CAPSTONE PROJECT 04

Is this Song the Next Big Hit?

PREDICTION OF SONG POPULARITY

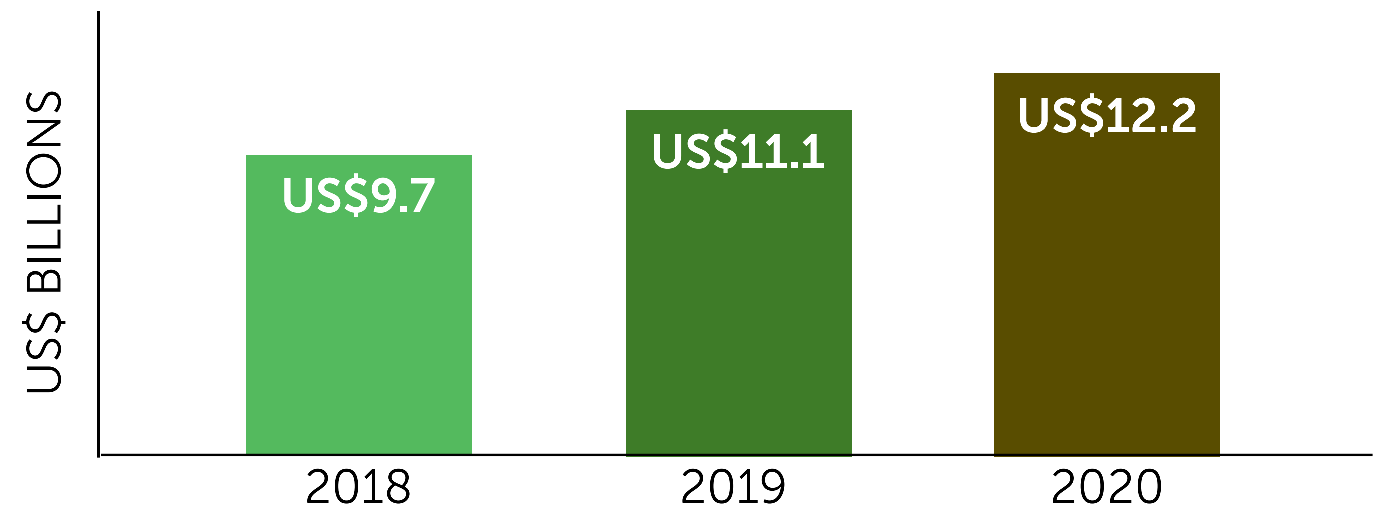
BY NOEL RODRIGUES

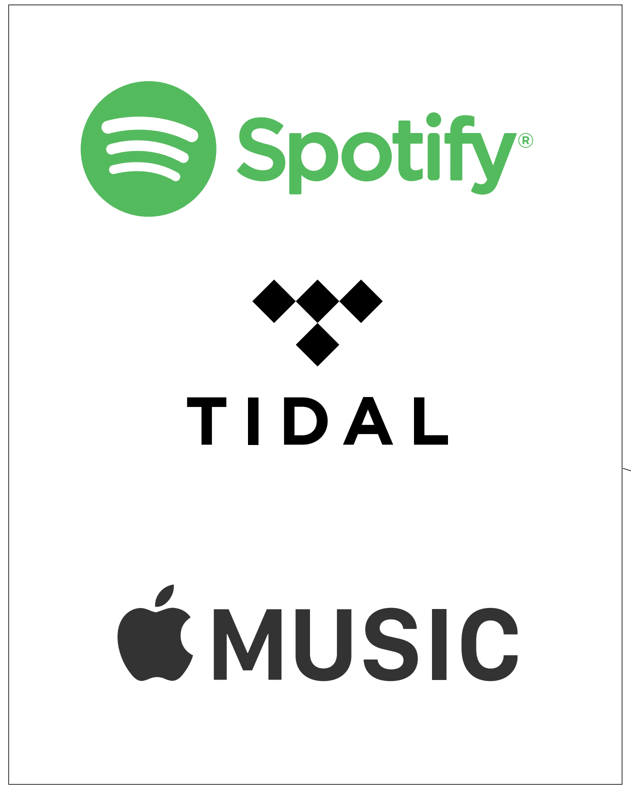
CONTENTS

1. INTRODUCTION

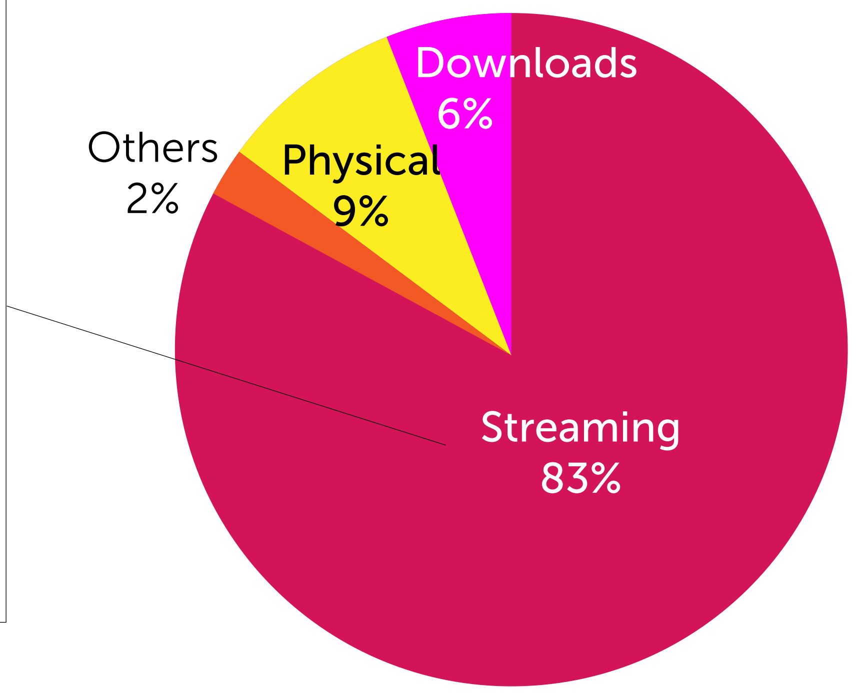
- The U.S. music industry was worth US$12.15 billion in 2020.[1]
- Streaming services accounted for 83% of this market, generating US$$10.07 billion.[2]
- Every music company and artist are trying to grab a slice of the pie, making an average annual salary of US$36,000 to millions for big-named artists such as Beyoncé.[3]
- The rise of "music analytics" has seen music companies embracing data science to analyse trends and predict what the next big hit might be.[4]
- No longer about raw talent, but about employing big data in choosing a song whose genre and lyrics have relevance.

U.S. MUSIC INDUSTRY REVENUES





U.S. MUSIC INDUSTRY REVENUES 2020



1. https://www.musicbusinessworldwide.com/the-us-recorded-music-industry-grew-by-over-1bn-in-2020-but-faces-big-challenges-over-streamings-pricing-and-its-growth/
2. https://www.riaa.com/wp-content/uploads/2021/02/2020-Year-End-Music-Industry-Revenue-Report.pdf
3. https://www.ziprecruiter.com/Salaries/Recording-Artist-Salary
4. https://www.opentracker.net/article/data-science-music

2. OBJECTIVES

- **MY ROLE:**
  Data analyst in Arpeggios Analytics

- **TARGET AUDIENCE:**
  Independent labels and record companies

- **CLIENT'S PROBLEM:**
  How to know what the next hit song will be?

- **OBJECTIVE:**
  To predict the popularity of a song

arpaggio ANALYTIC

ROC-A-FELLA RECORDS

GRAVE9 RECORDS

drkrec

MASTER RECORDS

ATLANTIC

~~SONY MUSIC~~ ~~EMI~~ ~~UNIVERSAL~~ ~~UNIVERSAL MUSIC~~

3. METHODOLOGY

- Data Source: Kaggle — over 174,000 chart-hitting songs which were collected from Spotify Web API[1]
- Time Period: Songs from 1921 to 2020 (100 years)[2]
- Prediction Metrics: Regression Model[3]
- ML Models: Linear Regression and K-Nearest Neighbor (KNN)
- Tools: Pandas, Numpy, Matplotlib, Seaborn and Scikit Learn via Jupyter Notebook IDE

4. PROCESS FLOW

- 4.1 Imports and Data Cleaning
- 4.2 EDA
- 4.3 Data Preparation
- 4.4 Training the Model
- 4.5 Testing the Model

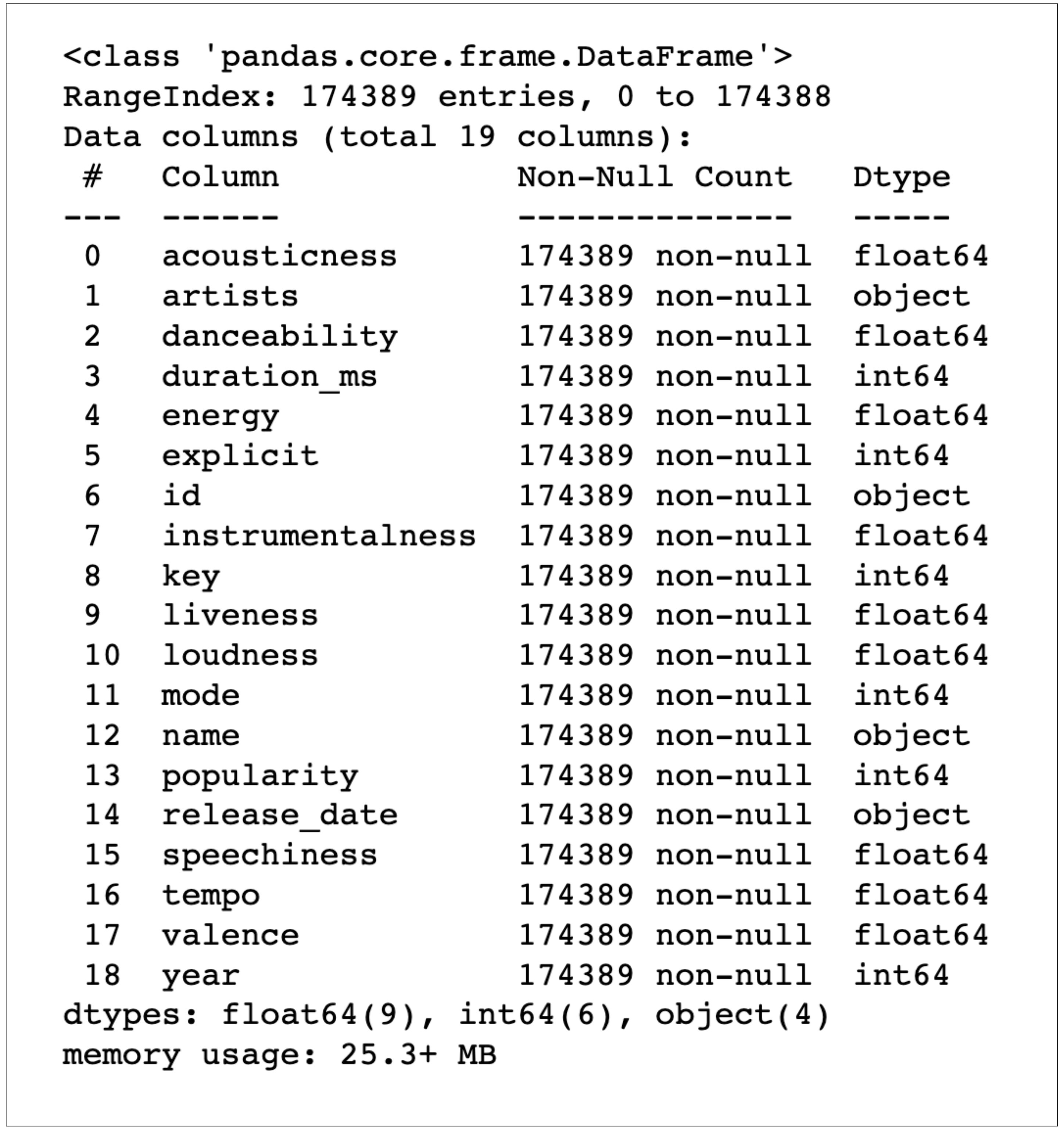
1. https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks
2. https://developer.spotify.com/documentation/web-api/reference/#endpoint-get-track
3. https://geekflare.com/choosing-ml-algorithms/

