Music Recommendations

Project Objective

The objective of this project is to develop a music recommendation system that predicts the perceived appreciation of music based on Twitter analyses and user feedback. This will involve leveraging contextual data from the nowplaying-RS dataset, which contains 11.6 million music listening events from 139K users and 346K tracks collected from Twitter. Additionally, sentiment values associated with hashtags will be analyzed to understand the emotional context.

Data Sources

nowplaying-RS Dataset

Contains rich context and content features of listening events, including timestamps, user location, tweet language, and sentiment analysis of hashtags. It provides insights into user preferences and cultural origins based on listening behavior.

Spotify Tracks Dataset

Includes audio features and genre information for Spotify tracks, enabling genre-based analysis and recommendation system development.

Project Scope

Data Exploration and Cleaning

- Understand the structure of the nowplaying-RS dataset.
- Address missing values, outliers, and inconsistencies in the data.
- Analyze sentiment values associated with hashtags to extract emotional context.

Data Analysis and Visualization

- Perform exploratory data analysis (EDA) to uncover patterns and relationships within the datasets.
- Visualize data using graphs and charts to gain insights into user preferences and music trends.

Feature Engineering and Modeling

- Extract relevant features from the datasets for building predictive models.
- Develop machine learning models to predict music appreciation based on user context and content features.

Final Report

- Compile final project report integrating previous renderings, including conclusions and future directions.
- Clean and comment code on GitHub, refine model and modeling pipeline if time permits.

Data Exploration and Cleaning

In the first step of this project to gain a comprehensive understanding of the datasets, we will examine the following tables to describe their structures.

The **“sentiment_values.csv”** dataset provides sentiment analysis results for hashtags encountered in the listening events. It includes sentiment scores from various lexicons (Vader, AFINN, Opinion Lexicon, SentiStrength) such as minimum, maximum, sum, and average scores. The hashtag column links sentiment scores to specific hashtags associated with the listening events:

Name of the Column	**Variable's Type**	**Description**	**Categorical / Quantitative**
hashtag	Float	Hashtag associated with the sentiment information	Quantitative
vader_min	Float	Minimum sentiment score using Vader lexicon	Quantitative
vader_max	Float	Maximum sentiment score using Vader lexicon	Quantitative
vader_sum	Float	Sum of sentiment scores using Vader lexicon	Quantitative
vader_avg	Float	Average sentiment score using Vader lexicon	Quantitative
afinn_min	Float	Minimum sentiment score using AFINN lexicon	Quantitative

afinn_max	Float	Maximum sentiment score using AFINN lexicon	Quantitative
afinn_sum	Float	Sum of sentiment scores using AFINN lexicon	Quantitative
afinn_avg	Float	Average sentiment score using AFINN lexicon	Quantitative
ol_min	Float	Minimum sentiment score using Opinion Lexicon	Quantitative
ol_max	Float	Maximum sentiment score using Opinion Lexicon	Quantitative
ol_sum	Float	Sum of sentiment scores using Opinion Lexicon	Quantitative
ol_avg	Float	Average sentiment score using Opinion Lexicon	Quantitative
ss_min	Float	Minimum sentiment score using SentiStrength Lexicon	Quantitative
ss_max	Float	Maximum sentiment score using SentiStrength Lexicon	Quantitative
ss_sum	Float	Sum of sentiment scores using SentiStrength Lexicon	Quantitative
ss_avg	Float	Average sentiment score using SentiStrength Lexicon	Quantitative

Table 1. variables of 'sentiment_values' dataset

The **"user_track_hashtag_timestamp.csv"** dataset captures associations between users, tracks, hashtags, and timestamps of listening events. Key columns include user_id, track_id, hashtag, and created_at. This dataset enables us to explore user engagement and interactions with music content through hashtags on social media:

Name of the Column	Variable's Type	Description	Categorical / Quantitative
user_id	Object	Identifier for the user	Unique Values
track_id	Object	Identifier for the track	Unique Values

hashtag	Object	Hashtag associated with the listening event	Categorical
created_at	Object	Timestamp of the listening event	N/A

Table 2. variables of 'user_track_hashtag' dataset

The **"context_content_features.csv"** dataset contains contextual and content features related to music listening events collected from Twitter. It includes information such as coordinates, instrumentalness, liveness, speechiness, danceability, valence, loudness, tempo, acousticness, energy, mode, key, artist_id, place, geo, tweet_lang, track_id, created_at, lang, and time_zone. These variables encompass a range of audio features and contextual information associated with each listening event:

Name of the Column	**Variable's Type**	**Description**	**Categorical / Quantitative**
coordinates	Object	N/A	Unique Values
instrumentalness	Float	Indicating the absence of vocals in a track	Quantitative
liveness	Float	Indicating the presence of an audience in a track	Quantitative
speechiness	Float	Indicating the presence of spoken words in a track	Quantitative
danceability	Float	Describing the suitability of a track for dancing	Quantitative
valence	Float	Indicating the musical positiveness	Quantitative
loudness	Float	The overall loudness of a track in decibel	Quantitative
tempo	Float	Beat per minute representation of the overall estimated tempo of a track	Quantitative
acousticness	Float	Describing whether the track is acoustic or not	Quantitative

energy	Float	Describing the intensity level of a track	Quantitative
mode	Float	Indicating the main key of a track with 1.0 and 0.0	Categorical
key	Float	Representing the group of notes by using standard Pitch class notation	Categorical
artist_id	Object	Identifier for the artist	Unique Values
place	Object	N/A	
geo	Object	N/A	
tweet_lang	Object	Indicating in which language the tweet was written.	Categorical
track_id	Object	Identifier for the track	Unique Values
created_at	Object	N/A	
lang	Object	Hashtag Language (?)	
time_zone	Object	representing the time zone of the area where the listening event occurred.	
user_id	Float	Identifier for the user	Unique Values
id	Int	N/A	Unique Values

Table 3. variables of 'context_content_features' dataset

Data Cleaning

We begin by cleaning the dataset **"sentiment_values.csv"** using the following steps:

Renaming misplaced columns: As can be seen in Fig. 1, the hashtag column is misplaced and it should be in the first column, then we have 4 columns that are not explicitly specified. We rename them to the sentiment scores. This renaming step enhances the interpretability of the dataset, providing clearer and more meaningful column names that align with the specific sentiment analysis metrics used.

Dropping Unnecessary Columns: Furthermore, the columns 'vader_min', 'vader_max', 'vader_sum', 'afinn_min', 'afinn_max', 'afinn_sum', 'ol_min', 'ol_max', 'ol_sum', 'ss_min', 'ss_max', and 'ss_sum' should be dropped because they do not provide valuable information

for the analysis and modeling process. These columns represent minimum, maximum, and sum values for sentiment scores ('vader', 'afinn', 'ol', and 'ss'), but since their values are already available in separate columns ('vader_score', 'afinn_score', 'ol_score', and 'ss_score'), including them would introduce redundancy and increase the complexity of the dataset without adding meaningful insights. Therefore, removing these redundant columns helps streamline the dataset and focus on the relevant sentiment score metrics for further analysis.

Addressing missing values: In the subsequent step, we address missing values within the sentiment scores (vader_score, afinn_score, ol_score, ss_score) by imputing them with average values where available. This approach ensures that our data remains representative and complete. The remaining unnecessary score columns (vader_avg, afinn_avg, ol_avg, ss_avg) are then dropped, as they do not offer additional valuable information for our analysis. Additionally, any rows containing missing values are dropped entirely to maintain data integrity and consistency.

Correlation matrix: Upon visualizing the data using a correlation matrix (Fig. 2), we observed strong correlations among certain sentiment scores ('vader_score', 'afinn_score', 'ss_score'), with correlation coefficients exceeding 80%. This high correlation indicated redundancy in the information captured by these scores. To address this, we made the decision to drop these three highly correlated scores, retaining only the 'ol_score', which we then renamed to 'sentiment_score' for clarity and focus.

				hashtag	vader_min	vader_max	vader_sum	vader_avg	afinn_min	afinn_max	afinn_sum	afinn_avg	ol_min	ol_max	ol_sum	ol_avg	ss_min	ss_max	ss_sum	ss_avg
relaxtime	0.8	0.8	2.4	0.8	NaN	NaN	NaN	NaN	0.7375	0.7375	0.7375	0.7375	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
melovechilicheese	0.8	0.8	0.8	0.8	NaN	NaN	NaN	NaN	0.9000	0.9000	0.9000	0.9000	1.0	1.0	1.0	1.0	0.8	0.8	0.8	0.8
greatmusic	0.8	0.8	2.4	0.8	1.0	1.0	1.0	1.0	0.8875	0.8875	0.8875	0.8875	1.0	1.0	1.0	1.0	0.8	0.8	0.8	0.8
rockballad	0.7	0.7	0.7	0.7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
amonamarth	0.3	0.3	0.3	0.3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	0.0	0.0	0.0	NaN	NaN	NaN	NaN

Figure 1. First 5 rows of 'sentiment_values.' dataset showing the misplaced columns

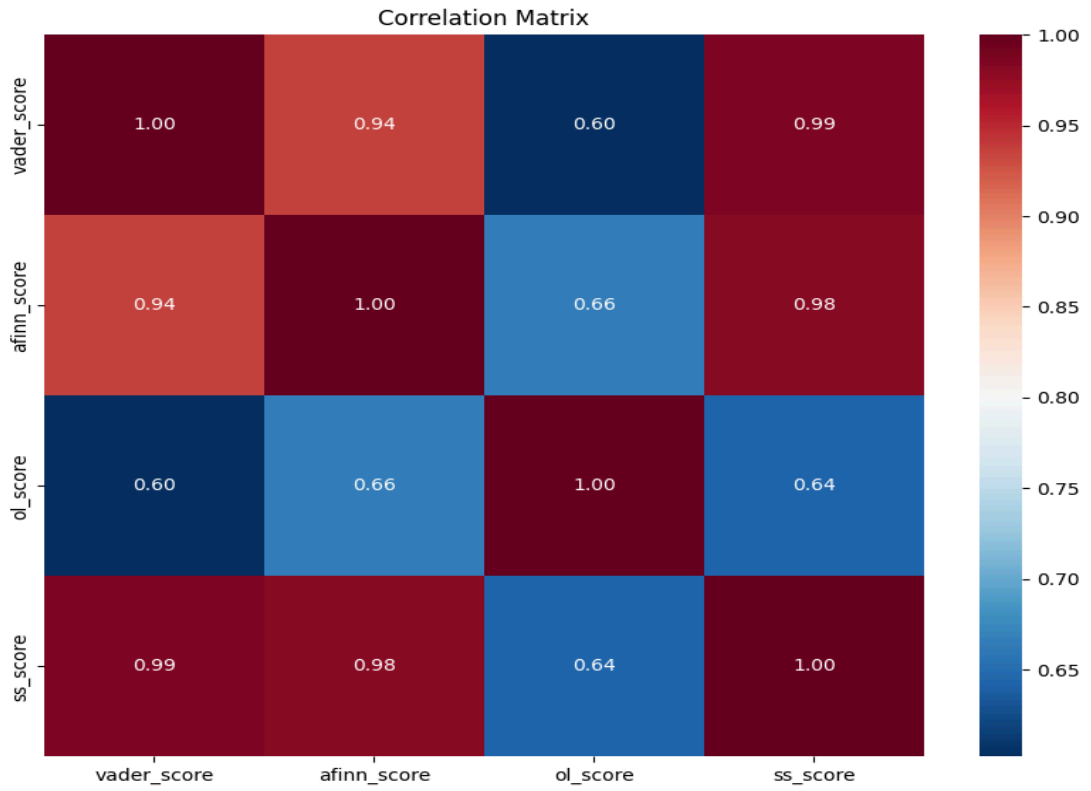


Figure 2. Correlation matrix between the columns of 'sentiment_values' dataset including vader_score, afinn score, ol_score and ss_score

By executing these data preprocessing steps, we ensured that our dataset was cleaned, consolidated, and optimized for further analysis, with a specific emphasis on retaining meaningful sentiment information while eliminating redundancy.

