UNIVERSITY OF PISA

Artificial Intelligence and Data Engineering

Data Mining and Machine Learning

Music popularity across borders

Feyzan Çolak, Noemi Cherchi

Academic Year 2023/2024

Contents

Contents **1**

1 Introduction **2**
1.1 Objective . . . 2
1.2 Dataset . . . 2
1.3 Audio Features . . . 3

2 Data Preprocessing **4**
2.1 Dataset Overview . . . 4
2.2 Grouping Regions . . . 5
2.3 Outlier Detection . . . 6
2.4 Encoding Regional Groups . . . 9

3 Data Visualization **10**

4 Algorithms **17**
4.1 Logistic Regression . . . 18
4.2 Decision Tree . . . 19
4.3 Random Forest . . . 20
4.4 Naïve Bayes . . . 22
4.5 K-Nearest Neighbors . . . 24

5 Literature Review **26**

6 Results **27**
6.1 Results of the Models . . . 27
6.2 Global Feature Importance Analysis . . . 28
6.3 Regional Feature Importance Analysis . . . 29

7 Conclusion **35**

Introduction

The music industry plays a significant role in today's globalized world as it has been growing rapidly over the years. It also faces the unique challenge of addressing diverse audiences across different countries. Each country has different tastes in music which are shaped by lots of factors. For a record label, it is crucial to understand these preferences for a successful music promotion and marketing, for example to decide which song of an album should become a single.
The ability to predict the popularity of a song before it is released can provide a significant competitive privilege.
More on the code can be found on the GitHub repository at the following link.

Objective

This project has two main goals: one is to predict the popularity of a song based on its audio features, and the other is to identify the most important audio features that contribute to a song's popularity in different countries.
In particular, specific audio features have been taken in consideration, such as danceability, key, loudness, mode, speechiness and acousticness. By leveraging Data Mining and Machine Learning Techniques, this project will provide insights into the factors determining the popularity of songs in different countries.

Dataset

The final dataset used in this project is obtained by combining two different and extensive datasets.

The first dataset was taken from the following Kaggle link.

This dataset is called Spotify Charts and contains information about the most popular songs in different countries. It is a complete dataset of all the "Top 200" and "Viral 50" charts published globally by Spotify. Spotify publishes a new chart every 2-3 days. This is Spotify's entire collection since January 1, 2017. The included key details are the song's name, artist, spotify id, rank, and the number of streams in each country. This dataset is helpful to undertand how musical preferences change across different parts of the world.

The second dataset was created by individuating all the unique song IDs from the first dataset and using the Spotify API to gather the audio features of more than 200K songs. The audio features that are used in this project are $danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, duration(ms), timesignature$. These features are essential for analyzing the musical elements that contribute to a song's popularity in each country.

Before starting the preprocessing phase, the two datasets were merged togethere by the Spotify ID. This ensured that the audio features from the Spotify API were correctly associated with the corresponding songs and their popularity metrics in different countries.

Audio Features

Feature	Data Type	Description
Danceability	float	Describes how suitable a track is for dancing based on a combination of musical elements. A value of 0.0 is least danceable and 1.0 is most danceable.
Energy	float	Measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale.
Key	integer	The key the track is in. Integers map to pitches using standard Pitch Class notation. If no key was detected, the value is -1.
Loudness	float	The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. It is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typically range between -60 and 0 dB.
Mode	integer	Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.
Speechiness	float	Speechiness detects the presence of spoken words in a track. Speech-like the recording (e.g. audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.
Acousticness	float	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
Instrumentalness	float	Predicts whether a track contains no vocals. "Ooh" and "aah" sounds are treated as instrumental in this context. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.
Liveness	float	Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.
Valence	float	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive, while tracks with low valence sound more negative.
Tempo	float	The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.
Duration (ms)	integer	The duration of the track in milliseconds.
Time Signature	integer	An estimated time signature. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure). The time signature ranges from 3 to 7 indicating time signatures of "3/4", to "7/4".

Data Preprocessing

In this section, we describe the Data Preprocessing steps performed on the merged dataset. The preprocessing phase is crucial in preparing the data for further analysis by cleaning it and transforming it into a suitable format for modeling. This is done in Python using the Pandas library.

Dataset Overview

We start by loading and exploring the dataset using the following commands:

```
# Display the first few rows of the dataset
df.head()

# Display basic information about the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26172626 entries, 0 to 26172625
Data columns (total 26 columns):
 #   Column            Dtype
---  ------            -----
 0   title             object
 1   rank              int64
 2   date              object
 3   artist            object
 4   track_id          object
 5   region            object
 6   chart             object
 7   trend             object
 8   streams           float64
 9   danceability      float64
 10  energy            float64
 11  key               int64
 12  loudness          float64
 13  mode              int64
 14  speechiness       float64
 15  acousticness      float64
 16  instrumentalness  float64
 17  liveness          float64
 18  valence           float64
 19  tempo             float64
...
 24  duration_ms       int64
 25  time_signature    int64
dtypes: float64(10), int64(5), object(11)
memory usage: 5.1+ GB
```

Figure 2.1: Basic information.

```
# Summary statistics of the dataset
df.describe()
```

	rank	streams	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence
count	2.617263e+07	2.032132e+07	2.617263e+07	2.617263e+07	2.617263e+07	2.617263e+07	2.617263e+07	2.617263e+07	2.617263e+07	2.617263e+07	2.617263e+07	2.617263e+07
mean	8.092288e+01	5.526276e+04	6.853916e-01	6.434528e-01	5.333580e+00	-6.256671e+00	5.730547e-01	1.097356e-01	2.579785e-01	1.427076e-02	1.694213e-01	5.228329e-01
std	5.918614e+01	2.095901e+05	1.376300e-01	1.672001e-01	3.635615e+00	2.531198e+00	4.946342e-01	1.001090e-01	2.467141e-01	8.827835e-02	1.291109e-01	2.232246e-01
min	1.000000e+00	1.001000e+03	0.000000e+00	0.000000e+00	0.000000e+00	-6.000000e+01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	2.900000e+01	3.546000e+03	6.000000e-01	5.350000e-01	2.000000e+00	-7.492000e+00	0.000000e+00	4.270000e-02	5.620000e-02	0.000000e+00	9.340000e-02	3.490000e-01
50%	6.700000e+01	9.566000e+03	7.040000e-01	6.620000e-01	6.000000e+00	-5.867000e+00	1.000000e+00	6.710000e-02	1.770000e-01	0.000000e+00	1.200000e-01	5.220000e-01
75%	1.310000e+02	3.541000e+04	7.840000e-01	7.700000e-01	8.000000e+00	-4.545000e+00	1.000000e+00	1.360000e-01	3.960000e-01	2.910000e-05	1.970000e-01	6.970000e-01
max	2.000000e+02	1.974970e+07	9.880000e-01	1.000000e+00	1.100000e+01	3.795000e+00	1.000000e+00	9.660000e-01	9.960000e-01	9.930000e-01	1.000000e+00	9.990000e-01

Figure 2.2: Summary statistics of the dataset.

Missing Values Detection

We check for missing values in the dataset using the following methods:

```
# Check for missing values in each column
df.isna().sum(axis=0)
```

The columns with missing values are: streams, title and artist. These columns are not useful for the analysis and can be removed.

Other columns such as date, chart, trend, etc... don't contribute to the analysis and can be removed.

After removing these columns, we check again for missing values to ensure a clean dataset and confirm that there are no missing values left. This allowed us to reduce the number of columns from 26 to 16.

Grouping Regions

Countries are then grouped into 11 geographical regions: Northern Europe, Eastern Europe, Western Europe, Southern Europe, North America, Latin America, East Asia, South Asia, Middle East, Oceania, and Africa. A mapping function is used to assign each country to a group.

```
# Mapping of countries to regions
regions_mapping = {
    'Latin America': ['Argentina', 'Bolivia', ...],
    'North America': ['Canada', 'United States'],
    'Northern Europe': ['Denmark', 'Finland', ...],
    ...
}

# Update the 'region'
for region, countries in regions_mapping.items():
    df.loc[df['region'].isin(countries), 'region'] = region
```

In cases where the same song appears more than once in a region, we calculate the mean rank and add a new column `frequency` to represent how many times the song appears in the chart.

```
# Group by 'track_id' and 'region' to calculate the mean of 'rank' and the frequency (count of occur
df_grouped = df_filtered.groupby(['track_id', 'region']).agg({
    'rank': 'mean',    # Mean of rank
    'track_id': 'size' # Count of occurrences
}).rename(columns={'rank': 'rank_mean', 'track_id': 'frequency'}).reset_index()
```

The chart dataset initially had about 20 million rows. By identifying the unique songs IDs for each region, we managed to decrease to 200K rows.

