SC1015

Spotify Music Analysis

Graciella Theodora (U2110112E)
Lee Jia Wen (U2110242G)
Lim Wan Loong (U2110848A)


Motivation

PLAYLIST

Top 100 tracks currently on Spotify

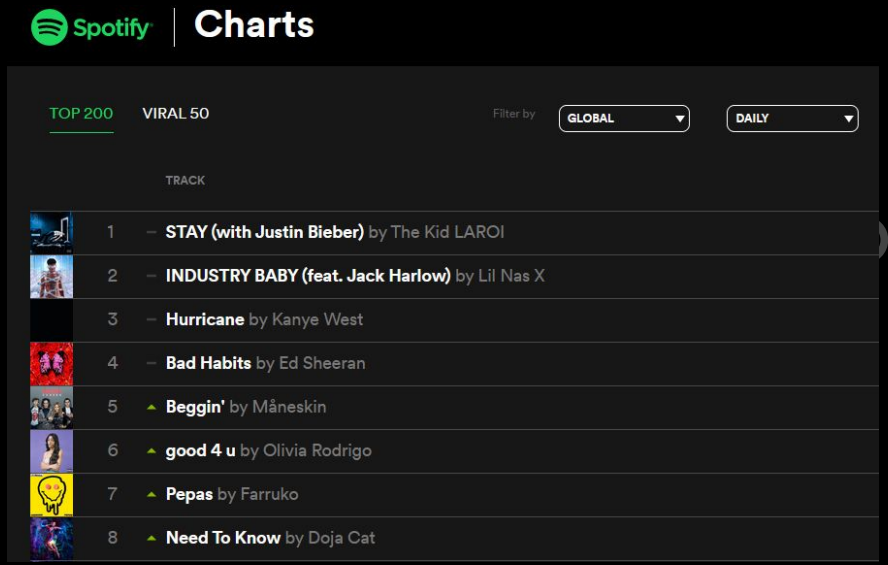
Created by: Spotify • 100 songs, 6 hr 12 min

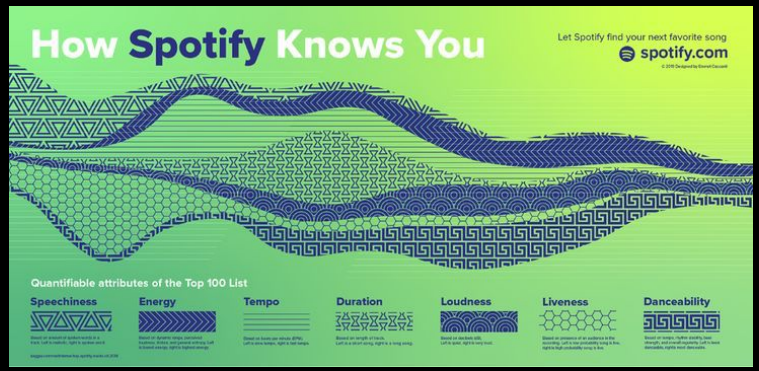
PLAY FOLLOWING

Filter Download

Motivation


Spotify for Developers


Spotify
Charts
TOP 200
VIRAL 50
Filter by
GLOBAL
DAILY
TRACK
1 STAY (with Justin Bieber) by The Kid LAROI
2 INDUSTRY BABY (feat. Jack Harlow) by Lil Nas X
3 Hurricane by Kanye West
4 Bad Habits by Ed Sheeran
5 Beggin' by Måneskin
6 good 4 u by Olivia Rodrigo
7 Pepas by Farruko
8 Need To Know by Doja Cat


How Spotify Knows You
Let Spotify find your next favorite song
spotify.com
Quantifiable attributes of the Top 100 List
Speechiness
Energy
Tempo
Duration
Loudness
Liveness
Danceability

Motivation

Spotify for Developers

How important are the attributes in predicting popularity

Whether there are any attributes that can be tweaked to increase popularity


Data Preparation
& Cleaning

01 Spotipy

Python library for the **Spotify Web API**, to gain access to **music data** provided by the Spotify platform

```
import spotipy
import spotipy.util as util

from spotipy.oauth2 import SpotifyClientCredentials
```

02 Fetch Playlist Tracks

Created a function to get the **track id** and **track name**

```python
def fetch_playlist_tracks(sp, username, playlist_id):
    """
    Returns the tracks for the given playlist.
    """

    offset = 0
    tracks = []

    # Make the API request
    while True:
        content = sp.user_playlist_tracks(username, playlist_id, fields=None, limit=100, offset=offset, market=None)
        tracks += content['items']

        if content['next'] is not None:
            offset += 100
        else:
            break

    track_id = []
    track_name = []
    track_genre = []

    for track in tracks:
        track_id.append(track['track']['id'])
        track_name.append(track['track']['name'])
        #track_genre.append(track['track']['genre'])

    # Create the final df
    df_playlists_tracks = pd.DataFrame({"track_id":track_id, "track_name": track_name})
    return df_playlists_tracks
```

	track_id	track_name
0	7qiZfU4dY1lWllzX7mPBI3	Shape of You
1	0VjIjW4GlUZAMYd2vXMi3b	Blinding Lights
2	1rgnBhdG2JDFTbYkYRZAku	Dance Monkey
3	0e7ipj03S05BNilyu5bRzt	rockstar (feat. 21 Savage)
4	7qEHsqek33rTcFNT9PFqLf	Someone You Loved
...	...	...
786	4EWCNWgDS8707fNSZ1oaA5	Heartless
787	67iAlVNDDdddxqSD2EZhFs	I'm Gonna Be (500 Miles)
788	1dGr1c8CrMLDpV6mPbImSI	Lover
789	2iUmqdfGZcHIhS3b9E9EWq	Everybody Talks
790	4DpNNXFMMxQEKl7r0ykkWA	Play Date

03 Extract Audio Features

Created a function, to get the audio features like **danceability, energy, tempo, popularity** etc.

```
def fetch_audio_features(sp, username, playlist_id):
    playlist = fetch_playlist_tracks(sp, username, playlist_id)
    index = 0
    audio_features = []

    while index < playlist.shape[0]:
        audio_features += sp.audio_features(playlist.iloc[index:index + 50, 0])
        index += 50

    features_list = []
    for features in audio_features:
        features_list.append([features['danceability'],
                              features['energy'], features['tempo'],
                              features['loudness'], features['valence'],
                              features['speechiness'], features['instrumentalness'],
                              features['liveness'], features['acousticness']])

    df_audio_features = pd.DataFrame(features_list, columns=['danceability', 'energy',
                                                             'tempo', 'loudness', 'valence',
                                                             'speechiness', 'instrumentalness',
                                                             'liveness', 'acousticness'])

    df_playlist_audio_features = pd.concat([playlist, df_audio_features], axis=1)
    df_playlist_audio_features.set_index('track_id', inplace=True, drop=True)

    return df_playlist_audio_features
```

track_name	danceability	energy	tempo	loudness	valence	speechiness	instrumentalness	liveness	acousticness
Shape of You	0.825	0.652	95.977	-3.183	0.931	0.0802	0.000000	0.0931	0.58100
Blinding Lights	0.514	0.730	171.005	-5.934	0.334	0.0598	0.000095	0.0897	0.00146
Dance Monkey	0.826	0.593	98.083	-6.401	0.541	0.0976	0.000161	0.1700	0.68800

01 Overview of data set

- 791 data points
- No-null values

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 791 entries, 0 to 790
Data columns (total 18 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   track_id           791 non-null    object
 1   track_name_x       791 non-null    object
 2   popularity         791 non-null    int64
 3   track_uri          791 non-null    object
 4   artist_name        791 non-null    object
 5   artist_popularity  791 non-null    int64
 6   artist_genres      791 non-null    object
 7   album              791 non-null    object
 8   track_name_y       791 non-null    object
 9   danceability       791 non-null    float64
 10  energy             791 non-null    float64
 11  tempo              791 non-null    float64
 12  loudness           791 non-null    float64
 13  valence            791 non-null    float64
 14  speechiness        791 non-null    float64
 15  instrumentalness   791 non-null    float64
 16  liveness           791 non-null    float64
 17  acousticness       791 non-null    float64
dtypes: float64(9), int64(2), object(7)
memory usage: 111.4+ KB
```

02 Drop unnecessary columns

- track_id, album, track_name_y, track_uri

	track_name_x	popularity	artist_name	artist_popularity	danceability	energy
	Shape of You	89	Ed Sheeran	96	0.825	0.652
	Blinding Lights	95	The Weeknd	97	0.514	0.730
	Dance Monkey	68	Tones And I	78	0.826	0.593
	rockstar (feat. 21 Savage)	88	Post Malone	91	0.585	0.520
	Someone You Loved	90	Lewis Capaldi	82	0.501	0.405

tempo	loudness	valence	speechiness	instrumentalness	liveness	acousticness
95.977	-3.183	0.931	0.0802	0.000000	0.0931	0.58100
171.005	-5.934	0.334	0.0598	0.000095	0.0897	0.00146
98.083	-6.401	0.541	0.0976	0.000161	0.1700	0.68800
159.801	-6.136	0.129	0.0712	0.000070	0.1310	0.12400
109.891	-5.679	0.446	0.0319	0.000000	0.1050	0.75100

03 Recast numeric values that are categorical

- **Categorical variables:** Speechiness, instrumentalness & liveness
- **Instrumentalness:** predicts whether a track contains no vocals.

  Values > 0.5 : No vocals (instrumental); Values <0.5 : Vocal

instrumentalness
0.000000
0.000095
0.000161
0.000070
0.000000