Next, we detail the preprocessing steps applied to the **"user_track_hashtag_timestamp.csv"** dataset. This preprocessing aims to ensure data cleanliness, integrity, and relevance for subsequent analyses. We followed the steps below for cleaning this dataset:

Loading the Dataset: The dataset was loaded into a Pandas DataFrame (df2) from the provided file path. This initial step allowed us to access and manipulate the dataset using Python's data manipulation tools.

Handling Null Values: Null values in the 'hashtag' column were identified and removed using the dropna() function. This step was crucial for maintaining data integrity and ensuring that subsequent analyses were not compromised by missing values.

Filtering Tracks by Usage Count: To focus on tracks with significant user interactions, tracks that were played less than 50 times were filtered out. This filtering process helped reduce noise in the dataset and prioritize tracks with higher usage counts for further analysis.

Merging with Cleaned Sentiment Dataset: The cleaned dataset was merged with sentiment scores obtained from a separate dataset (df1). This merge was based on the

common 'hashtag' column using an inner join, ensuring that only hashtags present in both datasets were retained. The merged dataset (df_sentiment) combined information about user interactions with tracks and associated sentiment scores.

Confirming Dataset Integrity: Several checks were performed to validate the integrity of the merged dataset. These checks included verifying the absence of null values, confirming the dataset's dimensions, and examining the distribution of hashtags. These steps ensured that the merged dataset was complete, consistent, and ready for further analysis.

The last dataset is **"context_content_features.csv"**. The following steps have been taken to clean this dataset:

Loading Necessary Columns: We load the dataset and limit it to the necessary columns to reduce memory usage and computational overhead. This step ensures that only relevant data is loaded into memory, improving processing efficiency.

Removing Tracks with Fewer Plays: Tracks that were played fewer than 50 times are removed. This filtering helps focus the analysis on more popular tracks, which are likely to provide more meaningful insights and trends. Additionally, removing infrequently played tracks can help reduce noise in the data.

Dropping Unnecessary Columns & Removing Null Values: Unnecessary columns such as coordinates, id, place, and geo and also all null values are dropped. These columns do not provide valuable information and we use 'time_zone' instead of 'coordinates', 'place', and 'geo', and 'user_id' instead of 'id'.

Filtering English Language Entries: The dataset is filtered to include only English language entries. This step ensures consistency in language for analysis and modeling purposes, as mixing multiple languages could introduce complexity and potentially skew results.

Merging with Sentiment Data: The dataset is merged with df_sentiment based on specified columns (track_id, created_at, user_id). This merge combines sentiment information with the context content features, enriching the dataset with additional insights about user sentiment towards tracks.

Converting and Dropping Columns: Certain columns like hashtag and user_id are converted to string type for consistency and ease of handling. Additional unnecessary columns are dropped post-merge to streamline the dataset and remove redundant information.

Filtering USA Time Zones: The dataset is filtered to include only USA time zones and their names are simplified. This step focuses the analysis on a specific geographical region, providing more targeted insights relevant to a particular audience or market segment.

Creating Binary Sentiment Column: A binary sentiment column is created based on the sentiment_score. This simplifies sentiment analysis by categorizing sentiment as either

positive or negative, making it easier to interpret and incorporate into further analysis or modeling tasks.

Reordering Columns: Columns are reordered to create the MVP dataset, arranging them in a logical sequence for better readability and usability. This step helps organize the data for easier analysis and model building.

One-Hot Encoding Time Zone: The time_zone column is one-hot encoded to convert categorical data into numerical format, which is required by many machine learning algorithms. This transformation preserves the ordinal relationship between different time zones while preventing the model from interpreting them as continuous variables. The original time_zone column is then dropped to avoid multicollinearity issues.

Spotify Tracks Dataset:

Name of the Column	Variable's Type	Description	Categorical / Quantitative
track_id	Object	Identifier for the track	Unique Values
artists	Object	The names of the artists	Categorical
album_name	Object	The name of the album in which the track appears	Categorical
track_name	Integer	The name of the track	Quantitative
popularity	Integer	The popularity of the track on a scale from 0 to 100, based on the total number of plays and recency	Quantitative
duration_ms	Integer	The length of the track in milliseconds	Quantitative
explicit	Boolean	Indicates whether the track has explicit lyrics	Categorical
danceability	Float	Describing the suitability of a track for dancing	Quantitative
energy	Float	Describing the intensity level of a track	Quantitative
key	Integer	Indicating the main key of a track with 1.0 and 0.0	Quantitative
loudness	Float	The overall loudness of a track in decibel.	Quantitative
mode	Integer	Indicating the main key of a track with 1.0 and 0.0	Quantitative
speechiness	Float	Indicating the presence of spoken words in a track .	Quantitative
acousticness	Float	Describing whether	Quantitative

		the track is acoustic or not	
instrumentalness	Float	Indicating the absence of vocals in a track	Quantitative
liveness	Float	Indicating the presence of an audience on a track	Quantitative
valence	Float	Indicating the musical positiveness	Quantitative
tempo	Float	Beat per minute representation of the overall estimated tempo of a track	Quantitative
time_signature	Integer	An estimated time signature, indicating how many beats are in each bar	Quantitative
track_genre	Object	The genre in which the track belongs	Categorical

Table 4. Variables of 'spotify_tracks' Dataset

Unlike the previous datasets, **"Spotify Tracks Dataset"** doesn't require a very detailed cleaning process. Only two steps are pursued to reach our final dataset.

Loading the Dataset: Upon loading the dataset as 'Spotify', we see that it includes 114000 rows and 20 columns.

Handling Duplicates and Null Values: The result of the duplicated method showed us that 450 duplicates exist in the dataset. After dropping the duplicates, null values were searched and only one row of 3 missing values was found and the dataset was reduced to 113549 rows and 20 columns.

These preprocessing steps collectively prepare the datasets for analysis or modeling tasks, ensuring data quality, consistency, and relevance for deriving meaningful insights or building predictive models.

Data Analysis and Visualization

nowplaying-RS Dataset

The purpose of this Exploratory Data Analysis (EDA) is to understand the underlying patterns and relationships within the nowplaying-RS dataset. We aim to explore the data,

identify and address issues such as multicollinearity and data imbalance, and prepare the dataset for further predictive modeling.

We start by examining the distribution of the target variable sentiment to understand its balance and the distribution of other features to understand their spread and central tendency. As Fig.3 shows the data is highly imbalanced, with a significant majority of positive sentiments (1). This imbalance can affect model training, leading to biased predictions. Therefore, balancing the dataset is necessary.

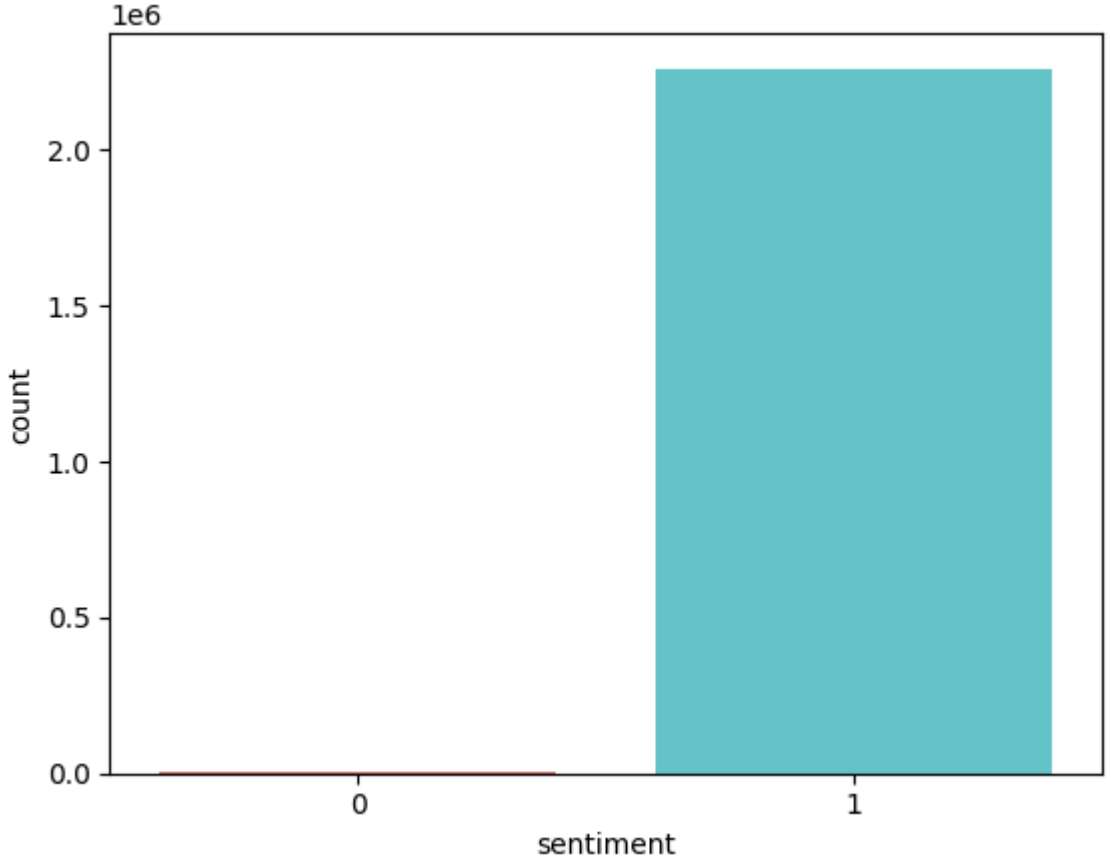


Figure 3. Sentiment distribution

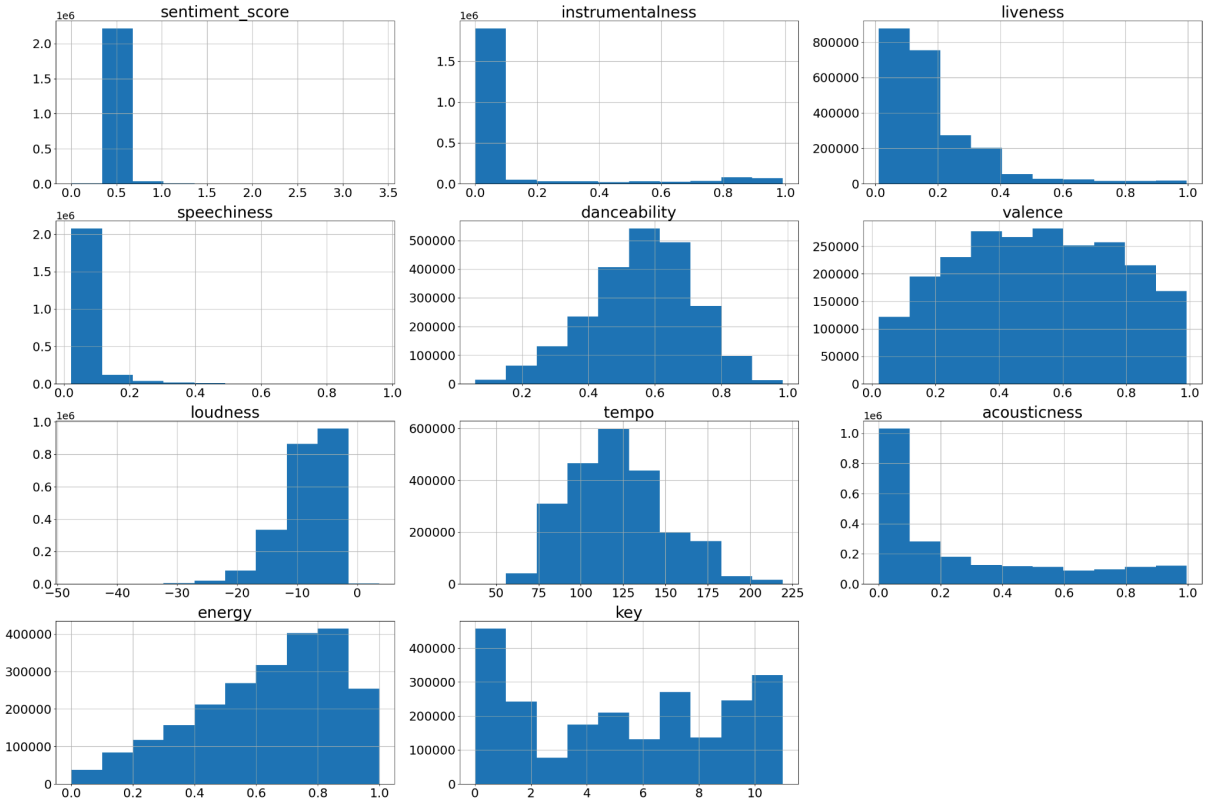


Figure 4. Distribution of the quantitative audio features in the nowplaying-RS

Then, we create a subset of the dataset with continuous variables and visualize their distributions using histograms (Fig. 4). Each feature exhibits different distribution patterns (some features are normally distributed with positive or negative skew). All features need to be normalized or scaled before modeling.