We visualize the distribution of songs for each regional group.

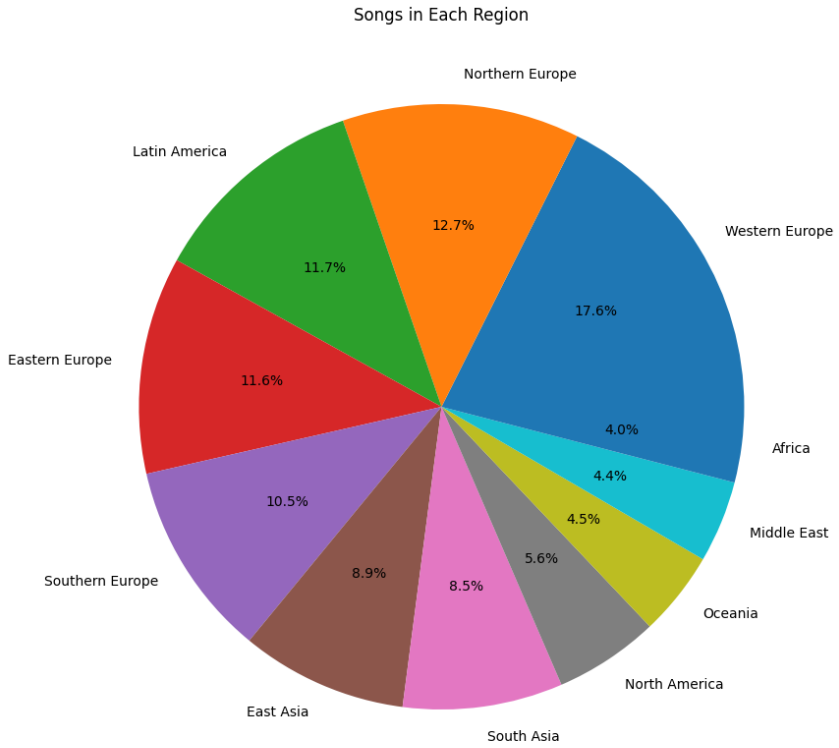


Figure 2.3: Distribution of songs across regions.

Now the song IDs can be removed since they are not needed for further analysis.

Outlier Detection

We plot box plots for each feature to visualize the distribution and detect outliers. Based on this analysis, we decide whether to remove outliers.

```
# List of features
features = ['danceability', 'energy', 'key', 'loudness', 'mode', 'speechiness', 'acousticness',
'instrumentalness', 'liveness', 'valence', 'tempo', 'duration_ms', 'time_signature']

# Create box plots for each feature
plt.figure(figsize=(15, 10))
for i, feature in enumerate(features, 1):
    plt.subplot(4, 4, i)
    sns.boxplot(x=df_final[feature])
    plt.title(f'Box Plot of {feature}')
plt.tight_layout()
plt.show()
```

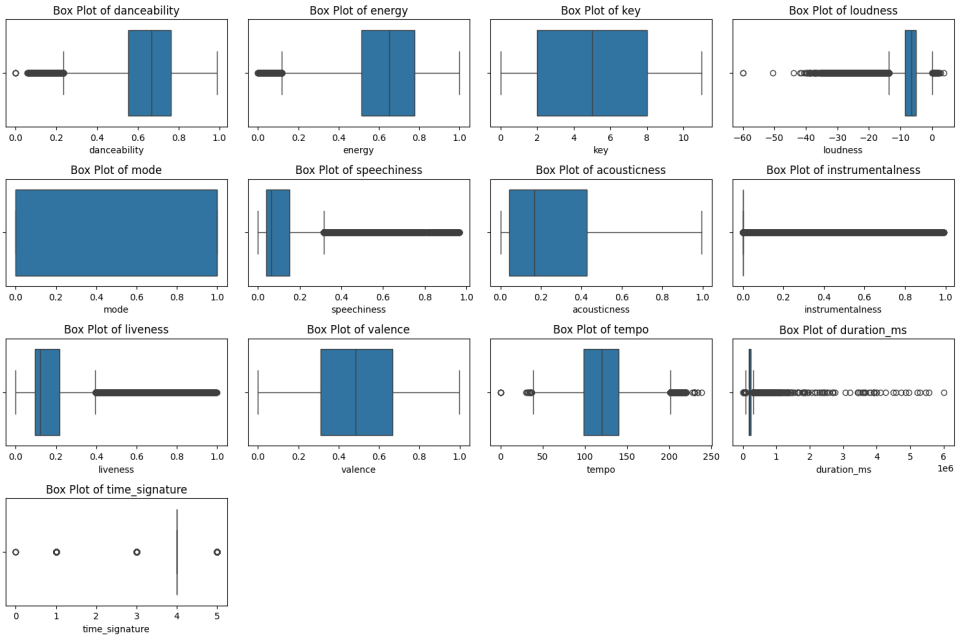


Figure 2.4: Box plot of audio features.

The box plots in Figure 2.4 show the distribution of various features in the dataset and help identify potential outliers.

- **Loudness**: The loudness feature has numerous outliers on the lower end, extending below -40 dB. This suggests that some tracks in the dataset have exceptionally low loudness levels, possibly due to poor audio quality or artistic choice.
- **Speechiness and Instrumentalness**: Both of these features exhibit a long tail of outliers. Speechiness has many values close to 0, with a few outliers extending towards 1, indicating tracks that are more spoken-word focused. Instrumentalness shows a dense concentration near 0 but also has numerous outliers, possibly representing fully instrumental tracks.
- **Liveness, and Valence**: Liveness shows some outliers, particularly above 0.8, which could represent live recordings. Valence, indicating the musical positivity, shows a broad range without many extreme outliers.
- **Tempo and Duration (in ms)**: Tempo has several outliers at the higher end above 200 BPM, while duration has many high outliers above 4 million milliseconds (about 66 minutes). These extreme values in duration suggest that some tracks are abnormally long, potentially representing extended live performances or podcasts.
- **Time Signature**: This feature contains categorical values, and outliers are represented by infrequent time signatures (e.g., 1, 3, and 5), which are less common in modern music compared to the standard 4/4 time signature.

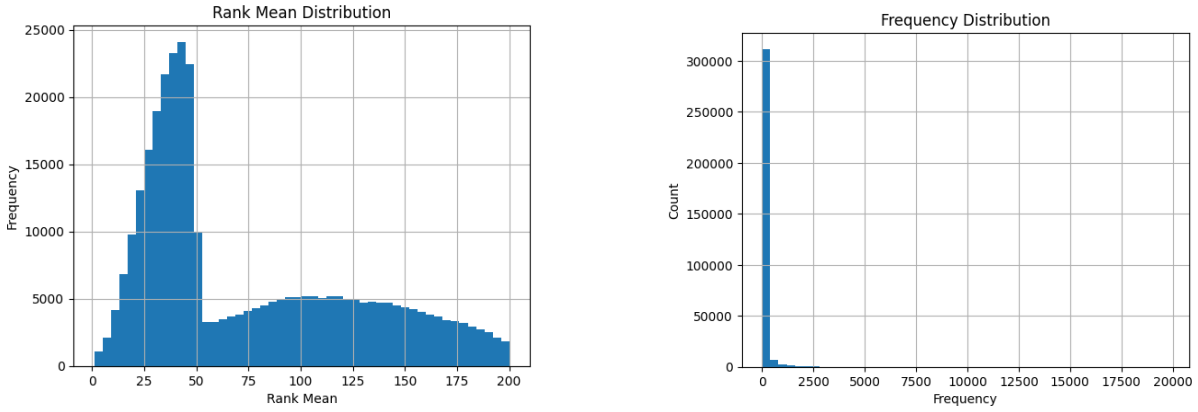
Outliers can potentially distort the analysis and the behavior of models.