4.1 Imports and Data Cleaning

ORIGINAL DATASET

- 174,389 row of records
- 19 columns of attributes
- No null values
- Some of the key attributes are:

  - Acousticnesss
  - Energy
  - Instrumentalness
  - Liveness
  - Mode
  - Speechiness
  - Tempo



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 174389 entries, 0 to 174388
Data columns (total 19 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   acousticness      174389 non-null  float64
 1   artists           174389 non-null  object
 2   danceability      174389 non-null  float64
 3   duration_ms       174389 non-null  int64
 4   energy            174389 non-null  float64
 5   explicit          174389 non-null  int64
 6   id                174389 non-null  object
 7   instrumentalness  174389 non-null  float64
 8   key               174389 non-null  int64
 9   liveness          174389 non-null  float64
 10  loudness          174389 non-null  float64
 11  mode              174389 non-null  int64
 12  name              174389 non-null  object
 13  popularity        174389 non-null  int64
 14  release_date      174389 non-null  object
 15  speechiness       174389 non-null  float64
 16  tempo             174389 non-null  float64
 17  valence           174389 non-null  float64
 18  year              174389 non-null  int64
dtypes: float64(9), int64(6), object(4)
memory usage: 25.3+ MB
```

REMOVE 'RELEASE_DATE'

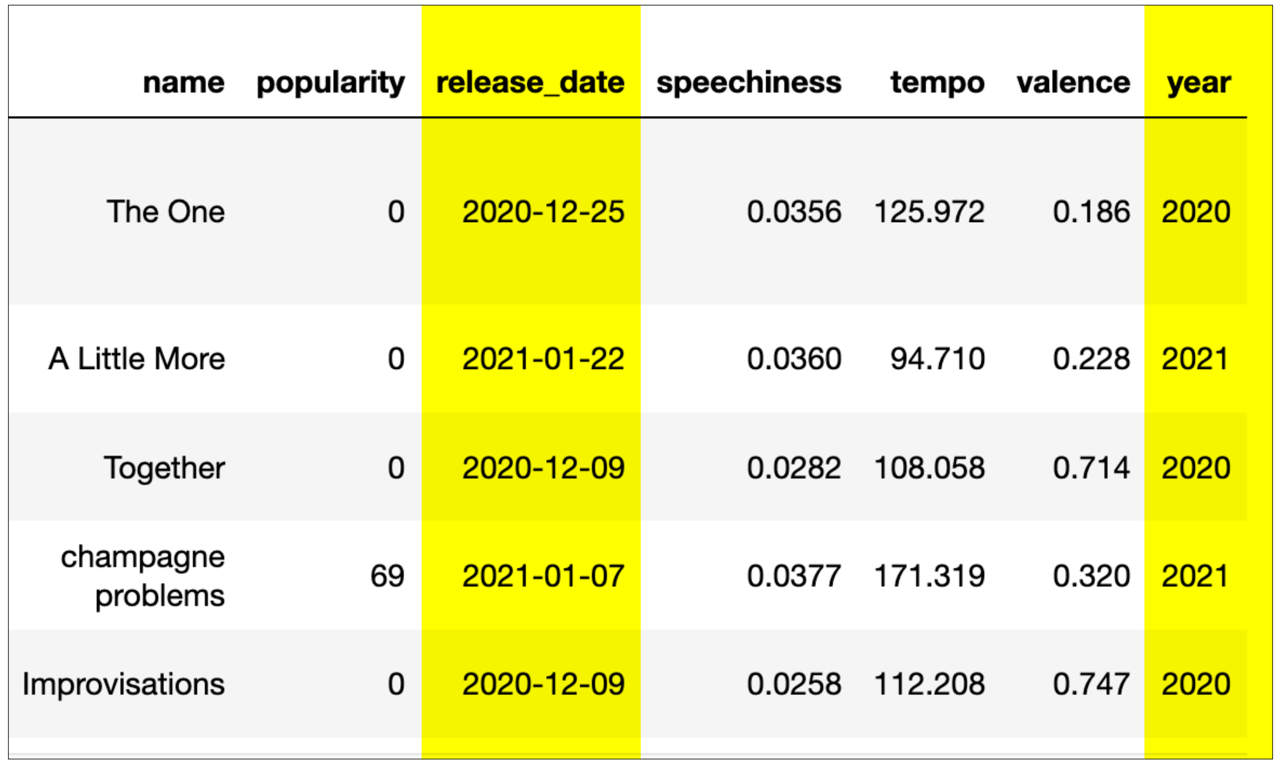
- No difference between 'release_date' and 'year'

REMOVE DUPLICATES

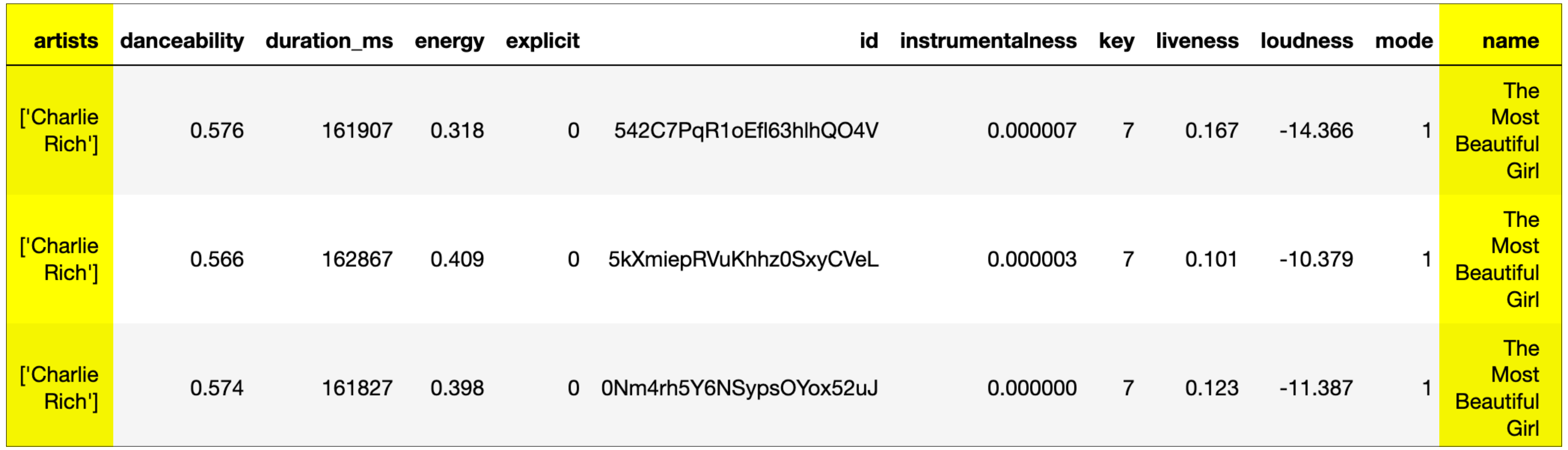
- Find songs with same 'artists' and (song) 'name'
- There were 14,948 duplicates
- Kept the song with the highest 'popularity' and remove its duplicates

REMOVE ZERO VALUES

- Remove all rows with zero values from columns 'tempo' and 'popularity'
- Reason being that 'tempo' with 0 value means the song is 'dead'
- A 'popularity' value of 0 means that the song shouldn't even be on this list



name	popularity	release_date	speechiness	tempo	valence	year
The One	0	2020-12-25	0.0356	125.972	0.186	2020
A Little More	0	2021-01-22	0.0360	94.710	0.228	2021
Together	0	2020-12-09	0.0282	108.058	0.714	2020
champagne problems	69	2021-01-07	0.0377	171.319	0.320	2021
Improvisations	0	2020-12-09	0.0258	112.208	0.747	2020



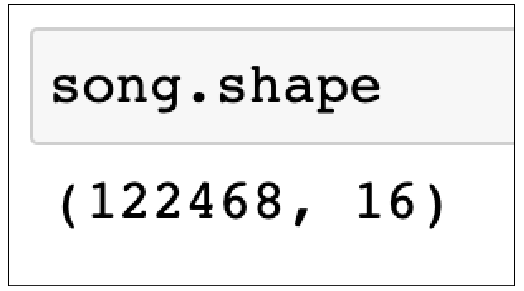
artists	danceability	duration_ms	energy	explicit	id	instrumentalness	key	liveness	loudness	mode	name
['Charlie Rich']	0.576	161907	0.318	0	542C7PqR1oEfl63hlhQO4V	0.000007	7	0.167	-14.366	1	The Most Beautiful Girl
['Charlie Rich']	0.566	162867	0.409	0	5kXmiepRVuKhhz0SxyCVeL	0.000003	7	0.101	-10.379	1	The Most Beautiful Girl
['Charlie Rich']	0.574	161827	0.398	0	0Nm4rh5Y6NSypsOYox52uJ	0.000000	7	0.123	-11.387	1	The Most Beautiful Girl

'ENERGY' VS 'LOUDNESS'

- Both are highly correlated and have similar attributes
- This heatmap shows a positive correlation of 0.78
- To avoid multicollinearity, 'loudness was removed and 'energy' was kept'

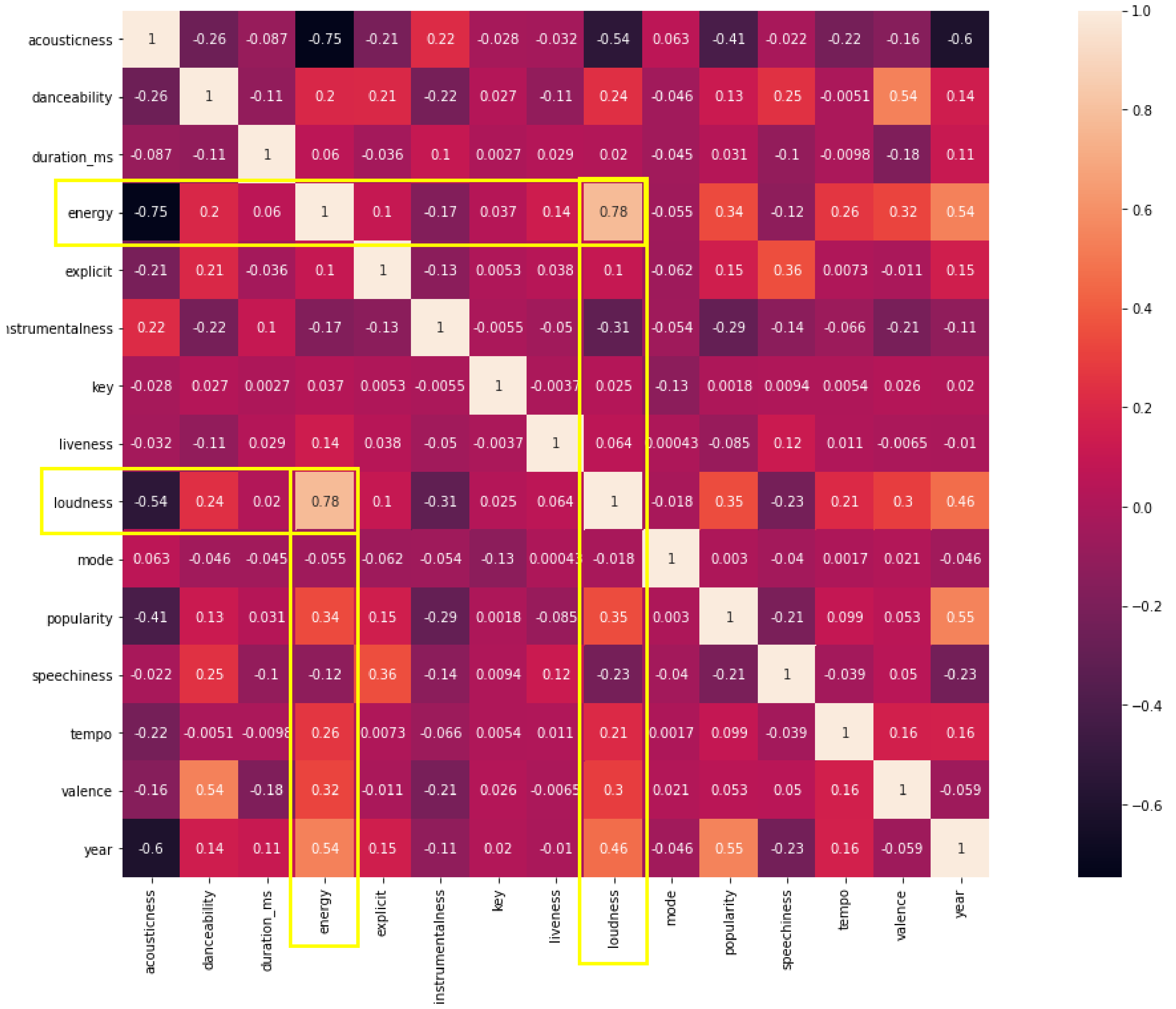
DATASET AFTER CLEANED

- 122,468 row of records (previously 174,389 rows)
- 16 columns of attributes (previously 19 columns)