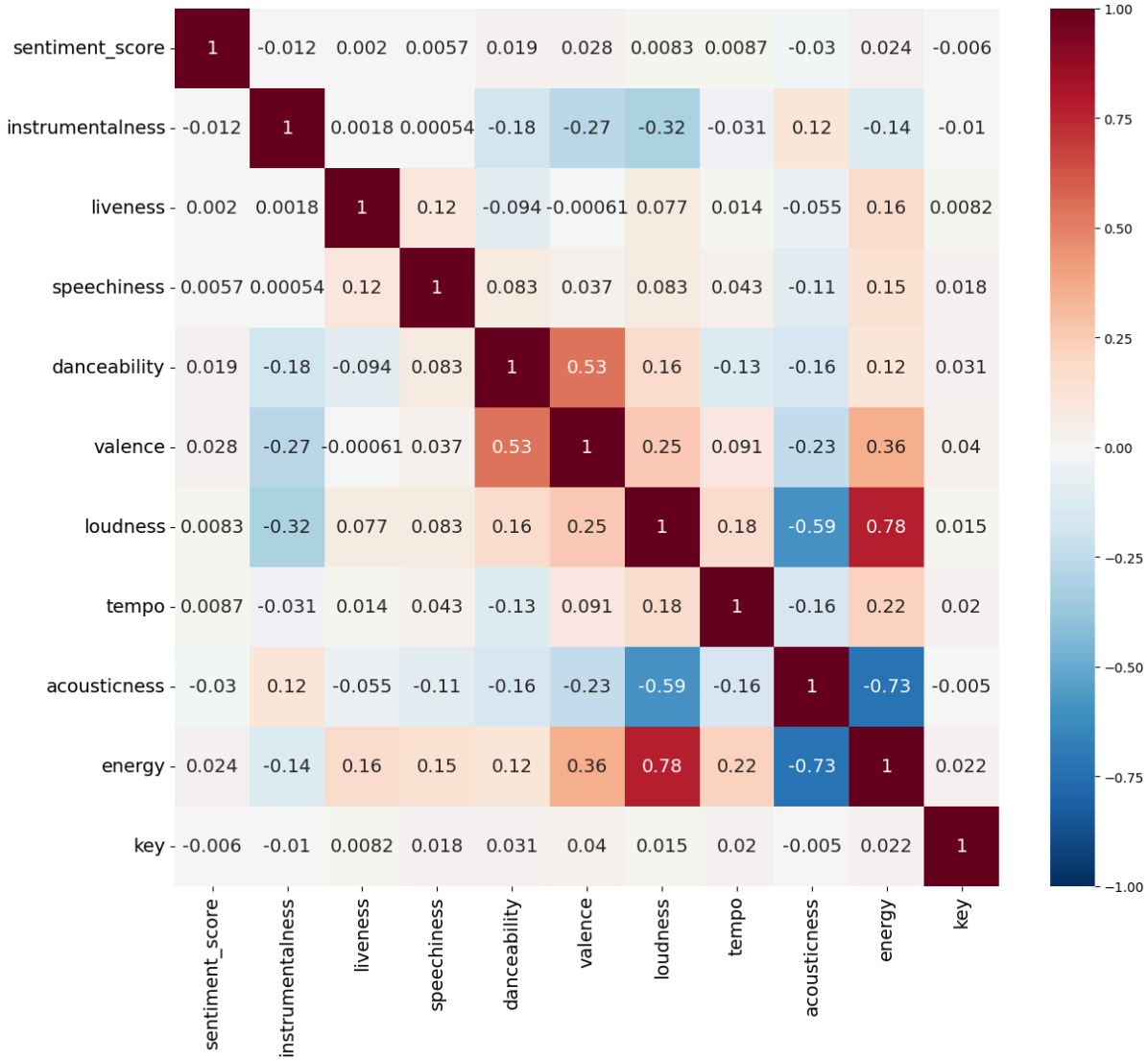


Figure 5. Correlation matrix between the chosen qualitative values of nowplaying-RS dataset

For the next step, we analyzed the correlations between numerical features to identify potential multicollinearity, which could affect model performance and interpretation. The heatmap (Fig. 5) shows the correlation coefficients between features. Notably, loudness and energy exhibit a high correlation, indicating multicollinearity. High multicollinearity can lead to instability in model coefficients, making it difficult to interpret their impact on the target variable. Apart from that, there is a strong negative correlation between acousticness and energy. Additionally, there is a moderate positive correlation between danceability and valence, as well as a moderate negative correlation between acousticness and loudness.

The strong correlation between **energy** and **loudness** and between **acousticness** and **energy** suggests multicollinearity. This can be addressed by removing one of the correlated features or using dimensionality reduction techniques.

Spotify Tracks Dataset

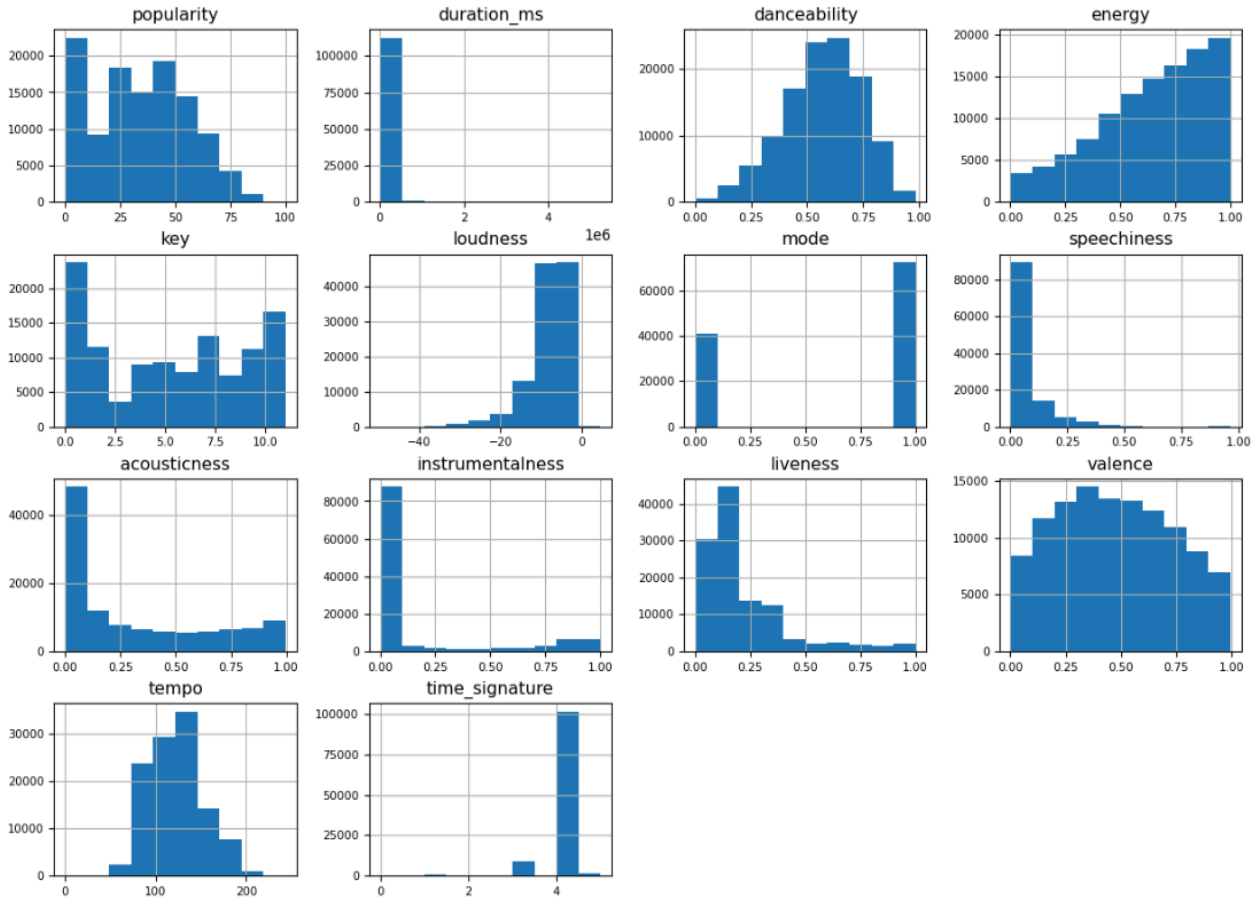


Figure 6. Distribution of qualitative features in the Spotify Tracks dataset (left to right, top to bottom: popularity, duration_ms, danceability, energy, key, loudness, speechiness, acousticness, instrumentalness, liveness, valence, tempo and time_signrature)

Similar to the nowplaying-RS dataset, Fig. 6 shows variables like danceability, valence, and tempo exhibiting quasi-normal distribution. However, the other features are distributed very unequally, notably, duration_ms, instrumentalness, and speechiness. This suggests the presence of a few extreme values in this category.

It must be noted that the tracks in this dataset are not chosen completely at random, as there are exactly 1000 entries for each of the 114 genres. This might explain why some variables don't seem to be randomly or normally distributed, notably energy in this dataset.

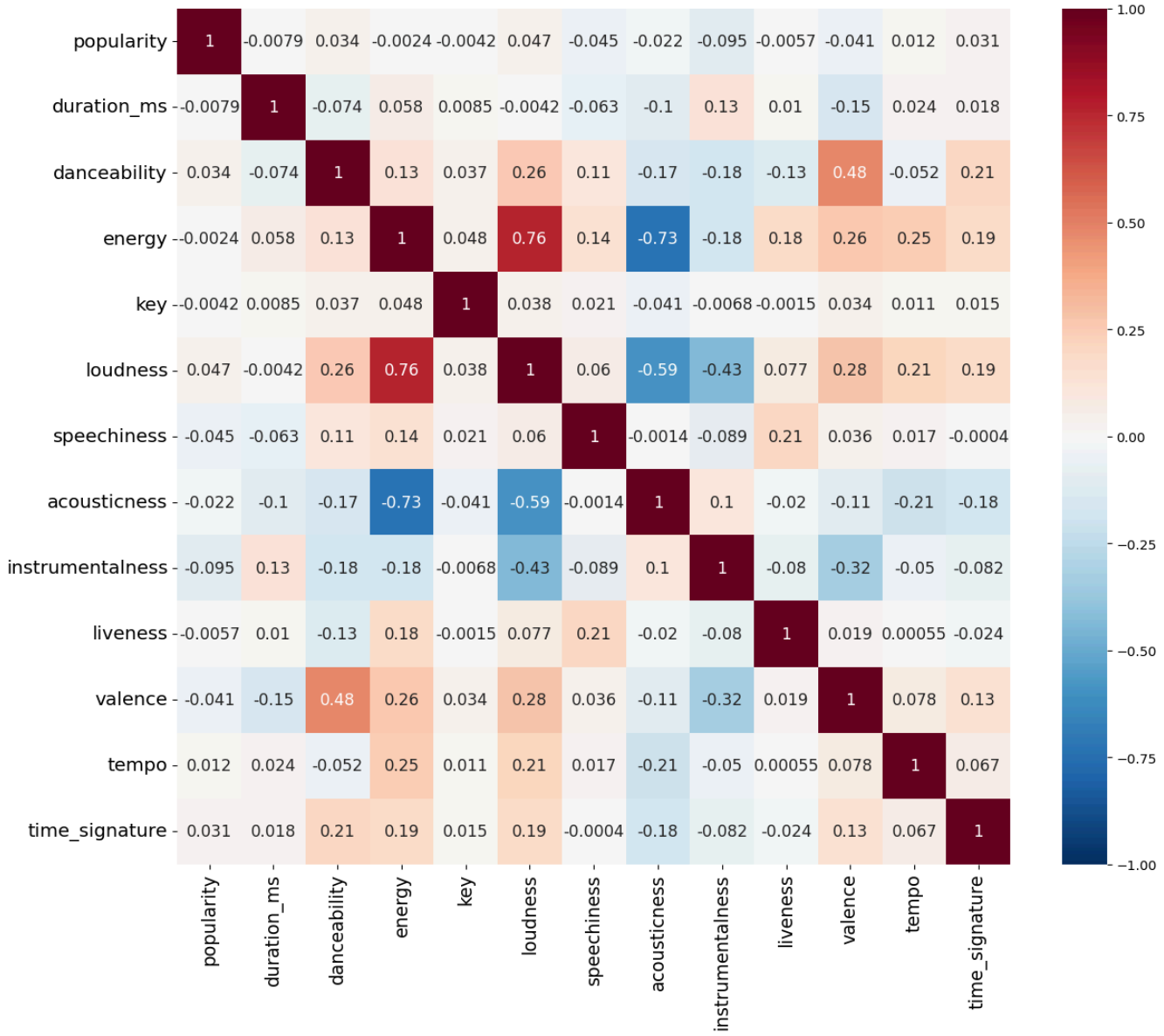


Figure 7. Correlation matrix between the chosen qualitative values of Spotify Track dataset