To remove outliers in a more secure way, we use the Mahalanobis distance to detect and eliminate extreme values.

```
# Subset of the data for the relevant features
data_features = df_final[features]

# Mean and covariance of the features
mean = np.mean(data_features, axis=0)
cov_matrix = np.cov(data_features.values.T)
inv_cov_matrix = np.linalg.inv(cov_matrix)

# Function to compute Mahalanobis distance for each point
def mahalanobis_distance(row, mean, inv_cov_matrix):
    return mahalanobis(row, mean, inv_cov_matrix)

# Apply the function to each row of the dataset
data_features['mahalanobis_dist'] = data_features.apply(lambda row: mahalanobis_distance(row,
mean, inv_cov_matrix), axis=1)

# Set threshold using chi-square distribution (95% confidence level)
threshold = chi2.ppf(0.95, df=data_features.shape[1]-1)

# Identify outliers (those with Mahalanobis distance greater than the threshold)
data_features['outlier'] = data_features['mahalanobis_dist'] > threshold
```

The Mahalanobis distance is a useful metric for detecting outliers in multivariate data. By calculating the distance of each data point from the mean in a multidimensional space, we can identify observations that are significantly different from the rest of the dataset. In this case, the threshold is set based on the chi-square distribution to ensure a 95% confidence level. The number of outliers is then counted to assess the impact of removing them from the dataset. We found that the number of outliers is 61, which is a small fraction of the total dataset, so they can be removed without significantly affecting the analysis.

This is the distribution of the rank mean and of the frequency of the songs in the dataset:



(a) Distribution of the rank mean.

(b) Distribution of the frequency of songs.

Figure 2.5: Comparison of the rank mean and frequency distributions.

The **rank mean** reflects the average position of a song on the charts. Most songs have a rank mean below 50, with a sharp decline after this point. This suggests that a large proportion of songs in the dataset perform well in terms of chart rankings. However, there is a tail of songs with higher rank means, indicating those that appear lower in the charts.

The distribution of the **frequency** of songs in the charts is heavily skewed to the left, with the majority of songs appearing in the charts only a few times. A small number of songs have appeared much more frequently, though this occurrence is rare, as indicated by the long tail of the distribution. This suggests

that while most songs have limited presence on the charts, a few songs maintain a significant chart presence over time.

Labeling Songs

We define a threshold to determine whether a song is popular or not. A binary column `Popular` is added where 0 represents non-popular songs and 1 represents popular songs:

```
# Calculate percentiles for rank_mean and frequency
rank_threshold = df['rank_mean'].quantile(0.4)  # 40th percentile
frequency_threshold = df['frequency'].quantile(0.6)  # 60th percentile

# Apply the thresholds
df['popular'] = 0
df.loc[(df['rank_mean'] < rank_threshold) & (df['frequency'] > frequency_threshold), 'popular'] = 1
```

We discovered that about 80% of the songs in the dataset are non-popular, while the remaining 20% are popular.

We can delete the rank_mean and frequency columns, since they're not needed for further analysis.

Encoding Regional Groups

We encode the `Region` column into numerical values:

```
df['region']=le.fit_transform(df['region'])
label_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
print(label_mapping)
```

The regional groups are encoded as follows:

```
{'Africa': 0, 'East Asia': 1, 'Eastern Europe': 2, 'Latin America': 3, 'Middle East': 4,
'North America': 5, 'Northern Europe': 6, 'Oceania': 7, 'South Asia': 8,
'Southern Europe': 9, 'Western Europe': 10}
```

The final dataset contains cleaned and transformed features, with only numerical values, ready for analysis. The features include popularity, encoded regional groups and the audio features.

Data Visualization

The final dataset has 189297 rows × 15 columns. The columns are as follows:

danceability	**energy**	**key**	**loudness**	**mode**
speechiness	**acousticness**	**instrumentalness**	**liveness**	**valence**
tempo	**duration_ms**	**time_signature**	**region**	**popular**

Table 3.1: Column names in the dataset

By visualizing and exploring the final dataset, we can underline the differeces with the raw dataset. This shows that the preprocessing phase was fundamental to obtain a clean dataset.

```
# Display the first few rows of the dataset
df.head()

# Display basic information about the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 189297 entries, 0 to 189296
Data columns (total 15 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   region            189297 non-null  int64
 1   danceability      189297 non-null  float64
 2   energy            189297 non-null  float64
 3   key               189297 non-null  int64
 4   loudness          189297 non-null  float64
 5   mode              189297 non-null  int64
 6   speechiness       189297 non-null  float64
 7   acousticness      189297 non-null  float64
 8   instrumentalness  189297 non-null  float64
 9   liveness          189297 non-null  float64
 10  valence           189297 non-null  float64
 11  tempo             189297 non-null  float64
 12  duration_ms       189297 non-null  int64
 13  time_signature    189297 non-null  int64
 14  popular           189297 non-null  int64
dtypes: float64(9), int64(6)
memory usage: 21.7 MB
```

Figure 3.1: Basic information.

One of the first thing to notice is how the memory usage has been reduced significantly. This is due to the removal of columns that were not useful for the analysis. In fact, the number of columns has been reduced from 26 to 15.

```
# Summary statistics of the dataset
df.describe()
```

	count	mean	std	min	25%	50%	75%	max
region	189297.0	5.571758	3.288120	0.0	3.0000	6.000000	9.000000	10.000
danceability	189297.0	0.649613	0.154390	0.0	0.5500	0.667000	0.764000	0.988
energy	189297.0	0.632489	0.195288	0.0	0.5100	0.652000	0.779000	1.000
key	189297.0	5.335055	3.604398	0.0	2.0000	6.000000	8.000000	11.000
loudness	189297.0	-7.405815	3.389996	-60.0	-8.8220	-6.818000	-5.249000	3.795
mode	189297.0	0.563068	0.496008	0.0	0.0000	1.000000	1.000000	1.000
speechiness	189297.0	0.118607	0.119773	0.0	0.0391	0.063700	0.157000	0.966
acousticness	189297.0	0.283864	0.275255	0.0	0.0511	0.189000	0.458000	0.996
instrumentalness	189297.0	0.049584	0.178130	0.0	0.0000	0.000002	0.000341	0.993
liveness	189297.0	0.179596	0.143091	0.0	0.0959	0.122000	0.216000	0.996
valence	189297.0	0.499454	0.232413	0.0	0.3180	0.493000	0.680000	0.999
tempo	189297.0	121.429386	28.596888	0.0	98.9270	120.041000	139.980000	238.816
duration_ms	189297.0	214141.510304	85004.396711	29668.0	177458.0000	205087.000000	238147.000000	6011575.000
time_signature	189297.0	3.951114	0.328212	0.0	4.0000	4.000000	4.000000	5.000
popular	189297.0	0.172475	0.377794	0.0	0.0000	0.000000	0.000000	1.000

Figure 3.2: Summary statistics of the dataset.

The popular songs represent about the 20% of the dataset.

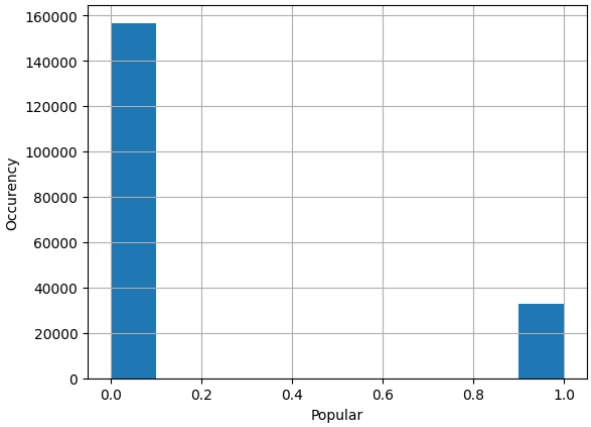


Figure 3.3: Distribution of popular songs.

Each audio feature is distributed in this way:

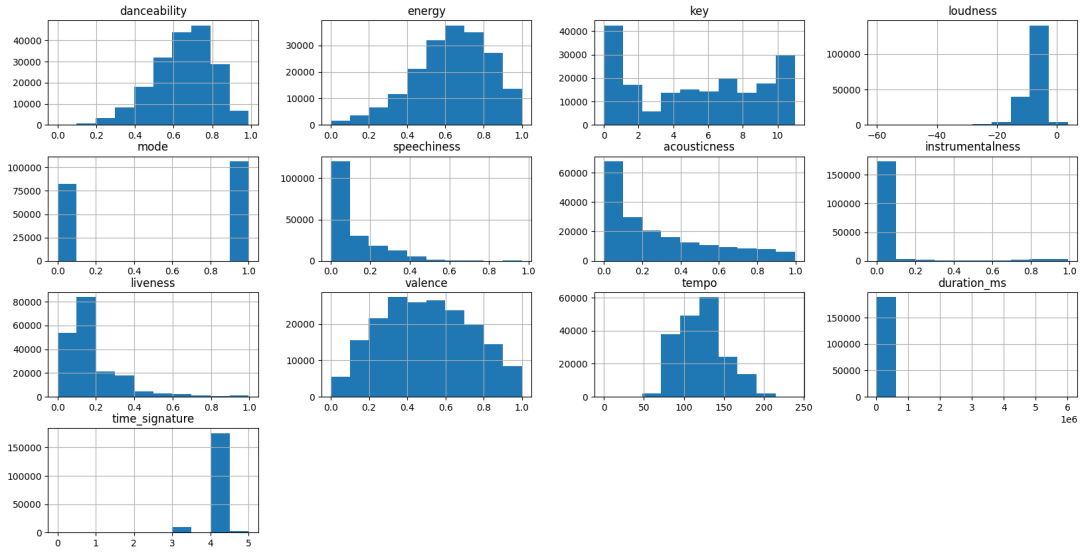


Figure 3.4: Distribution of audio features.