```
bins =[-np.inf,0.5,1]
instrumentalness_cat = pd.cut(songdf.instrumentalness,bins,
                              labels=['Vocal', 'No Vocals'])
songdf.insert(11,'instrumental', instrumentalness_cat)
```

instrumental
Vocal
Vocal
Vocal
Vocal
Vocal

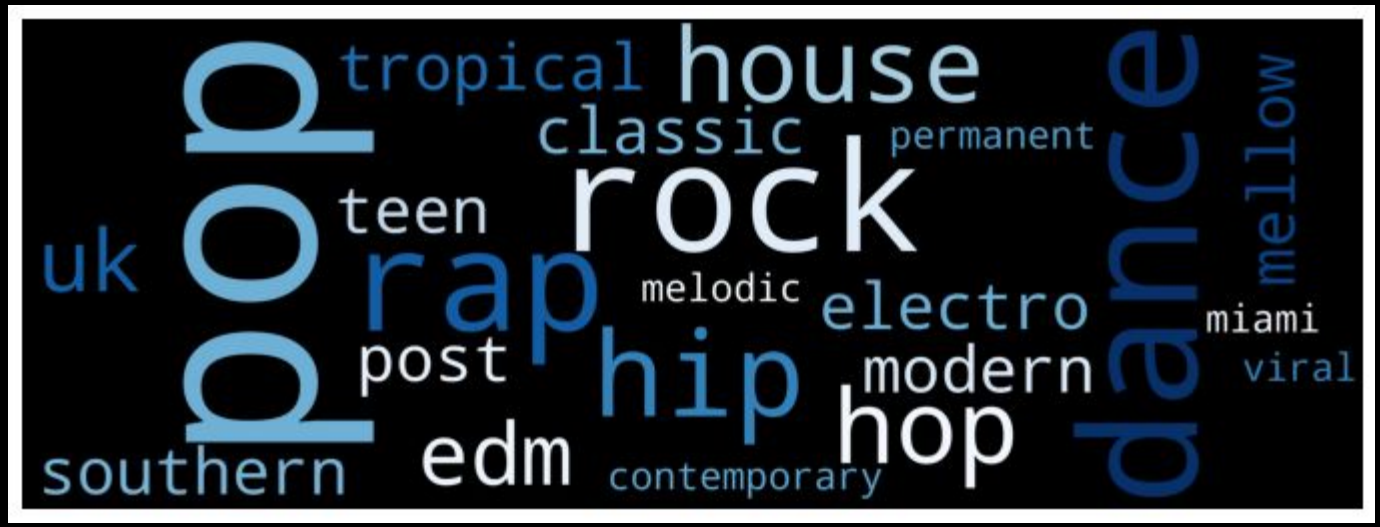
04 One Hot Encoding of Categorical Variables for ML

speechiness	instrumental
Words	Vocal
Words	Vocal
Words	Vocal
Words	Vocal
Words	Vocal
...	...
Words	Vocal
Words	Vocal
Words	Vocal
Words	Vocal
Words	Vocal

speech_Mix	speech_Words	instrumental_No Vocals	instrumental_Vocal
0.0	1.0	0.0	1.0
0.0	1.0	0.0	1.0
0.0	1.0	0.0	1.0
0.0	1.0	0.0	1.0
0.0	1.0	0.0	1.0
...	...	...	...
0.0	1.0	0.0	1.0
0.0	1.0	0.0	1.0
0.0	1.0	0.0	1.0
0.0	1.0	0.0	1.0
0.0	1.0	0.0	1.0


Exploratory
Data Analysis

Word Cloud

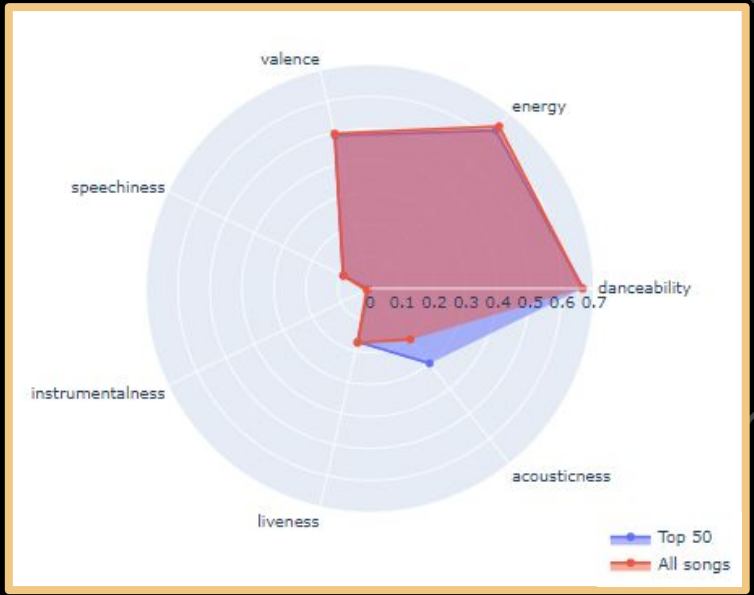

pop
tropical
house
classic
permanent
rock
teen
uk
rap
melodic
electro
miami
post
hip
modern
viral
hop
dance
mellow
southern
edm
contemporary

Radar Plot

Top 50 songs were compared against the rest of the songs

Insights:

- Top 50 songs have **higher acoustic values**
- Other attributes are similar as songs are of similar type (top song tracks)

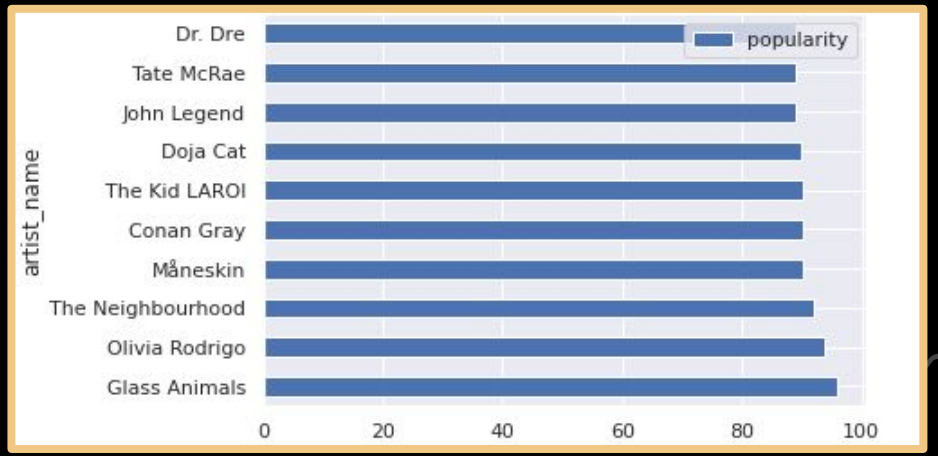


Univariate visualisation 1: Top Artists

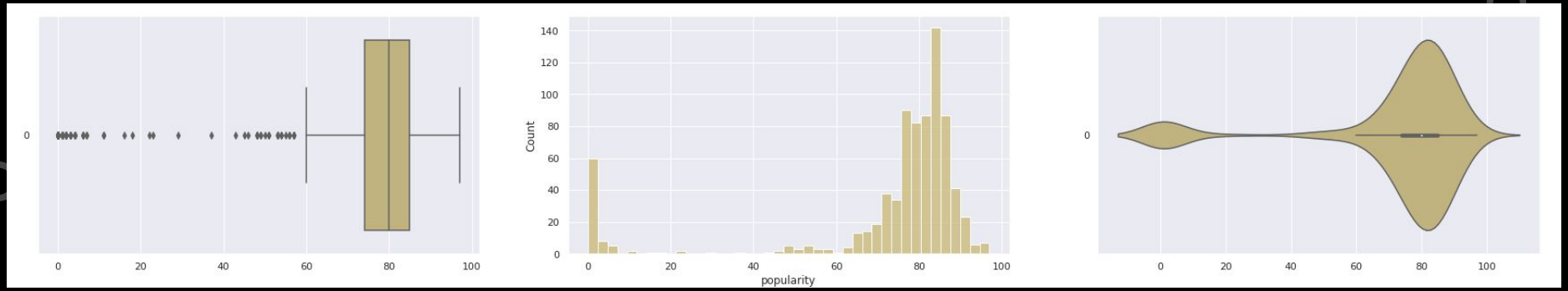
Artists who produce songs with the **highest mean popularity**

Insights:

- Glass Animals, Olivia Rodrigo & The Neighbourhood produce the **most popular songs** on average



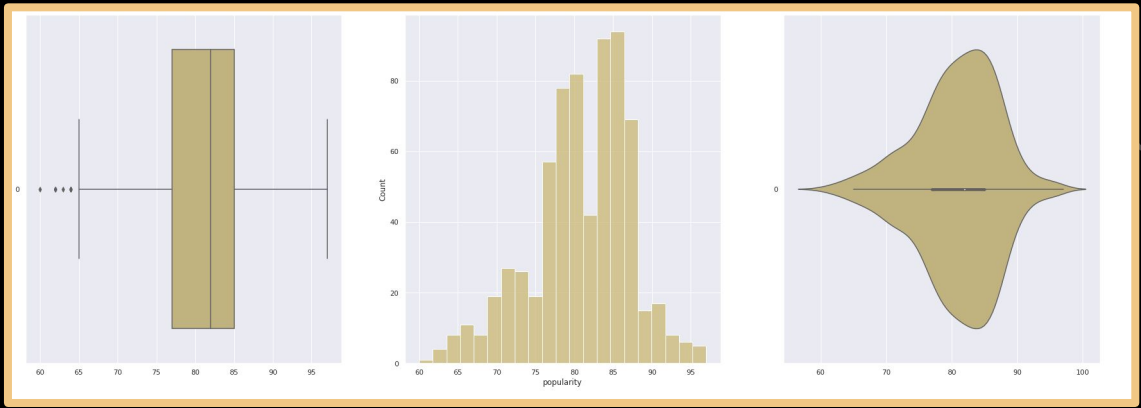
Univariate Visualisation 2: Popularity