The correlation pattern of the Spotify dataset shows similarities to the nowplaying-RS dataset. The energy-loudness correlation and the negative acousticness-energy correlation are also observed here. (Fig. 7). As this dataset also contains the categorical variable “explicit”, which takes either the value 0 or 1, we can see another slight correlation: Tracks labeled as “explicit” score slightly higher in terms of “speechiness”. A possible explanation might be that rap songs are more “speechy” and also tend to be more explicit.

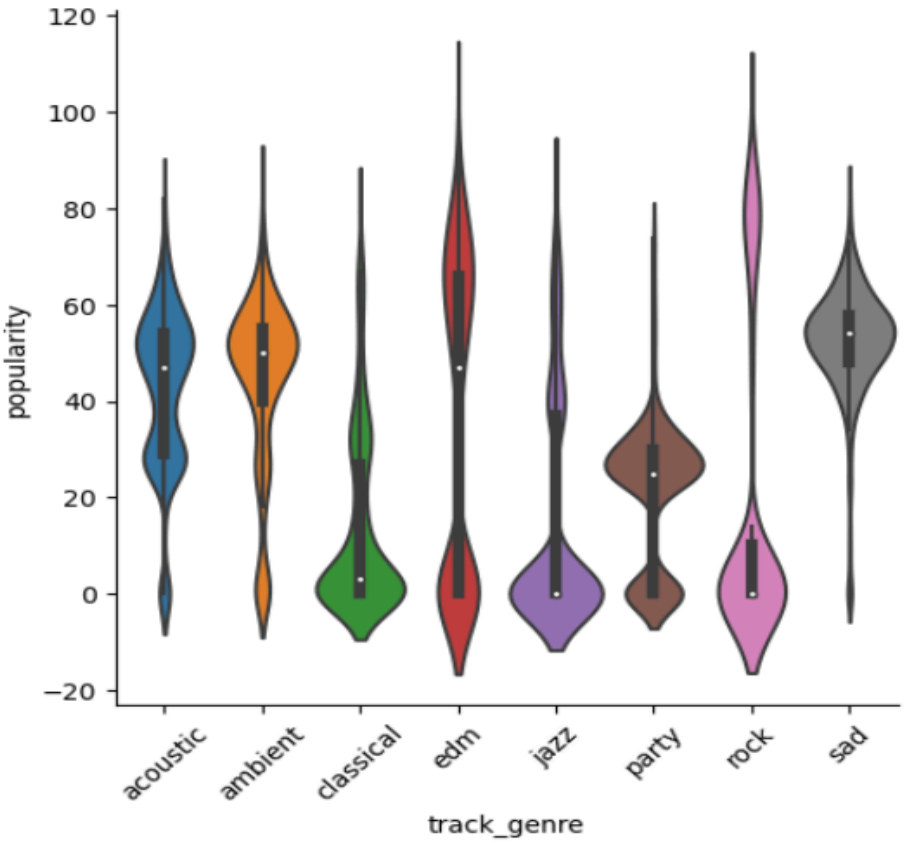


Figure 8. Popularity of the tracks by genres

Figure 8 shows the visualization of an exemplary selection of genres and their respective popularity from the Spotify dataset. As per the dataset's documentation, "[t]he popularity is calculated by algorithm and is based, for the most part, on the total number of plays the track has had and how recent those plays are."
This gives us an insight into possible interpretations of how the popularity is distributed across the genres. For example, EDM seems to have equally many very popular tracks and less popular ones. This might suggest that there are some well known hits in this genre that are listened to by a broader audience, and some lesser known ones that are enjoyed by fans of the genre. A similar effect can be seen with the jazz and classical genres: There are a few songs that are considered quite popular, but most of the songs are possibly lesser known and enjoyed by connoisseurs.

Feature Engineering and Modeling

Feature Engineering

The goal of our project is to present a music recommendation system by considering users' previous listening events. Previously, nowplaying-RS dataset was explained in detail and although it has vast information about the content and context of the tracks, one cannot find song-related information, such as track names, albums, artists, and popularity. However, a

better understanding of the dataset and representing a visually appealing result should also include these. Therefore, we used Twitter and Spotify APIs to enrich our dataset with additional information.
We figured out that Twitter API does not allow us to fetch user-related data with a basic subscription; that's why, Spotify API is used. This allows us to work with our track-id and artist-id columns. Firstly, both columns were included but no matching data was found. This led us to try the columns separately, and only track-id requests fetched responses from the endpoint. Figure 9 shows an example of fetched data. Although genres are gathered from the API as well, mismatches between songs and genres were observed. The reason can be that Spotify is also using a machine learning algorithm to infer information about lesser known songs.

	track_name	album_name	artists	popularity	track_id
0	5 Little Monkeys	100 Fun Songs for Kids Vol. 1	The Little Sunshine Kids	63.0	f05db4f79ab13d87afdc57a32845ed09
1	Dead n Gone	Dead n Gone	Luci4	65.0	a1b80335546946037f44c19f05dead35
2	EGGSTREME DUCK PHONK	EGGSTREME DUCK PHONK	sma$her	57.0	eb4a326f68aa037875e73358d896fd54
3	abcdefu	abcdefu	GAYLE	75.0	8bc882a8ec16fed7fda5b786df3463d1
4	Fruit Bat	Bluey the Album	Bluey	55.0	39a89224e3d775e6c25caa5274ac8571
...	...	...	...	...	...
28867	Cruel Summer	KIDZ BOP 2024	Kidz Bop Kids	57.0	84330c06e1df025ed3afb69b3fda0b57
28868	容易受傷的女人	Coming Home (Remastered 2019)	Faye Wong	44.0	fa6a56ade1528de51362a87af7f66b84
28869	772 Love Pt. 3 (Your Love)	772 Love Pt. 3 (Your Love)	YNW Melly	54.0	778ebf8b9ba90de31972486776e2977b
28870	Six in a Bed	50 Silly Songs	The Little Sunshine Kids	64.0	6f267ec8e92a287a04bb4a05726c7b82
28871	Dance Monkey	KIDZ BOP Party Playlist!	Kidz Bop Kids	57.0	42ffc105744c630858cdf48680ed21dc

Figure 9. Track_name, album_name, artists and popularity columns gathered by using Spotify API

Model Development Strategy

Based on Context and Content Features

For the first part of this project, we want to develop machine learning models to predict music appreciation based on user context and content features.

1. **Preprocessing and Initial Modeling Efforts:**

Building on the preprocessing insights gained from the exploratory data analysis (EDA), the first steps in modeling involved addressing the challenges identified, such as class imbalance and feature scaling.

Data Splitting: The dataset was split into training and testing sets using a standard 80/20 split, ensuring that both sets are representative of the overall data distribution.

Feature Scaling: Given the range of scales across different features, particularly those related to audio properties like loudness and tempo, normalization was essential. We applied

StandardScaler to transform the feature values to have zero mean and unit variance, which helps in stabilizing the learning process and improves convergence in algorithms that are sensitive to feature scaling.

2. **Logistic Regression Baseline Model:**

As a first step, a logistic regression model was used as the baseline. This model's simplicity allowed us to quickly gauge the initial performance and identify major issues, such as the inability to effectively classify the minority class.

Features selected for modeling exclude columns such as 'sentiment', 'sentiment_score', 'user_id', and 'track_id' to avoid data leakage.

- **Model Training and Evaluation**: The logistic regression model was trained on the scaled training data. The performance on the test set indicated a high accuracy of approximately 99.68%, but further examination using a confusion matrix and classification report revealed that the model failed to predict any instances of the minority class accurately.

3. **Addressing Class Imbalance:**

Given the severe class imbalance highlighted by the logistic regression outcomes, strategies to balance the dataset were implemented:

- **SMOTE (Synthetic Minority Over-sampling Technique)**: To overcome the issue of class imbalance, SMOTE was applied to the training data. This technique generates synthetic samples from the minority class to achieve a balanced dataset, which helps improve the classifier's ability to detect the minority class.

4. **Ensemble Modeling and Comparison:**

To capture more complex patterns and reduce the variance of our predictions, we employed ensemble techniques such as Random Forest, Decision Tree, and XGBoost, comparing their performances.

- **Model Configurations and Training**: Each model was configured with a basic set of hyperparameters. The models were trained on the SMOTE-enhanced and scaled training data, ensuring they learned to recognize patterns across a balanced class distribution.
- **Model Evaluations:** The performance of each model was evaluated on the scaled test data. Metrics such as accuracy, precision, recall, and F1-score were particularly considered to assess each model's ability to generalize and correctly classify instances of both classes.

- **Comparison of Models:**

Logistic Regression:

Accuracy: 77.30%
Precision and recall for the minority class (negative sentiment) were improved but still not satisfactory.

Decision Tree:

Accuracy: 99.22%
Significant improvement in recall for the minority class, indicating better handling of class imbalance.

Random Forest:

Accuracy: 99.24%
Further improvement in both precision and recall for the minority class compared to the Decision Tree.

XGBoost:

Accuracy: 99.00%
High accuracy and improved performance on the minority class, but slightly lower than Random Forest.

5. **Hyperparameter Optimization Using Randomized Search:**

The Random Forest model showed promising results in preliminary tests. To further enhance its performance, RandomizedSearchCV was utilized to optimize several hyperparameters. This approach searches over specified parameter values to find the best combination for improving model performance.

- **Parameter Grid:** A comprehensive grid of parameters including the number of trees (n_estimators), tree depth (max_depth), minimum samples to split a node (min_samples_split), and minimum samples at a leaf node (min_samples_leaf) was defined.

- **Search Process:** The search involved fitting various configurations of the Random Forest model on a subset of the training data (2% used for faster computation), using SMOTE for balancing and cross-validation to ensure robustness.

- **Best Model Selection:** The best performing model parameters (n_estimators: 200, max_depth: 30, min_samples_split: 2, min_samples_leaf: 2, max_features: 'log2) were selected based on cross-validated accuracy scores.

6. Analysis of Feature Importance from Optimized Random Forest Model

Following the hyperparameter tuning using RandomizedSearchCV, the optimized Random Forest model provided a detailed view into the relative importance of each feature in predicting music appreciation. The significance of understanding and visualizing feature importance lies in its ability to highlight which features contribute most significantly to the model's decision-making process. This insight is crucial for both refining the model further and for providing actionable insights to stakeholders regarding which aspects of the music or user feedback most influence listener appreciation.