This is the boxplot of the audio features:

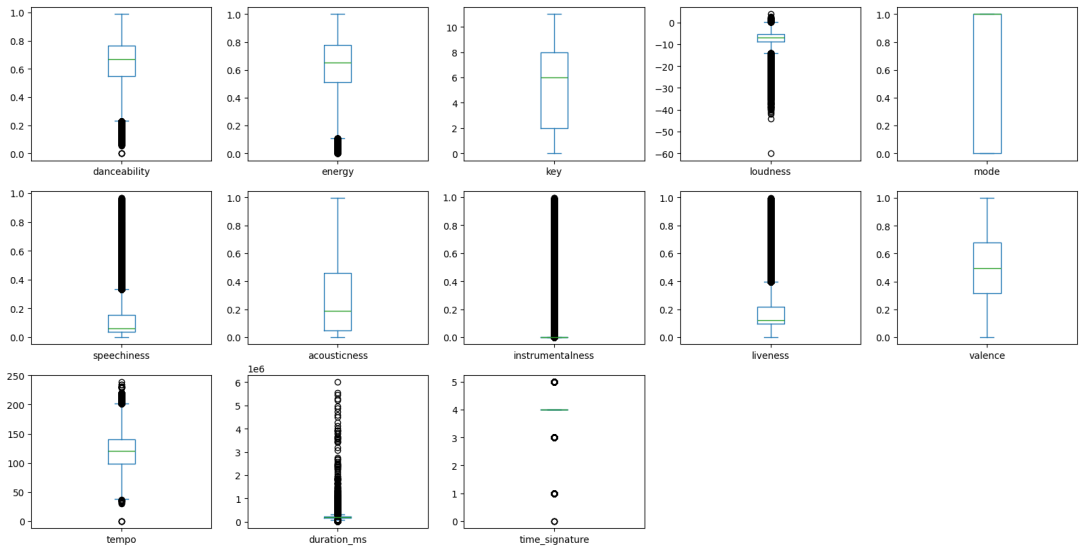


Figure 3.5: Boxplot of audio features.

Many features have outliers (especially loudness, instrumentalness, speechiness, and duration_ms), which means they have songs that are very different from the "average" in terms of these attributes. Features like mode and time_signature have almost no variation, thus they may not be useful for predicting the target variable (popularity).

This is the heatmap of the correlation matrix:

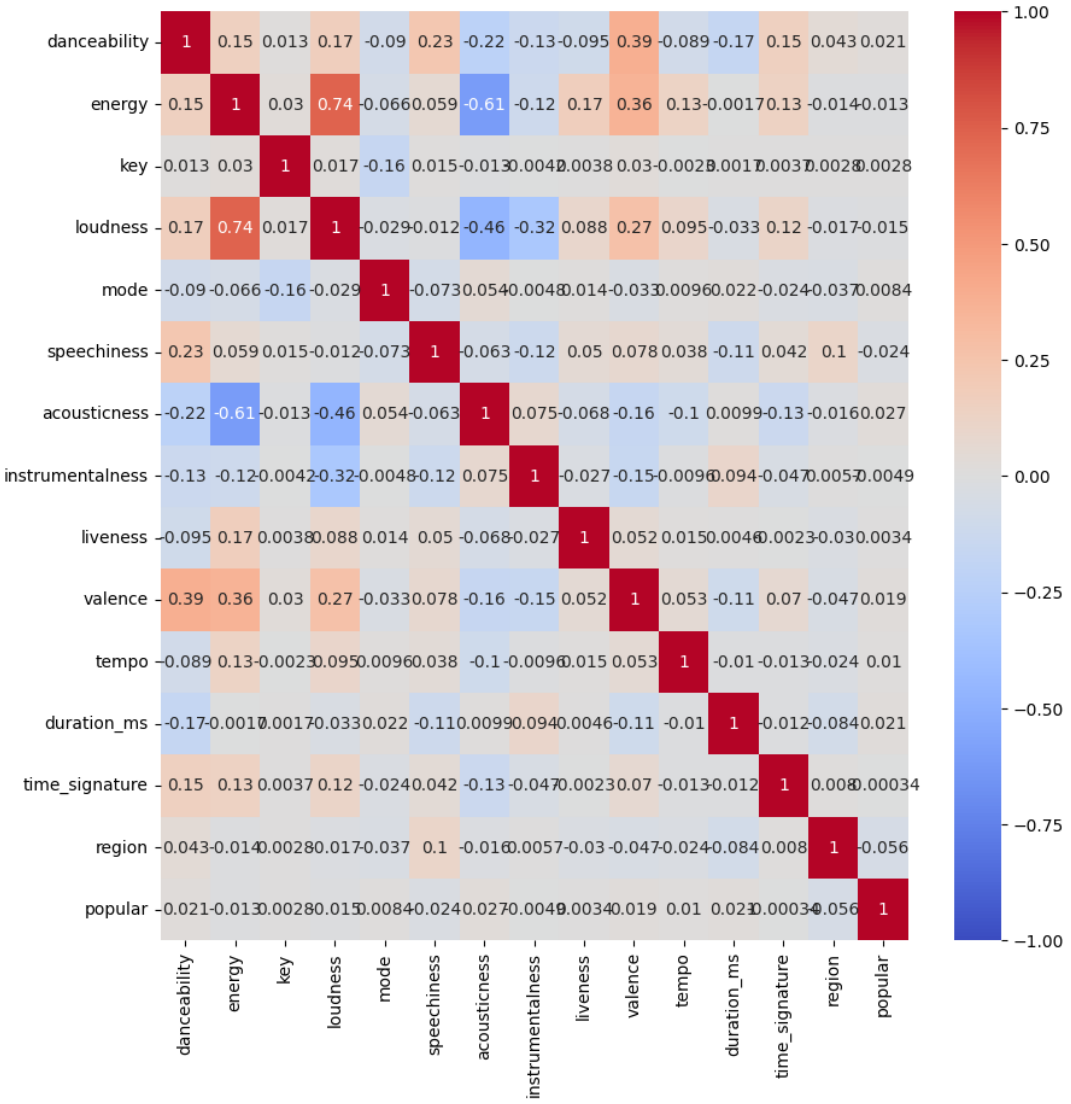


Figure 3.6: Heatmap of the correlation matrix.

The heatmap shows that the audio features have a mix of positive and negative correlations with each other. The most notable correlations are:

- Energy and Loudness: strong positive correlation (0.74). Loud songs tend to also have high energy, which makes sense musically.
- Energy and Acousticness: strong negative correlation (-0.61). Acoustic songs tend to have lower energy levels, which aligns with expectations as acoustic songs are often more mellow.

This also shows us that no single feature stands out as highly predictive of song popularity based on these correlations, which suggests that popularity might be determined by a complex interaction of features rather than individual attributes.