```
song.shape
```

```
(122468, 16)
```

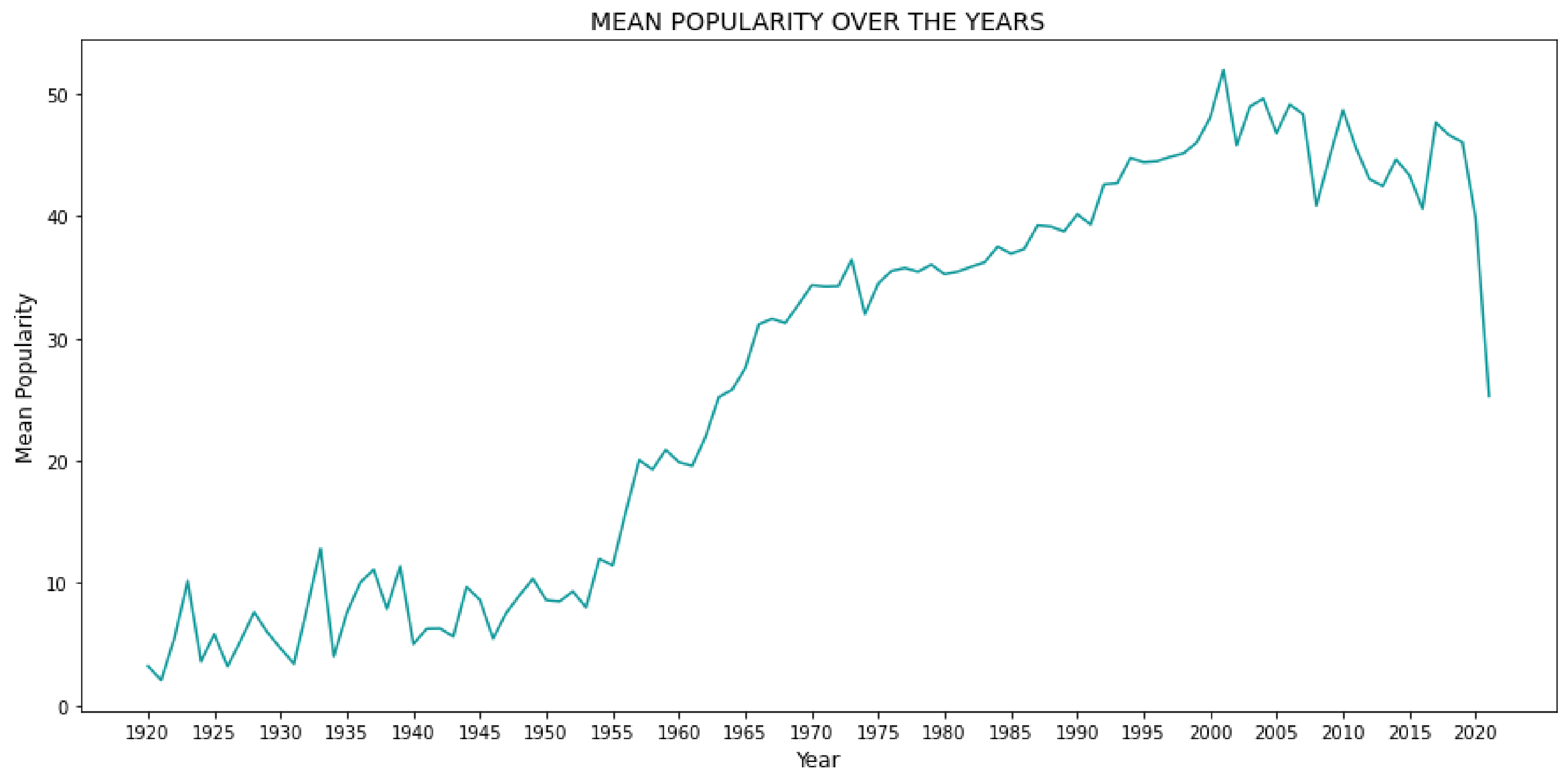


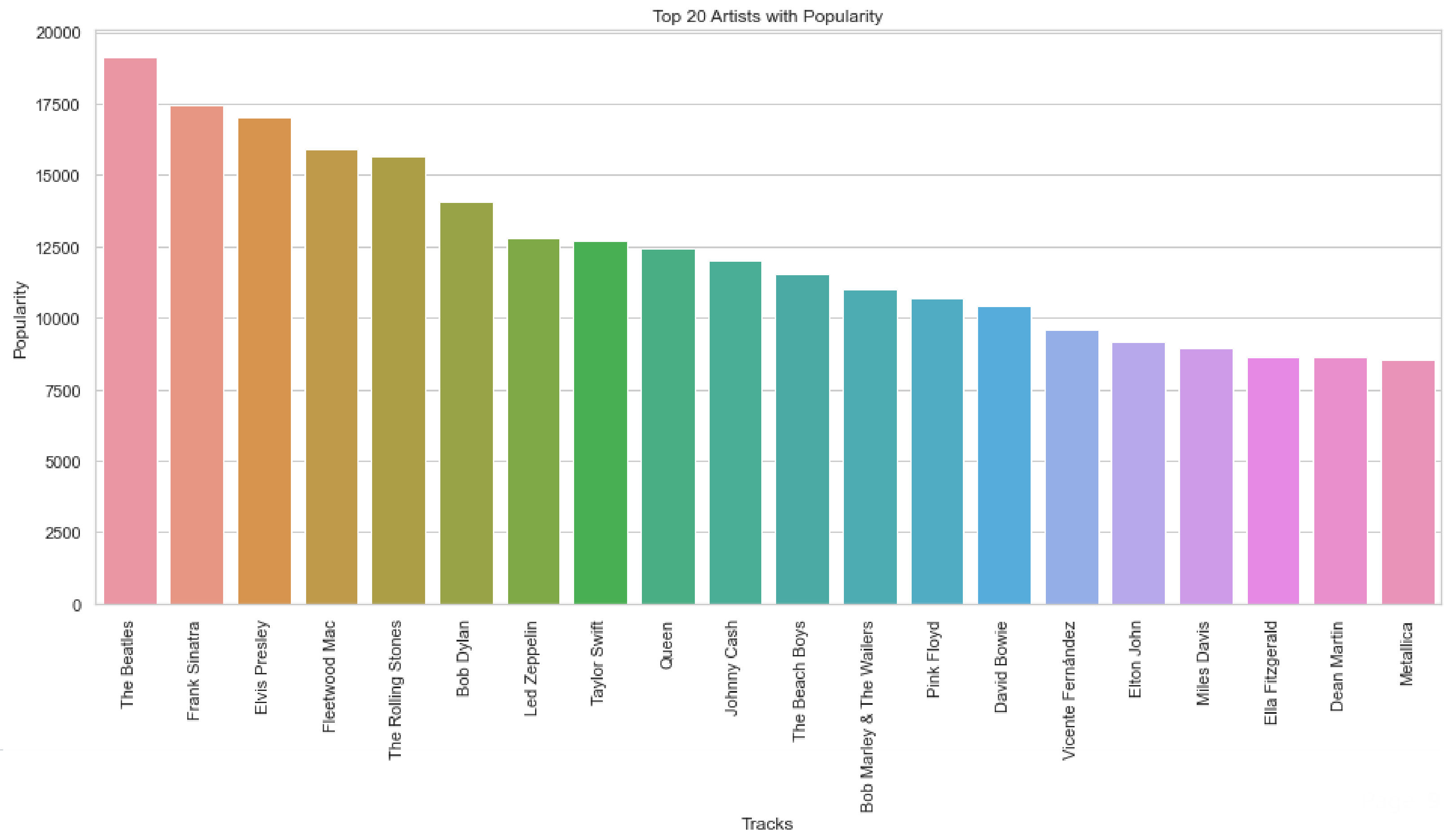
	acousticness	danceability	duration_ms	energy	explicit	instrumentalness	key	liveness	loudness	mode	popularity	speechiness	tempo	valence	year
acousticness	1	-0.26	-0.087	-0.75	-0.21	0.22	-0.028	-0.032	-0.54	0.063	-0.41	-0.022	-0.22	-0.16	-0.6
danceability	-0.26	1	-0.11	0.2	0.21	-0.22	0.027	-0.11	0.24	-0.046	0.13	0.25	-0.0051	0.54	0.14
duration_ms	-0.087	-0.11	1	0.06	-0.036	0.1	0.0027	0.029	0.02	-0.045	0.031	-0.1	-0.0098	-0.18	0.11
energy	-0.75	0.2	0.06	1	0.1	-0.17	0.037	0.14	0.78	-0.055	0.34	-0.12	0.26	0.32	0.54
explicit	-0.21	0.21	-0.036	0.1	1	-0.13	0.0053	0.038	0.1	-0.062	0.15	0.36	0.0073	-0.011	0.15
ıstrumentalness	0.22	-0.22	0.1	-0.17	-0.13	1	-0.0055	-0.05	-0.31	-0.054	-0.29	-0.14	-0.066	-0.21	-0.11
key	-0.028	0.027	0.0027	0.037	0.0053	-0.0055	1	-0.0037	0.025	-0.13	0.0018	0.0094	0.0054	0.026	0.02
liveness	-0.032	-0.11	0.029	0.14	0.038	-0.05	-0.0037	1	0.064	.00043	-0.085	0.12	0.011	-0.0065	-0.01
loudness	-0.54	0.24	0.02	0.78	0.1	-0.31	0.025	0.064	1	-0.018	0.35	-0.23	0.21	0.3	0.46
mode	0.063	-0.046	-0.045	-0.055	-0.062	-0.054	-0.13	0.0004	-0.018	1	0.003	-0.04	0.0017	0.021	-0.046
popularity	-0.41	0.13	0.031	0.34	0.15	-0.29	0.0018	-0.085	0.35	0.003	1	-0.21	0.099	0.053	0.55