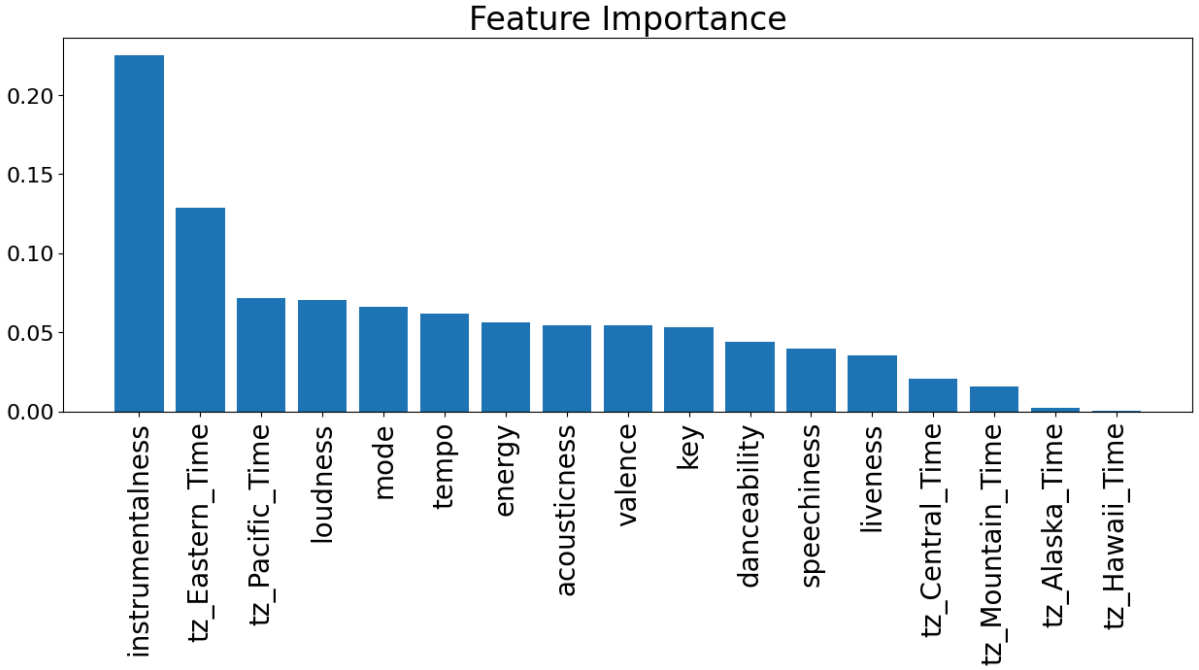


Figure 10. Feature importance

- **Plotting Feature Importance:**

Purpose: The feature importance plot was generated as a part of the model evaluation to visualize and rank the features based on their impact on the model's predictions. This step is essential for interpreting the model's behavior, particularly in understanding how different features influence the predictive performance.

Methodology: Using the feature_importances_ attribute of the trained Random Forest model, we extracted the importance scores for each feature. These scores represent the mean decrease in impurity (usually calculated by the Gini index) contributed by each feature across all trees in the forest.

- **Key Observations:**

High Importance Features: Features such as instrumentalness, loudness, and time-related features like Eastern Time and Pacific Time often appear at the top of the list. High scores in instrumentalness and loudness suggest a strong preference or aversion by listeners towards instrumental music and the loudness of tracks, which directly impacts their appreciation.

Temporal Features: The importance of time zone features (e.g., Eastern Time, Pacific Time) indicates significant geographical variations in music appreciation, possibly reflecting cultural or regional musical preferences.

Lower Importance Features: Features appearing towards the bottom of the importance ranking, such as speechiness or valence, might have a lesser direct impact on the prediction of music appreciation in the context of this specific dataset.

- **Strategic Implications:**

Model Refinement: Understanding which features are most influential allows for the optimization of data collection and preprocessing in future iterations of modeling work. For example, focusing on enhancing the quality and range of data related to the most influential features could improve model performance.

Business Insights: For stakeholders, knowing which features affect music appreciation the most can guide strategic decisions, such as marketing different genres of music in specific regions or adjusting music production elements to align with listener preferences.

In conclusion, the feature importance analysis not only enriches our understanding of the dataset and model but also provides a clear direction for both technical improvements and strategic business actions. This step is indispensable for advancing toward a more targeted and effective music recommendation system.

7. Implementation of Optimized Random Forest Model with Class Weights

In this phase of the project, we utilized the best parameters identified from a previous grid search optimization process to configure the Random Forest model. Additionally, instead of relying on SMOTE for balancing the dataset, we explored the use of class weights to manage the imbalance directly within the model training process. This approach allows us to directly influence the model's handling of class imbalance through the learning algorithm itself rather than altering the dataset's composition.

Class weights were manually specified with a greater weight assigned to the minority class (0). The weight ratio (10:1) was chosen to counteract the severe class imbalance by increasing the penalty for misclassifying the minority class, thus focusing the model's learning toward better recognition of these under-represented samples.

- **Model Evaluation:**

Performance Metrics:

Upon prediction with the test set, the model achieved an **accuracy** of approximately 99.42%. While this high accuracy is consistent with previous models, the key distinction in this evaluation lies in the improved recognition of the minority class.

Precision, Recall, and F1-Score: The model exhibited a substantial increase in recall for the minority class (0.92), with a corresponding F1-score of 0.50. This indicates a successful identification of the minority class compared to earlier models without class weights, where recall was significantly lower.

Confusion Matrix Analysis:

The confusion matrix provided further insight, showing that 1,330 out of 1,451 instances of the minority class were correctly identified, with only 121 misclassifications. Although there were 2,510 false positives, the substantial increase in true positives for the minority class marks a significant improvement in the model's capability to recognize less appreciated music tracks.

This approach demonstrated that adjusting class weights within the Random Forest algorithm is an effective strategy for enhancing model sensitivity toward the minority class in an imbalanced dataset. By directly modifying the learning algorithm's focus, we achieved a substantial improvement in detecting the less frequently represented class without resorting to synthetic data augmentation methods like SMOTE. This method not only preserves the original dataset's integrity but also ensures that the model remains robust and generalizable to real-world data distributions.

8. **Feature Engineering for Optimized Random Forest Model**

In the ongoing development of our music sentiment analysis model, we applied advanced feature engineering techniques to refine the predictive capabilities of our Random Forest classifier, which had been previously optimized with class weights.

- **Feature Reduction**

Initial steps involved the removal of features with minimal impact on model performance to streamline the dataset. Specifically, time zone features such as 'tz_Hawaii_Time', 'tz_Alaska_Time', and 'tz_Mountain_Time' were eliminated based on their low importance scores derived from earlier analyses.

- **Interaction Term Creation**

To capture potential synergistic effects between features, an interaction term was introduced between 'instrumentalness' and 'loudness'. This decision was informed by the hypothesis that the interaction between these audio features might provide deeper insights into the factors influencing music sentiment, reflecting the complex nature of musical perception.

- **Polynomial Feature Engineering**

Polynomial transformations were applied to 'instrumentalness' and 'loudness' to explore non-linear dependencies that could better capture the nuances of the dataset. This step involved elevating these features to the second degree, which allows the model to consider both the individual effects and the combined impact of these features squared and as an interaction term.

- **Model Implementation**

The RandomForestClassifier, configured with previously optimized parameters and adjusted class weights to address the class imbalance, was retrained on this newly engineered feature set. The class weights were specifically tuned to improve the model's sensitivity to the minority class, thereby enhancing its predictive accuracy across classes.

- **Model Evaluation**

Post-enhancement, the model demonstrated an accuracy of approximately 99.42% on the test set, mirroring the performance achieved prior to these feature engineering steps. Despite similar accuracy metrics, the nuanced engineering efforts were critical in maintaining high performance while adjusting the model to handle newly integrated features effectively.

- **Results**:

Accuracy: 99.42%
Precision for class 0 (Minority class): 35%
Recall for class 0: 91%
F1-score for class 0: 50%
Confusion Matrix: Demonstrated the model's capability to accurately identify 1,327 out of 1,451 instances for the minority class, alongside robust performance for the majority class.

These outcomes validate the effectiveness of our feature engineering strategy. Despite similar accuracy levels to earlier models, the introduction of interaction and polynomial features supports the robustness of our model, ensuring it remains effective even as the complexity of the feature set increases. This approach not only sustains model performance but also enriches the model's ability to generalize across different musical sentiments, thus reinforcing our analytical framework for sentiment prediction in music based on Twitter data.

9. Deep Learning Model Implementation and Evaluation

Here, we describe the implementation of a deep-learning model for sentiment classification. We focus on the architecture, the rationale behind specific choices in the model design, the interpretation of the results obtained, and a comparison with the performance of the Random Forest model.

- **Deep Learning Model Architecture**

The architecture of the deep learning model was designed with multiple layers to capture complex patterns in the data. Here are the key components:

Input Layer: The input dimension matches the number of features in the training data.

Hidden Layers:

First Hidden Layer: 128 neurons with ReLU activation function. ReLU helps in dealing with the vanishing gradient problem, making the network train faster.

Dropout Layer: 50% dropout rate to prevent overfitting by randomly setting half of the input units to 0 at each update during training time.

Second Hidden Layer: 64 neurons with ReLU activation.
Dropout Layer: Another 50% dropout rate.

Third Hidden Layer: 32 neurons with ReLU activation.

Dropout Layer: 50% dropout rate.

Output Layer: A single neuron with a sigmoid activation function. The sigmoid function is suitable for binary classification as it outputs a probability value between 0 and 1.

The model was compiled with the Adam optimizer and binary crossentropy loss function:

Adam Optimizer: Adam is a popular optimization algorithm that combines the advantages of two other extensions of stochastic gradient descent, AdaGrad, and RMSProp. It works well in practice and is efficient.

Binary Crossentropy Loss: This loss function is ideal for binary classification tasks as it measures the difference between two probability distributions – the predicted output and the actual output.

- **Interpretation of Results**

Accuracy: The model achieved a high test accuracy of 98.05%. This indicates that the model is performing well on the test data in terms of overall accuracy.

Precision and Recall: The precision for class 0 is low, indicating a high number of false positives. However, the recall for class 0 is very high, indicating that almost all actual class 0 samples are identified. The precision and recall for class 1 are both very high, indicating excellent performance for the majority class.

Confusion Matrix: The confusion matrix shows that there are a significant number of false positives for class 0. This is expected due to the high imbalance in the dataset.

- **Training History**

The training history plots show the model's performance over the epochs:

Model Accuracy Plot: The training and validation accuracy improve steadily, indicating that the model is learning effectively.

Model Loss Plot: The training loss decreases consistently, and the validation loss shows some fluctuations but generally follows the training loss trend, indicating a good fit without significant overfitting.

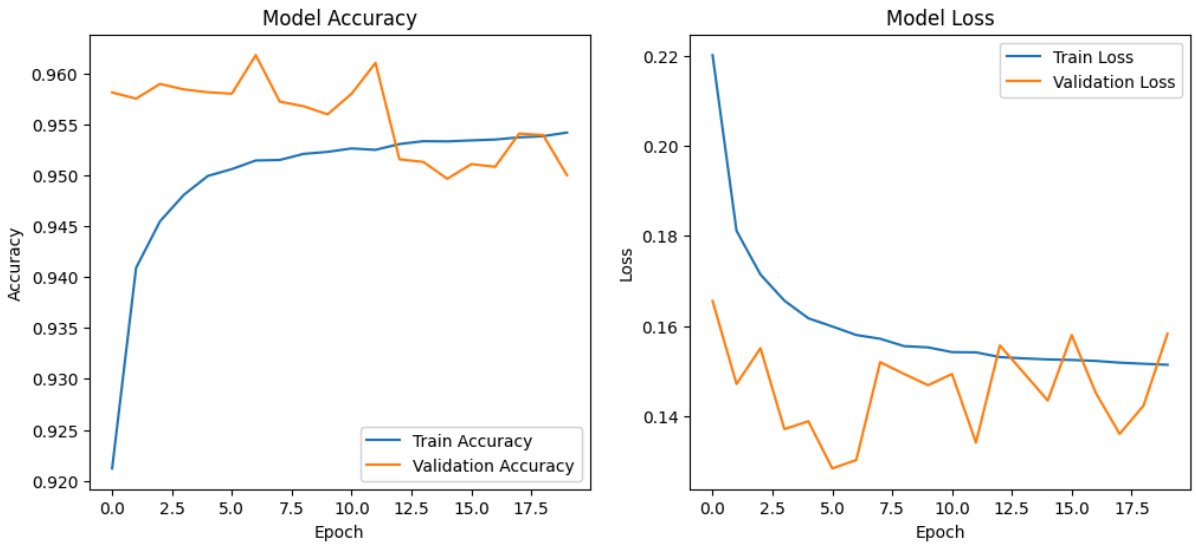


Figure 11. Training and Validation Accuracy and Loss for the Deep Learning Model

- **Comparison with Random Forest Model**

Despite the deep learning model's high test accuracy, its performance is worse than the Random Forest model in several aspects:

Class 0 Precision and Recall: The deep learning model's precision for class 0 is significantly lower than the Random Forest model's precision. This means the deep learning model produces more false positives for class 0 compared to the Random Forest model.

