27th July 2023

Statistical Learning Project

Group 03:

- Angelo Mandara (2077139)
- Tito Tamburini (1837335)
- Antonio Rocca (1813055)
- Claudiu Gheorghiu (1845227)

Music Time Period Classification

1. Introduction

We believe a fundamental part of everyone’s life is music, also we believe that nostalgia is a very strong feeling, so the goal of the study is to create a machine learning model that can correctly classify songs by decade in order to improve the listener experience. Specifically, we want to help the music industry to create playlists or recommendations systems, to provide a way to track the cultural and social influences that changed the music over time and to find recurring features to predict the next big music movement.

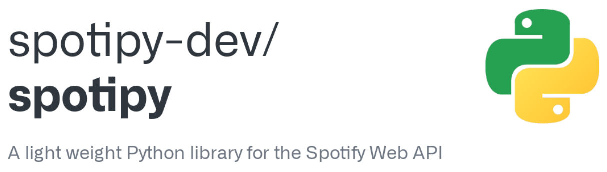
We’ll use a dataset built with the Spotify API via the Spotify Python library: Spotipy.

After creating and cleaning the dataset, we will use multiple classification models to train and evaluate predictions’ performance. We are going to demonstrate the potential of machine learning in analyzing music data and offer insights into how popular music changed over time.

---

2. The dataset

To collect the data we implemented a Spotipy script module in Python. Spotipy grants access to all the Spotify songs. We wrote a scalable script that created a relatively big dataset (9425 songs) for the six decades 60s, 70s, 80s, 90s, 00s, 10s.


spotipy-dev/
spotipy

A light weight Python library for the Spotify Web API

In order to gain access to Spotify via the Spotipy API, the first thing we need to do is to create an account Spotify for Developers to get the credentials for the Authentication. After retrieving a CLIENT_ID and an SECRET_CLIENT_ID, we can connect to Spotify via the Spotipy library in Python, by creating the Spotify python object:

```
spotify = spotipy.Spotify(
    client_credentials_manager=SpotifyClientCredentials(
        client_id=CLIENT_ID,
        client_secret=SECRET_CLIENT_ID)
    )
```

After obtaining the connection with our credentials to the Spotify API we can retrieve the playlists that groups the song by each decade:

```
decade = '2010s'
playlists =search_playlists_by_decade(decade,spotify)
```

The following code returns a list of playlists which have in the title the decade.

The next step is to retrieve all the songs in each found playlist:

```
tracks_to_insert = get_all_playlists_tracks(playlists,decade,spotify)
```

The code above returns a list of Spotify tracks that are identified by the tuple = (Spotify_Uri, track_name, decade), from these tracks just retrieved we need to extract the audio features.

To extract the audio features from each retrieved tracks, we used this part of the code, which uses the Spotipy function: *audio_features()*, which extracts from a list of Spotify tracks their audio features:

```
tracks_to_insert = audio_features_all_tracks(tracks_to_insert,spotify)
```

Now the object `tracks_to_insert` contains a list of dictionaries that represents the rows that we will use in our dataset.

The final step is to insert the new rows inside the dataset:

```
spotify_songs = pd.read_csv('spotify_songs.csv')

spotify_songs = pd.concat([spotify_songs,tracks_to_insert])
```

After that we can write `spotify_songs` pandas dataframe in a .csv file, and repeat the process for each decade we want to include in our analysis.

It follows the description of a row in our dataset:

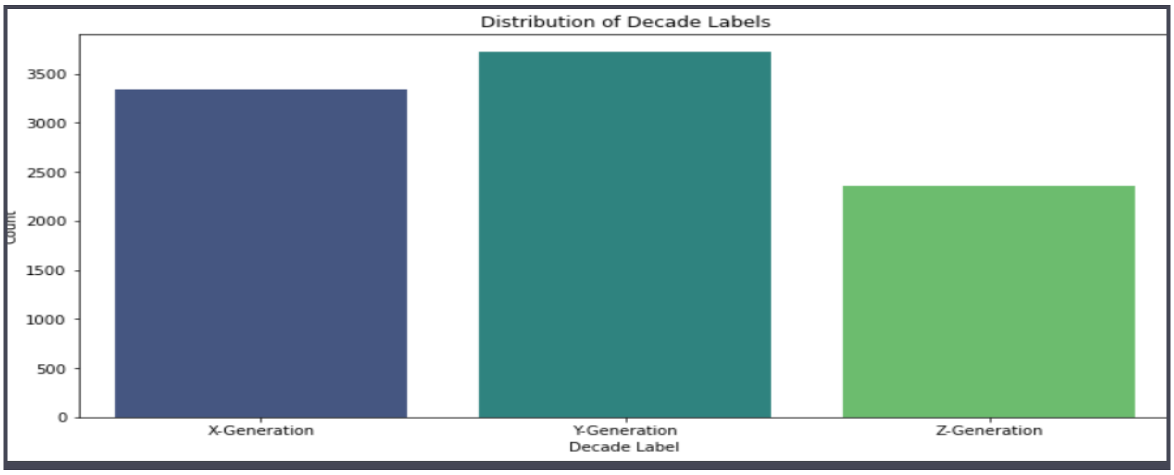
- ***track_uri***: unique identifier for each track or song in our dataset. It is a string type, typically used to reference and identify individual songs;
- ***track_name***: name or title of each track. It is also a string type and provides human-readable information about the song;
- ***decade***: target variable that we want to predict. It indicates the decade in which each song belongs. It is a categorical variable, representing discrete values corresponding to different decades (e.g., '60s', '70s', '80s', etc.);
- ***danceability***: measures how suitable a song is for dancing based on musical elements like rhythm, beat, and tempo. It ranges from 0 to 1, where higher values indicate higher danceability;
- ***energy***: measure of intensity and activity in a song. It represents the presence and intensity of dynamic elements such as loudness and percussiveness. Similar to danceability, it ranges from 0 to 1, with higher values indicating higher energy;
- ***key***: key or tonality of the song. It is a categorical variable with values representing different musical keys (e.g., C major, D minor, etc.);
- ***loudness***: overall volume of a song in decibels. It is a continuous numerical feature, with higher values indicating louder songs;
- ***mode***: modality of the song, indicating whether it is in a major (1) or minor (0) key;
- ***speechness***: presence of spoken words in the song. It ranges from 0 to 1, with higher values indicating more spoken words and lower values representing more instrumental music;
- ***acousticness***: proportion of acoustic sounds in the song. It ranges from 0 to 1, with higher values indicating a more acoustic or unplugged sound;
- ***instrumentalness***: likelihood that a song is instrumental, meaning it has no vocals. It also ranges from 0 to 1, with higher values indicating a higher probability of being instrumental;
- ***liveness***: probability that a song was recorded in a live concert setting or in a studio;
- ***valence***: Valence represents the musical positiveness conveyed by a song. It ranges from 0 to 1, with higher values indicating more positive and happy-sounding music;
- ***tempo***: speed or pace of a song in beats per minute (BPM). It is a continuous numerical feature that provides information about the song's rhythm and speed.

Since working on the data the models had big troubles in terms of accuracy, since near decades share more or less similar features, we passed from 6 decades to three groups of twenty years: X-Generation (60s plus 70s), Y-Generation (80s plus 90s) and Z-Generation (00s plus 10s).

3. Data Analysis

In this section, we'll look at how decade labels are distributed, show how specific features are distributed by decade, and examine how the features evolve over time. We'll also define "boringness" score and explain how the Kruskal-Wallis H test may be used to evaluate how numerical features relate to the label "decade."

Counterplot that represents the frequency of songs from different decades in our dataset.



In the box plots we are comparing the X-generation (60s-70s) to the Z-generation (80s-90s) and the Z-generation (00s-10s) for the features ‘danceability’, ‘energy’, ‘loudness’, ‘acousticness’, ‘speechiness’, and ‘valence’, we can observe distinct differences.

- Danceability:
    - Z Generation (2000-2020): Songs are more danceable, likely due to the prevalence of dance-oriented genres.
    - X and Y Generations (60s-70s-80s-90s): Songs may be less danceable, offering a broader range of genres.

- Energy:
    - Z Generation (2000-2020): Similar energy levels as the Y generation, known for dynamic and high-energy music.
    - X Generation (60s-70s): Possibly lower energy levels compared to Z and Y generations, featuring diverse genres.

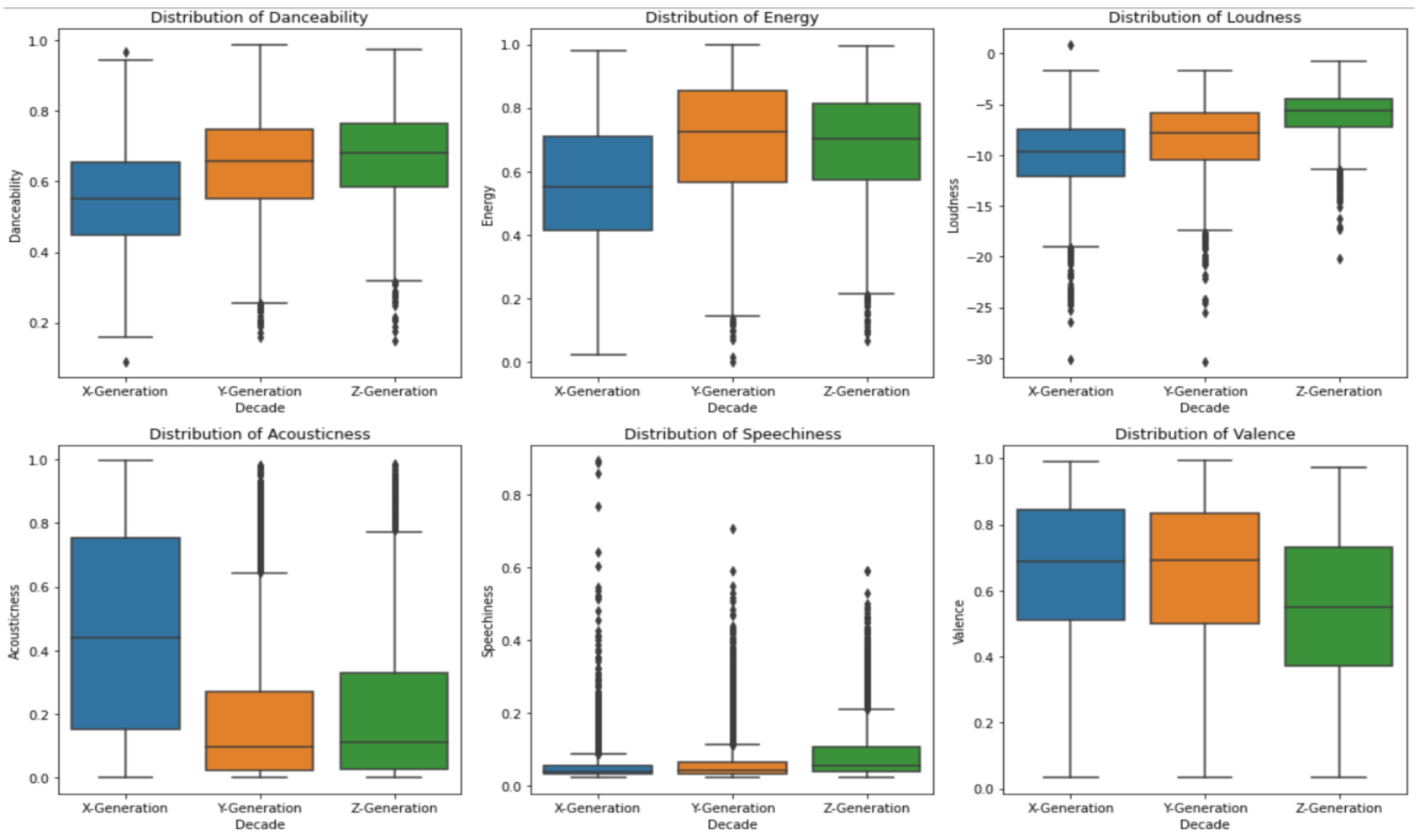
- Loudness:

    - Z Generation (2000-2020): Louder songs, reflecting modern production styles.
    - X and Y Generations (60s-70s-80s-90s): Songs with lower loudness, aligned with the technology and production of those eras.

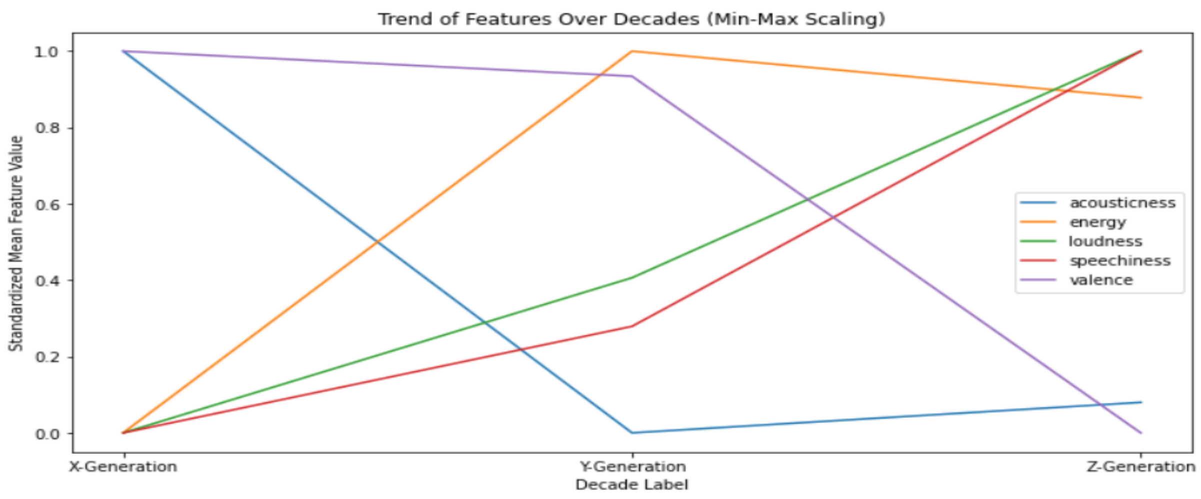
- Acousticness:
    - X Generation (60s-70s): Higher acousticness, as these decades emphasized acoustic instruments.
    - Z Generation (2000-2020): Lower acousticness, due to increased use of electronic and synthesized sounds.

- Speechiness:
    - Z Generation (2000-2020): Higher speechiness, possibly due to the rise of rap and spoken-word genres.
    - X and Y Generations (60s-70s-80s-90s): Lower speechiness, reflecting diverse vocal styles in various genres.

- Valence:
    - X Generation (60s-70s): Higher valence, with music often conveying positivity and happiness.
    - Z Generation (2000-2020): Lower valence, reflecting a broader emotional range in modern music.



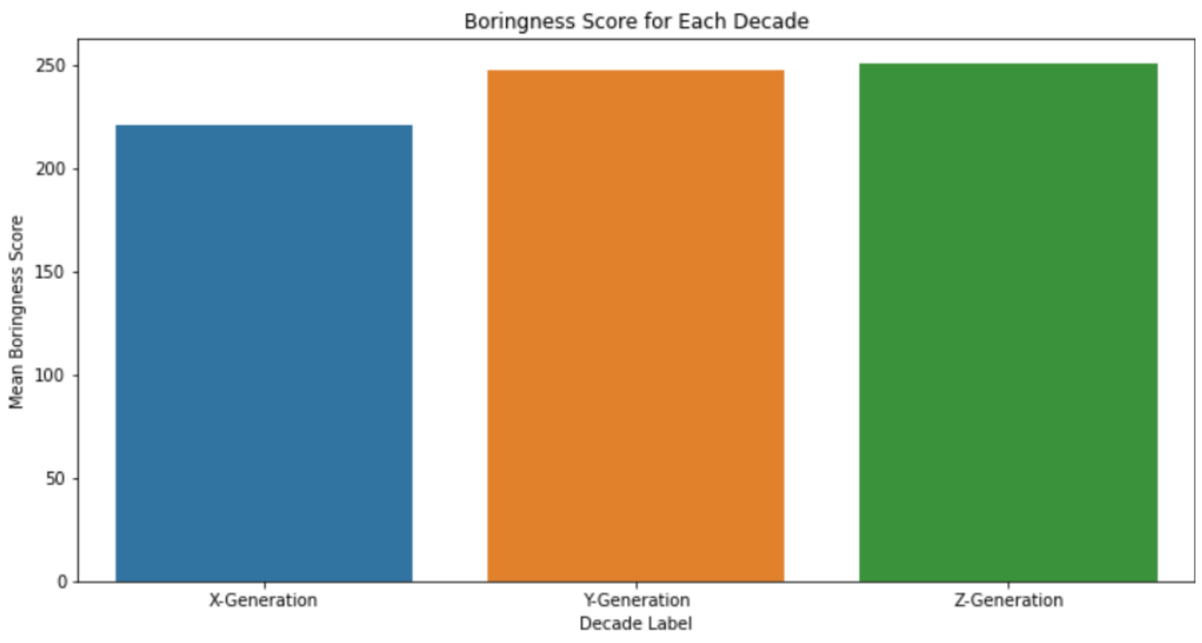
To analyze how the distribution of a specific feature changes over time, we created line plots where each line represents the trend of the feature over the years or decades.



The trend we can see reflects the analysis of the boxplots we did before.

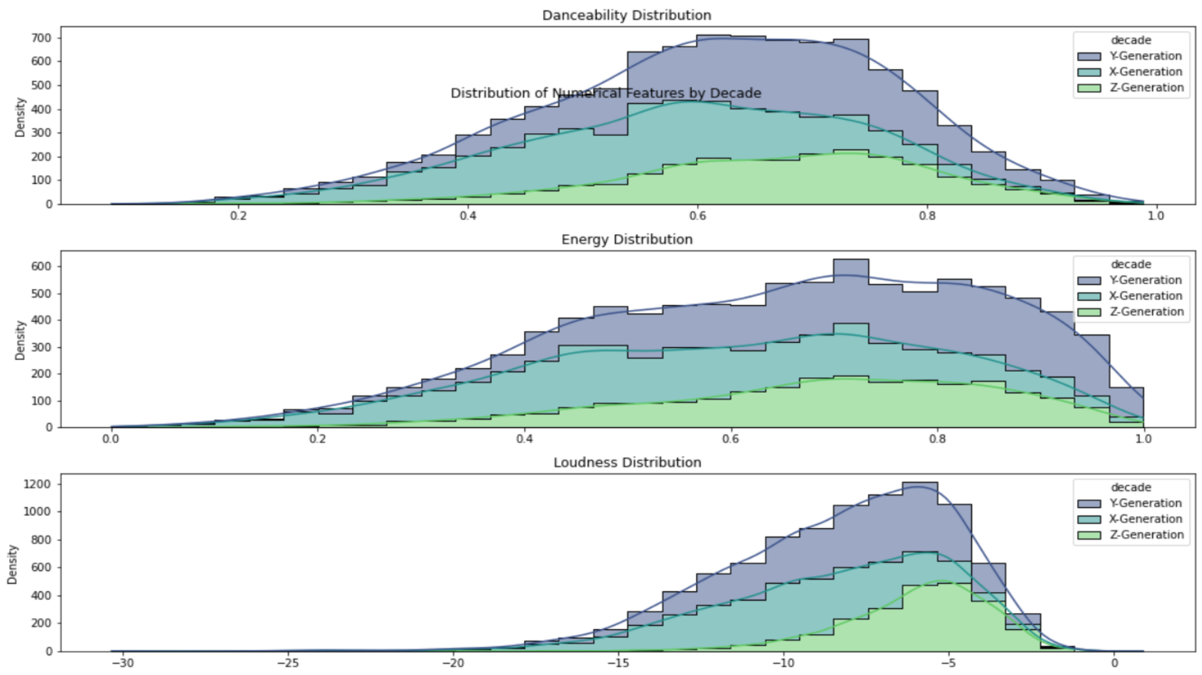
We have devised a simple equation to calculate the "boringness" score of a song, which helps assess how "boring" a period is. The equation takes into account the features "loudness," "tempo," "energy," and "danceability."

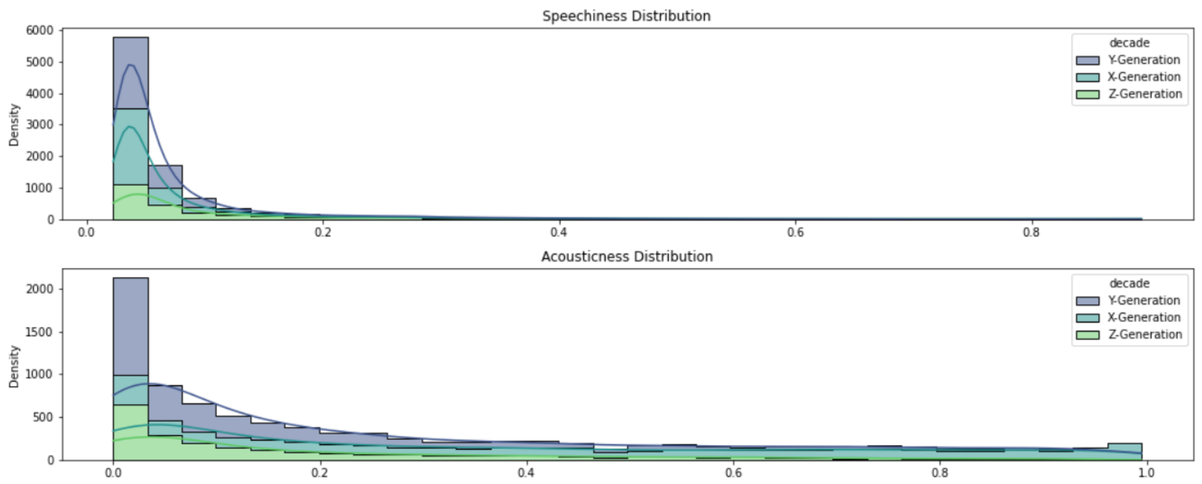
$$boringness = loudness + tempo + (energy \cdot 100) + (\text{danceability} \cdot 100)$$



The "boringness" score is a simple numerical representation of how exciting a song is based on its features. Lower scores suggest more vibrant songs, ideal for a lively party. X Generation songs have lower scores, Y Generation higher than X, and Z Generation even higher, indicating a shift in musical preferences towards less energetic music over time.

To plot the distribution of each numerical variable within each 'decade' group, we can create a set of histograms or density plots for each feature. This will allow us to visualize how the values of each feature are distributed across different 'decade' categories.



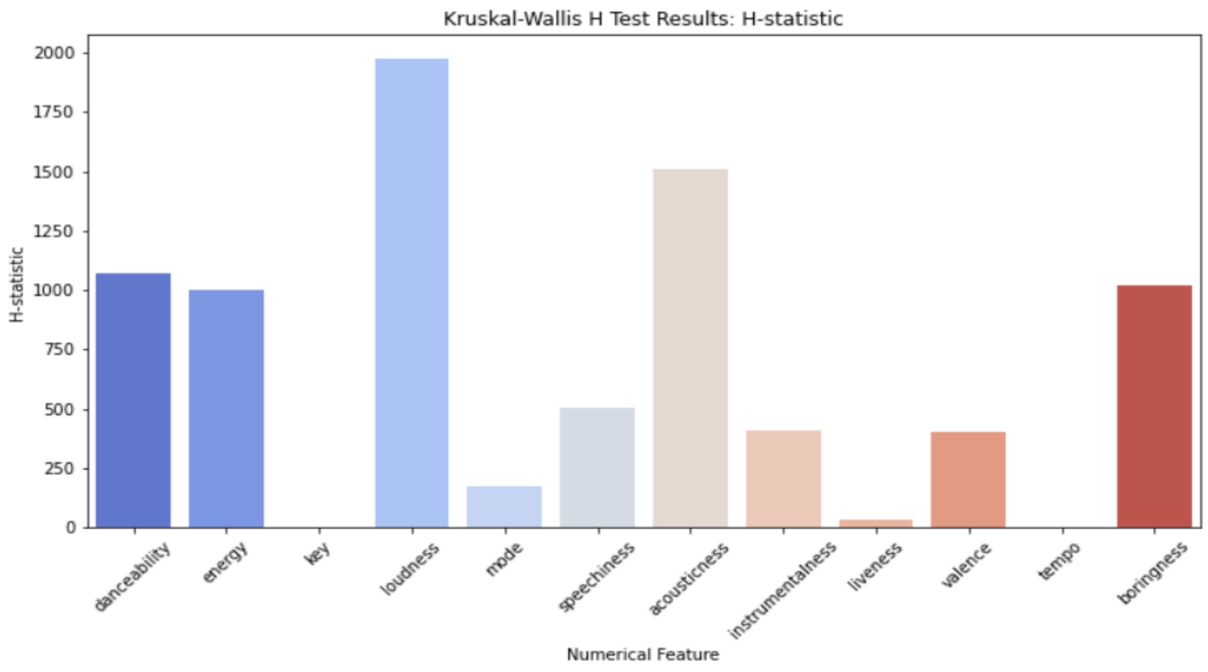


The density plot shows the distribution of each numerical feature within each 'decade' group. When the density is higher, it means that there are more data points (elements) in that particular 'decade' category for that specific feature. This happens when there is an imbalance in the number of data points across the 'decade' groups, the shape is very similar.

The Kruskal-Wallis H test can be used to determine how each numerical feature and the term "decade" are related to one another. This non-parametric test will enable us to ascertain whether there are any notable discrepancies in the characteristics' means across the various 'decade' groupings.

Each numerical feature will be subjected to the Kruskal-Wallis H test to determine whether the feature distribution varies noticeably between various decades.

With the use of these analyses and visualizations, we can learn more about the distribution and evolution of features through time and how they connect to our dataset's 'decade' label.



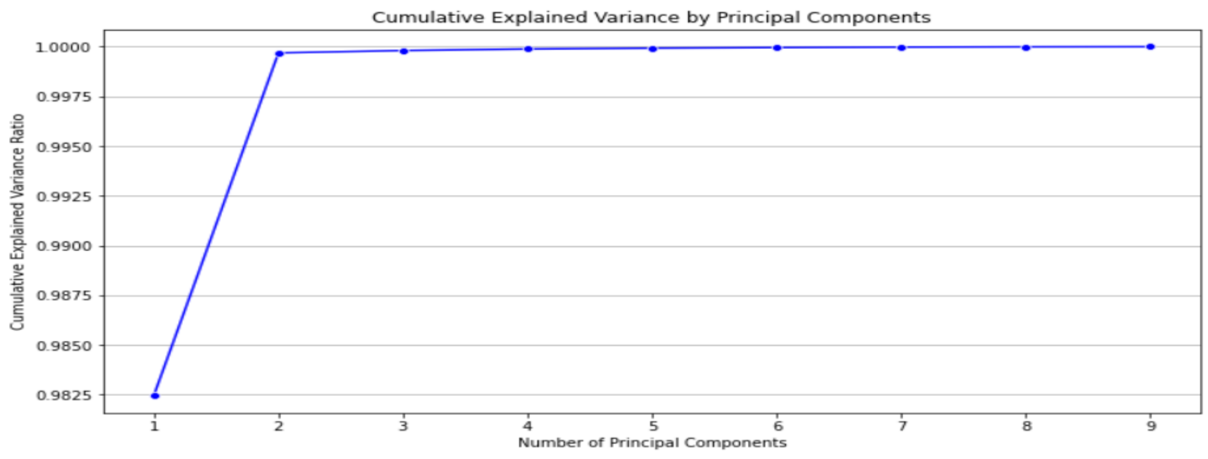
To interpret the H-statistic:

- Magnitude of Difference : The H-statistic is a numerical value that quantifies the overall difference in ranks among the groups. A larger H-statistic means that the numerical feature exhibits more variability or differences across the 'decade' groups.
- Significance: The H-statistic is meaningful in the context of the p-value obtained from the Kruskal-Wallis test. If the p-value is small (typically less than the chosen significance level, often 0.05), then the large H-statistic is likely to be significant. It means there is strong evidence to reject the null hypothesis and conclude that there are significant differences in the distributions of the numerical feature among the 'decade' groups.

4. Model & Methods

4.1 PCA

In our case, we applied PCA to the dataset with the hope of reducing the dimensionality of the features and potentially discovering underlying patterns related to different decades. However, in our case this technique was not helpful.

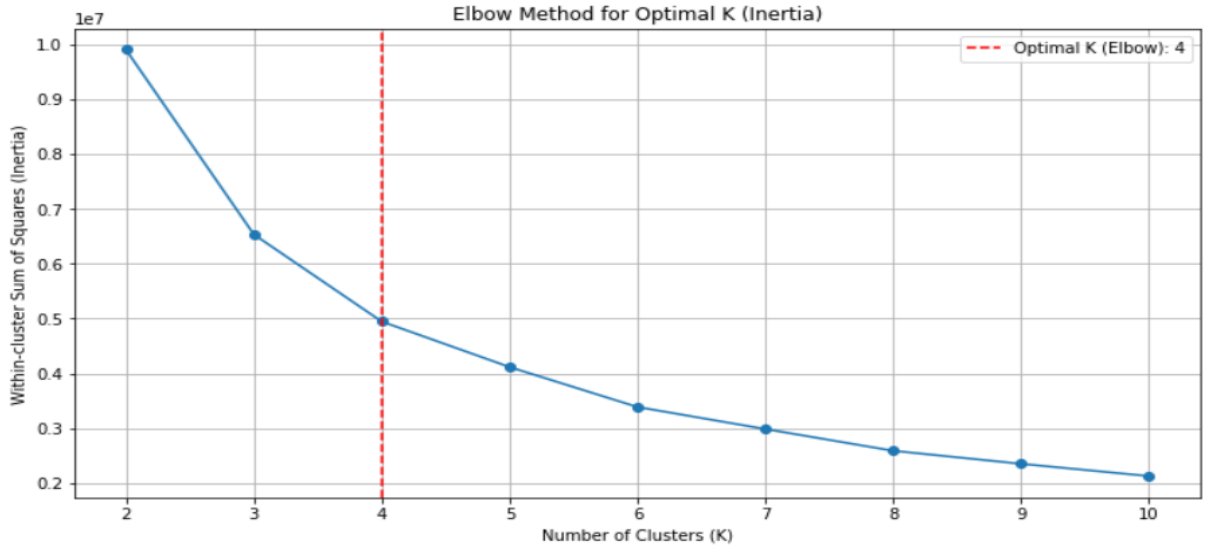


The relationship between the features and the target variable (decade) might be complex and not well-represented by linear transformations. PCA assumes linear relationships between features, and if the dataset exhibits nonlinear interactions, PCA might not effectively uncover the underlying patterns.

4.2 KMEANS++

After attempting unsupervised clustering with KMeans++, we discovered that the elbow method's recommended number of clusters was 4, and not 3 as we wanted.

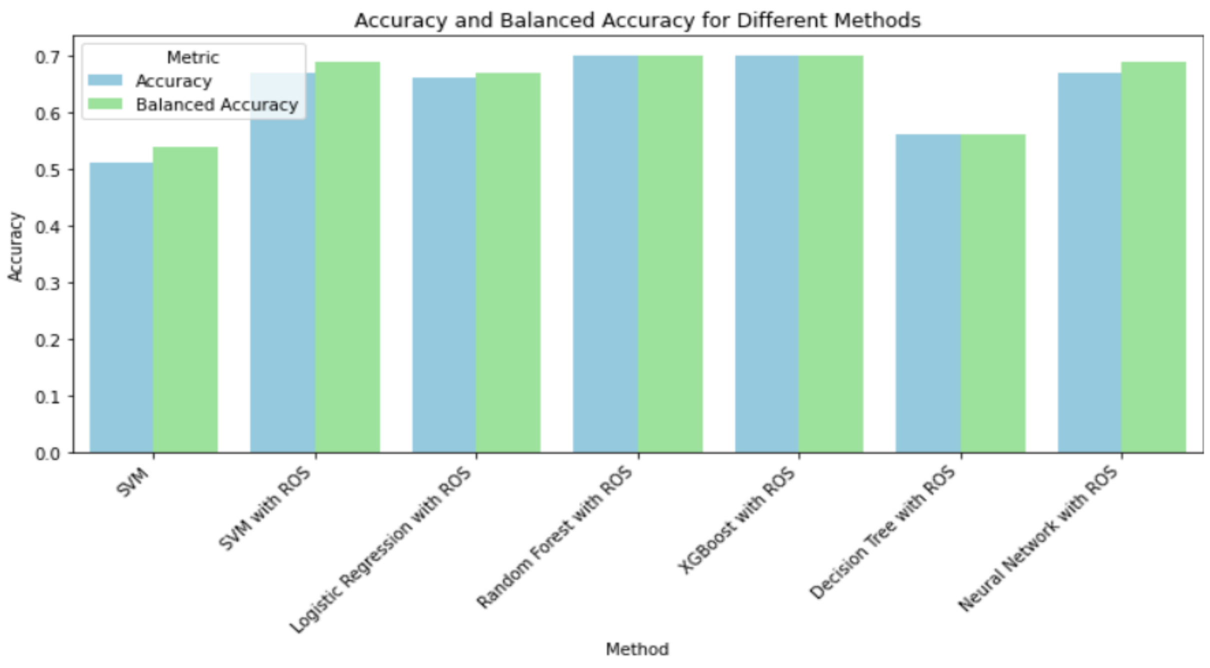
The outcomes were less encouraging when accuracy was used to assess the clustering performance. The clustering accuracy was found to be 0.46, indicating that the KMeans++ clusters did not closely match the three decades we were looking to identify (X-generation,Y and Z).



The data may not show clearly defined clusters based on the chosen features, as suggested by the relatively poor accuracy. It's likely that the features we selected aren't sufficiently different to properly divide the three decades. The data's inherent complexity or the overlapping feature distributions across decades, among other things, could have led to the inadequate grouping outcomes.

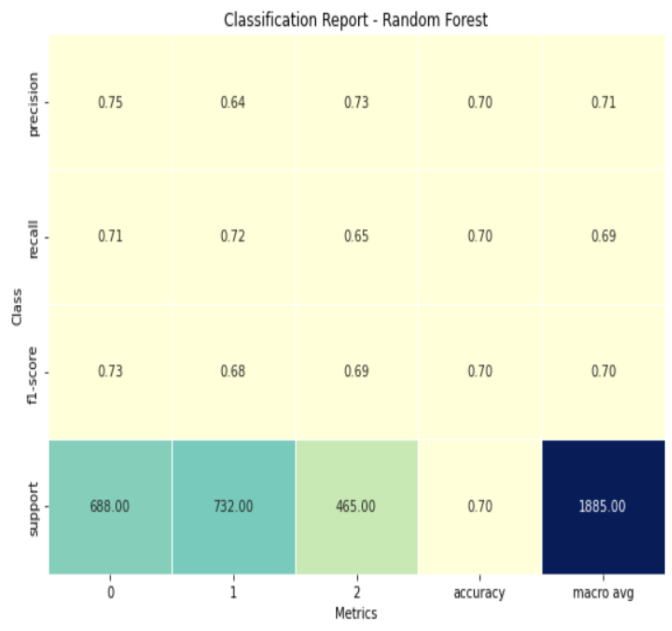
4.3 MODELS

In our analysis, we tackled the class imbalance issue by using Random Over-Sampling (ROS) with SMOTE to balance the dataset(was not so unbalanced, but the Z class has less elements). We then tried different classification methods, evaluating their performance using both accuracy and balanced accuracy metrics. This approach helped us choose the best model, which turned out to be the **Random Forest** classifier. Despite some other methods having similar accuracy values, Random Forest demonstrated superior overall performance and robustness.



We obtained the optimum hyperparameters by performing a Randomised Search on the Random Forest parameters to fine-tune the model. We will now train the Random Forest classifier on the resampled training data using these optimized parameters, and we will assess its performance on the test set.

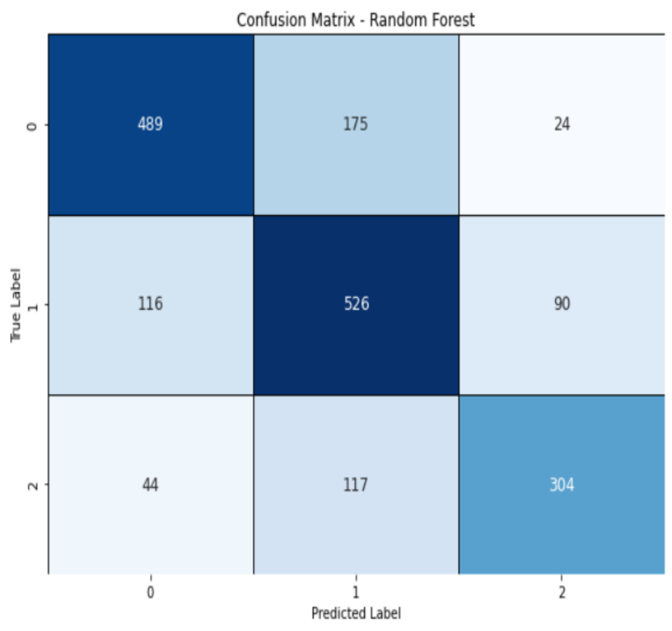
The number of accurate and inaccurate predictions for each class will be shown in the confusion matrix, while the precision, recall, and F1-score values for each class will be given in the classification report. We may evaluate the Random Forest model's performance using these criteria across generations.



Classification Report - Random Forest

Class	0	1	2	accuracy	macro avg
precision	0.75	0.64	0.73	0.70	0.71
recall	0.71	0.72	0.65	0.70	0.69
f1-score	0.73	0.68	0.69	0.70	0.70
support	688.00	732.00	465.00	0.70	1885.00

Metrics



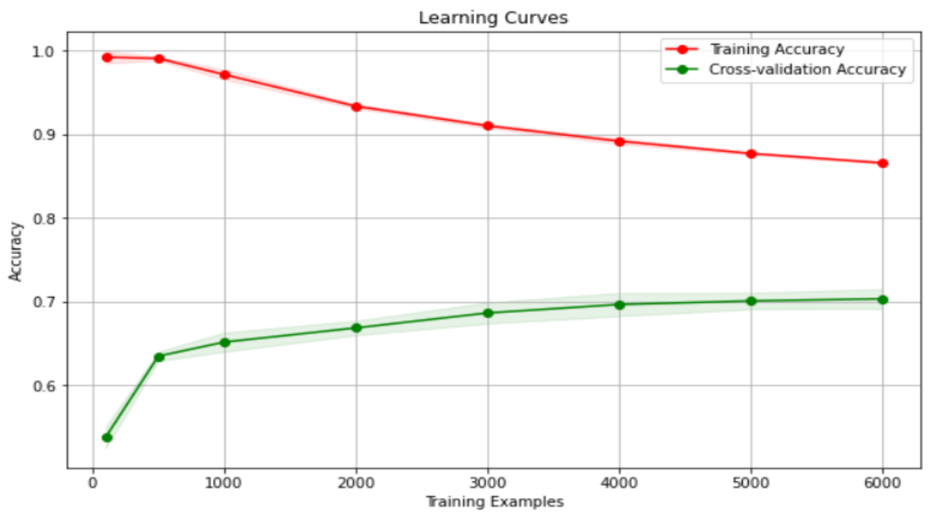
Confusion Matrix - Random Forest

True Label \ Predicted Label	0	1	2
0	489	175	24
1	116	526	90
2	44	117	304

4.4 LEARNING CURVES

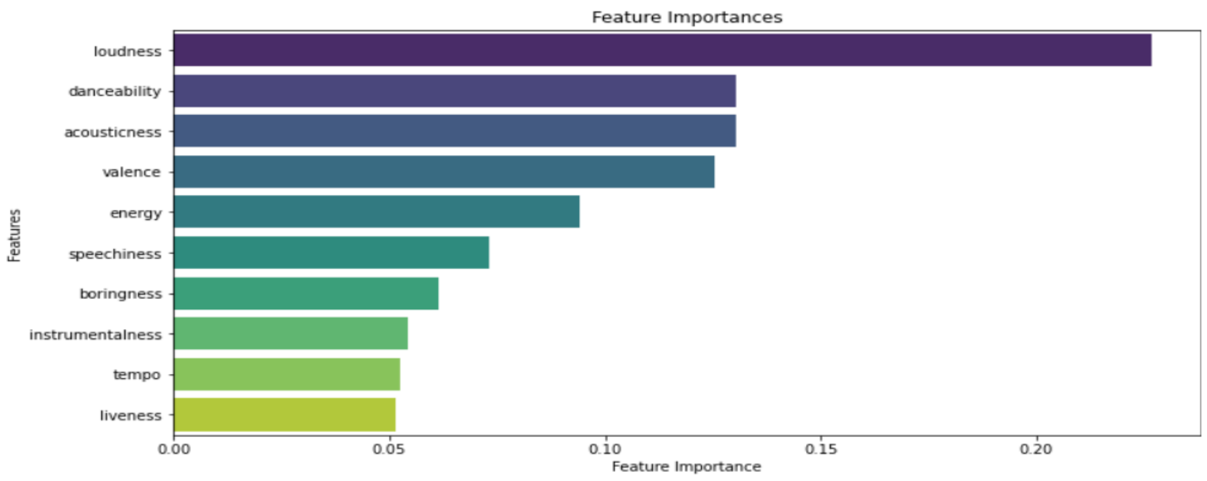
An important technique for assessing a machine learning model's performance is its learning curve. They offer insight on how the performance of the model varies as the volume of training data rises. We may determine whether a model is overfitting or underfitting by graphing the accuracies of the training and cross-validation (or test) sets against the size of the training set.

Our Random Forest classifier initially appeared to be overfitted because the training accuracy was continuously 1 while the cross-validation accuracy was below par. This suggested that the model was having trouble generalizing to new data and was having trouble remembering the training data. To solve this problem, we carefully adjusted the hyperparameters **max_depth**, **min_samples_split**, and **min_samples_leaf**. We noticed a considerable decrease in overfitting as a result, and the learning curve illustrates how the model performed better on unobserved data.



4.5. INTERPRETABILITY

We examined the feature importances obtained from the Random Forest classifier, which ranks the features based on their significance in predicting the decade labels. The bar plot reveals the following ranking of features from the most to the least.



By understanding the order of feature importances, we can gain valuable insights into the distinct musical characteristics that define each decade and use this knowledge to better interpret our model's predictions.

---

5. IML papers of reference:

1. https://cs229.stanford.edu/proj2013/Bartok-Final-F.pdf: gives an idea about which models would be most effective for this kind of task and how to handle the noise from the data. This paper is about a song time period classification project done for the Computer Science course in Stanford University.
2. https://towardsdatascience.com/spotify-api-audio-features-5d8bcbd780b2: clearly explains how to use and implement the Spotify Python library, Spotipy, for a data science project.
3. https://www.ee.columbia.edu/\\\~dpwe/pubs/ismir05-svm.pdf: focuses on the type of distance and evaluation measurements used for the project.
4. https://www.researchgate.net/publication/362948512_Classification_of_Music_Genres_using_Feature_Selection_and_Hyperparameter_Tuning: is about a different task, but gives

an important help to understand relevant features for the study and how to tune hyperparameters.