- Large number of songs with **0 popularity**
- Skewness: -2.24

Univariate Visualisation 2 : Popularity

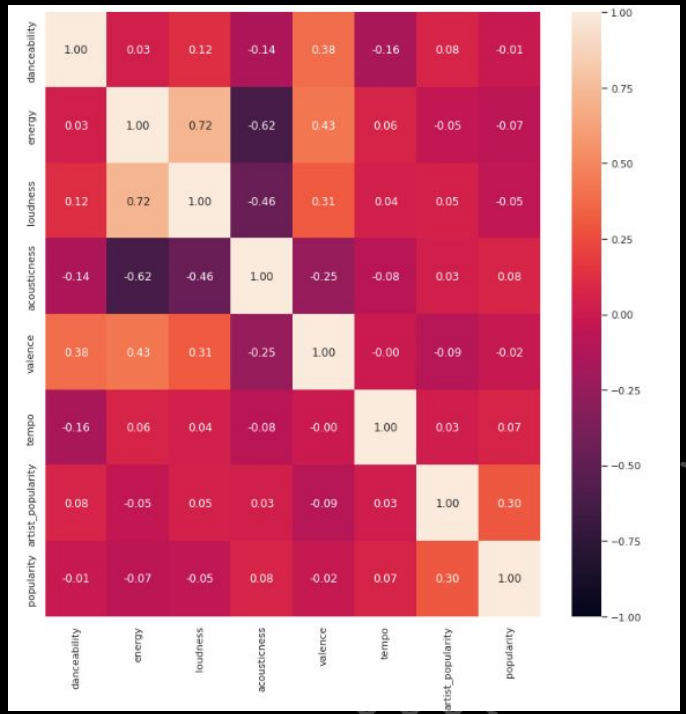
- Outliers removed using **IQR method**
- Popular songs like “Demons - Imagine Dragon” & “Mirrors - Justin Timberlake” had ‘0’ popularity → **highly unlikely**
- Skewness: -0.52 (previously -2.24)



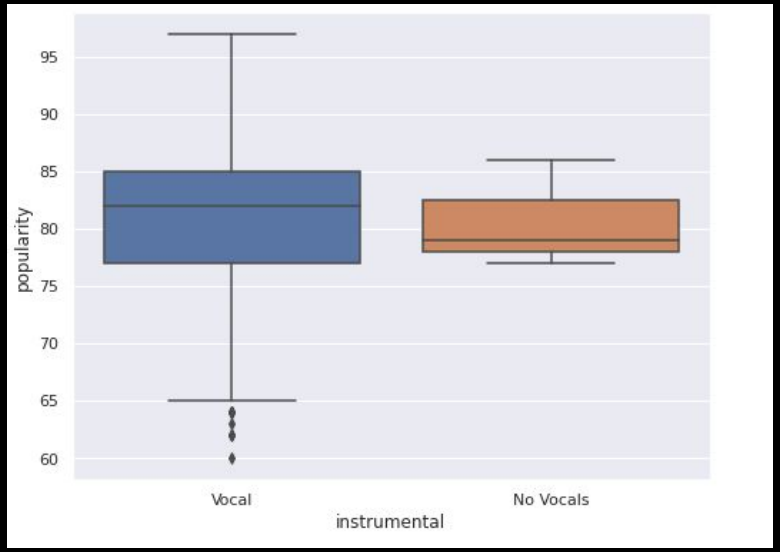
Multivariate visualisation 1: Numeric Features

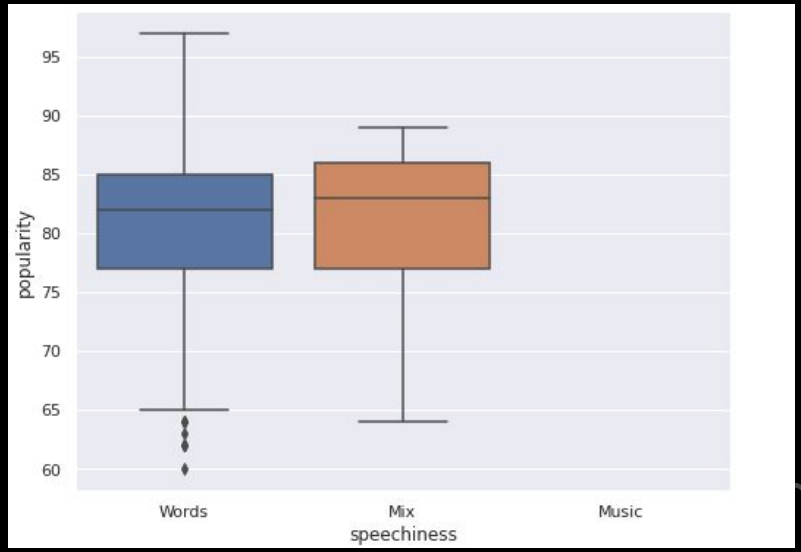
Highest Correlation with Popularity	
Features	**Correlation**
Artist popularity	0.30
Acousticness	0.08
Energy/Tempo	0.07

- Most features **do not have linear relationship** with popularity



Bivariate visualisation 2: Categorical Features





- Songs with **vocals** have **higher popularity** (**higher median**)


Machine Learning

Linear Regression

Prediction of the popularity using a single variable of numerical data

- All the R^2 value are close to 0, suggesting that the prediction is largely wrong
- Mean MSE is about 38

Linear Regression

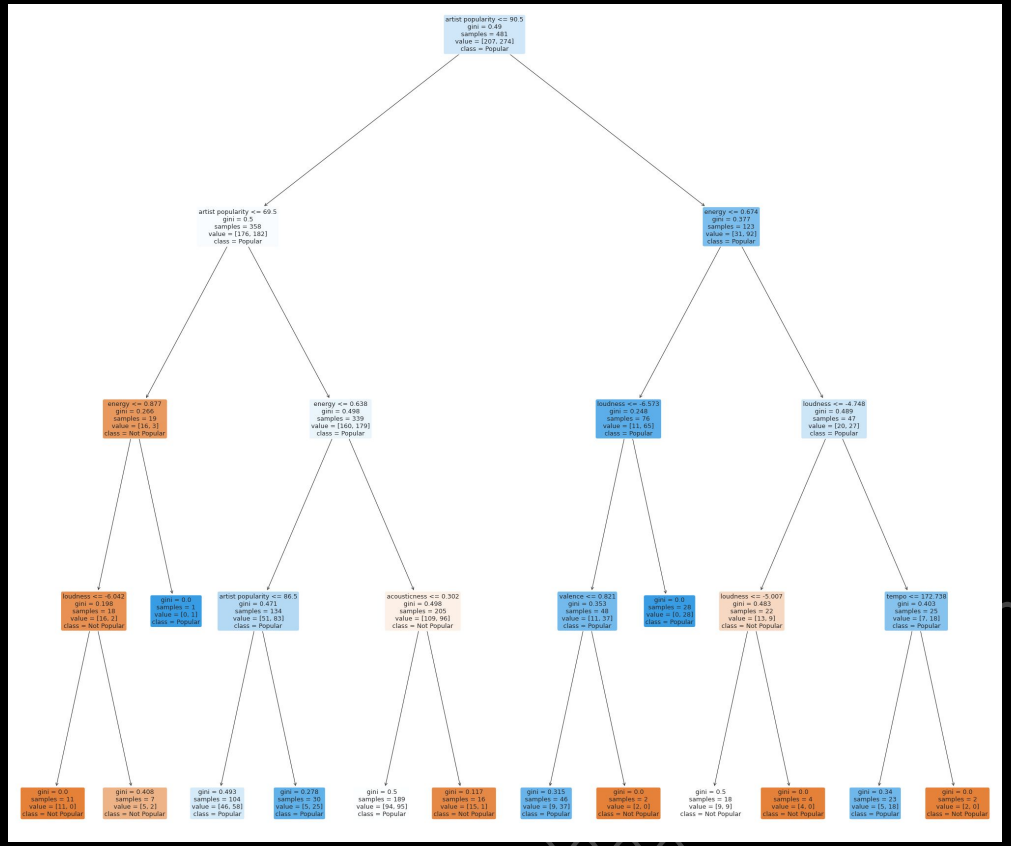
Prediction of the popularity using ALL the variables

Model		Linear Regression	Decision Tree Regression	Random Forest Regression
Train data	R^2	0.099676	0.23019	0.76263
	MSE	34.33655	29.35885	9.05292
Test Data	R^2	0.09764	0.04897	0.11104
	MSE	40.27984	42.45269	39.68204

Decision Tree

Create decision tree using all numerical and categorical data (ohe data)

Popular	=>80
Not Popular	<80



Decision Tree - Accuracy

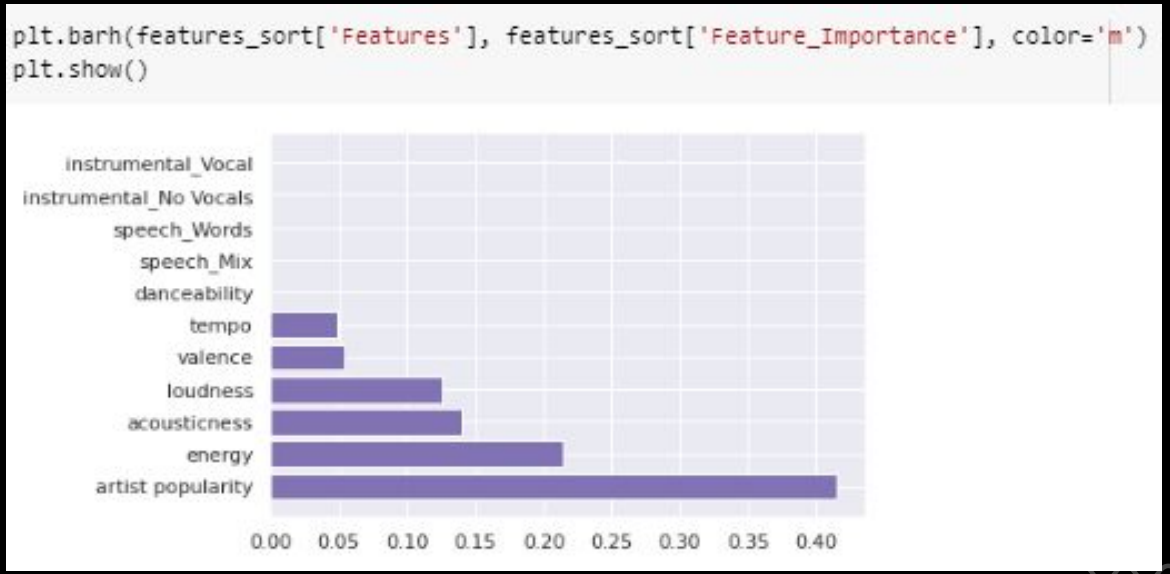
```
Train Data
Accuracy  :       0.6636363636363637