Overall Performance: The Random Forest model, with an accuracy of 99.42%, outperforms the deep learning model in terms of precision, recall, and F1-score for both classes.

- **Reasons for Worse Performance of Deep Learning Model**

Several factors could contribute to the deep learning model's worse performance compared to the Random Forest model:

Imbalanced Data: The deep learning model, even with SMOTE balancing, struggles to handle the extreme imbalance in the dataset. Class 0 has very few samples compared to class 1, making it difficult for the model to learn effectively.

Model Complexity: Deep learning models are complex and require large amounts of data to generalize well. While the dataset is large, the imbalance could hinder the model's ability to learn meaningful patterns for the minority class.

Overall, while the deep learning model shows promise, the Random Forest model remains the superior choice for this specific task, given its better handling of the class imbalance and overall performance metrics.

Interpretation of the results

Using interpretability tools, we analyzed the feature importance and model behaviors to gain insights into the results of our models. For the Random Forest model, feature importance was plotted, revealing that features such as 'instrumentalness', and 'loudness' had significant impacts on the prediction of sentiment. This understanding allowed us to engineer features more effectively and adjust the model parameters to enhance performance.

For the deep learning model, although the architecture was designed to capture complex patterns in the data, the results showed that it did not outperform the Random Forest model. The deep learning model achieved a high accuracy but struggled with precision and recall for the minority class, indicating potential issues with class imbalance that were not fully mitigated by SMOTE.

Scientific and Business Conclusions

The success of the Random Forest model highlights its robustness and ability to handle class imbalance effectively when combined with feature engineering and class weighting techniques. This model's interpretability, through feature importance scores, provides valuable insights for business decisions, such as understanding which song attributes are most predictive of positive sentiment.

In contrast, the deep learning model, despite its sophisticated architecture, did not achieve better performance than the Random Forest model. This result underscores the importance of choosing the right model for the problem at hand and ensuring adequate handling of class imbalances. The complexity of deep learning models may not always translate to better performance, especially in scenarios where simpler, well-tuned models like Random Forests can provide more reliable and interpretable results.

From a business perspective, the insights derived from the Random Forest model can inform strategies for music recommendation systems, targeted marketing campaigns, and product development focused on enhancing user engagement and satisfaction. Understanding the key drivers of positive sentiment in music can help businesses tailor their offerings to better meet user preferences and improve overall user experience.

Based on only Content Features

A similar strategy to the previous one was pursued but only audio features of the rows were used. The main idea was to employ the Spotify dataset for recommending songs for the users in nowplaying-RS since the Spotify dataset does not include any context information for establishing a similarity.

Audio features are instrumentalness, liveness, speechiness, danceability, valence, loudness, tempo, acousticness, energy, mode, and key of the tracks. Since these give an idea about the tracks, these features can collectively capture the mood and emotional content of the music track. The strategy followed is summarized below.

1- Scaling the Features

All features were given in the different scales, that's why it is important to normalize the data to have a zero mean or unit variance. Here, StandardScaler from the Sklearn-preprocessing package was employed to have zero mean.

2- Reducing the dimensionality with Principal Component Analysis (PCA)

PCA aims to reduce the noise and redundancy in the data by reducing the dimensionality, hence, it may provide a more robust ML model. In the figure below, the explained variance ratio and the cumulative explained variance ratio can be observed.

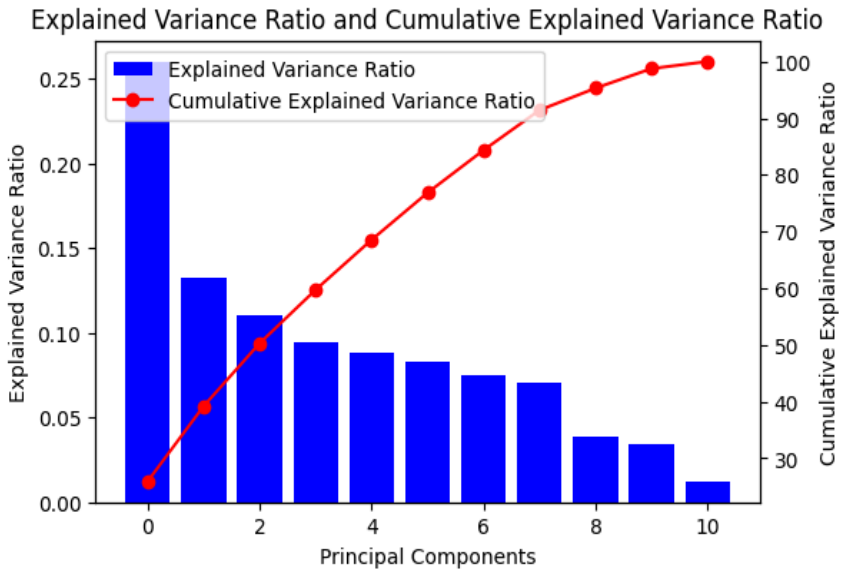


Figure 12. PCA analysis with audio features of nowplaying RS dataset.

In total we have 11 audio features, PCA shows that we could reduce it to 8 audio features which provides more than 90 percent of variance. However, the difference in dimensionality (3 features) might not significantly impact the computational load or model performance in your specific application. Therefore, it was decided to continue with all features without using PCA.

3 - Oversampling for balancing the dataset

Various types of oversampling exist for preventing imbalance in datasets (Figure 11). In our modelings, we compared the efficiency of Random Over Sampler (ROS), ADASYN, and SMOTE. The performance of the Random Forest Classification(n_estimators=10, random_state=42, n_jobs=-1) was compared. While positive sentiments were classified with 99% f1 scores for all, 25%, 34 %, and 35% f1 scores were obtained for the negative sentiments, respectively. Therefore, we continued the modeling with SMOTE.

ROS:

Accuracy: 0.9815986363806756

	precision	recall	f1-score	support
0	0.14	0.96	0.25	1451
1	1.00	0.98	0.99	452048
accuracy			0.98	453499
macro avg	0.57	0.97	0.62	453499
weighted avg	1.00	0.98	0.99	453499

ADASYN:

Accuracy: 0.9880440750696253

	precision	recall	f1-score	support
0	0.20	0.95	0.34	1451
1	1.00	0.99	0.99	452048
accuracy			0.99	453499
macro avg	0.60	0.97	0.67	453499
weighted avg	1.00	0.99	0.99	453499

SMOTE:

Accuracy: 0.9886328305023826

	precision	recall	f1-score	support
0	0.21	0.95	0.35	1451
1	1.00	0.99	0.99	452048
accuracy			0.99	453499
macro avg	0.61	0.97	0.67	453499
weighted avg	1.00	0.99	0.99	453499

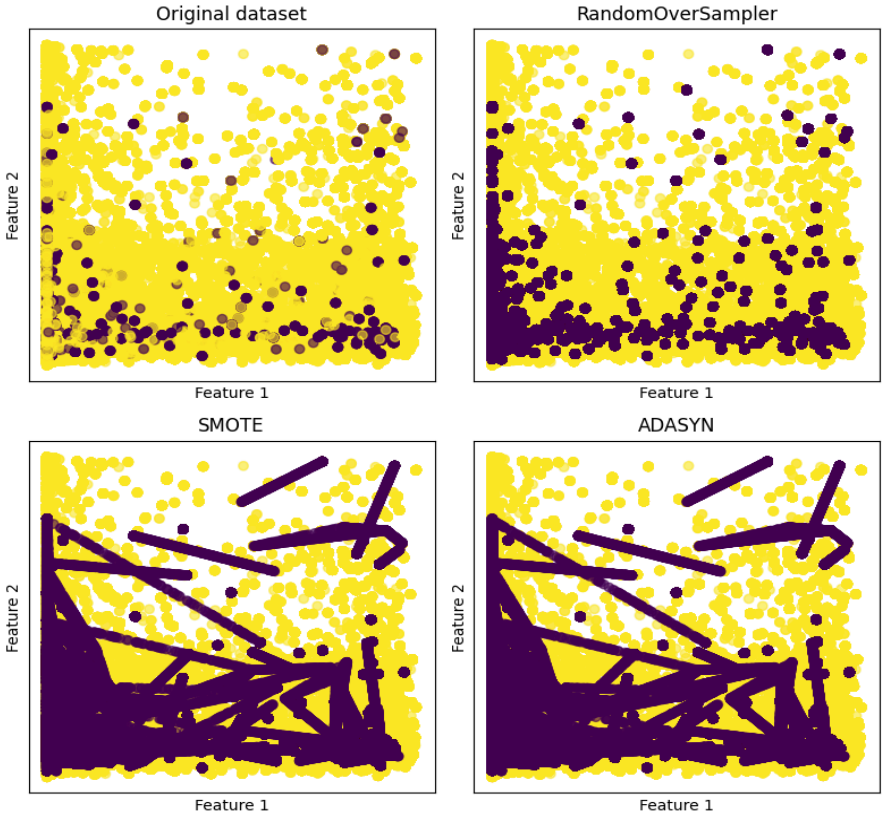


Figure 14. Oversampling methods show the change in the dataset by employing 2 features.

4- ML Models

Random Forest, Logistic Regression, XGBoost, and Dense Neural Networks were the models used. Unfortunately, ML modeling with content features was the last strategy that we used and there was not enough time to enhance the performance by using different hyperparameters, dense layers, and loss functions to obtain an optimum performance for using it in the recommendation system. Below, the summary of the models can be seen.

Model Description	F1-score (Negative Class)	F1-score (Positive Class)
Logistic Regression	2%	86%
Random Forest (n_estimators=10, random_state=42, n_jobs=-1)	35%	99%
Grid Search with Random Forest (best estimator from grid search)	35%	99%
XGBoost (use_label_encoder=False, eval_metric='logloss', eta=0.6)	34%	99%
Dense Layers: 2 Hidden Layers (1: 64 neurons, activation: ReLU; 2: 32 neurons, activation: ReLU), Output Layer (1 neuron, activation: Sigmoid), epochs=100, batch_size=32, validation_split=0.2	33%	99%

Genre Detection with Machine Learning

The #nowplaying dataset initially only contained IDs for each track and no further identifiers of any kind as to what the name of the song was. Furthermore, the dataset had been created in 2014 and the format of the track_id had been changed in the meantime which meant it was not possible to resolve the IDs to track names. The only known features about each track were the numerical metrics such as valence, danceability… etc., but it is impossible to imagine what a given song would *sound* like based on these features alone. This meant that the dataset was hard to interpret for a human, which also meant that there would be very little information to rely on in terms of interpretability and performance assessment of the recommendation model.

Thus, the idea arose to try and assign a genre to each track. This would provide at least a category based on which one can get a rough idea of what a track might sound like. Furthermore, having genre categories assigned to the tracks might even improve the performance of the recommendation system. To achieve this, we decided to train machine learning models on the Spotify dataset, where the tracks were already annotated with genre tags, and later apply it to the #nowplaying dataset to determine track genre.

What even is genre?

For the purpose of this project, we made the initial assumption that musical tracks can be put in groups based on shared features or properties; said group can be referred to as *genre*. Some of these features include the ones that the Spotify API provides: For example, we can assume that tracks belonging to the Dance genre have on average higher danceability scores, and tracks from the Acoustic genre would likely overall have very high acousticness scores.

Crucially, it has to be noted that both of our datasets do not contain information about certain aspects that might be highly influential on genre assignment. Examples of this might be the origin country of the artist or instruments used. Nevertheless, we wanted to try and train a deep learning model on the data we had available, hoping it still would be sufficient for a successful detection.

Preprocessing

The track_genre column in the Spotify dataset contains 1000 tracks of each of the 114 different genres. At first, attempts were made to train the model on this data frame as is and attempt class prediction with the full set of 114 genres. The results however were less than convincing, the accuracy scores were <0.15. Upon further analysis, we noticed that some of the pre-assigned genre categories were very similar and could therefore not be discerned with accuracy (an example of this would be the labels “electro” and “electronic”). In order to remedy this, we decided to combine the 114 genres into 9 broader genre groups, expecting these would be more distinctive from one another. Furthermore, we dropped some of the original categories of track_genre and their respective entries altogether because they didn’t qualify as a genre at all - an example would be the labels “german”, which is hard to define because there is German pop, German pop, etc, but simply because two songs have German text or are performed by German artists does not mean they share particularly many features apart from that.

After assigning the new genre groups, we created a new stratified subset which included 7000 tracks from each of the groups to maintain a balanced dataset. When splitting the data into test and training sets, we also made sure it was stratified.

Traditional Machine Learning Methods

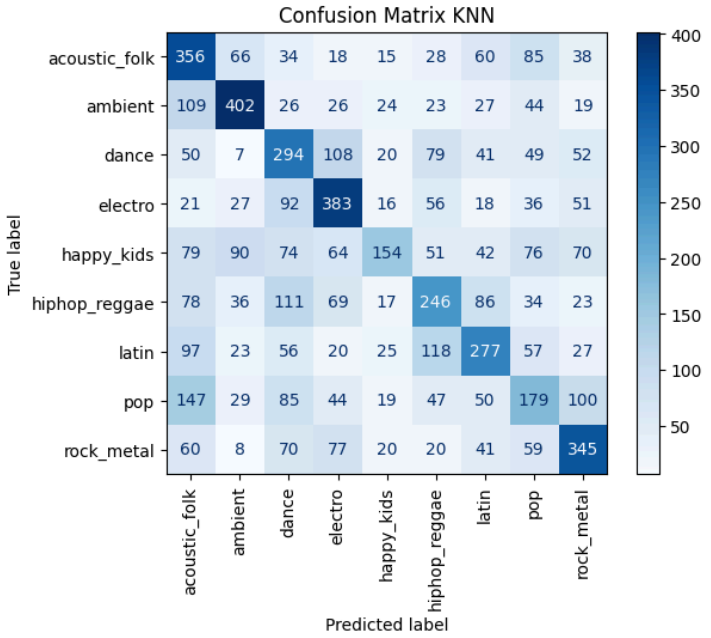
The first attempts at genre classification were made using “traditional” machine learning methods such as K-Nearest Neighbor, Decision Tree, Random Forest, and Support Vector Machine.

The parameters for each of these models were chosen via GridSearchCV to optimize for accuracy.

The classification reports and confusion matrices for each of the models are displayed below:

K-nearest Neighbor Classification Report and Confusion Matrix

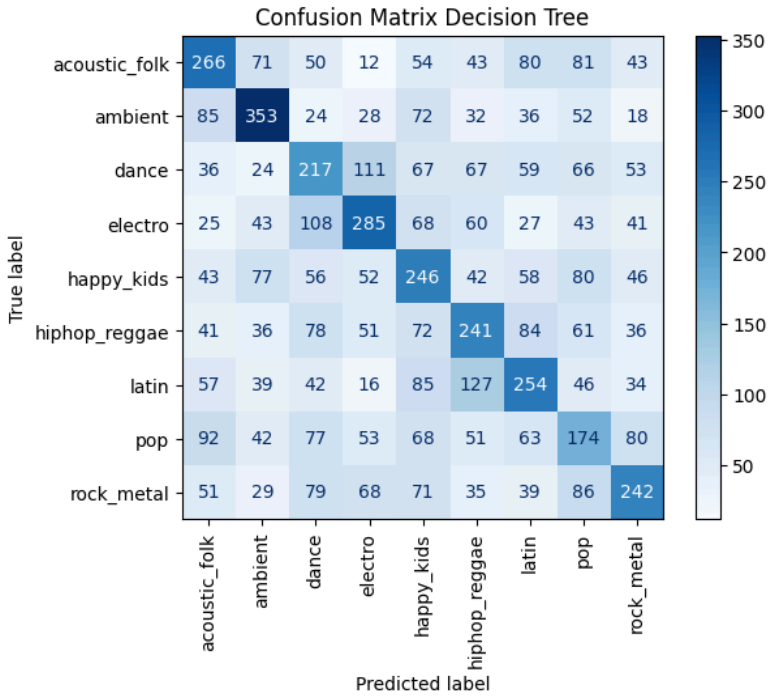
Parameters: n_neighbors = 30, metric = "manhattan"



	precision	recall	f1-score	support
acoustic_folk	0.36	0.51	0.42	700
ambient	0.58	0.57	0.58	700
dance	0.35	0.42	0.38	700
electro	0.47	0.55	0.51	700
happy_kids	0.50	0.22	0.30	700
hiphop_reggae	0.37	0.35	0.36	700
latin	0.43	0.40	0.41	700
pop	0.29	0.26	0.27	700
rock_metal	0.48	0.49	0.48	700
accuracy			0.42	6300
macro avg	0.43	0.42	0.41	6300
weighted avg	0.43	0.42	0.41	6300

Decision Tree Classification Report and Confusion Matrix

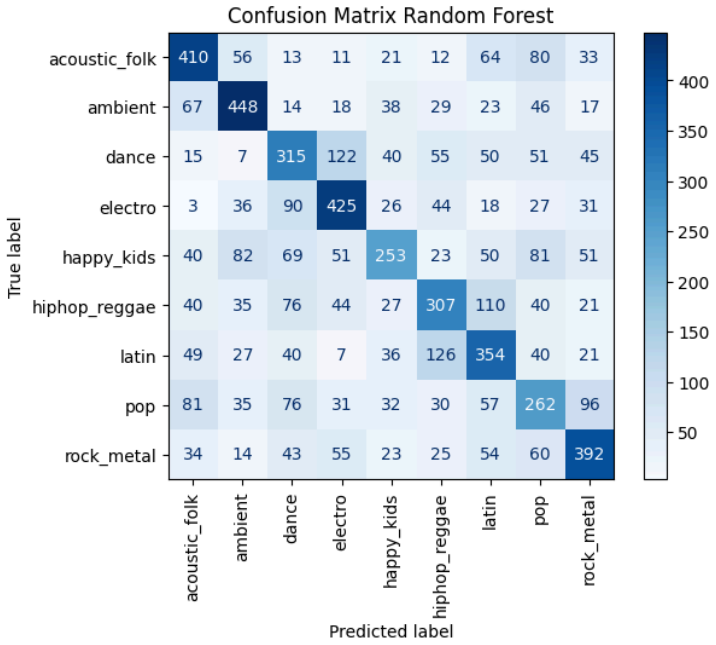
Parameters: 'criterion': 'gini', 'splitter': 'best'



	precision	recall	f1-score	support
acoustic_folk	0.38	0.38	0.38	700
ambient	0.49	0.50	0.50	700
dance	0.30	0.31	0.30	700
electro	0.42	0.41	0.41	700
happy_kids	0.31	0.35	0.33	700
hiphop_reggae	0.35	0.34	0.34	700
latin	0.36	0.36	0.36	700
pop	0.25	0.25	0.25	700
rock_metal	0.41	0.35	0.37	700
accuracy			0.36	6300
macro avg	0.36	0.36	0.36	6300
weighted avg	0.36	0.36	0.36	6300

Random Forest Classification Report and Confusion Matrix

Parameters: 'max_depth': 20, 'min_samples_leaf': 1



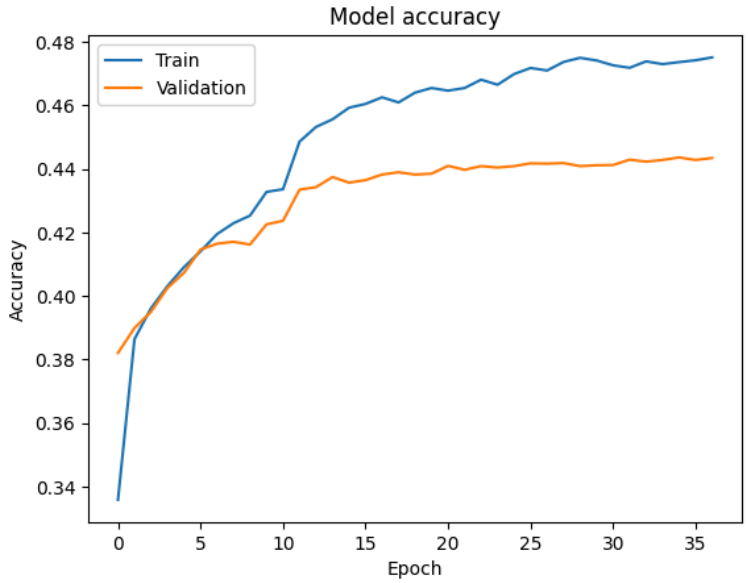
	precision	recall	f1-score	support
acoustic_folk	0.55	0.59	0.57	700
ambient	0.61	0.64	0.62	700
dance	0.43	0.45	0.44	700
electro	0.56	0.61	0.58	700
happy_kids	0.51	0.36	0.42	700
hiphop_reggae	0.47	0.44	0.45	700
latin	0.45	0.51	0.48	700
pop	0.38	0.37	0.38	700
rock_metal	0.55	0.56	0.56	700
accuracy			0.50	6300
macro avg	0.50	0.50	0.50	6300
weighted avg	0.50	0.50	0.50	6300

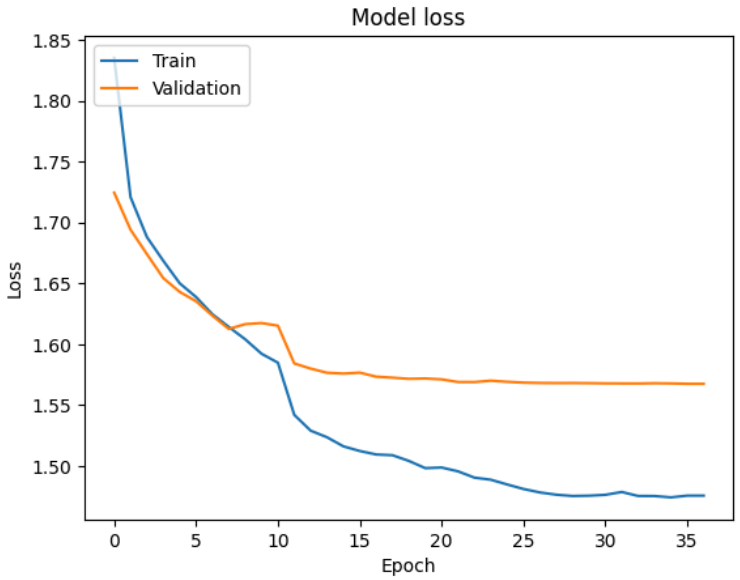
As is visualized in the figures above, the Random Forest models yielded the best results overall with an f1-score of 0.50.