speechiness	-0.022	0.25	-0.1	-0.12	0.36	-0.14	0.0094	0.12	-0.23	-0.04	-0.21	1	-0.039	0.05	-0.23
tempo	-0.22	-0.0051	-0.0098	0.26	0.0073	-0.066	0.0054	0.011	0.21	0.0017	0.099	-0.039	1	0.16	0.16
valence	-0.16	0.54	-0.18	0.32	-0.011	-0.21	0.026	-0.0065	0.3	0.021	0.053	0.05	0.16	1	-0.059
year	-0.6	0.14	0.11	0.54	0.15	-0.11	0.02	-0.01	0.46	-0.046	0.55	-0.23	0.16	-0.059	1

4.2 EDA

POPULARITY

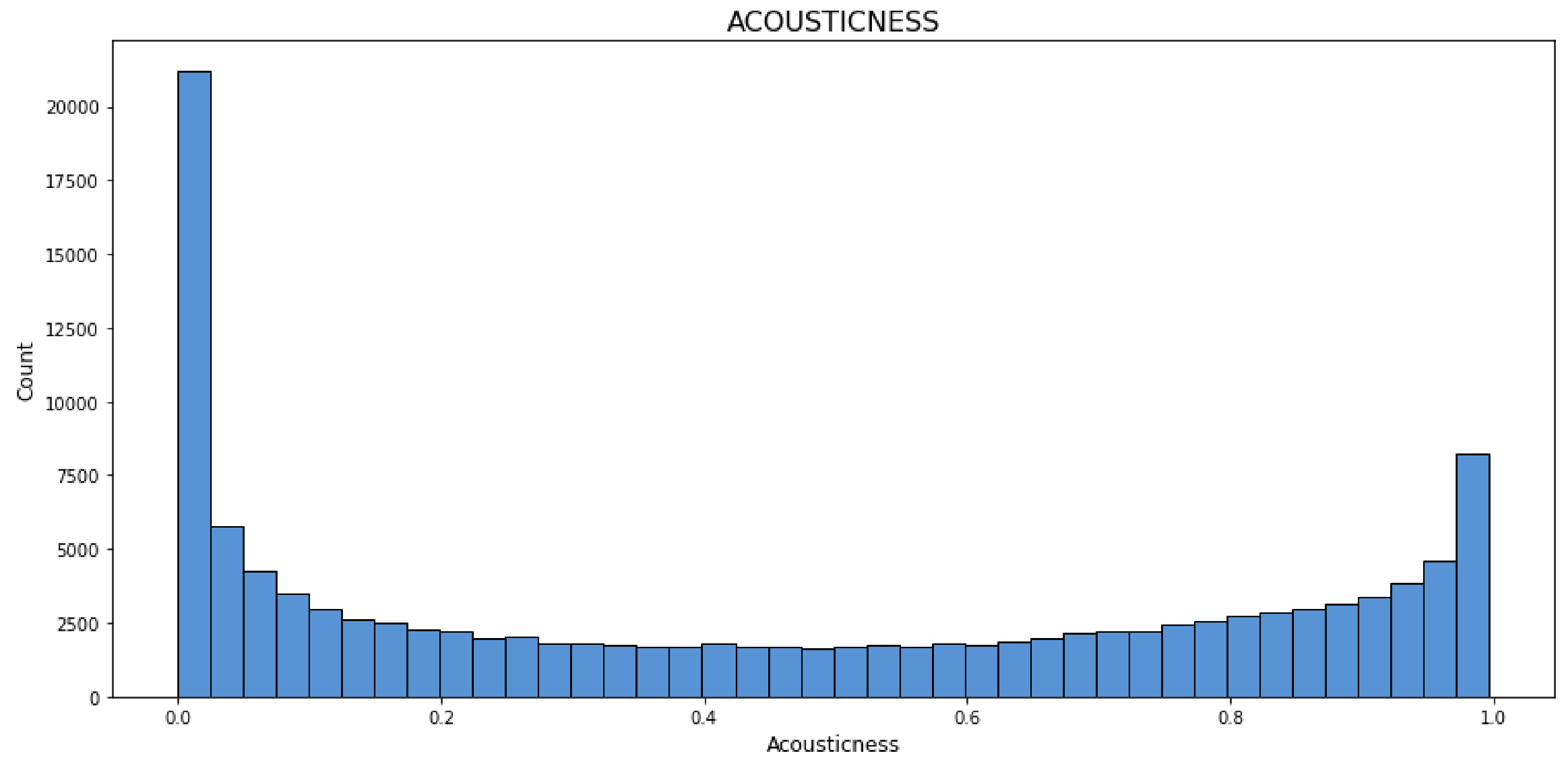
- Popularity of songs has increased since 1920.
- After World War II, music became increasingly more popular as the economy grew.
- Reached a peak at 2000 and started dipping. One plausible reason could be the advent of digital music, as people stop buying CDs and started downloading music illegally *(yay, Napster!)*.
- The most popular artist over a century is The Beatles (from 1921 - 2020)

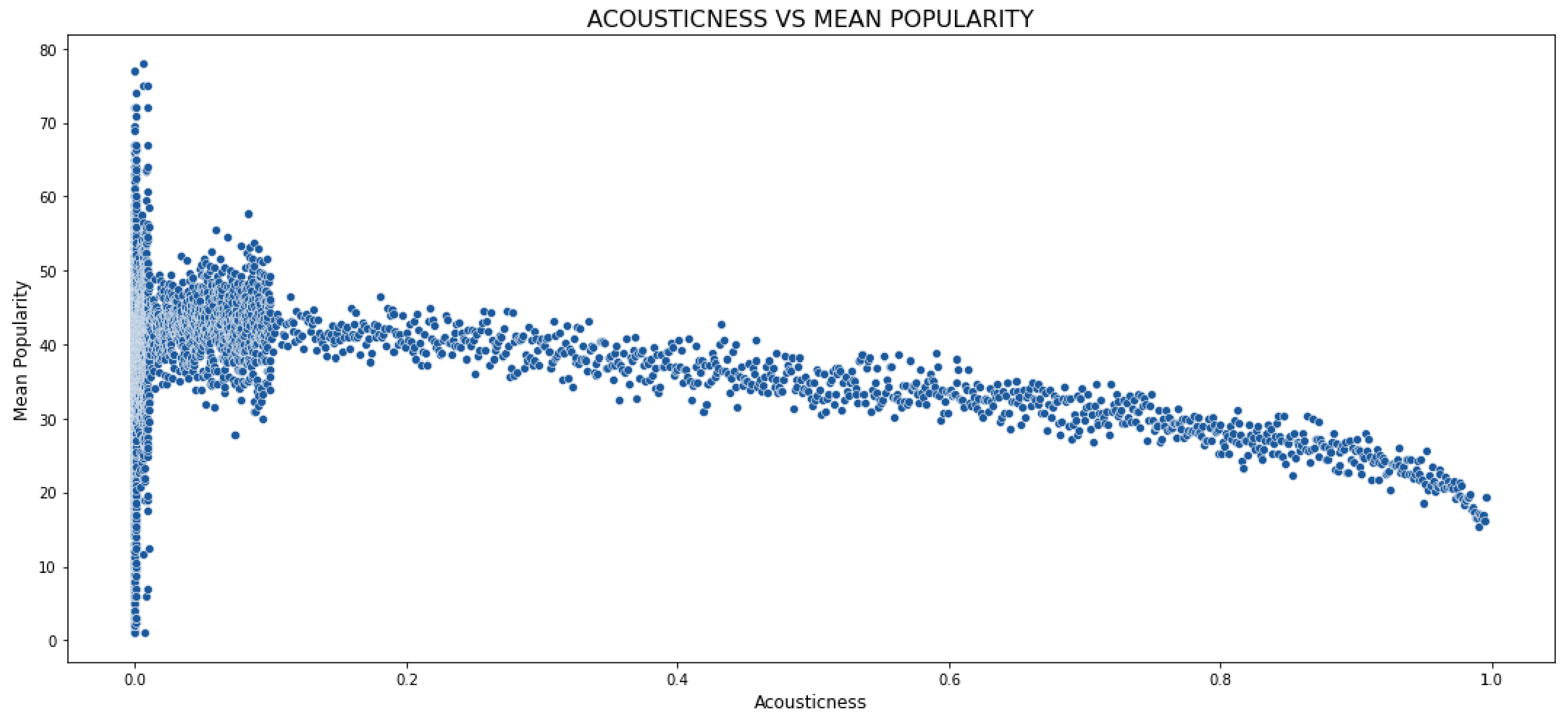




ACOUSTICNESS

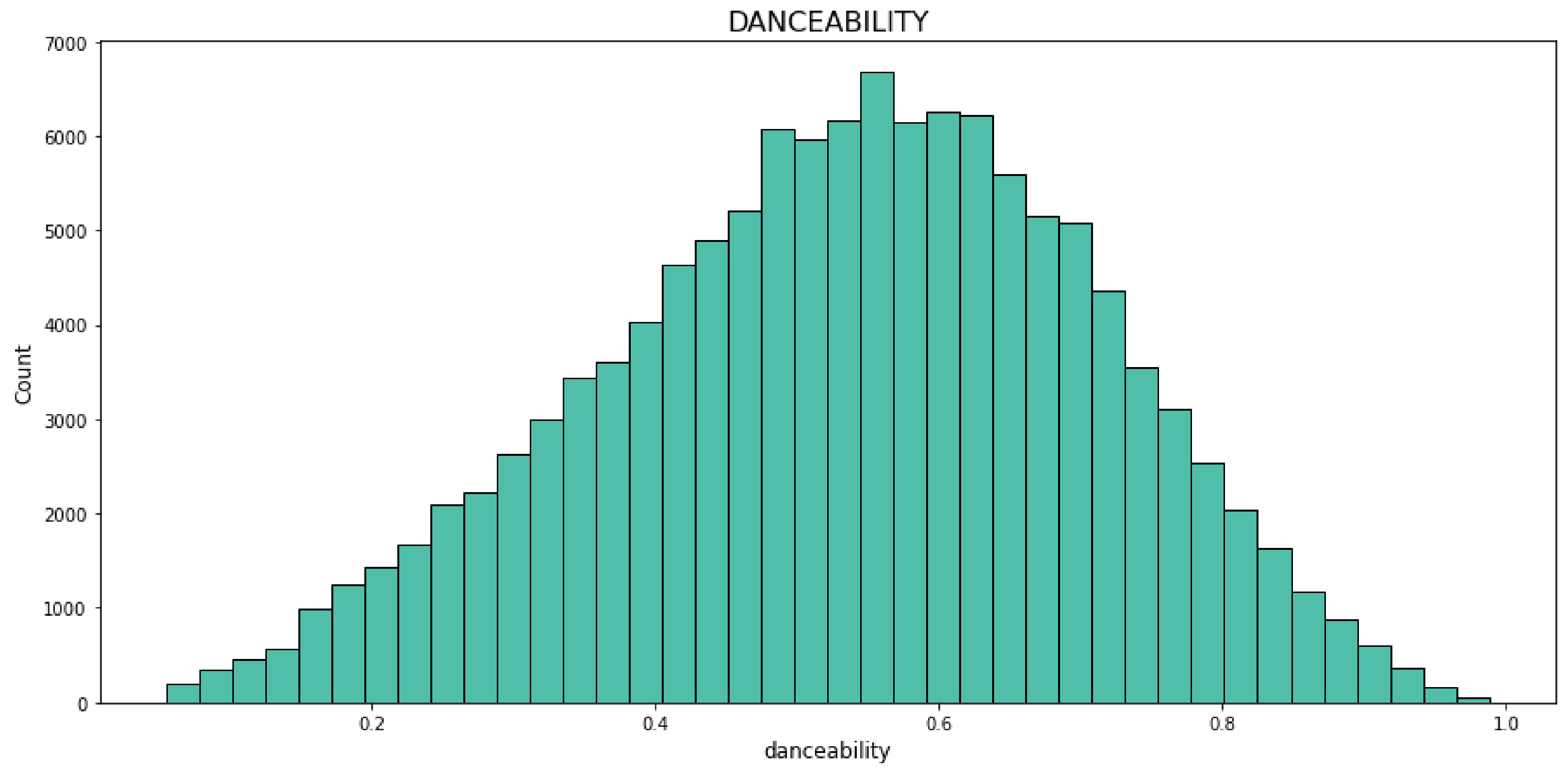
- Ranges from 0 to 1.
- A higher acousticness value means that the song is more "unplugged" and natural-sounding.
- A acousticness level closer to zero means that the song is heavily produced with many digital effects.
- The scatter plot shows that people prefer songs that are heavily produced and studio-engineered, which is the majority of pop songs we hear on the radio.

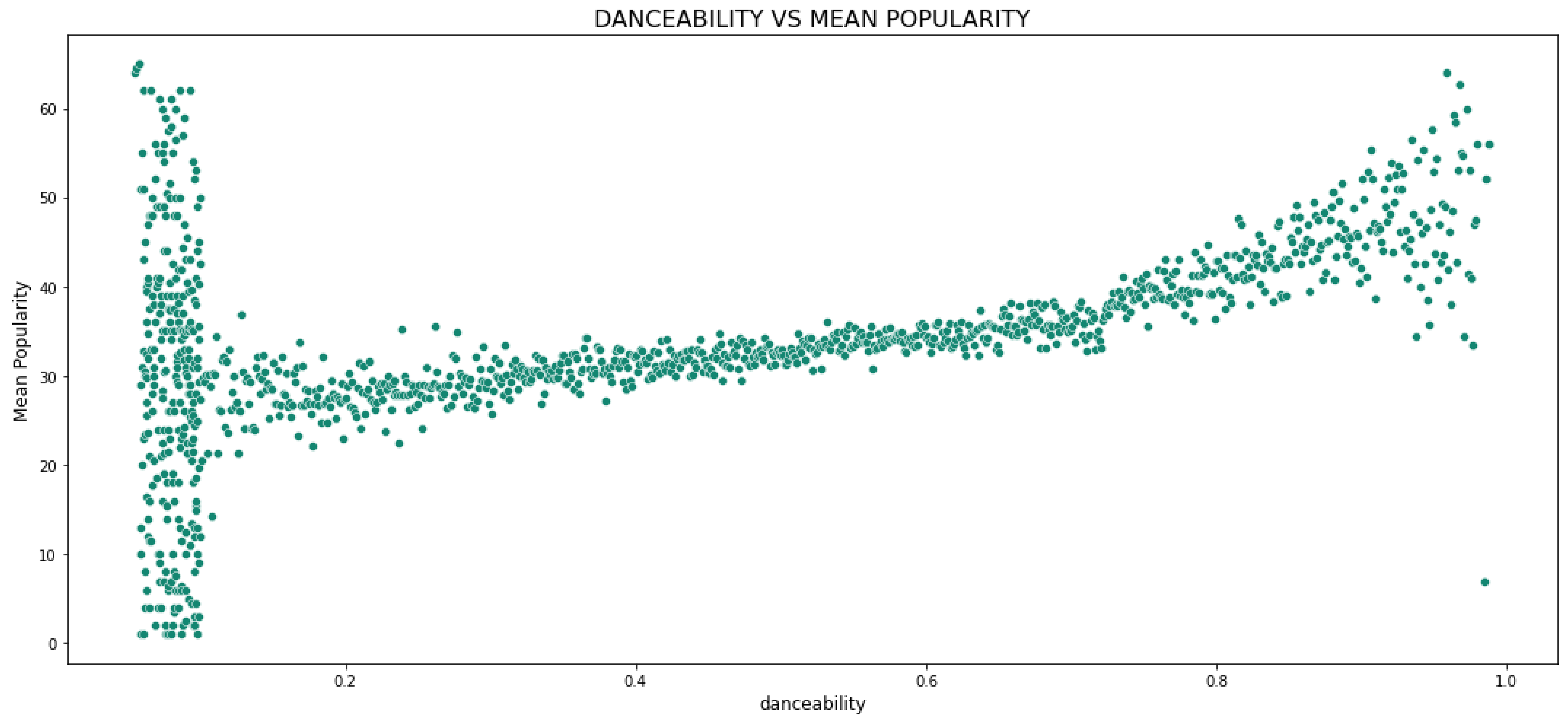




DANCEABILITY

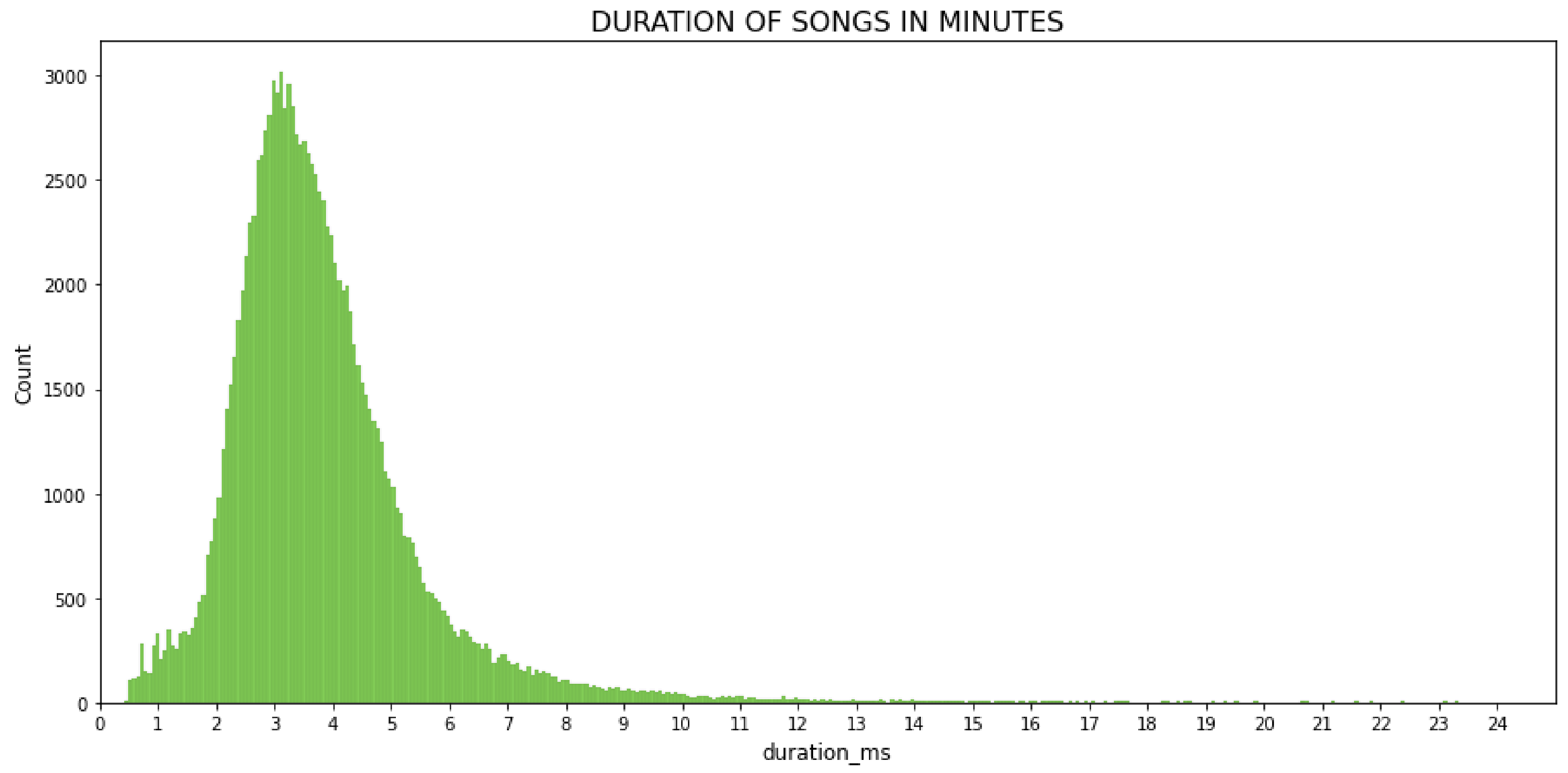
- Ranges from 0 to 1 and is normally distributed.
- A value of 0 is least danceable and 1 is most danceable.
- The skew on the left of least danceable songs reflect that sad, heartbreak songs are popular.
- At the same time, the danceable songs on the right end indicate that listeners enjoy rhythmic and up-tempo songs too.





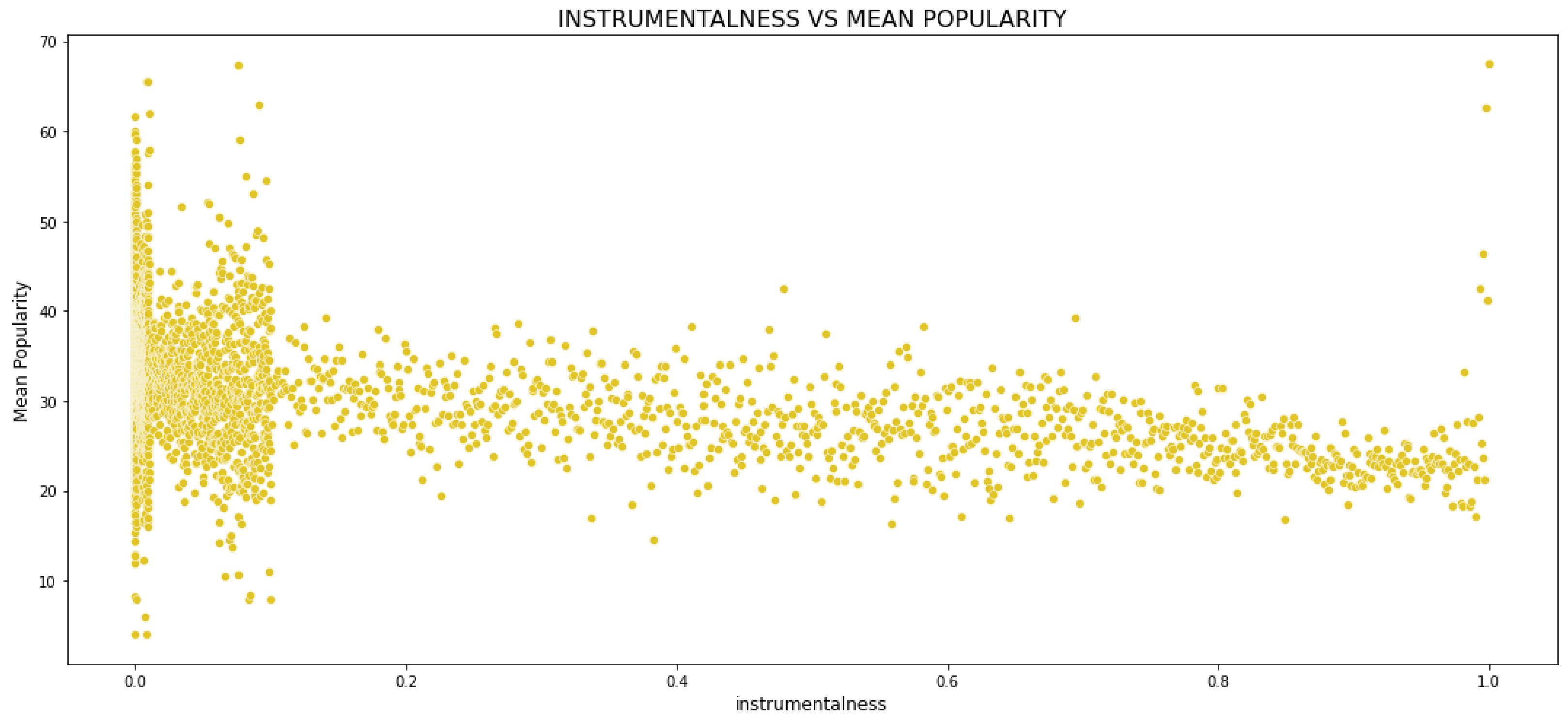
DURATION

- All songs last from 5 seconds to 90 minutes
- Most songs are between 2 to 5 minutes.
- 3.5 minutes is the holy grail duration for a pop song
- Of course, we do have some hit songs that go way longer like Bohemian Rhapsody, which is 6 minutes and American Pie by Don McLean which is over 8 minutes



INSTRUMENTALNESS

- Value being close to 1 means there are no vocals (i.e. instrumental music)
- The scatter plot shows that the instrumental values skew towards zero, which means that songs with vocals are more popular than instrumental music. *After all, who likes elevator music?*



4.3 Data Preparation

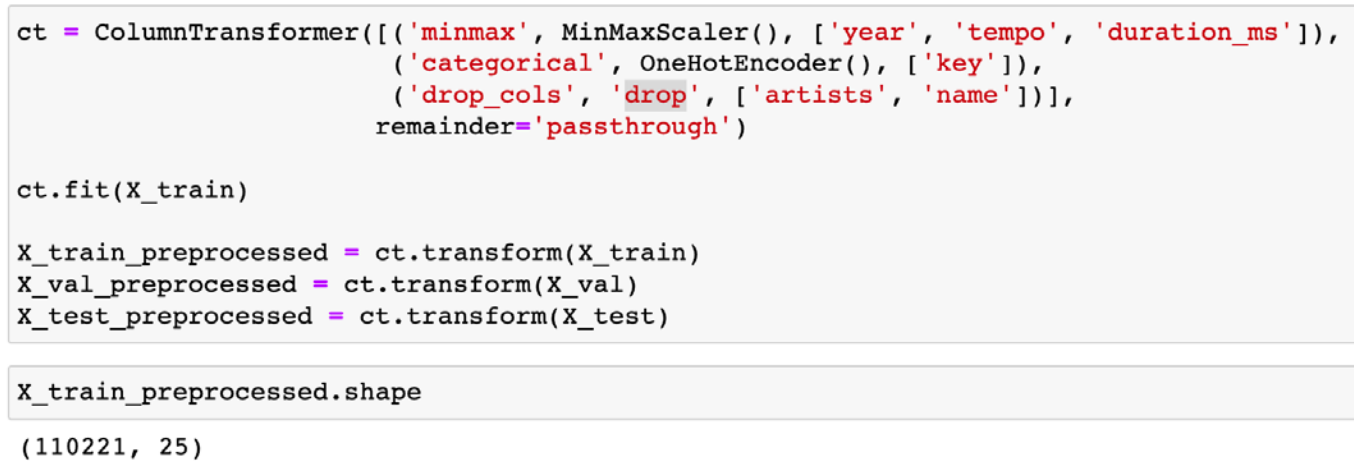
- First, we will divide our independent and dependent variable into two separate variables.
- Second, we will split the data into training and testing datasets.

```python
# First, we will divide our independent and dependent variable by dropping 'popularity' (i.e. independent variable).
# By sorting them by 'year'

X = song.sort_values(by='year').drop(columns=['popularity']).copy()
y = song.sort_values(by='year')['popularity'].copy()
```

```python
# Second, we will split the data into training and testing datasets.

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, shuffle=False)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, shuffle=False)
```

4.4 Training the Model

- Employ various preprocessing tasks such as:
- ColumnTransformer to scale the year, tempo and duration (i.e. convert to 0-1 scale)
- One-hot encoding (i.e create 10 columns taking 0/1 values)
- Drop ‘artists’ and ‘name’ as they cannot be converted to numerical values



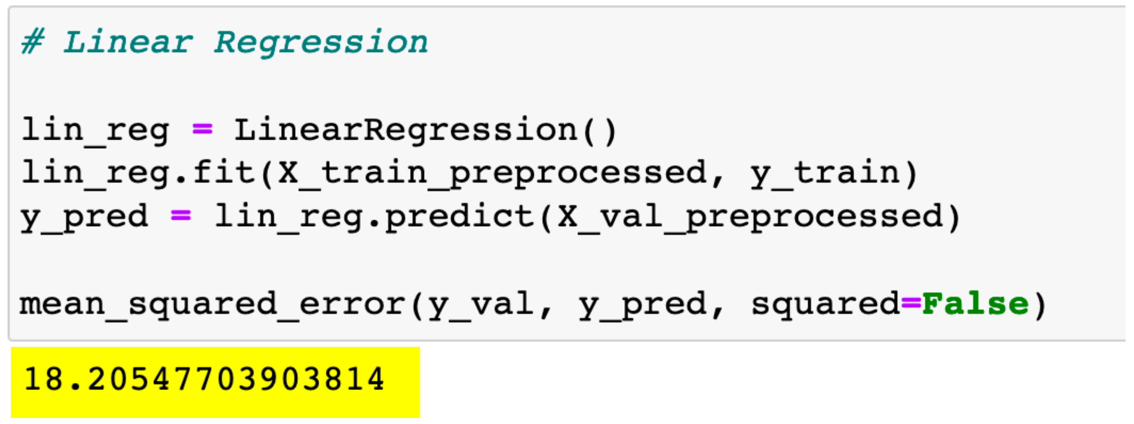
```
ct = ColumnTransformer([('minmax', MinMaxScaler(), ['year', 'tempo', 'duration_ms']),
                        ('categorical', OneHotEncoder(), ['key']),
                        ('drop_cols', 'drop', ['artists', 'name'])],
                       remainder='passthrough')

ct.fit(X_train)

X_train_preprocessed = ct.transform(X_train)
X_val_preprocessed = ct.transform(X_val)
X_test_preprocessed = ct.transform(X_test)
```

