257                   244                 12               84
412                  3045                  6               53
716                  1188                  0                1
746                   686                  2                6
951                  1320                  0               26
..                    ...                ...              ...
472                   892                  0               17
509                  2297                  0               29
114                 41751                 25               32
447                  8879                  0              107
208                   332                  5               41

     in_deezer_charts in_shazam_charts
257                 2                9
412                 1               32
716                 0                1
746                 0               15
951                 0                0
..                ...              ...
472                 0                0
509                 0                0
114                 0              666
447                 0                1
208                 0               19

[666 rows x 5 columns]
     in_spotify_playlists  in_spotify_charts  in_apple_charts  \
682                   601                  0               73
617                  5481                  0               30
497                   680                  0               15
278                  7613                 33               90
451                  3788                  0                3
..                    ...                ...              ...
193                    86                 11               33
880                   685                  2                0
124                   457                 24              116
441                 25653                  0              132
110                 26792                 32              113

     in_deezer_charts in_shazam_charts
682                 0                0
617                 0                0
497                 0                0
278                15               55
451                 0                0
..                ...              ...
193                 0                1
880                 0               30
124                 3                2
441                 0                0
110                 0              458

[286 rows x 5 columns]
257    118810253
412    571386359
```

```
716     312622938
746     146363130
951     133895612
          ...
472      65362788
509     116903579
114    1205951614
447     663832097
208      70106975
Name: streams, Length: 666, dtype: object
682     154119539
617     489945871
497      51641685
278     782369383
451     520034544
          ...
193      30343206
880     129314708
124     330346424
441    1449779435
110    1093605526
Name: streams, Length: 286, dtype: object
```

In [27]:
```
# Making a pipeline
pipe = Pipeline([
    ('scale', MinMaxScaler()),
    ('predict', LinearRegression(n_jobs = -1)) # Basic linear regression
])

# Now, fit the ol training data
pipe.fit(x_train, y_train)
```

Out[27]:



In [28]:
```
# Alright, now, we have fit the data, and should be good to predict values
y_pred = pipe.predict(x_test) # Returns its predictions for y values
# Calculate the r squared score (averages between 0.6 and 0.7)
r2 = r2_score(y_test, y_pred)
print(r2)
```

```
0.6701846046919
```

Questions

Why subset?

At first, I guessed kind of randomly, assuming songs in playlists or in charts were more likely to have more streams. This was partially correct, after running some hotmaps, i found the

following:

- Spotify playlists is by far the best parameter, removing it drops the average from an .65 to around the .2 area.
- None of the other programs (apple music, shazam, etc) had any relation with playlist - streams, so those got wacked
- Being in the charts is in general a good indicator of more streams So, my subset is made purely of how many charts and spotify playlists the song is in

---

Did it perform well?

Well, kind of, averaging around a 0.6 - 0.7 is better than half (I think), so its not too bad. I thought I would be able to get something better, but after looking at the data, it makes sense why I couldn't.

All the features related to the type of music (bpm, valence, energy, etc) had almost zero correlation with the streams, so that removed almost half of the columns. Another big issue is that the streams are total for all time, not for this year, which means that the year column is mostly useless, and all the charts/playlists are docked a decent amount, because people play a lot of old music.

Why R-Squared?

I'm lazy.

No, but really, I've already used r squared, and have a better feel for what is good vs what is bad. Also, r^2 can show negative correlations to any negative number, so when I was trying out artists as a numeric column, it had a crazy low r^2 score (like -3.2) which means it was just an awful feature to use.

Finally, this is a regression task, and r squared is for regression, so it's a good metric to choose. (I tried out a mean error squared as well, but I didn't understand it's output that much)

In [ ]:

In [ ]: