# Song popularity prediction

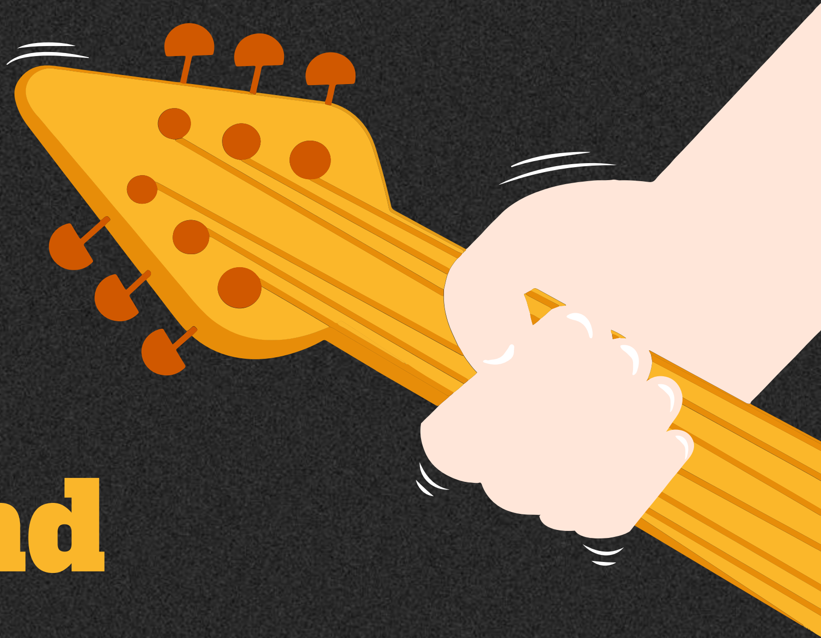
# For aspiring songwriters

SC1015 Mini-Project by Team 8: Ye Chuan and Elaine

# TABLE OF CONTENTS

# 01

# Motivation and dataset



# Motivation

## Aspiring songwriters

- Predict our song written will become successful
- Find out what makes our song successful

## Most popular music streaming platform as of 2023

- Popularity of song on Spotify would be an accurate measure of song success

# Problem Definition

"What are the key song attributes that would determine if a song would become popular on Spotify?"

Response: Popularity
Predictors: Song attributes

# About the dataset

**Extracted from:** kaggle
**Dataset compiled by:** Zaheen Hamidani

## What is in the dataset

232,725 song from 26 different genres as well as 18 different song attributes.

02

# Data Exploration and Cleaning

# General Data Exploration and Cleaning

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 232725 entries, 0 to 232724
Data columns (total 18 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   genre             232725 non-null  object
 1   artist_name       232725 non-null  object
 2   track_name        232725 non-null  object
 3   track_id          232725 non-null  object
 4   popularity        232725 non-null  int64
 5   acousticness      232725 non-null  float64
 6   danceability      232725 non-null  float64
 7   duration_ms       232725 non-null  int64
 8   energy            232725 non-null  float64
 9   instrumentalness  232725 non-null  float64
 10  key               232725 non-null  object
 11  liveness          232725 non-null  float64
 12  loudness          232725 non-null  float64
 13  mode              232725 non-null  object
 14  speechiness       232725 non-null  float64
 15  tempo             232725 non-null  float64
 16  time_signature    232725 non-null  object
 17  valence           232725 non-null  float64
dtypes: float64(9), int64(2), object(7)
memory usage: 32.0+ MB
```

General information about the entries

```
pd.isnull(raw).sum()

genre               0
artist_name         0
track_name          0
track_id            0
popularity          0
acousticness        0
danceability        0
duration_ms         0
energy              0
instrumentalness    0
key                 0
liveness            0
loudness            0
mode                0
speechiness         0
tempo               0
time_signature      0
valence             0
dtype: int64
```

Checking for NaN/ Null values

# General Data Exploration and Cleaning

Finding out what are the different types music genres avaliable

```
[ ] raw["genre"].nunique()
```

27

27 track genres instead of 26?

Let's use the `.unique()` method below to list the genres

```
raw["genre"].unique()
```

```
array(['Movie', 'R&B', 'A Capella', 'Alternative', 'Country', 'Dance',
       'Electronic', 'Anime', 'Folk', 'Blues', 'Opera', 'Hip-Hop',
       "Children's Music", 'Children’s Music', 'Rap', 'Indie',
       'Classical', 'Pop', 'Reggae', 'Reggaeton', 'Jazz', 'Rock', 'Ska',
       'Comedy', 'Soul', 'Soundtrack', 'World'], dtype=object)
```

Duplicated genres

| | |
|---|---|
| Comedy | 9681 |
| Soundtrack | 9646 |
| Indie | 9543 |
| Jazz | 9441 |
| Pop | 9386 |
| Electronic | 9377 |
| Children’s Music | 9353 |
| Folk | 9299 |
| Hip-Hop | 9295 |
| Rock | 9272 |
| Alternative | 9263 |
| Classical | 9256 |
| Rap | 9232 |
| World | 9096 |
| Soul | 9089 |
| Blues | 9023 |
| R&B | 8992 |
| Anime | 8936 |
| Reggaeton | 8927 |
| Ska | 8874 |
| Reggae | 8771 |
| Dance | 8701 |
| Country | 8664 |
| Opera | 8280 |
| Movie | 7806 |
| Children's Music | 5403 |
| A Capella | 119 |

Name: genre, dtype: int64

Before cleaning

| | |
|---|---|
| Children's Music | 14756 |
| Comedy | 9681 |
| Soundtrack | 9646 |
| Indie | 9543 |
| Jazz | 9441 |
| Pop | 9386 |
| Electronic | 9377 |
| Folk | 9299 |
| Hip-Hop | 9295 |
| Rock | 9272 |
| Alternative | 9263 |
| Classical | 9256 |
| Rap | 9232 |
| World | 9096 |
| Soul | 9089 |
| Blues | 9023 |
| R&B | 8992 |
| Anime | 8936 |
| Reggaeton | 8927 |
| Ska | 8874 |
| Reggae | 8771 |
| Dance | 8701 |
| Country | 8664 |
| Opera | 8280 |
| Movie | 7806 |
| A Capella | 119 |

Name: genre, dtype: int64

After cleaning

# General Data Exploration and Cleaning

### Duplicated Tracks

The `track_id` is said to be a unique identifier of each track on Spotify.

```
print("Number of duplicated track_id", pop_country["track_id"].duplicated().sum())
```

```
Number of duplicated track_id 306
```

| | genre | artist_name | track_name |
|---|---|---|---|
| 109008 | Pop | Devin Dawson | All on Me |
| 212354 | Country | Devin Dawson | All On Me |
| 112896 | Pop | Devin Dawson | Asking for a Friend |
| 212781 | Country | Devin Dawson | Asking For A Friend |

Repeated song names

```
[ ] # Sort entries in decscending order of popularity
    # Next, drop duplicated track_id and only keep th
    print(f"No. of Songs before dropping: {pop_countr
    pop_country = pop_country.sort_values(by="popular
    print(f"No. of Songs after dropping: {pop_country

    print(f"Entries with duplicated track_id {pop_cou
```

```
No. of Songs before dropping: 15659
No. of Songs after dropping: 15353
Entries with duplicated track_id 0
```

Remove Duplicates

# Exploring the distribution of Numerical Song Attributes

**Such as:**

## General Numerical Song Attributes

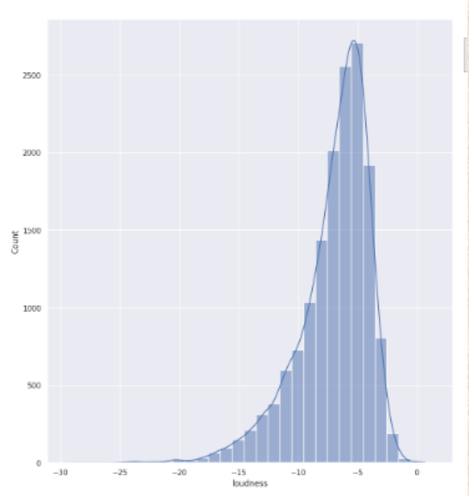
- Tempo
- Loudness
- Duration

## Spotify determined Numerical Song Attributes

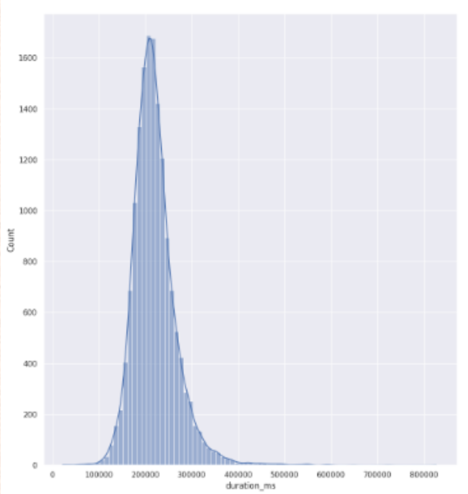
- Energy
- Instrumentalness
- Acousticness
- Speechiness
- Danceability
- Liveness
- Valence

# Exploring the distribution of Numerical Song Attributes

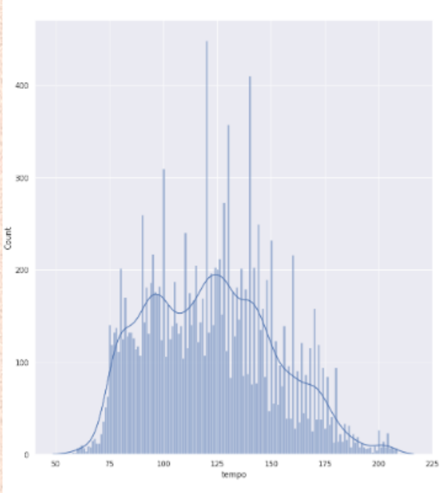
## Such as:




Loudness



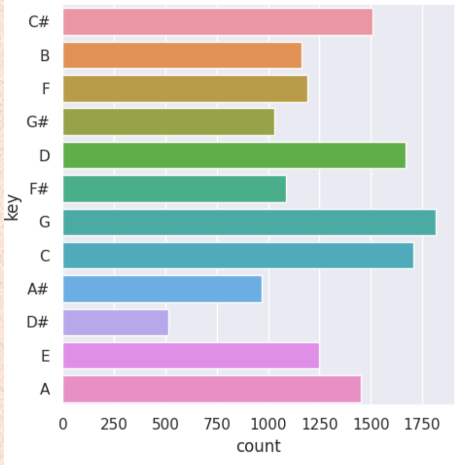

Duration_ms



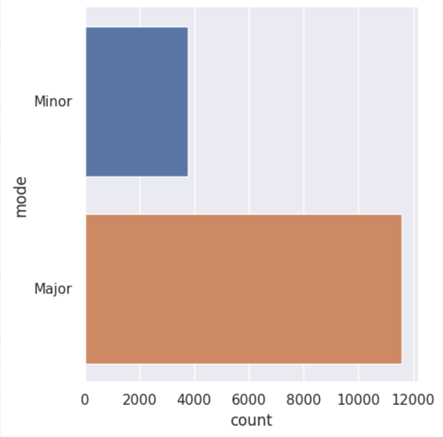

Tempo

# Exploring the distribution of Categorical General Song Attributes

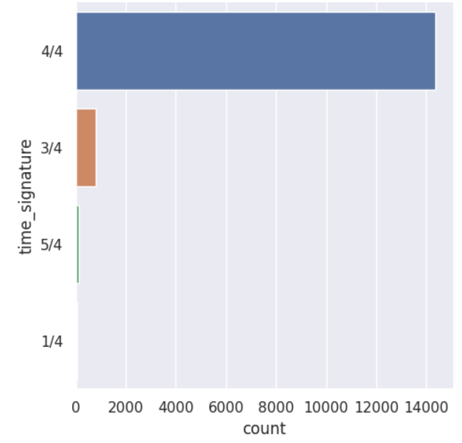
## Such as:




key



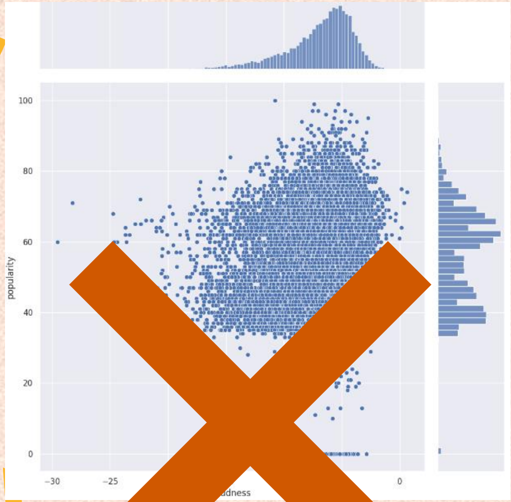

mode

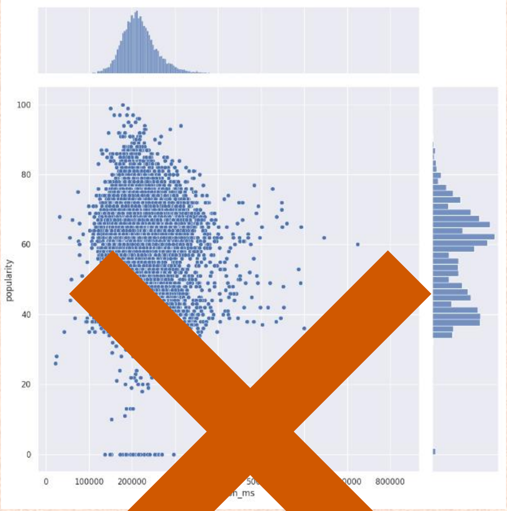



time_signature

# Exploring the correlation between song attributes and popularity

Numerical Variables





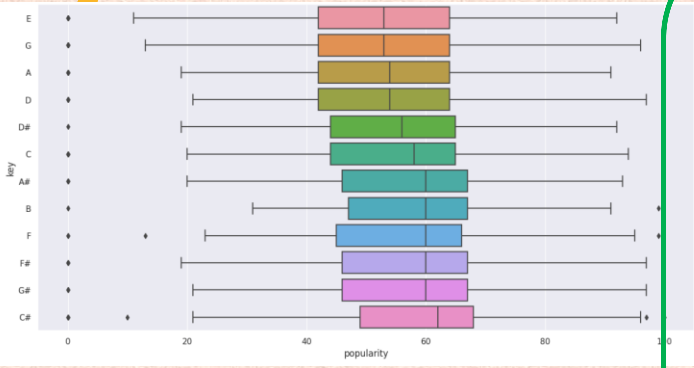
Loudness

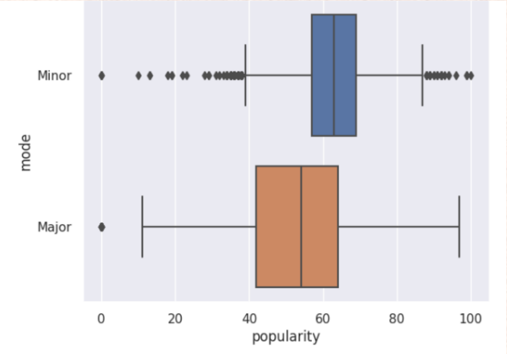
Duration_ms

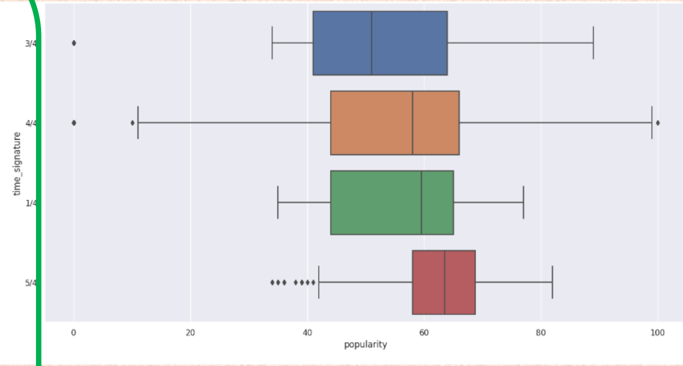
Tempo

# Exploring the correlation between song attributes and popularity

Categorical Variables




key

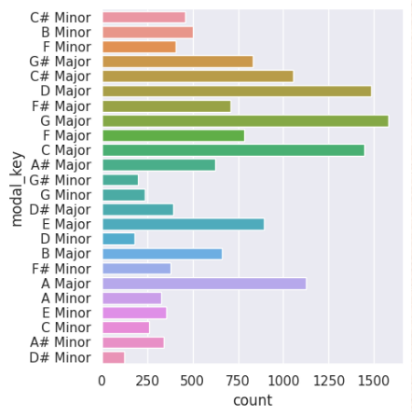
mode

Time_signature

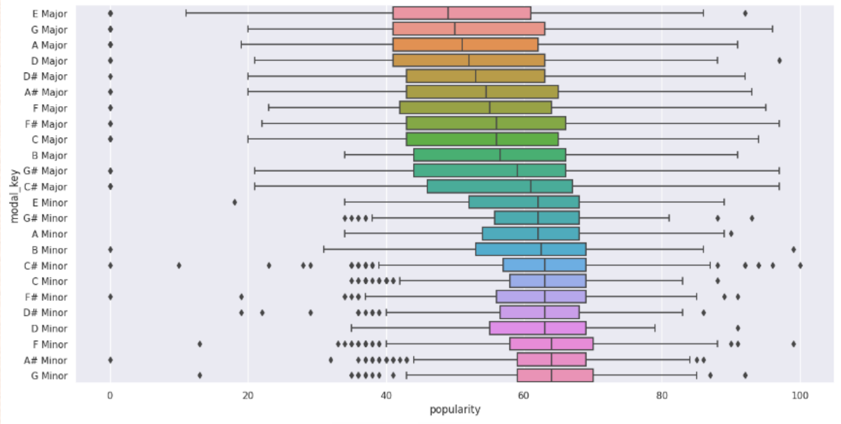
# Creation of new categorical variable

## modal_key

**Created by merging key and mode.**
**Rational: In general, it is more common to have songs categorized in terms of it's key and mode (major/minor) together instead of having it's key and mode separated.**




Distribution of modal_key




Relationship with popularity

# Reducing size of dataset

**Rational: The scope of the dataset is too wide. We have to narrow our focus on the specific genre and language of our interest.**

## Genre of interest: Pop and Country

```
[ ]  pop_country = raw.loc[(raw["genre"]=="Pop") | (raw["genre"]=="Country")]
     pop_country["genre"].value_counts()

     Pop        9386
     Country    8664
     Name: genre, dtype: int64
```

Keeping entries with pop or country as genres only

# Reducing side of dataset

**Rational: The scope of the dataset is too wide. We have to narrow our focus on the specific genre and language of our interest.**

## Language of Interest : English

```python
pop_country = pop_country[pop_country["track_name"].apply(detect_english)]

print("After Removing Non-English Titles")
pop_country["genre"].value_counts()
```

```
After Removing Non-English Titles
Country     8072
Pop         7587
Name: genre, dtype: int64
```

Removing non-English titled songs

03

# Song Lyrics

Addition of song attributes

# Obtaining Song Lyrics

## API Calls

Genius API

- Response to song search queries
- Does **not** provide lyrics in plaintext
- Provides a URL to the lyrics page

## Web Scrape

Genius Lyrics

- BeautifulSoup4
- Multiple Separated Chunks
- Removal of section tags (i.e. [Chorus], [Verse1])

# Obtaining Song Lyrics

## Time Taken

**9+ Hours**

**15000+ Songs**

```
05:54:42 [1/15659]: Cam - My Mistake
05:54:44 [2/15659]: Kevin Fowler - That Girl
05:54:46 [3/15659]: Roger Miller - Chug-A-Lug
05:54:47 [4/15659]: M. Ward - Chinese Translation
05:54:49 [5/15659]: Chris Cagle - Anywhere But Here
05:54:50 [6/15659]: Cassadee Pope - I Wish I Could Break Your Heart
05:54:53 [7/15659]: Mark Chesnutt - Too Cold At Home
05:54:54 [8/15659]: Gary Allan - Get Off On The Pain
05:54:56 [9/15659]: Hank Williams, Jr. - Old Habits
05:54:57 [10/15659]: Ryan Adams - When The Stars Go Blue
05:55:00 [11/15659]: Adam Sanders - Sippin' on the Good Times
05:55:01 ARTIST MISMATCH: Adam Sanders - Sippin' on the Good Times <-SEAR
```

# 04 NLP

Natural Language Processing

# Tokenization

**Pre-Processing**

Dash Removal (e.g. oh-oh-oh)

**Tokenization**

Break into smallest units of meaning

| | lyrics | tokenized |
|---|---|---|
| 107804 | Yeah, breakfast at Tiffany's and bottles of bu... | [Yeah, ,, breakfast, at, Tiffany, 's, and, bot... |
| 107802 | You got me some type of way (Hmm)\nAin't used ... | [You, got, me, some, type, of, way, (, Hmm, ),... |
| 107829 | Oh, she's sweet but a psycho\nA little bit psy... | [Oh, ,, she, 's, sweet, but, a, psycho, A, lit... |
| 107808 | Found you when your heart was broke\nI filled ... | [Found, you, when, your, heart, was, broke, I,... |
| 107838 | High, high hopes\nHad to have high, high hopes... | [High, ,, high, hopes, Had, to, have, high, ,,... |

# Stop Words

```
["a", "about", "ai", "ain", "am", "an", "and", "any", "are", "aren", "aren't", "n't", "as", "at", "be", "been", "but",
```

```
"doesn't", "doing", "doin'", "doin", "don", "don't", "down", "during", "durin'", "durin", "each", "few", "for", "from",
```

```
"for", "from", "further", "had", "hadn", "hadn't", "has", "hasn", "hasn't", "have", "haven", "haven't"
```

# Stemming

**Stemming** — Quick way to reduce related words to the same "stem

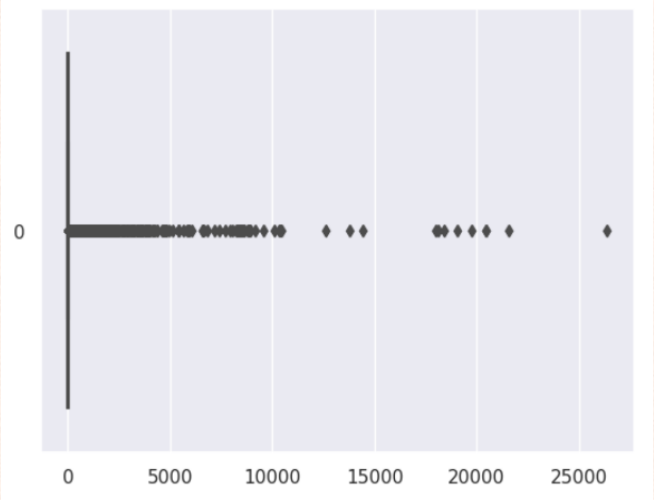
**Lemmatization** — Similar, but attempts to reduce to actual words

| | tokenized | stems |
|---|---|---|
| 107804 | [Yeah, breakfast, Tiffany, bottles, bubbles, G... | [yeah, breakfast, tiffani, bottl, bubbl, girl,... |
| 107802 | [got, type, way, Hmm, used, feelin, way, Mmm, ... | [got, type, way, hmm, use, feelin, way, mmm, m... |
| 107829 | [Oh, she, sweet, psycho, little, bit, psycho, ... | [oh, she, sweet, psycho, littl, bit, psycho, n... |
| 107808 | [Found, heart, broke, filled, cup, overflowed,... | [found, heart, broke, fill, cup, overflow, too... |
| 107838 | [High, high, hopes, high, high, hopes, living,... | [high, high, hope, high, high, hope, live, sho... |

# Bag-of-Words

Simple model to bag words together and count their frequencies


Bag-of-Words Model



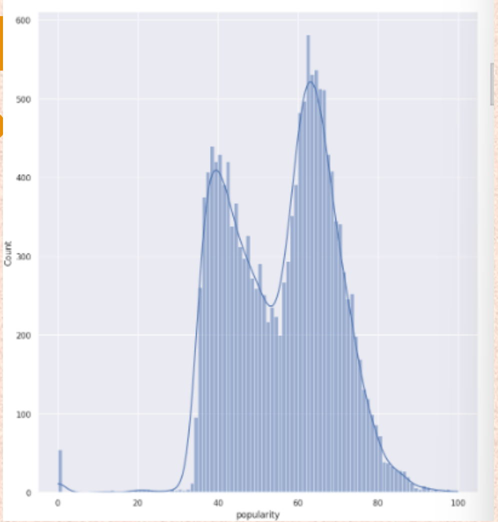

Frequency of words in our corpus

Image: https://www.freecodecamp.org/news/text-classification-and-prediction-using-bag-of-words-8aeb1396cded/

# 05 Machine Learning

Models used and Rational behind using these models

# Response variable




Numerical

**Finalising the range of values**

Since we want to split `popularity` ratings into **5 different category** without having significant issues later due to class imbalance, we shall split our data equally into the 5 categories.

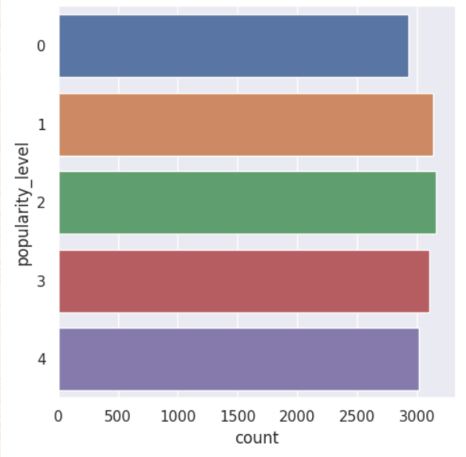
Each categories should have an estimate of around $\frac{15353}{5} = 3070.6$ entries.

Hence, we could split our data into the 5 categories as follows:

```
#Creating new column to store popularity as category
pop_country["popularity_level"] = 0 #'Low'

pop_country.loc[(pop_country["popularity"]>=42) & (pop_country["popularity"]<=51),
pop_country.loc[(pop_country["popularity"]>=52) & (pop_country["popularity"]<=61),
pop_country.loc[(pop_country["popularity"]>=62) & (pop_country["popularity"]<=67),
pop_country.loc[(pop_country["popularity"]>=68) & (pop_country["popularity"]<=100),

pop_country["popularity_level"].value_counts()
```

```
2    3161
1    3139
3    3104
4    3020
0    2929
Name: popularity_level, dtype: int64
```

Categorisation




Categorical

# Words Selection

# Chi-Squared Test

```
[ ]  selected_stems_mask = selector.get_support()
     selected_stems = stems[selected_stems_mask]
     selected_stems[390:415]
```

```
array(['about', 'abov', 'abram', 'abroad', 'abrupt', 'absentmind',
       'absinth', 'absolut', 'absolv', 'abstain', 'abstract',
       'abstractionist', 'absurd', 'abu', 'abus', 'abyss', 'ac', 'academ',
       'acapella', 'acapulco', 'accardi', 'acceler', 'accent', 'accept',
       'accessori'], dtype=object)
```

## 18271 Words → 15000 Words

# Feature Encoding

**Mode**

**Major → 1**
**Minor → 0**

**Encode categories into number to feed into model**

| | mode | mode_encoded |
|---|---|---|
| 107804 | Minor | 0 |
| 107802 | Minor | 0 |
| 107829 | Major | 1 |
| 107808 | Major | 1 |
| 107838 | Major | 1 |
| ... | ... | ... |
| 212806 | Major | 1 |
| 7471 | Major | 1 |
| 216687 | Major | 1 |
| 215996 | Major | 1 |
| 216430 | Major | 1 |

9093 rows × 2 columns

# Model selection – Using Cross-Validation

## Naïve Bayes Classifier

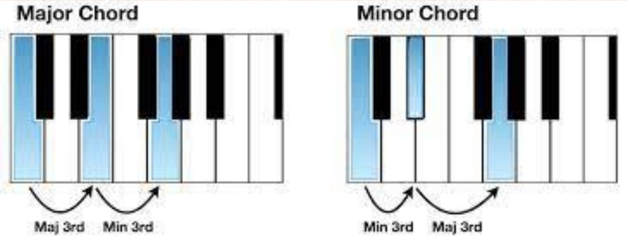
```
# 5-Fold Cross-Validation
bernoulli_scores = cross_val_score(bernoulli, features_set, pop_country["popularity_level"], cv=5)
multinomial_scores = cross_val_score(multinomial, features_set, pop_country["popularity_level"], cv=5)

print("Benoulli Test Set Accuracy:")
print(f"\tMean: {bernoulli_scores.mean()}")
print(f"\tStandard Deviation: {bernoulli_scores.std()}")


print("Multinomial Test Set Accuracy:")
print(f"\tMean: {multinomial_scores.mean()}")
print(f"\tStandard Deviation: {multinomial_scores.std()}")
```

```
Benoulli Test Set Accuracy:
	Mean: 0.3368496937654183
	Standard Deviation: 0.011992435647182501
Multinomial Test Set Accuracy:
	Mean: 0.3287115407527558
	Standard Deviation: 0.012285173361276032
```

06

# Outcome and insights

# Insights




**Accuracy: 25%**

Mode only



**Spotify determined Numerical Song Attributes**

Energ

Instrumentaln

Acc

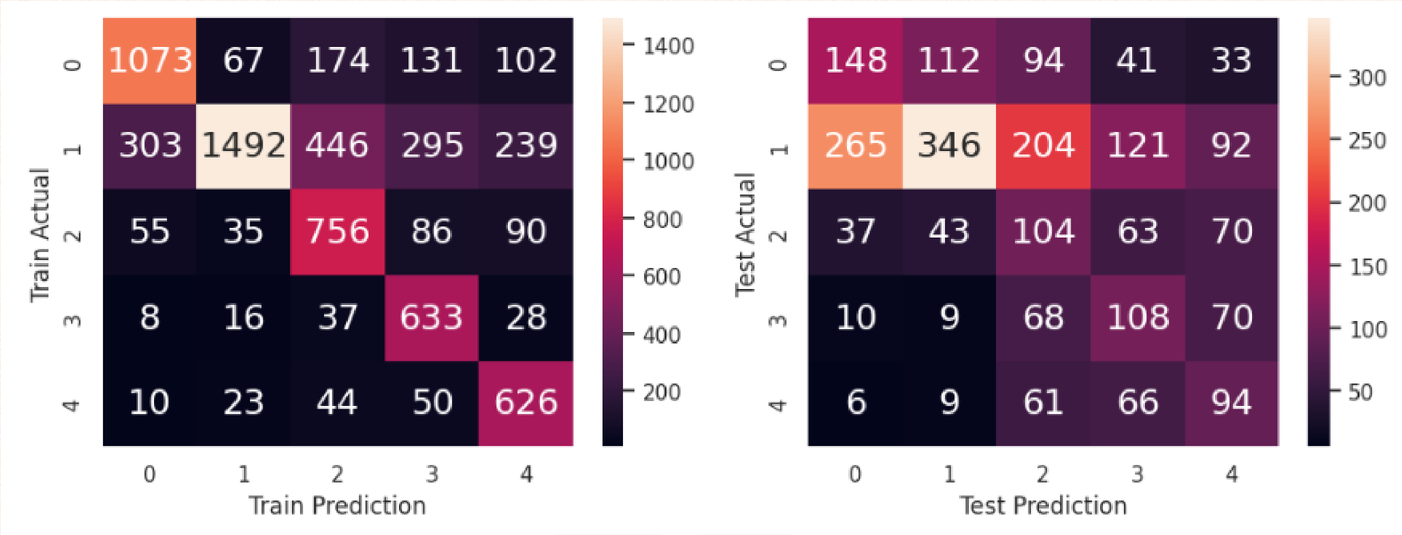
Speechiness

Danceability

Valence

Adding Lyric

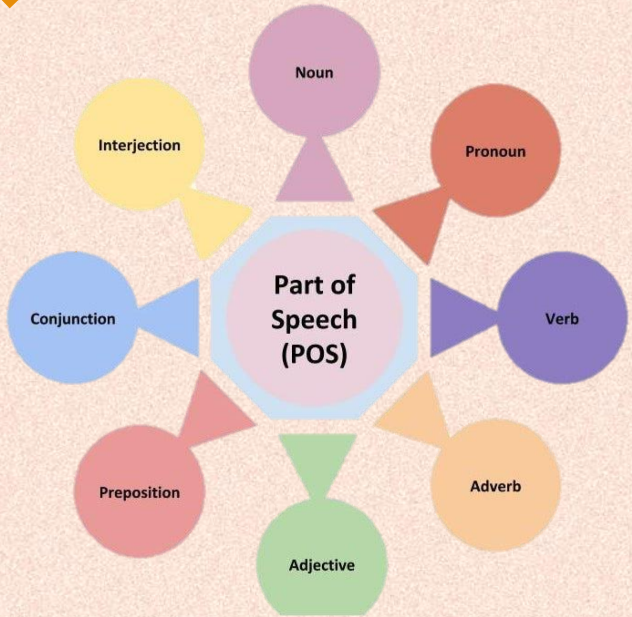
# Outcomes

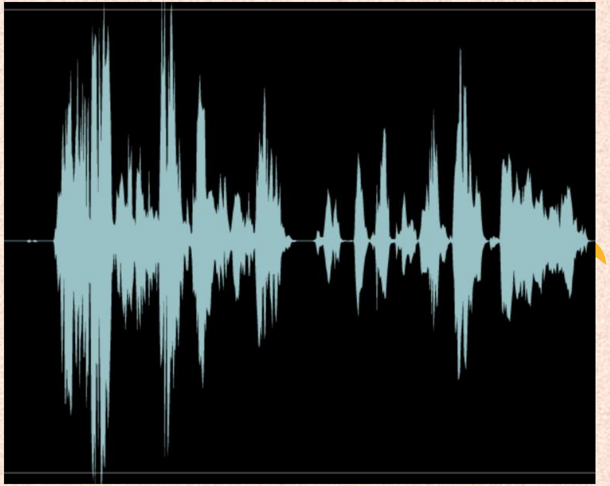



**Accuracy: 60%**

**Accuracy: 35%**

# Improvements



Audio Analysis

Image: https://suneelpatel18.medium.com/nlp-pos-part-of-speech-tagging-chunking-f72178cc7385

# Thank you