```
X_train_preprocessed.shape
```

```
(110221, 25)
```

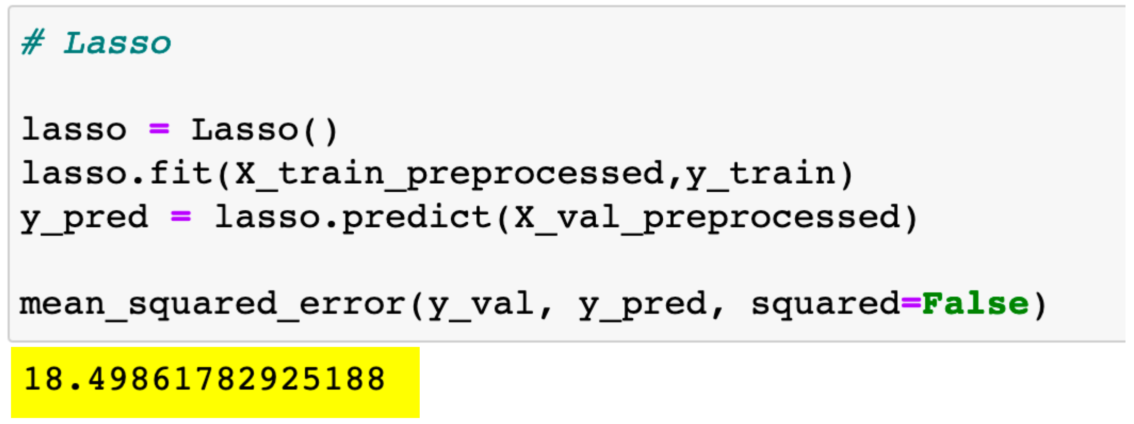
MEAN SQUARE ERROR (MSE)

- Check MSE with various regression models



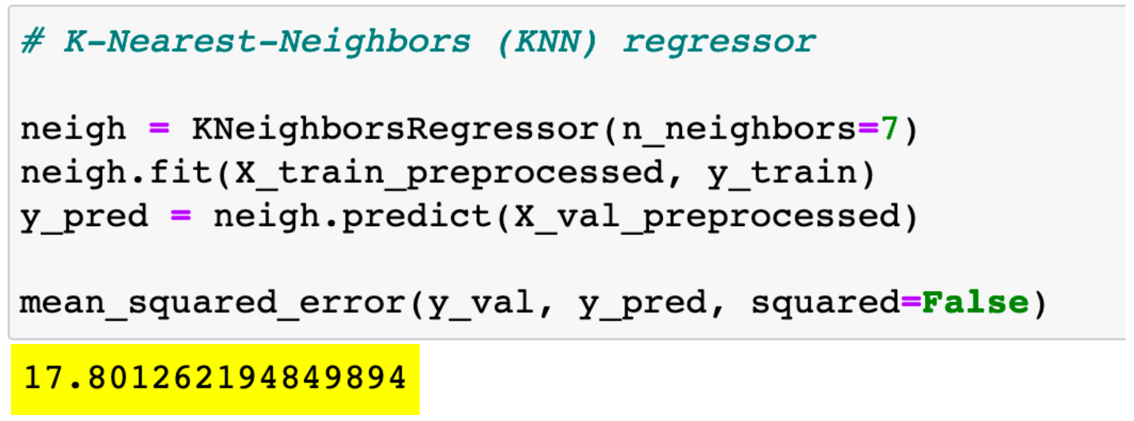
```
# Linear Regression

lin_reg = LinearRegression()
lin_reg.fit(X_train_preprocessed, y_train)
y_pred = lin_reg.predict(X_val_preprocessed)

mean_squared_error(y_val, y_pred, squared=False)
```

```
18.20547703903814
```



```
# Lasso

lasso = Lasso()
lasso.fit(X_train_preprocessed,y_train)
y_pred = lasso.predict(X_val_preprocessed)

mean_squared_error(y_val, y_pred, squared=False)
```

```
18.49861782925188
```



```
# K-Nearest-Neighbors (KNN) regressor

neigh = KNeighborsRegressor(n_neighbors=7)
neigh.fit(X_train_preprocessed, y_train)
y_pred = neigh.predict(X_val_preprocessed)

mean_squared_error(y_val, y_pred, squared=False)
```

```
17.801262194849894
```

4.5 Testing the Model

- I then train the model on a larger train set, composed of the former ‘X_train’ and ‘X_val’, and test its performances on ‘X_test’

```
X_train_old = X_train
X_train = pd.concat([X_train_old, X_val])
```

```
y_train_old = y_train.copy()
y_train = pd.concat([y_train_old, y_val])
```

```
# Defining a new dataframe that corresponds to X_train, but where popularity has not been dropped.
song_train = song.sort_values(by='year').loc[:120014].copy()
```

```
# At the plot 'Mean Popularity Over The Years', there is a sudden drop of popularity around the end of 2020.
# This may be due to Spotify's data not updating to the latest metrics.
# And since the popularity scores tend to lag on the Spotify API, we drop the songs from 2021.

X_test_no_2021=X_test[X_test['year']!=2021].copy()
y_test_no_2021=y_test[:X_test_no_2021.shape[0]].copy()
```

```
# Preprocessing the 'artists' column as seen before

artists_and_pop = {}

train_mean_pop = song_train['popularity'].mean()

for artist in X_train['artists'].unique():
    temp = song_train[song_train['artists'] == artist]['popularity'].copy()
    if len(temp) > 1:
        artists_and_pop[artist] = temp.mean()
    elif len(temp) == 1:
        artists_and_pop[artist] = train_mean_pop

X_train['artists'] = X_train['artists'].map(artists_and_pop)
X_test_no_2021['artists'] = X_test_no_2021['artists'].map(lambda artist: artists_and_pop.get(artist, train_mean_pop))
```

```
# Adding usual preprocessing steps

ct = ColumnTransformer([('minmax', MinMaxScaler(), ['year', 'tempo', 'duration_ms', 'artists']),
                        ('categorial', OneHotEncoder(), ['key']),
                        ('drop_cols', 'drop', ['name'])],
                       remainder='passthrough')

ct.fit(X_train)

X_train_preprocessed = ct.transform(X_train)
X_test_preprocessed = ct.transform(X_test_no_2021)
```

KNN REGRESSOR SCORE

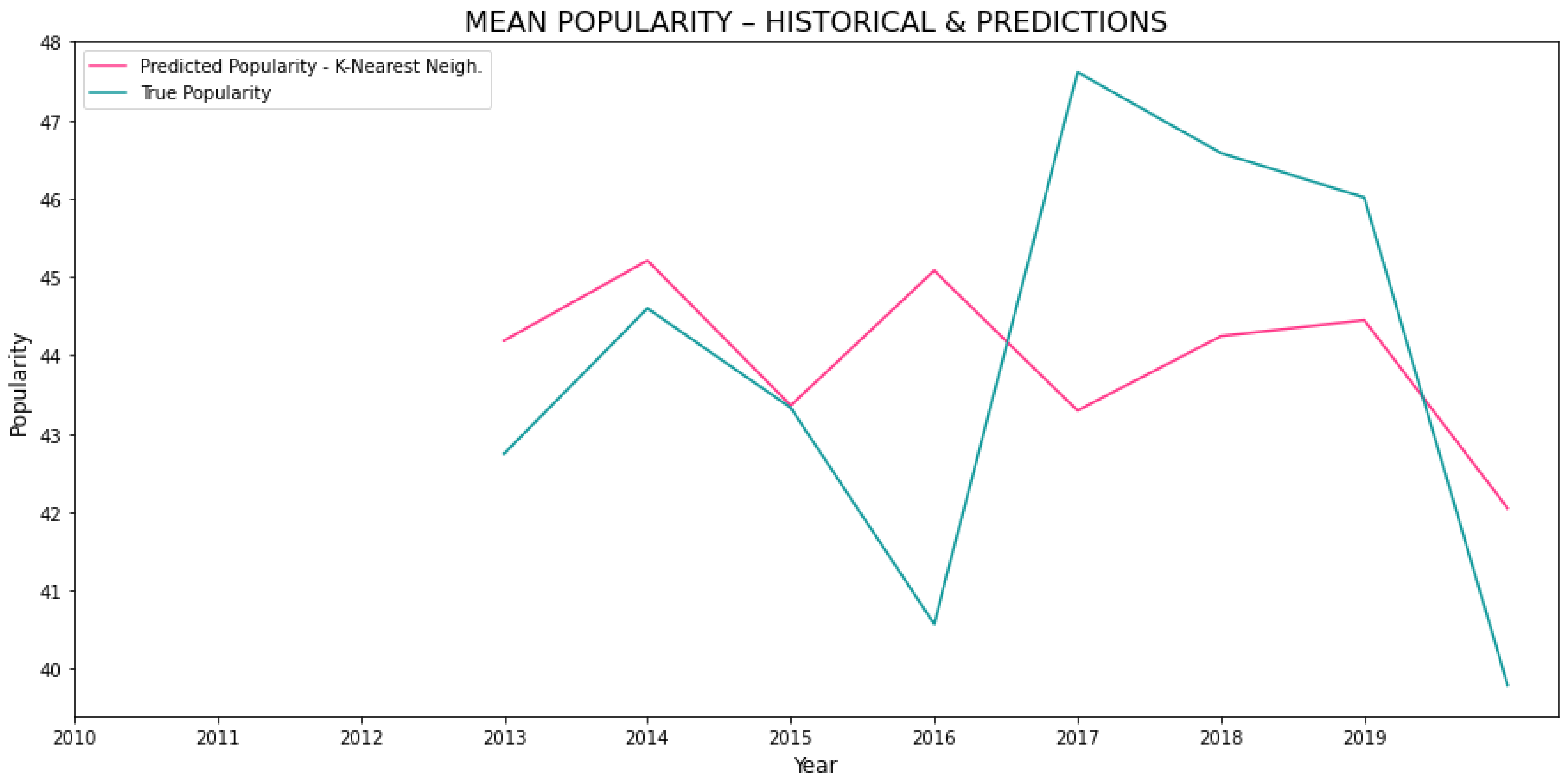
- With the new train set reconstituted, I scored my best model (KNN with 18 neighbors) on the test set.
- This is our MSE on the test set, ie. songs issued roughly between 2014 and 2020 (10% of the whole dataset).