Deep learning

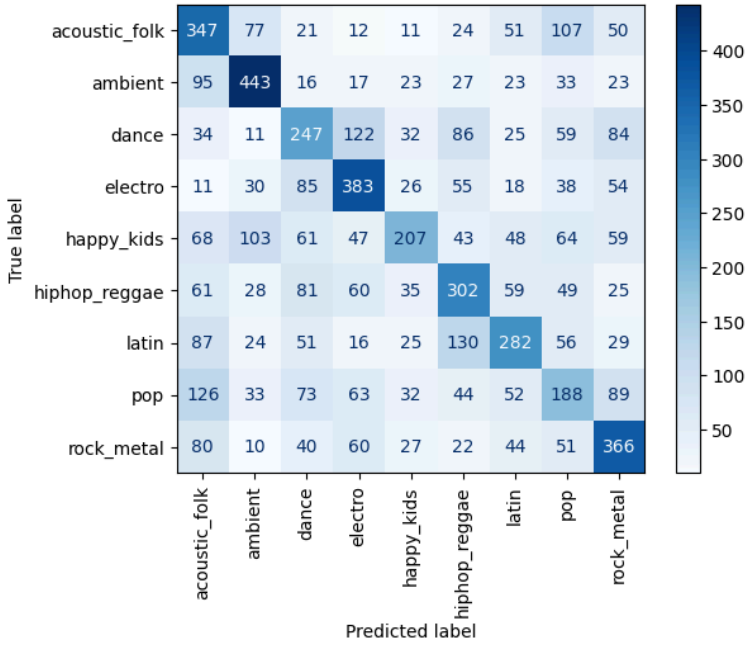
We wanted to see if a deep learning model could potentially outperform the “traditional” approaches and decided to build a sequential dense learning model with the following parameters:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 512)               6144

 dense_1 (Dense)             (None, 256)               131328

 dropout (Dropout)           (None, 256)               0

 dense_2 (Dense)             (None, 128)               32896

 dropout_1 (Dropout)         (None, 128)               0

 dense_3 (Dense)             (None, 64)                8256

 dropout_2 (Dropout)         (None, 64)                0

 dense_4 (Dense)             (None, 32)                2080

 dense_5 (Dense)             (None, 16)                528

 dense_6 (Dense)             (None, 9)                 153

=================================================================
Total params: 181385 (708.54 KB)
Trainable params: 181385 (708.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```



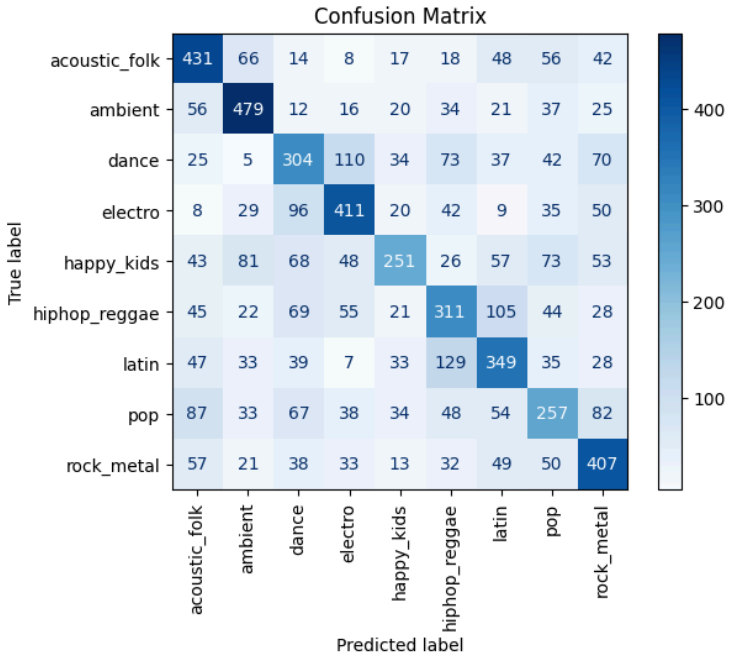


Neural Network Confusion Matrix



As the performance of the Dense network did not surpass the performance of the Random Forest model, we combined the model with the Random Forest model to create a stacked model, as visualized below:

Stacked Neural Network with Random Forest Confusion Matrix



Indeed, the stacked model had an improved performance in comparison to the dense neural network model alone.

We consequently applied the model onto the #nowplaying dataset in order to classify the genre of the songs:

```
df.predicted_genres.value_counts()
✓ 0.1s
rock_metal        490104
acoustic_folk     418517
pop               309748
dance             255991
electro           202302
ambient           173532
hiphop_reggae     153107
latin             137767
happy_kids         92411
```

Summary

Overall, we must concede that the results of our efforts are less than ideal, but also not a complete failure. The model would have likely performed better if we had additional features available and were not limited to the few features that both datasets shared. Ideally, we would have had audio data, but this would have not been attainable and required handling of an even bigger amount of data and high-end hardware resources, which would have transcended the scope of this project.

In experimenting with different classifier models, we could see big variances in how well the genres could be discerned from one another. For example, all models could classify songs

from the *ambient* category with relative confidence, but struggled with *pop*. A major factor for this is that genre categories are inherently not precisely defined and there is a lot of overlap between the features of various genres. By grouping genres into bigger and broader categories, we could counteract this effect to some extent and significantly improve classification accuracy. So, we can positively emphasize that the Classification Accuracy is well above random levels and considering it was a challenging task to infer genre based on very limited features and without actual audio data, even for an AI classifier.
However, as we deemed the classifier not quite reliable enough, we eventually decided against using it in our recommendation system. Nevertheless, we are convinced that its performance could have reached satisfactory quality if we had been able to acquire more metadata for the song recommendations.

Recommendation System

Recommendation systems are based on algorithms written by analyzing each user's historical data to suggest the most relevant products or items. Although there is a vast variety of recommendation systems, they mostly fall into the categories of content-based, collaborative, or hybrid systems (Figure 10).

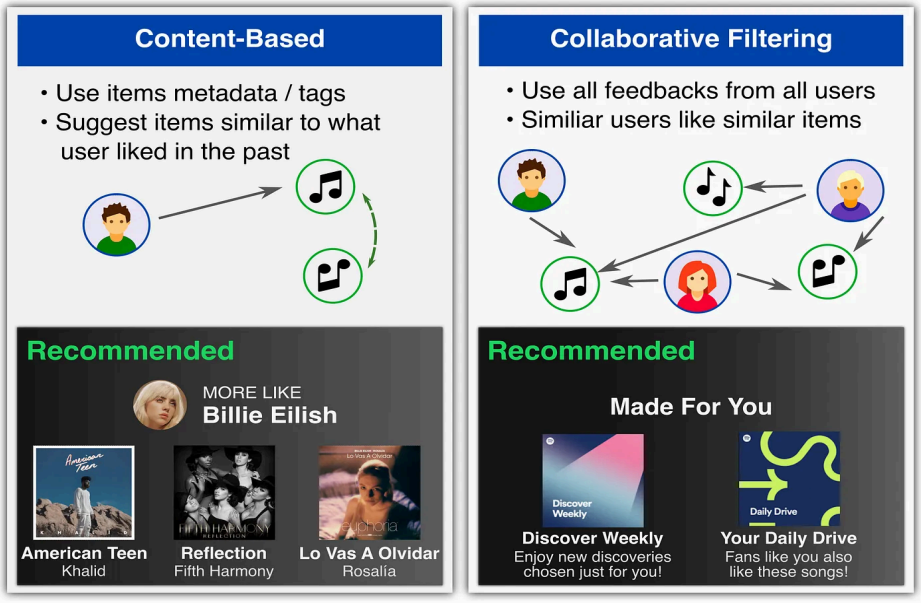


Figure ?. Content-Based and Collaborative Filtering [3]

Collaborative filtering predicts that the user will like items that similar users have liked in the past or the similar peer items will be recommended, based on the historical data of a user. It's a straightforward approach but the biggest problem with this filtering is cold start where there is not enough data for a brand new user.

Content-based filtering selects an item that is similar to the user's query. However, each user is evaluated independently and when there is a new user in the system, the system struggles with problems.
The hybrid method uses metadata of users, as well as user-item interactions, providing more optimal recommendations.

Our approach mainly employs content-based filtering. It consists of two parts: machine learning modeling to predict sentiment values and using cosine similarity to suggest tracks.
Figure 11 summarizes the system we employed. Firstly, nowplaying-RS dataset was utilized and random 5000 rows were sampled as a test dataset. Sentiment binary classification was performed using the best previously identified machine learning model, which was Random Forest classification. Audio features of the rows with the predicted-sentiment 1 were chosen to deploy. In the next step, random users were chosen by excluding the test data set from the now playing-RS. After ensuring the user is not included in our test dataset, we calculated the cosine similarity between the chosen users and the audio features of the predicted row's tracks.

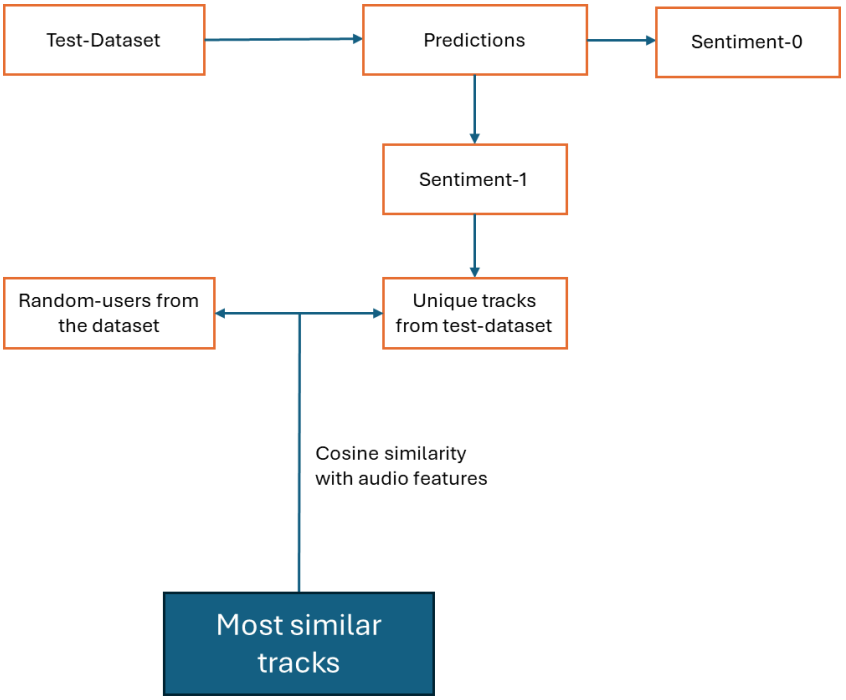


Figure ?. Our approach to recommend tracks

CONCLUSION

In this project, we aimed to develop a robust music recommendation system employing both machine learning techniques and content-based filtering. Preprocessing was pursued with

feature engineering to enhance the modeling. One of the biggest challenges was insufficient features in both datasets. While the nowplaying-RS dataset did not include the genres and track names, user-ids were missing in the Spotify dataset. Another challenge encountered was the imbalance in data points within the nowplaying-RS dataset and the complexity arising from the diverse genres present in the Spotify dataset, making it difficult to classify tracks accurately. Our ML models were predicting the appreciation of users based on the context and content features nowplaying-RS and classifying the genres of the tracks in Spotify.
Although we implemented various types of machine learning models including deep learning for both nowplaying-RS and Spotify datasets, further improvements are possible to increase the performance of the system and make a better recommendation. However, it should not be forgotten that we were limited by time and the calculation capacity of our computers.

References

[1] - Poddar, A., & Schedl, M. (2018). #nowplaying-RS: A New Benchmark Dataset for Building Context-Aware Music Recommender Systems . In *Proceedings of the Sound and Music Computing Conference 2018*. Retrieved from https://evazangerle.at/publication/poddar-smc-2018/poddar-smc-2018.pdf

[2] - Stäbler M. , (2022, February 2). Music Genre Prediction by Extracted Audio Features and Comparison to the Spotify API. Towards AI. https://pub.towardsai.net/music-genre-prediction-by-extracted-audio-features-and-comparison-to-the-spotify-api-ebf82232e31a

[3] - Towards Data Science. (n.d.). Recommender systems: A complete guide to machine learning models. *Towards Data Science*. Retrieved June 23, 2024, from https://towardsdatascience.com/recommender-systems-a-complete-guide-to-machine-learning-models-96d3f94ea748