TPR Train :       0.6927899686520376
TNR Train :       0.6233766233766234

FPR Train :       0.37662337662337664
FNR Train :       0.3072100313479624
<matplotlib.axes._subplots.AxesSubplot at 0x7fb84dfd1410>
```


```
Test Data
Accuracy  :       0.4855072463768116

TPR Test :        0.5540540540540541
TNR Test :        0.40625

FPR Test :        0.59375
FNR Test :        0.44594594594594594
<matplotlib.axes._subplots.AxesSubplot at 0x7fb84d7ce350>
```

Decision Tree - Accuracy

- Determining the important features in the prediction

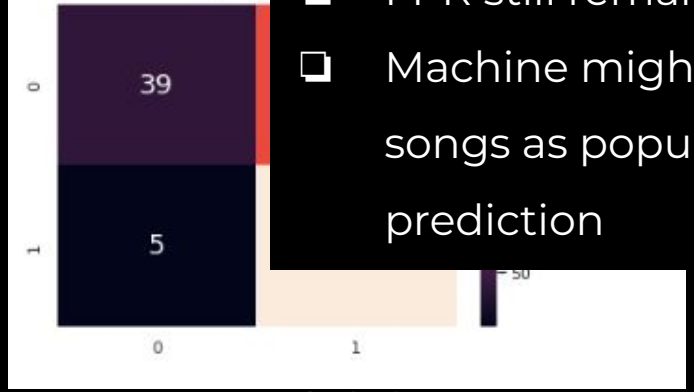
```
plt.barh(features_sort['Features'], features_sort['Feature_Importance'], color='m')
plt.show()
```



Decision Tree - Accuracy

- Using the Top 3 attributes

```
Train Data
Accuracy  :      0.6403326403326404

TPR Train :      0.9817518248175182
TNR Train :      0.188

FPR Train :      0.811
FNR Train :      0.018
<matplotlib.axes._subp
```




```
Test Data
Accuracy  :      0.5942028985507246

TPR Test :       0.8739495798319328
```

- The result did not really improve
- FPR still remains very high
- Machine might just be predicting everything songs as popular, which is not helping in our prediction

Random Forest Classification

- Tuning the hyperparameter using GridSearchCV

```
# Import GridSearch for hyperparameter tuning using Cross-Validation (CV)
from sklearn.model_selection import GridSearchCV

# Define the Hyper-parameter Grid to search on, in case of Random Forest
param_grid = {'n_estimators': np.arange(100,1001,100),   # number of trees 100, 200, ..., 1000
              'max_depth': np.arange(2, 11)}             # depth of trees 2, 3, 4, 5, ..., 10

# Create the Hyper-parameter Grid
hpGrid = GridSearchCV(RandomForestClassifier(),   # the model family
                      param_grid,                 # the search grid
                      cv = 5,                     # 5-fold cross-validation
                      scoring = 'accuracy')       # score to evaluate

# Train the models using Cross-Validation
hpGrid.fit(X_train, y_train.popularity_categorical.ravel())
```

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(),
             param_grid={'max_depth': array([ 2,  3,  4,  5,  6,  7,  8,  9, 10]),
                         'n_estimators': array([ 100,  200,  300,  400,  500,  600,  700,  800,
             scoring='accuracy')
```

```
# Fetch the best Model or the best set of Hyper-parameters
print(hpGrid.best_estimator_)

# Print the score (accuracy) of the best Model after CV
print(np.abs(hpGrid.best_score_))
```

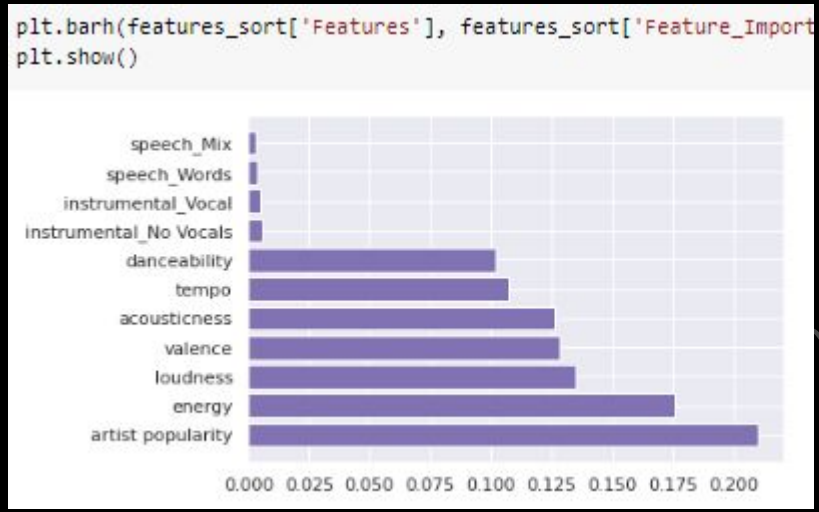
```
RandomForestClassifier(max_depth=7, n_estimators=600)
0.6363636363636365
```

Random Forest Decision Tree

- **Determining the important features in the prediction**

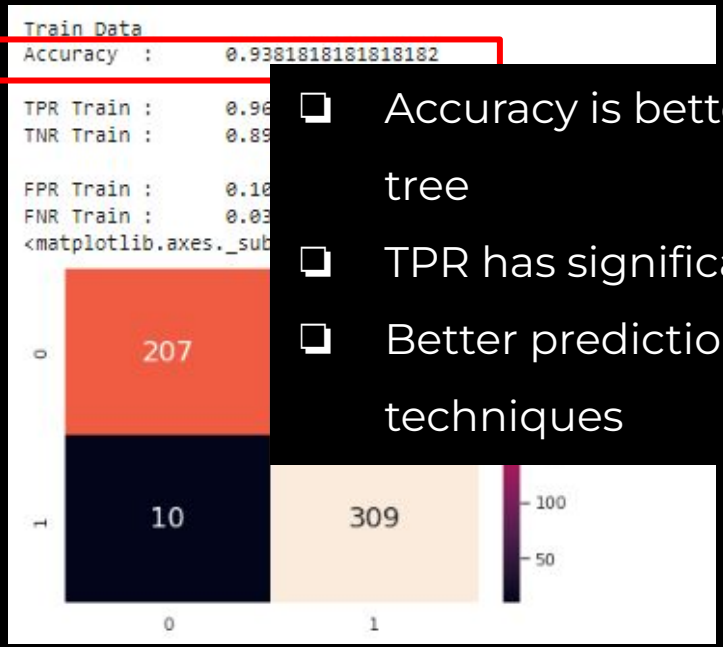
Insights:

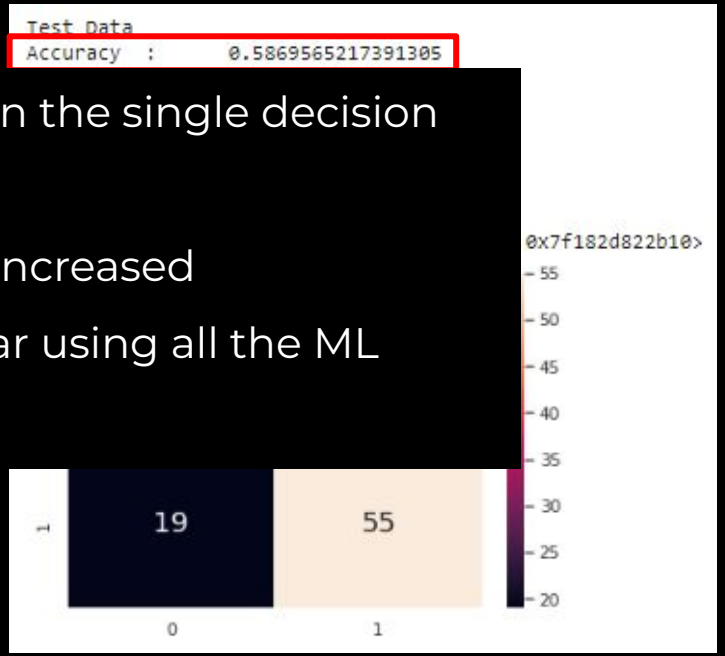
- Only few attributes are prominent in predicting the popularity of the songs
- Top three features are **loudness**, **energy** and **artist popularity**



Random Forest Decision Tree – Accuracy

- Using more random decision trees to predict the popularity





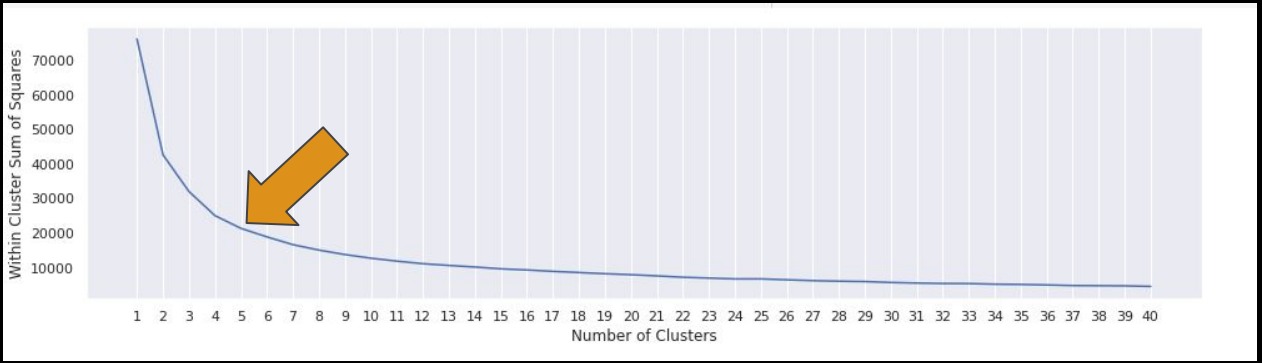
- Accuracy is better than the single decision tree
- TPR has significantly increased
- Better prediction so far using all the ML techniques

KMeans Clustering

Partition data into clusters for analysis

Step 1: Reduce within sum of squares

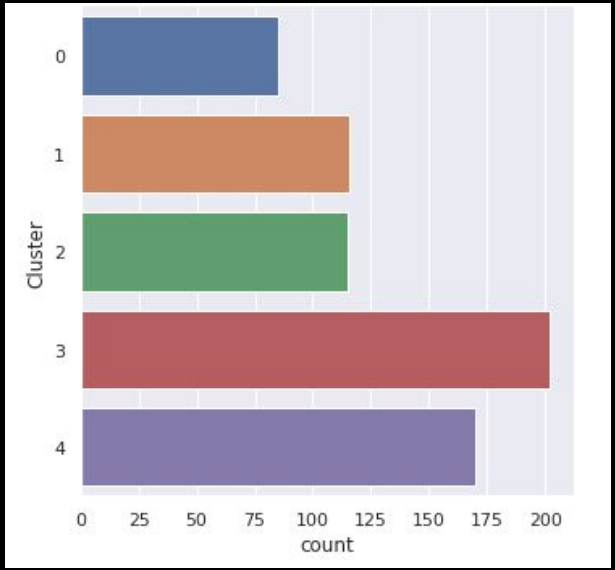
- **Cluster = 5** shows a steep decrease in within sum of squares

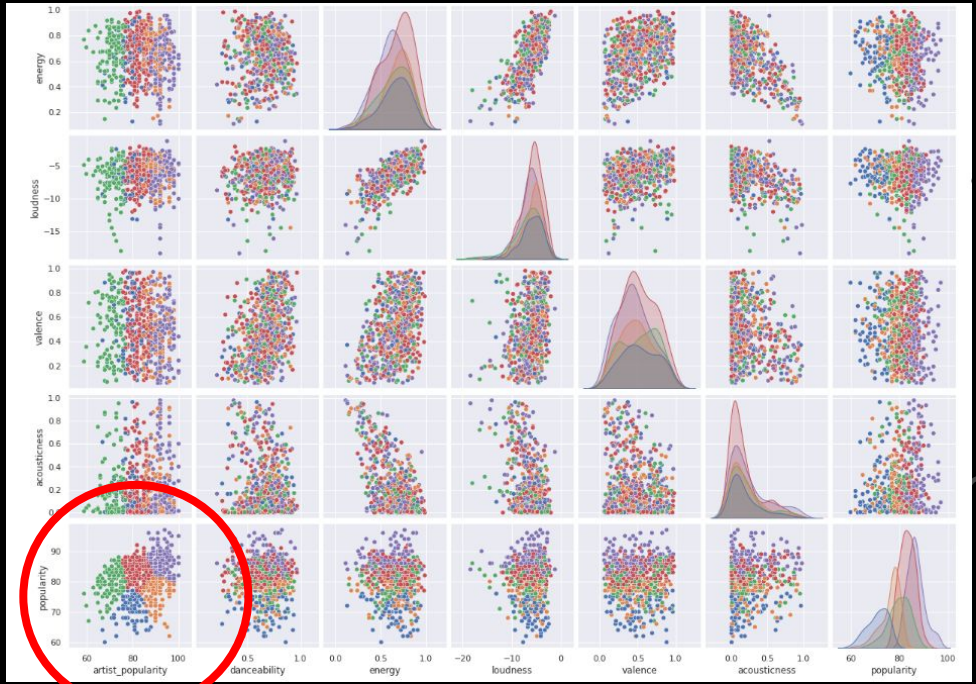


Elbow Plot

KMeans Clustering

Step 2: Visualise clusters

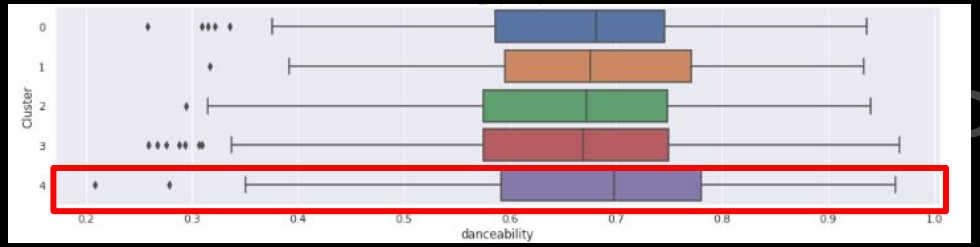


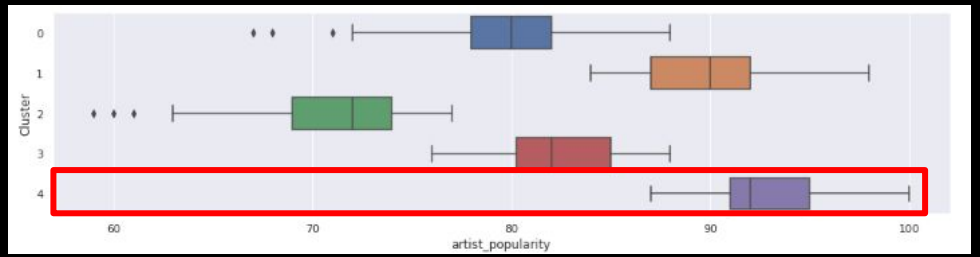


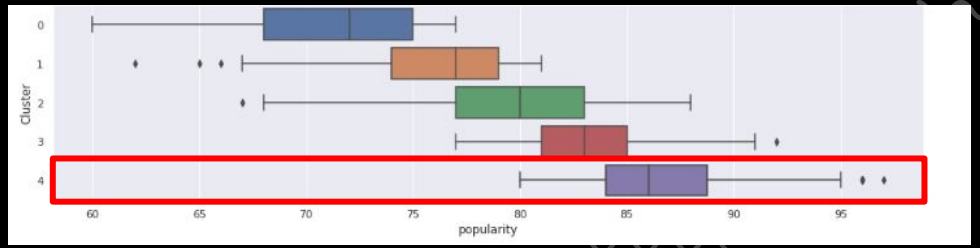
KMeans Clustering

Step 3: Analyse Clusters

Songs with **highest popularity** (purple box plot) tend to have **highest artist popularity** as well as **higher danceability**







Conclusion

Key Insights

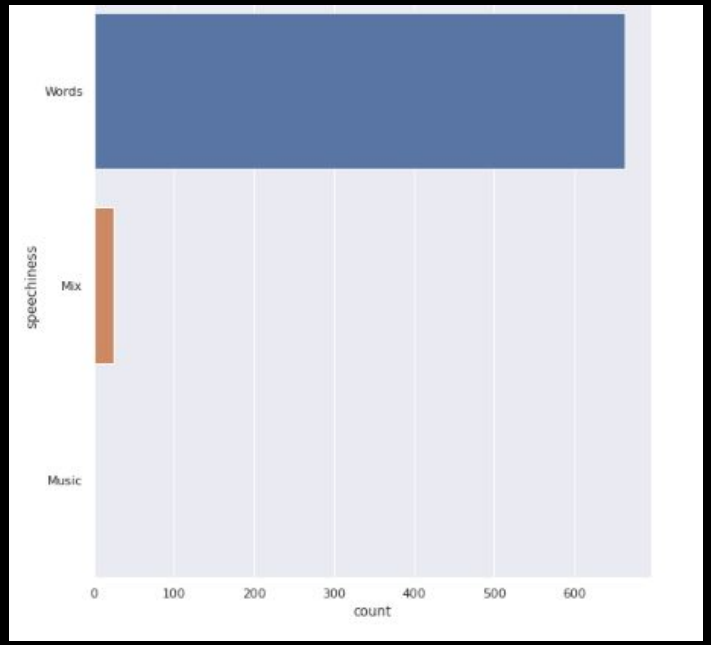
- Decision Trees / Random forest more ideal
- Predict songs at accuracy of 60% (at best)
- Important attributes (artist popularity seems to stand out in both ML models):

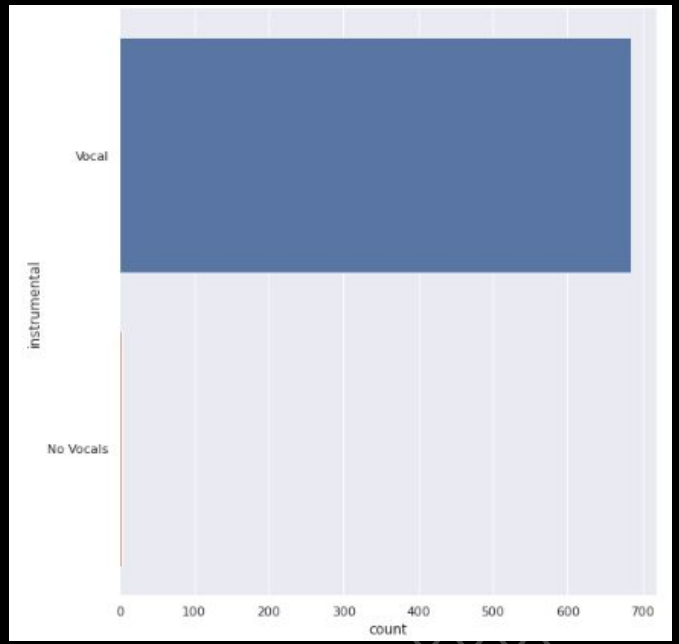
  Random forest: artist popularity (most important), energy

  Clustering : artist popularity, danceability
- Ideally, Energy and danceability of a song can be increased to increase a song's popularity, although it does not guarantee an increase as artist's popularity is still the most important factor

Limitations and Recommendations

Class imbalance of categorical variables





Limitations and Recommendations

Study Other Variables

- Social media following
- Record label
- Artists' network value
- Nostalgia score
- Historical data
- Year of release
- etc.

Break popularity into different subsets

- Region
- Demographics
- Streaming device
- Number of shared Spotify accounts
-

A song's popularity not solely determined by its numerical features

Limitations and Recommendations

- Class imbalance for categorical variables → due to relatively small dataset size/ lack of spread
- Unsure of how spotify's algorithm to derive popularity value
- Song's popularity may not be strictly defined by numeric variables → lyrics or words in a song could determine how catchy thus popular it is, which we could not measure → lack of data on play time, number of playlists songs added to etc.
- Model used to analyse non-numeric data from songs e.g., lyrics, most common words used in most popular songs etc.

Thank You

And continue to enjoy music

Last 2 mins → Finish strong

- What is the OUTCOME of your project? Did it solve your original problem? Anything interesting? → Limitations, expectations

1) Artist popularity influence popularity the most
2) Can also say that other attributes not really important, and spotify may have its own algorithm to calculate the popularity of the song, other than just solely based on the attributes

Conclusion i think can get from this website, https://towardsdatascience.com/predicting-popularity-on-spotify-when-data-needs-culture-more-than-culture-needs-data-2ed3661f75f1

Last 2 mins → Finish strong

- What are your data-driven INSIGHTS and recommendations / views towards the target problem? (cite the data then mention the insight)

  → Talk about other possible ML models to use

  → Popularity numbers based on spotify's algorithm, not sure how it is calculated

  → Talk about how songs still are affected primarily on artist popularity

  → Other attributes that could be more important in predicting song popularity?

  → While some trends can be seen, the very nature of songs is that its popularity may not be able to be solely measured / captured using these numerical variables

Content Page

Next 3 min → set the stage

- You MUST mention how you collected / curated / cleaned / prepared the data for this problem.

  1) Discuss steps taken in jupyter notebook to extract data into csv file

  2) Discuss data cleaning steps:

  → Check that there are no duplicates (all unique track id)

  → Convert categorical variables e.g. speechiness, instrumental, live performance (converted from numeric to categorical)

  → Drop unnecessary columns e.g., artist genres, track id etc.

  → And more ..

- Present your Exploratory Data Analysis and some initial data-driven Insights from the dataset.

1) Discuss the steps of our EDA, and what kind of insight we get, especially the popularity part, when we get the data from the top hit playlist, but got some songs with 0 popularity which is abit

2) Can try take a look at the example video, see how they present their eda

3) Can try to see what kind of interesting data or fact that we can get from our data, can be anything, dont need to restrict yourself

- Did you only use tools and techniques learned in this course? What ELSE did you learn / try?

1) Explain the techniques we used, and also the additional stuff (e.g., radar plot, etc.)

- You MAY also mention how you are planning to set up the Analysis / ML problem for this case.

→ One hot encoding of categorical variables

→ Correlation of variables for linear regression

→ Remove outliers using IQR method as popularity data is left skewed (IQR used for skewed data) → show reduce in skewness of popularity data

Next 3 mins → core analysis

- If you used ML (regression, classification, or something else); mention mainly WHICH one(s).

1) Regression (univariate and multivariate, decision tree regression and random forest regression)
2) Decision tree classification
3) Random forest classification
4) Clustering analysis

Next 3 mins → core analysis

- You may now briefly CLARIFY why and how the ML problem(s) aim(s) to solve your objective.

  1) Regression -> predict popularity value

  2) Decision tree -> predict if a song is popular or not (popular being >80?), most important attributes used in decision tree

  3) Clustering -> look out for attributes defining most popular songs

- How did you apply ML technique(s) to SOLVE your problem? Which model(s), how and why?

- Did you only use tools and techniques learned in this course? What ELSE did you learn / try?

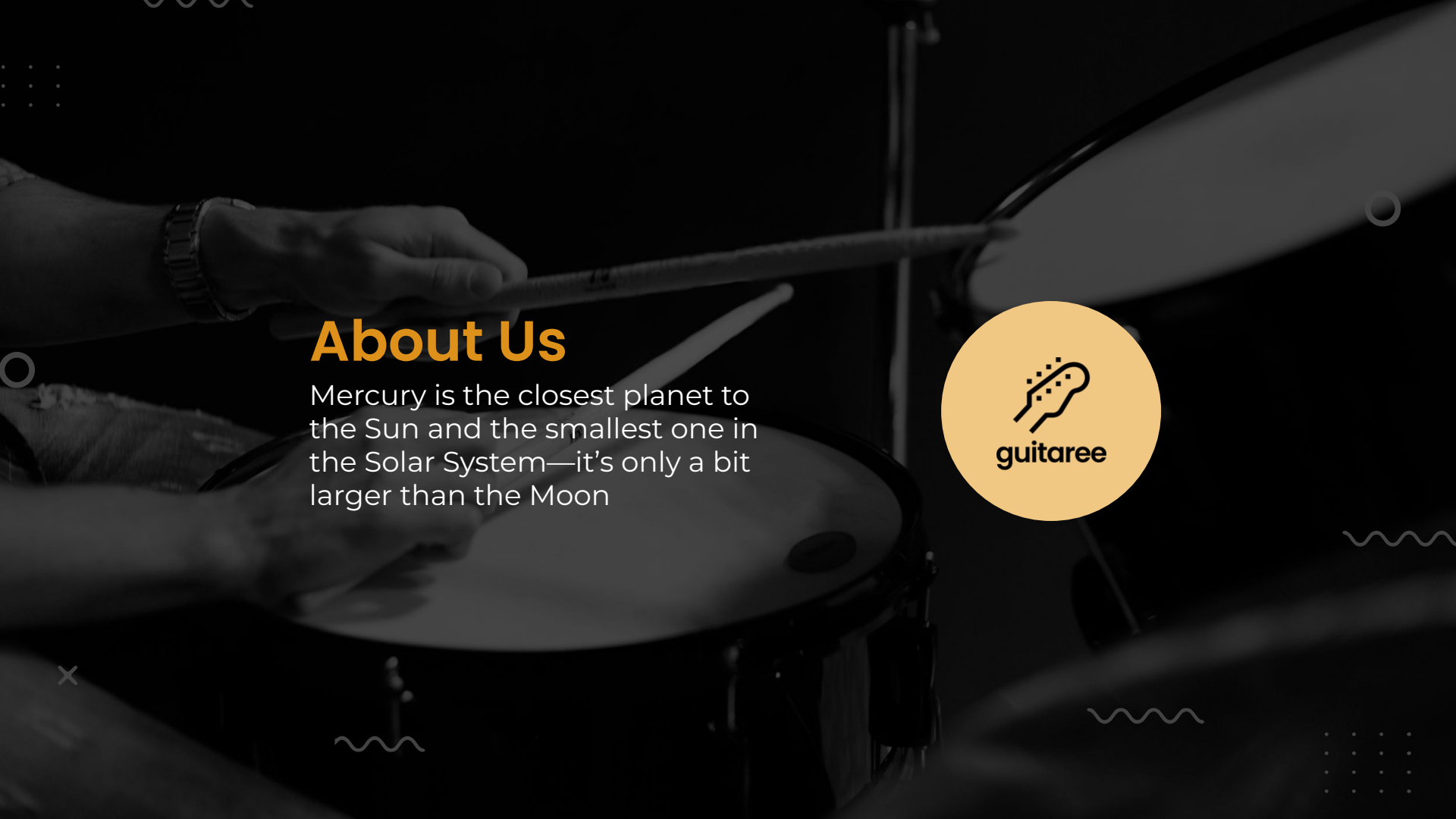
Agenda

01
About Our Company
Here you could describe the topic of the section

02
Our Services/Products
Here you could describe the topic of the section

03
Our Clients
Here you could describe the topic of the section

04
Expected Projection
Here you could describe the topic of the section



About Us

Mercury is the closest planet to the Sun and the smallest one in the Solar System—it's only a bit larger than the Moon

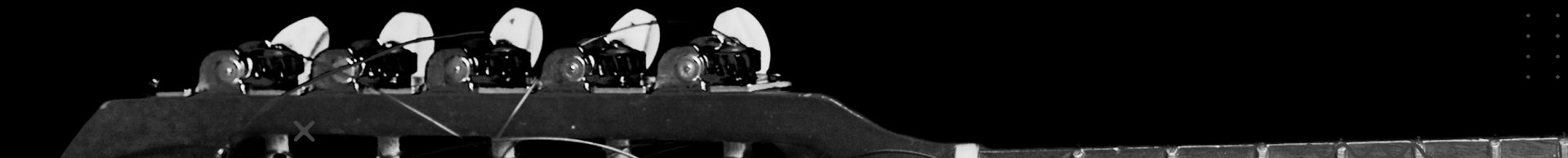
guitaree

Our History

2005
Jupiter is the biggest planet in the Solar System

2008
Venus has a beautiful name, but it’s terribly hot

2010
Despite being red, Mars is actually a cold place

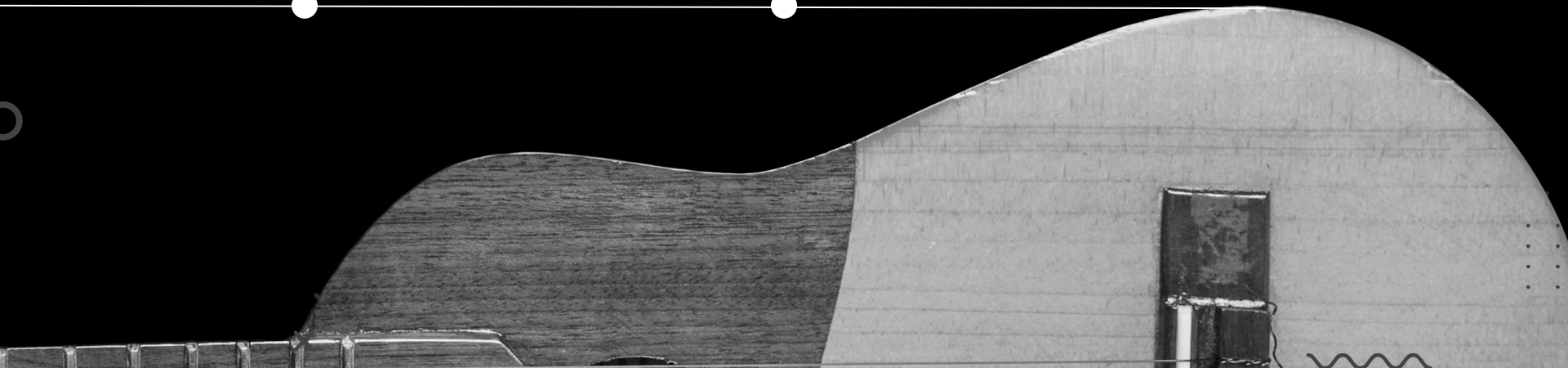


Our History

2012
Mercury is the closest planet to the Sun

2018
Neptune is the farthest planet from the Sun

And still counting!



Our Philosophy

Mission

Despite being red, Mars is a cold place. It’s full of iron oxide dust

Vision

Jupiter is a gas giant and the biggest planet in the Solar System

Values

Mercury is the closest planet to the Sun and the smallest one

“This is a quote. Words full of wisdom that someone important said”

—Someone Famous

Sometimes, Reviewing Concepts Is a Good Idea

Mercury

Mercury is the closest planet to the Sun

Venus

Venus has a beautiful name, but is hot

Mars

Despite being red, Mars is a cold place

Jupiter

It’s the biggest planet in our Solar System

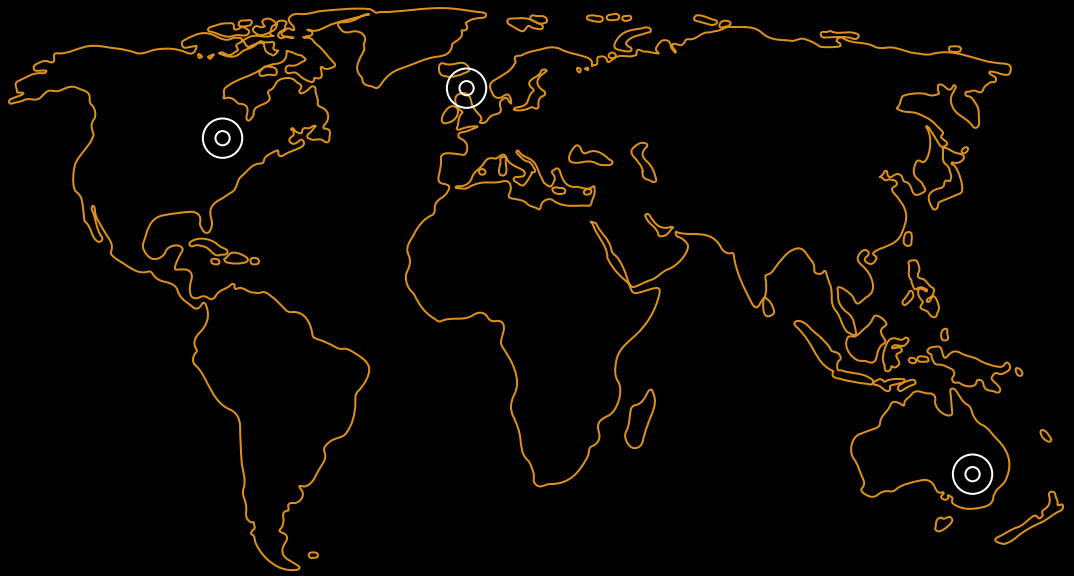
Saturn

Saturn is the ringed one and a gas giant

Neptune

Neptune is the farthest planet from the Sun

Locations



Find out where all our offices around the world are located

This Is a Table

	Mass	Diameter	Gravity
Mercury	0.06	0.38	0.38
Mars	0.11	0.53	0.38
Saturn	95.2	9.4	1.16

Future Projects

Project 02
Venus has the most beautiful name of all

Project 01
Mercury is the closest planet to the Sun

03 **Project**
Despite being red, Mars is a cold place

04 **Project**
Jupiter is a gas giant and the biggest

Our Services

Venus

Venus has a beautiful name

Mercury

Mercury is the smallest planet

Best Sellers

Venus has a beautiful name, but it’s terribly hot

Saturn is the ringed one and a gas giant

Despite being red, Mars is actually a cold place

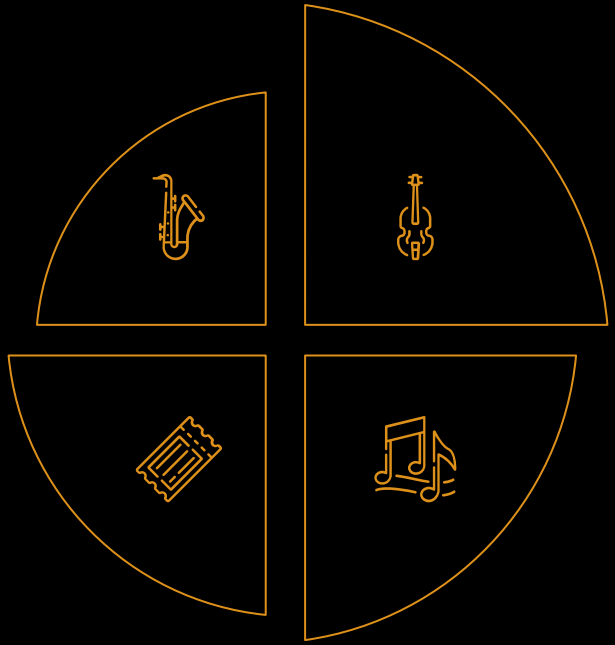
Our Strengths

Loyalty

Neptune is the farthest planet from the Sun

Reliability

Mercury is the smallest closest planet to the Sun

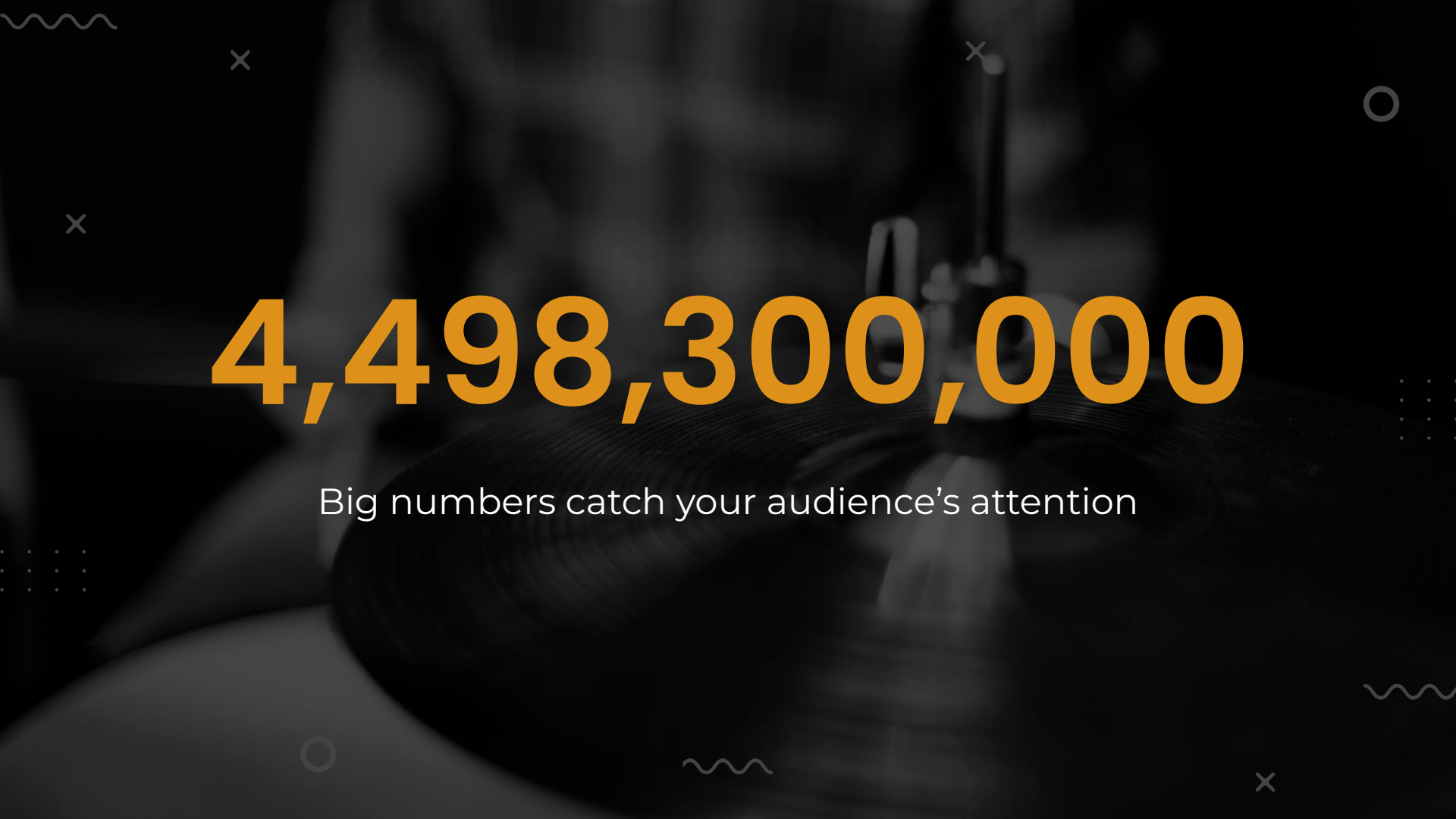


Efficiency

Despite being red, Mars is actually a very cold place

Commitment

Jupiter is the biggest planet in the Solar System


4,498,300,000
Big numbers catch your audience’s attention

Target

Gender

M 50%

W 80%

Music Genres

Alternative

Classical

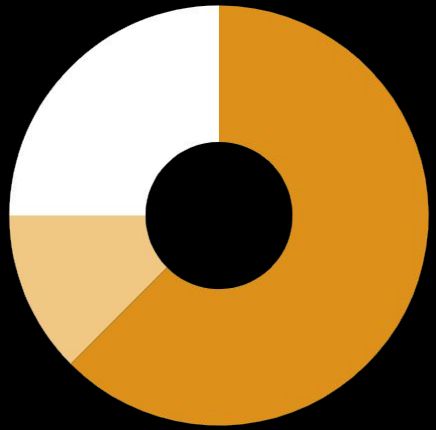
Dance

Interests

Sneak Peek

You can replace the image on the screen with your own work. Just delete this one, add yours and center it properly

Our Numbers



- Venus 25%
- Mars 12.5%
- Mercury 62.5%

To modify this graph, click on it, follow the link, change the data and replace it

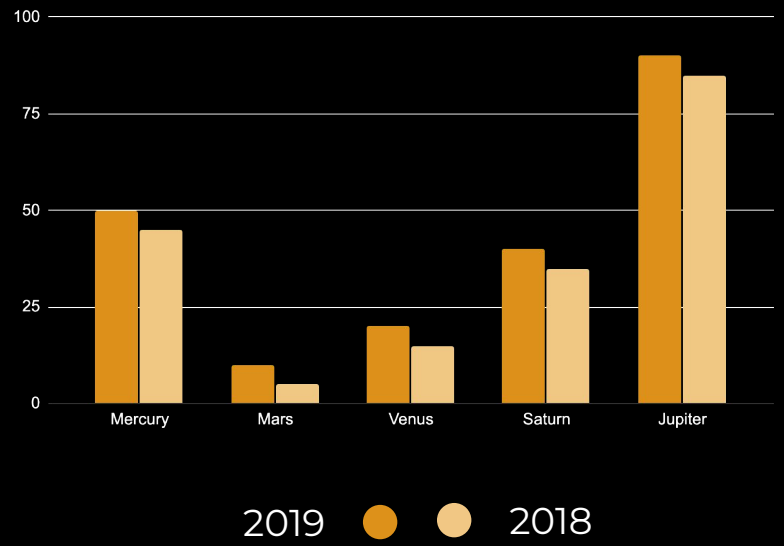
$28.87
average price per unit

30 days
avg. days on market

45 shops
in different countries

77 units
remaining inventory

Our Growth



To modify this graph, click on it, follow the link, change the data and replace it

$20,000
Expected income for 2020

100
New employees next year

Customer Testimonials

Helena James

“Mercury is the closest planet to the Sun and the smallest one”

Jenna Doe

“Venus has a beautiful name and is the second planet”

John Patterson

“Despite being red, Mars is actually a very cold place”

Awards

Mercury

Mercury is the closest planet to the Sun and the smallest one

Venus

Venus has a really beautiful name and is the second planet

Our Team

John James

You can replace the image on the screen with your own

Jenna Doe

You can replace the image on the screen with your own

Jane Patterson

You can replace the image on the screen with your own