We can also visualize the popularity of each feature in the different regions:

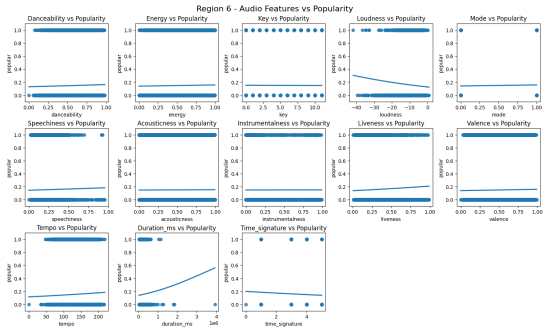


Figure 3.7: Popularity in Northern Europe

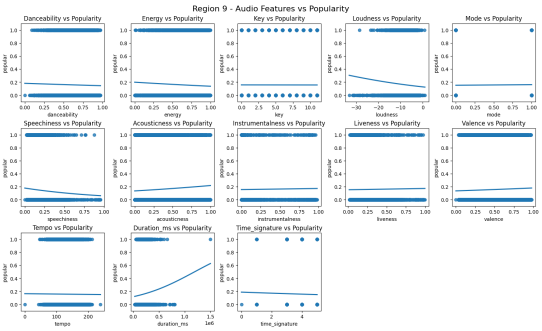


Figure 3.8: Popularity in Southern Europe

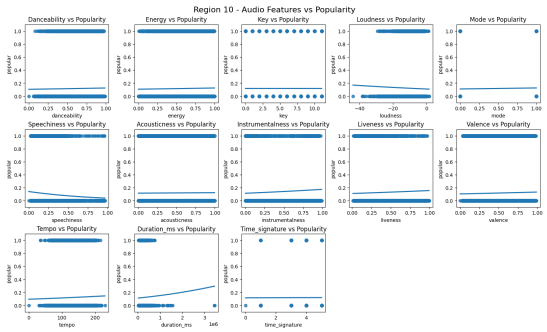


Figure 3.9: Popularity in Western Europe

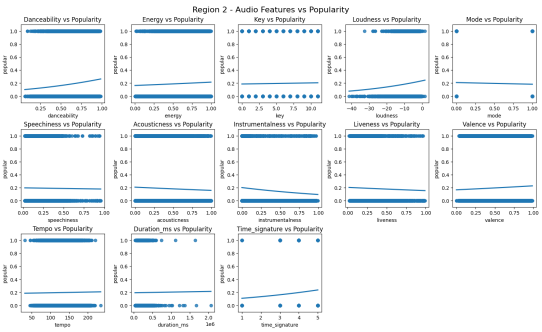


Figure 3.10: Popularity in Eastern Europe

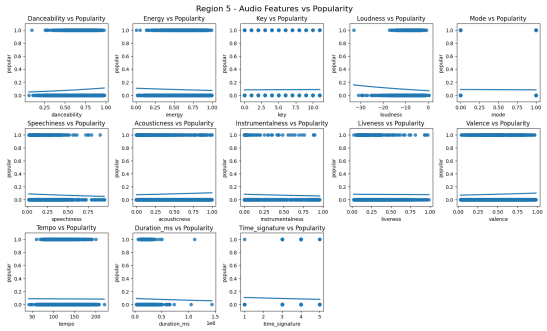


Figure 3.11: Popularity in North America

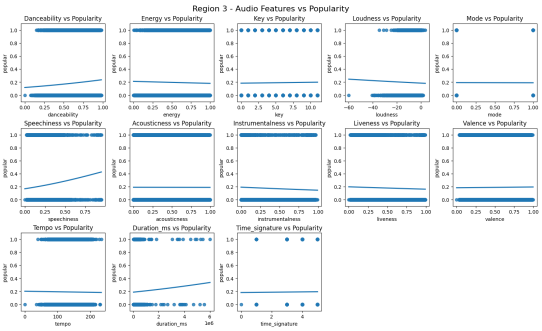


Figure 3.12: Popularity in Latin America

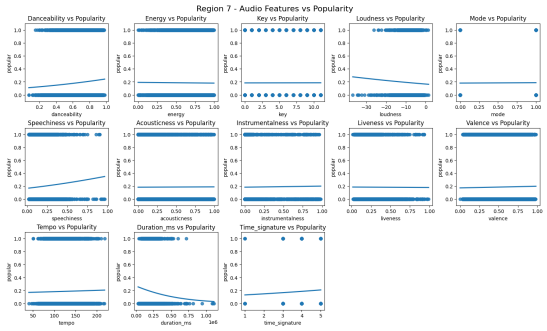


Figure 3.13: Popularity in Oceania

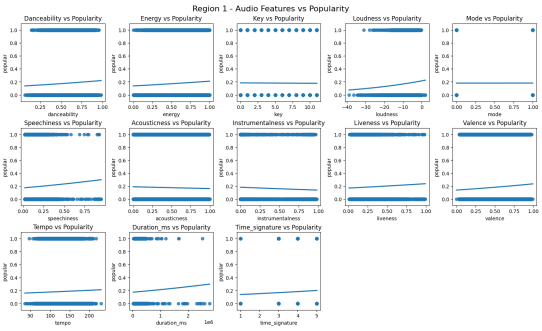


Figure 3.14: Popularity in East Asia

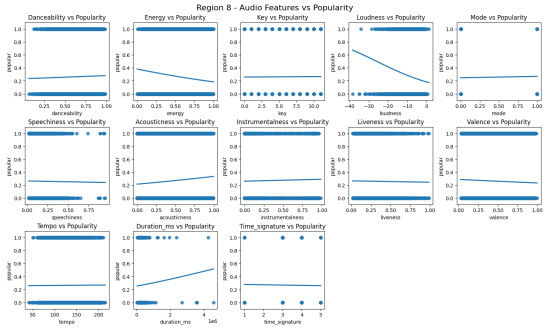


Figure 3.15: Popularity in South Asia

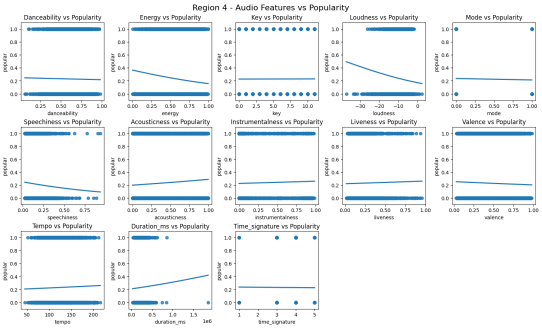


Figure 3.16: Popularity in the Middle East

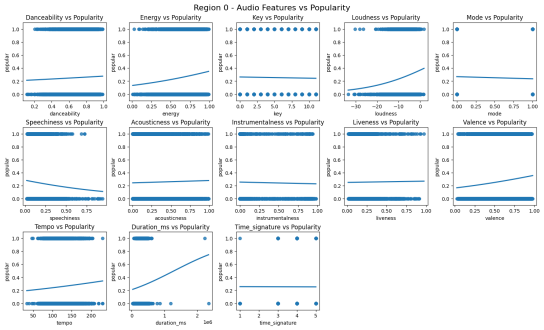


Figure 3.17: Popularity in Africa

This allows us to understand what features make a song popular or not in the different regions.

- **Northern Europe**: energy and danceability
- **Southern Europe**: acousticness and valence
- **Eastern Europe**: danceability
- **North America**: danceability and acousticness
- **Latin America**: speechiness and danceability
- **Oceania**: danceability and energy
- **East Asia**: valence
- **South Asia**: danceability and acousticness
- **Middle East**: danceability
- **Africa**: loudness, valence, acousticness, duration and tempo

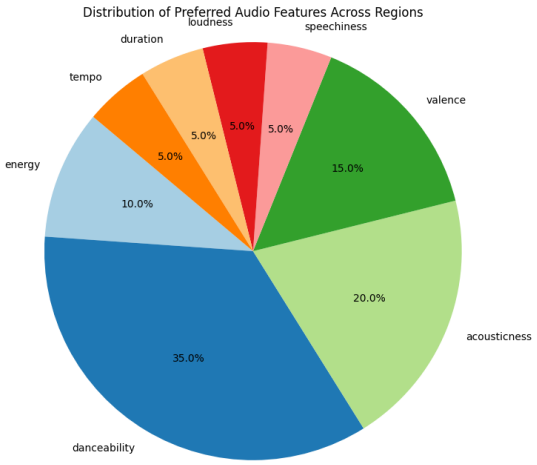


Figure 3.18: Global preferences

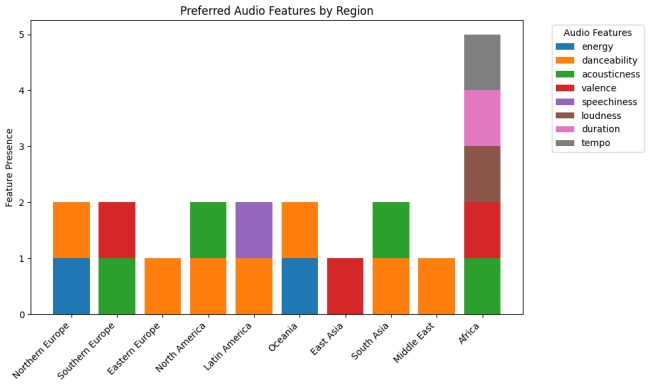


Figure 3.19: Regional preferences

These images show the regional preferences only based on the dataset distribution. In the next pages, we'll illustrate the results of the different Machine Learning models used.