```
neigh = KNeighborsRegressor(n_neighbors=18)
neigh.fit(X_train_preprocessed, y_train)
y_pred = neigh.predict(X_test_preprocessed)

mean_squared_error(y_test_no_2021, y_pred, squared=False)
```

```
24.41401760197151
```

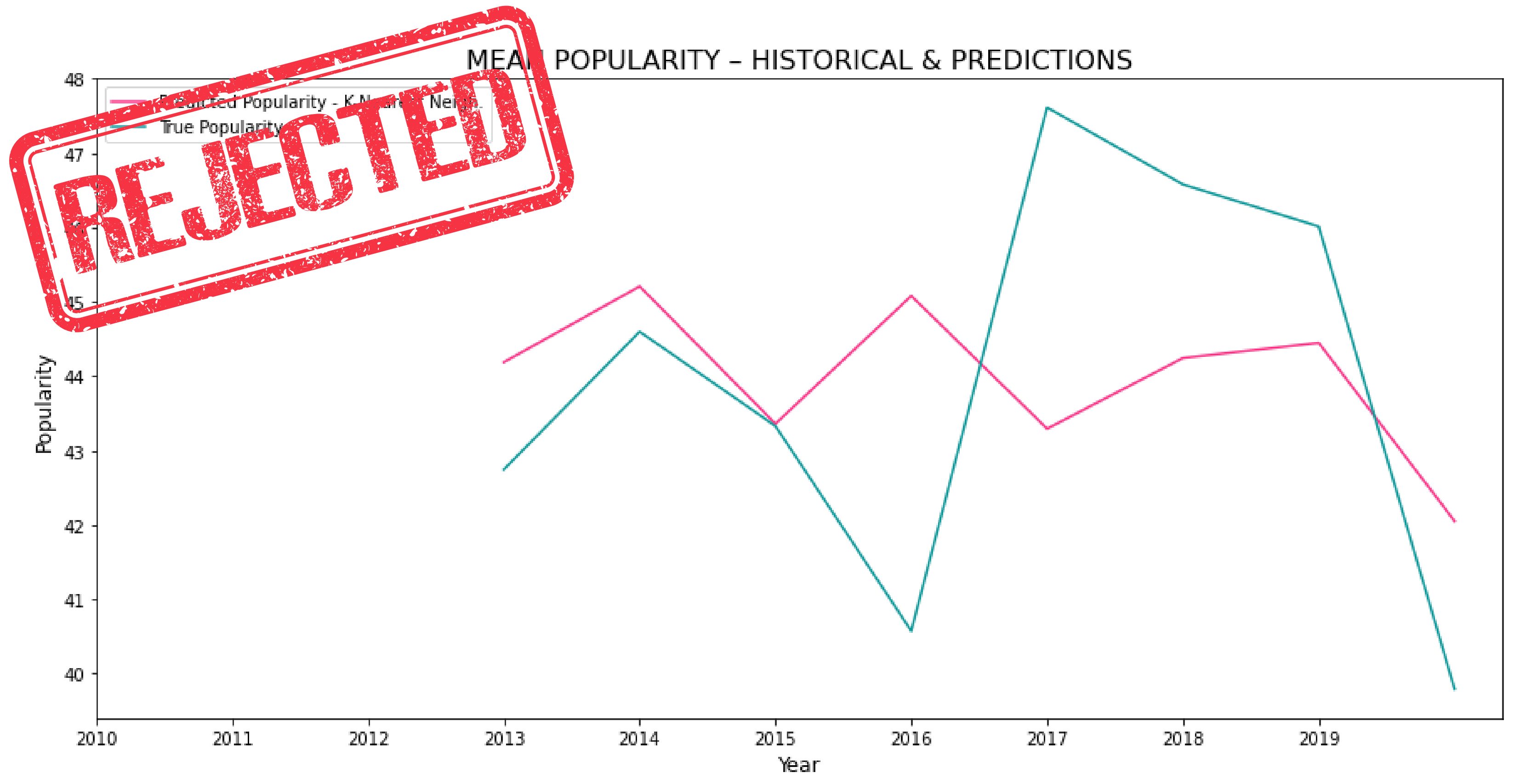
5. EVALUATION OF MODEL

- **We will plot the predicted popularity for the test set, and compare it to the true popularity for the train and test set, by sorting the songs.**



6. CONCLUSION

- We can see that my prediction is systematically higher than the real value. My model failed to adapt to sharp drop in 2016 and the spike in 2017.



7. FUTURE OPPORTUNITIES

- **ARTIST'S POPULARITY**
  Given more time and if a measurable data is available, I would include the artist's own popularity (not just the song) in the training. Current artists such as Beyoncé, Ed Sheeran and Taylor Swift do influence the song's popularity because of their star status.

- **TITLE & LYRICS**
  Another measurement would be on the song title or lyrics. There are NLP tools such as TF-IDF Vectorization which can rank the effectiveness of words, with several APIs offering such service. However, these are premium services and I'm too cheap to pay for them.

ACKNOWLEDGMENT

- Fredy Tantri
- A. M. Aditya
- Zaleha Ali for sharing her notes

Transcript of the presentation on the last page.

TRANSCRIPT

In my very first capstone project, I revealed that I was a rock-star wannabe.

Sadly, my dream didn't materialise. So I did what every failed musician would do... which is to work as a data analyst in the music industry.

SLIDE 02

Today's music artists like Beyoncé, Taylor Swift and Dua Lipa aren't just beautiful and talented, they have also possess another weapon in their arsenal — Data Science.

The modern music company today employ music analytics to predict trends, determine when's the best time to release music, find out the profiles of their listeners, set concert dates and more.

So can data science also be used to predict the next big hit?

SLIDE 03

The music industry in the US was worth over $12 billion last year.

Streaming services such as Spotify, Tidal and Apple Music accounted for 83% of this market, generating US$$10.07 billion in 2020.

Every music company and artist are trying to grab a slice of the pie, from making an average annual salary of US$36,000 to millions for big-named artists such as Beyoncé.

The past decade has seen music companies embracing data science, with the rise of "music analytics" helping big music companies to analyse trends and predict what the next big hit might be.

Producing the next big hit isn't about raw talent anymore. It's about employing big data and then choosing a song whose genre and lyrics have relevance and will go well with listeners.

SLIDE 04

Let's go through the fundamentals of this presentation.

I work for a data analytics firm called Arpeggios Analytics that specialises in music analytics.

We target small independent labels, which has surprisingly captured a quarter of this billion-dollar industry.

The big players such as Universal Music Group, Sony Music and EMI already have their own in-house music analytics department, so our clientele focus is on the independent labels.

Our clients each manages a stable of artist across a wide range of genres — from hip hop to pop to electronic dance music.

And their perennial issue or problem statement is:: How to know what will be the next hit song?

So our objective is to predict the popularity of a song using Machine Learning models.

SLIDE 05

Like many, I got my data from Kaggle, which in turn, got the data from Spotify Web API.

The data has about 174,000 songs over 100 year period from 1921 to 2020.

For this case, it will be a regression problem. So I will be using Machine Learning Models of Linear Regression and KNN.

The tools used would be the usual libraries.

Now, I shall go ahead with the work flow.

SLIDE 06

When I imported the dataset, I found a whopping 174,000 records over 19 columns of attributes.

Some of the key attributes are acousticness, energy and tempo. I'll go through each of the attribute later.

The great news was that there was no null values. However, as I pored through the data, I realised that it was rather 'dirty' with plenty of duplicates and zero values.

SLIDE 07

One of the first thing I noticed was that there was a Release_Date and Year. Upon closer scrutiny, I decided that they were both similar, so I removed Release_Date as I did not require the exact date of the song release.

Next, I found heaps of duplicates of artists and song name. Close to 15,000 duplicates.

So what I did was to keep the duplicated song with the highest popularity rating remove its lesser popular duplicates.

Next, I realised that for Tempo and Popularity, there were plenty of zero values. We can't have tempo with a zero value as it means the song is dead.

Same for popularity too... a zero value means the song shouldn't even be on the list.

SLIDE 08

There were two attributes that were peculiar — energy and loudness. Both attributes are talking about the same thing, which is intensity.

To confirm, I did a heat map and both attributes have a positive correlation of 0.78.

So to avoid multi-collinearity, I removed loudness and kept energy.

All in all, after the data was cleaned, there remained 122,000 rows of records, down from 174,000. And 16 columns, down from 19 previously.

SLIDE 09

The next step was Exploratory Data Analysis.

First off is the attribute 'popularity' as the objective is to find the song popularity.

When I I plotted the mean popularity over the entire 100 years.

After World War II, music became increasingly more popular as the economy grew.

It peaked at 2000 and started dipping. And one plausible reason could be the advent of digital music, as people stop buying CDs and started downloading music illegally.

Anyone remembers Napster?

And just for fun, I found out that The Beatles is the most popular artist over a century.

SLIDE 10

Acousticness measures if the song is natural-sounding or heavily produced with digital effects.

The scatter plot below tells us that people prefer songs which are heavily produced and studio-engineers, which is the majority of pop songs we hear on the radio.

SLIDE 11

The Danceability attribute is a normal bell curve, with the zero value being least danceable and one being most danceable.

But when we plot with mean popularity, we see that there is a skew on the left, which reflects that sad, heartbreak songs are very popular.

SLIDE 12

With regards to Duration, even though all the songs span between 5 seconds to 90 minutes, we find that most songs are between 2 to 5 minutes.

The holy grail duration for a pop song is 3 and half minutes.

I think that should be the optimal duration for Project Capstone presentations too.:)

Regarding Instrumentalness, the yellow scatter plot shows that songs with vocals are more popular that songs that are mainly instrumental. Which isn't surprising since instrumental music is much like elevator music.

SLIDE 13

Moving on quickly, I then prepared the data by dividing the independent and dependent variables into two separate data groups.

I then split the data into training and testing sets.

SLIDE 14

Before training the model, I employed ColumnTransformer and One-Hot Encoding to convert the values to a 0-1 scale.

I also dropped the columns which cannot be converted to numerical values like the artists name and song names.

After which, I did a quick check on the Mean Square Error using three regression models — Linear Regression, Lasso and KNN.

From the results, KNN had the lowest error of 17.8. So, I'll proceed with using KNN as my regression model.

SLIDE 15

I then trained the model with a larger train set, and scored my model with 18 neighbours.

This gave me a Mean Squared Error of 24.4, which is significantly higher than previously.

SLIDE 16

Finally, I plotted the predicted popularity for the test set and compared it with the true popularity data.

As you can see, my prediction is systematically higher than the real value.

The models fail to adapt to sharp drop in 2016 and the spike in 2017.

One possible reason for this discrepancy could be that there may be large outliers which won't transformed in the data.

And another possible reason for this poor regression model could be that the important attributes like artists name and song name were not included in the calculations.

SLIDE 17

In conclusion, my model failed in predicting the song popularity with the true data.

Moving forward, if more time was given, like 7 years, I would include the artist's own popularity. This is because top artists do influenced the song's popularity because of their star status.

Another thing I would improve on is to include measuring the song title and lyrics.

There are NLP tools such as TF-IDF Vectorization, which can actually rank the effectiveness of words.

However, these are premium services which I'm too cheap to pay at the moment.

SLIDE 18

Last but not the least, I would like to thank Fredy, Aditya and Zaleha for sharing her notes.