Algorithms

To further investigate the relationship between audio features, geographical regions, and song popularity, various machine learning algorithms were applied to the dataset. In particular, the same algorithms that are used in the literature review have been used. This approach allowed us to compare the algorithms we used not only amongst them, but also with the ones used in the literature review. In addition to those algorithms, also K-Nearest Neighbors (KNN) was used.
The algorithms include:

- Logistic Regression
- Decision Tree
- Random Forest
- Naive Bayes
- K-Nearest Neighbors

The initial steps were the same for all algorithms.

Implementation Steps

- **Data Preparation:** The dataset is split into X and Y: X containing the audio features and region, and Y containing the popularity column. The popularity column is used as the target variable. Then the dataset is split into training and testing sets using an 80/20 partition.
- **Feature Scaling:** The features are scaled using the *StandardScaler* from the *sklearn.preprocessing* library. The data is normalized to ensure that all feautres contribute equally to the distance calculation involved in the model, which helps improve the convergence of the algorithm.
- **Class Imbalance:** The dataset is checked for class imbalance, and if necessary, techniques such as oversampling or undersampling are applied to balance the classes.
- **Model Training:** The algorithm models were trained using the scaled training set. The model was configured with a maximum iteration limit of 2000 and class weight balanced to give equal importance to both classes during the training.
- **Cross-Validation:** To evaluate the model's performance more robustly, 5-fold cross-validation was performed. This technique involves splitting the data into five subsets (folds) and training the model on four folds while validating it on the remaining fold. This process is repeated five times, each time using a different fold for validation.
- **Predictions:** Predictions were made on the scaled test set, and probabilities were calculated for class membership.
- **Model Evaluation:** The model was evaluated using the confusion matrix, classification report, which included metrics such as precision, recall, and F1-score.
- **Feature Importance:** The feature importance of the model was determined by examining the coefficients of the models. Both feature importance for the whole dataset and feature importance for each specific region has been applied. As a result, dominant features to determine the popularity of the songs were found.
- **Regional Analysis:** Regional analysis was performed to better understand how different audio features contribute to make a song popular across various regions.

Logistic Regression

Logistic regression is a supervised machine learning algorithm that is used to compute classification problems with a binary output. It is based on the idea of modeling the probability of a certain class or event. Unlike linear regression, which predicts continuous values, logistic regression predicts categorical outcomes by applying a logistic function to the input data. In this case, it's used to predict whether a song will be popular or not, depending on the region and the audio features of the song. The output is either 0 or 1, whith 0 representing the song most probably will not be popular and 1 representing the song most probably will be popular.
To address potential class imbalance in the dataset, the *Synthetic Minority Over-sampling Technique (SMOTE)* was applied, which created synthetic samples of the minorty class (popular songs).

Accuracy: **85.42%**

Classification Report

	Precision	**Recall**	**F1-score**	**Support**
0	0.85	1.00	0.92	55340
1	0.00	0.00	0.00	9613
Accuracy		0.85		
Macro avg	0.43	0.50	0.46	64953
Weighted avg	0.73	0.85	0.78	64953

Table 4.1: Classification Report

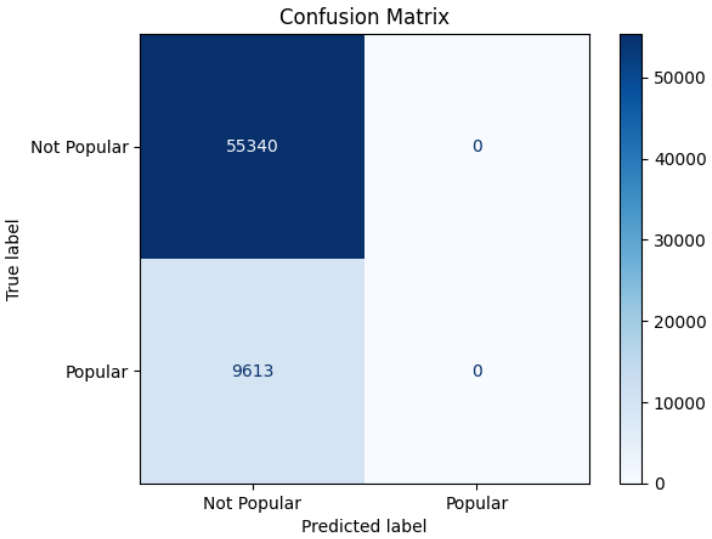


Figure 4.1: Confusion Matrix of Logistic Regression Model

The strong accuracy is due to the strong class imbalance. In fact, the confusion matrix shows that the model is not able to predict at all when a song will be popular. This happens despite having taking into consideration the class imbalance by using oversampling with the SMOTE technique.

Decision Tree

Decision Tree is a supervised machine learning algorithm that is used for both classification and regression tasks. It works by recursively partitioning the data into subsets based on the values of the input features. The goal is to create a model that predicts the value of the target variable by learning simple decision rules inferred from the data features. Decision trees are easy to interpret and visualize, making them popular for exploratory data analysis and decision-making tasks. In this project, Decision Tree was used with the SMOTE technique to address the class imbalance.

After splitting the dataset into test and train sets and fitting the model, the Decision Tree model was evaluated using the following metrics: accuracy, confusion matrix and classification report.

Accuracy: **68.64%**

Classification Report

	Precision	**Recall**	**F1-score**	**Support**
0	0.88	0.73	0.80	55340
1	0.21	0.42	0.28	9613
Accuracy			0.69	
Macro avg	0.55	0.58	0.54	64953
Weighted avg	0.78	0.69	0.72	64953

Table 4.2: Classification Report

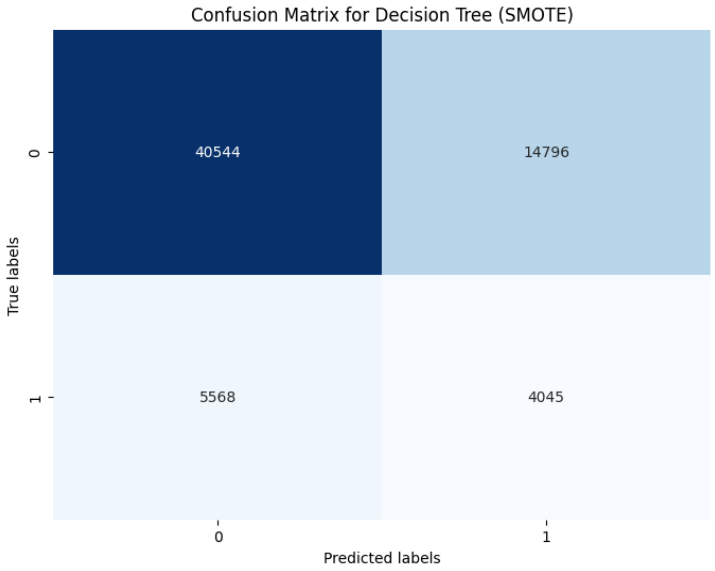


Figure 4.2: Confusion Matrix of Decision Tree Model

The Decision Tree model has a pretty good accuracy and the confusion matrix shows a better balance between the number of true positives and true negatives compared to Logistic Regression. This indicates that the Decision Tree model is better at predicting both popular and non-popular songs, but still the results could be improved.

The Decision Tree Classifier can be shown:

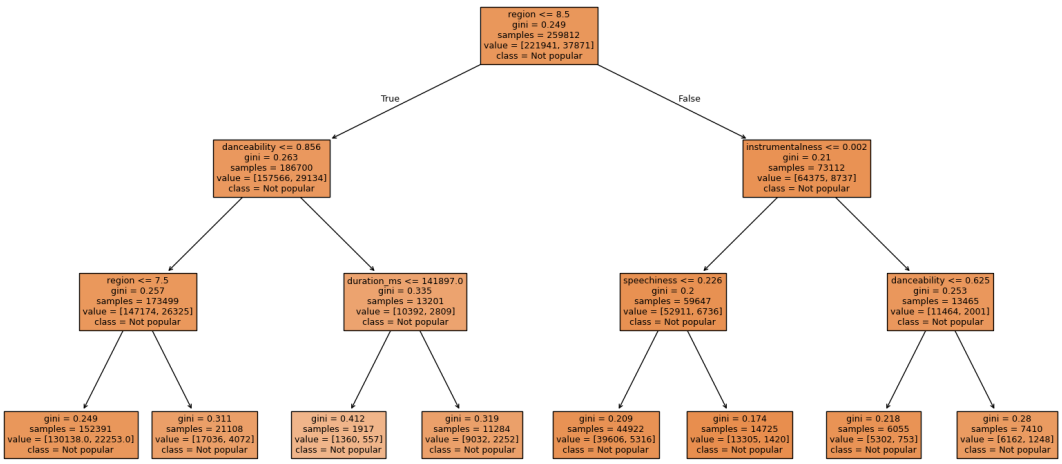


Figure 4.3: Decision Tree Classifier

Random Forest

Random Forest is an ensemble learning method that constructs a multitude of decision trees during training and outputs the mode of the classes as the prediction of the individual trees. It can be used for both classification and regression tasks. Random Forest is known for its robustness, scalability, and ability to handle high-dimensional data with ease. In this project, Random Forest was used with the SMOTE technique to address the class imbalance.
The concept of Random Forest is based on Bagging (Bootsrap Aggregating) which is a technique used to reduce the variance of the model by averaging the predictions of multiple models. Random Forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Mathematically, if $T_1, T_2, \ldots, T_B$ are the B decision trees in the forest, and $h(x)$ is the output prediction of a single tree for input x, the Random Forest prediction for classification is:

$$\hat{y} = \text{mode}\left(T_1(x), T_2(x), \ldots, T_B(x)\right)$$

For the regression, the final prediction is the average:

$$\hat{y} = \frac{1}{B}\sum_{i=1}^{B} T_i(x)$$

Accuracy: **83.02%**

Classification Report

	Precision	Recall	F1-score	Support
0	0.82	0.84	0.83	55340
1	0.84	0.82	0.83	55443
Accuracy			0.83	
Macro avg	0.83	0.83	0.83	110913
Weighted avg	0.83	0.83	0.83	110913

Table 4.3: Classification Report

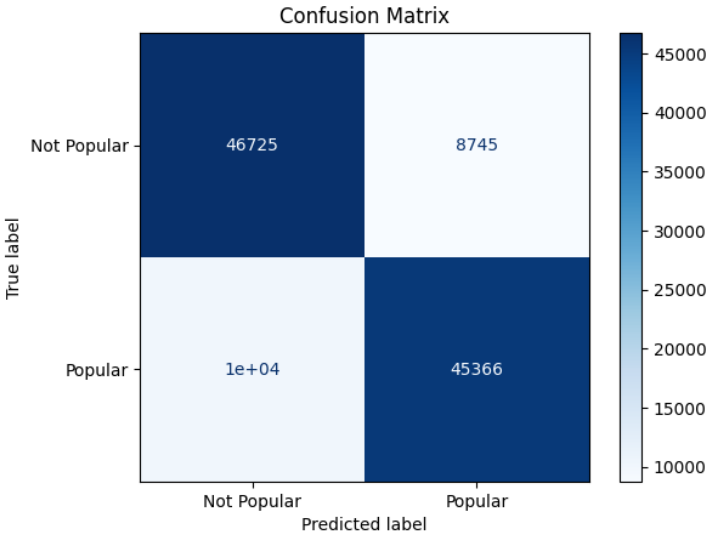


Figure 4.4: Confusion Matrix of Random Forest Model

In this case there's both a good accuracy of the model and a good result of the confusion matrix, which shows a good balance between the number of true positives and true negatives. This is why the Random Forest model is the one taken into account for the final results. Moreover, Random Forest is very well suited for feature importance analysis, which is crucial for this project.

Naïve Bayes

Naïve Bayes is a family of probabilistic algorithms based on Bayes' Theorem, which is used for classification tasks. The algorithm is particularly popular for its simplicity, efficiency, and effectiveness in dealing with large datasets. It is called "naïve" because it makes the assumption that the features are independent from each other given the class label. In this project Gaussian Naive Bayes is used to predict whether a song will be popular or not based on its audio features and release region by assuming that the features follow a normal distribution. To address the class imbalance, undersampling has a more balanced result than oversampling, so it was used in this case.

The Bayes' Theorem is expressed as:

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

Where:

- $P(C|X)$ is the posterior probability of class C given the features X.
- $P(X|C)$ is the likelihood of observing the features X given class C.
- $P(C)$ is the prior probability of class C.
- $P(X)$ is the prior probability of observing the features X.

To make predictions, Naïve Bayes calculates the posterior probability for each class and selects the class with the highest probability. The independence assumption simplifies the calculation of the likelihood:

$$P(X|C) = P(X_1|C) \cdot P(X_2|C) \cdots P(X_n|C)$$

Where $X_1, X_2, \ldots, X_n$ are the features.

Accuracy: **53.52%**

Classification Report

	Precision	**Recall**	**F1-score**	**Support**
0	0.54	0.46	0.50	9544
1	0.53	0.61	0.57	9450
Accuracy		0.54		
Macro avg	0.54	0.54	0.53	18994
Weighted avg	0.54	0.54	0.53	18994

Table 4.4: Classification Report

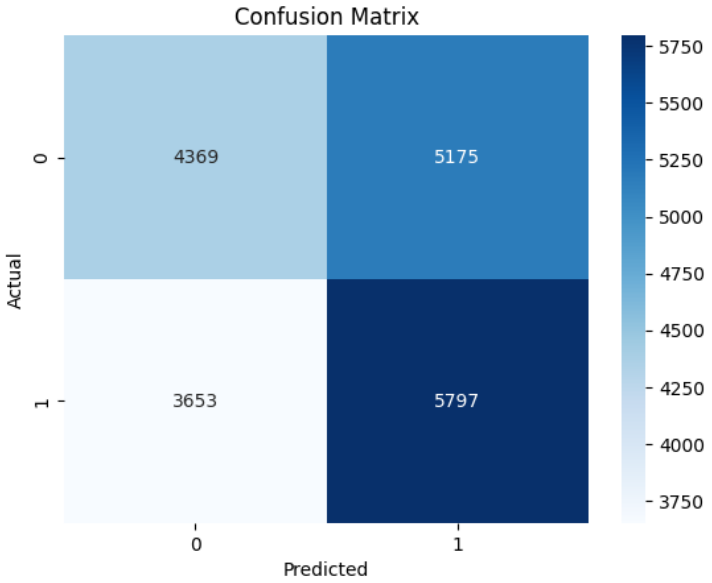


Figure 4.5: Confusion Matrix of Naive Bayes Model

In this case the accuracy is not too good, and the confusion matrix is quite balanced, but it could still be better. This is why the Naïve Bayes model is not the best one for this project.

K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a simple, instance-based learning algorithm that is used for classification and regression tasks. It is a non-parametric method that makes predictions based on the similarity of the input data to the training data. KNN is a lazy learning algorithm, meaning that it does not learn a model during training but instead stores the training data and makes predictions at runtime. The algorithm works by calculating the distance between the input data and the training data and selecting the K-nearest neighbors to make a prediction. The class label of the majority of the neighbors is assigned to the input data as the predicted class. In this project, KNN was used with oversampling, specificcally using the SMOTE technique, to address the class imbalance.

Accuracy: **80.73%**

Classification Report

	Precision	**Recall**	**F1-score**	**Support**
0	0.96	0.64	0.77	55470
1	0.73	0.97	0.83	55443
Accuracy			0.81	
Macro avg	0.84	0.81	0.80	110913
Weighted avg	0.84	0.81	0.80	110913

Table 4.5: Classification Report

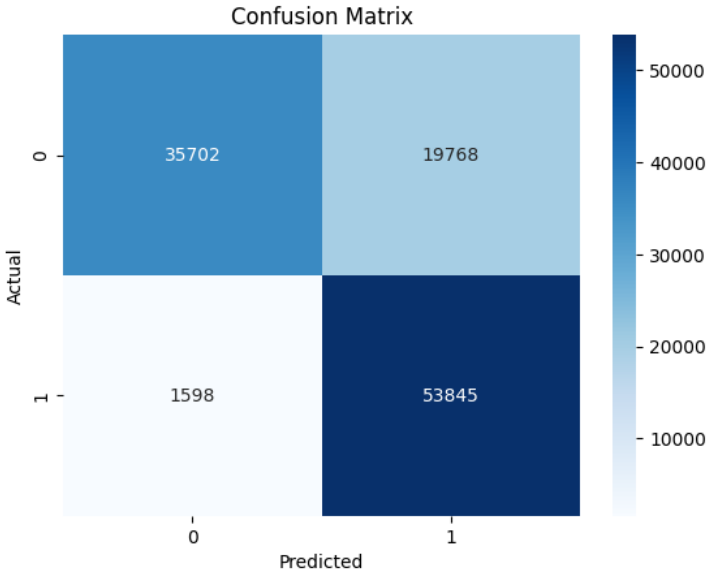


Figure 4.6: Confusion Matrix of KNN Model

The KNN algorithm allows us to calculate and plot the ROC curve, which is a graphical representation of the true positive rate against the false positive rate. This is useful for evaluating the performance of a classification model and comparing different models. The area under the ROC curve (AUC) is a measure of the model's ability to distinguish between the classes. A higher AUC value indicates a better-performing model.
The True Positive Rate (Sensitivity/Recall) on the y-axis shows how often the model correctly identifies a popular song (i.e., correctly predicts a "positive" class).
The False Positive Rate on the x-axis shows how often the model incorrectly predicts a song as popular when it's not (i.e., incorrectly predicts a "positive" class when it should have been negative).
The red dashed line represents a random classifier, meaning if the model was making random predictions, the ROC curve would follow this line (AUC = 0.5).
The ROC curve for the KNN model is shown below:

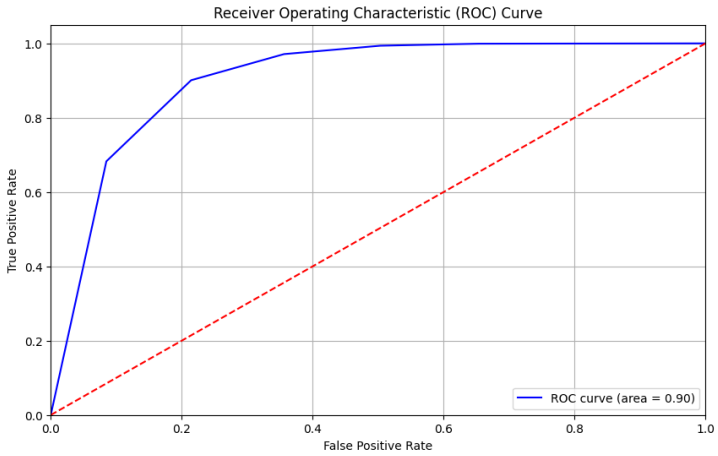


Figure 4.7: ROC Curve of KNN Model

An AUC of 0.90 means that the model can correctly rank songs by their popularity potential 90% of the time, demonstrating a strong ability to separate the classes. This suggests that the it's able to distinguish between popular and non-popular songs with high accuracy. This is a very good result, but the confusion matrix is more accurate on predicting the popular songs rather than the non popular ones.

Literature Review

To have a better understanding of the project, it is important to review the literature on the topic.
In particular, we decided to analyze an article extracted from the 2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA). The article is titled "Predicting a Hit Song with Machine Learning: Is there an apriori secret formula?" by authors: A. H. Raza and K. Nanath.
The authors collected data from the Billboard Hot 100 Charts from 2017 to 2019, and they classified a song as Hit when it appeared in the top 20 at the end of each month, as a Non-Hit if it was amongst the bottom ten songs at the end of each month. This different approach of collecting data allowed the authors to have a better composition of Hit songs vs Non-Hit songs (337 vs 310). To collect the audio features, they also used a Spotify API. The audio features collected were the same as in our dataset.
The authors used also the lyrics of the songs to extract the sentiment analysis. This showed that the sentiment of the lyrics of the Hit songs was in general more positive than the sentiment of the lyrics of the Non-Hit songs, but there wasn't a huge difference. The authors concluded that the sentiment analysis of the lyrics of the songs didn't have a significant impact on the prediction of the popularity of the songs.
They didn't study popularity of the songs across different regions, but they used the same approach to predict the popularity of the songs: they used Logistic Regression, Decision Tree, Random Forests and Naïve Bayes. The algorithm with the highest accuracy was Logistic Regression at 52.0%.
The authors discovered that Danceability was one of the most significant features, if not the most significant. Other features that proved to be significant were Energy, Valence, Tempo, Speechiness, and Loudness. At the same time the algorithms accuracy wasn't so high, which made it difficult to predict if a song will be popular or not with precision.

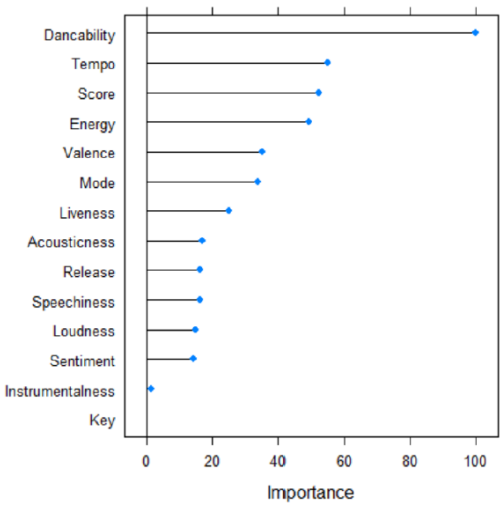


Figure 5.1: Feature Importance

Results

Results of the Models

Model	Accuracy	Weighted Average of Precision
Logistic Regression	85.42%	0.73
Decision Tree	68.64%	0.78
Random Forest	83.02%	0.83
Naïve Bayes	53.52%	0.54
KNN	80.73%	0.84

Table 6.1: Accuracy and Weighted Average of Precision of Different Models

According to the results shown in the table above, the model with the highest accuracy is Logistic Regression with 85.42%, but this is due to the high imbalance of the dataset, so it's not considered the best model.
Random Forest has the higher value of accuracy among the other models, and it also has the highest weighted average of precision. Furthermore, its confusion matrix has the highest number of correct predictions. This makes Random Forests the best model for this dataset with 83.02% accuracy.
Another pretty good result was obtained by KNN, with an accuracy of 80.73% and a weighted average of precision of 0.84.
In general, all the models have high accuracy values, which indicates that the dataset is well-suited for classification. Even the worst model, Naïve Bayes, has an accuracy of 53.52%, which is not bad.
Furthermore, our accuracy results are better than the article studied for the literature, in which the best algorithm, Logistic Regression, had an accuracy of 52%. This could be due to the fact that their dataset was smaller and less diverse than the dataset used in this study.

Global Feature Importance Analysis

It's important to understand the global importance of the features in the dataset to be able to make better predictions.

The feature importance is extracted from the Random Forest model, which is the best model for this dataset.

The Random Forest model shows that globally many features are important for predicting the popularity of a song: *acousticness* is the most important, but also *speechiness*, *valence*, *duration*, *tempo*, *instrumentalness*, *liveness*, *danceability*, *energy* and *loudness* have similarly high values of importance. The least important features are *key*, *mode* and *time signature*. These last two features were already shown to not be important in the correlation matrix.

On the other hand, all the other models show that *dancebility* is the most important feature, with a significant difference from the other features.

This difference can be attributed to the inherent complexity and flexibility of the Random Forest algorithm, which allows it to capture intricate patterns and interactions in the data that other models may not.
Also the literature supports the *danceability* as the most important feature for predicting the popularity of a song, with a significant difference from the other features.

Global Feature Importance

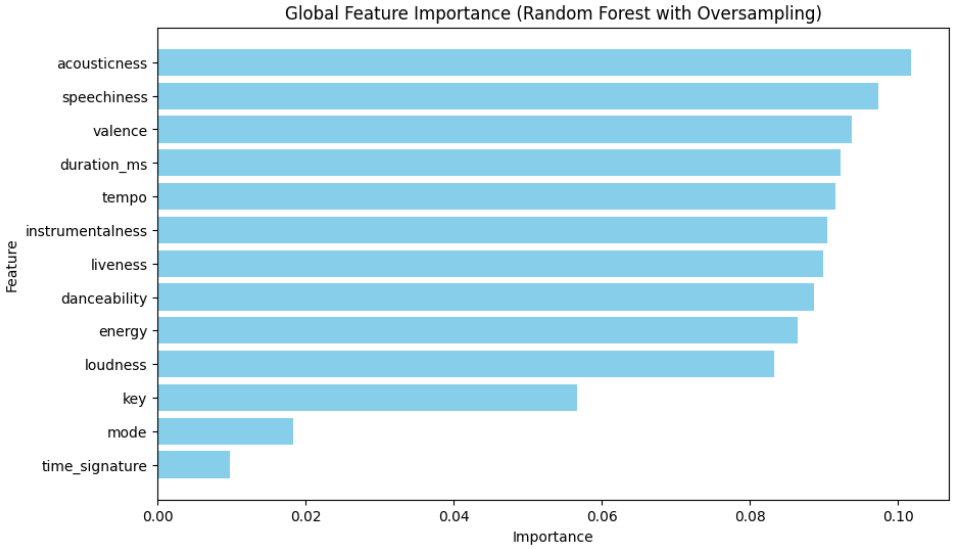


Figure 6.1: Global Feature Importance Using Random Forests Model

Regional Feature Importance Analysis

The feature importance is extracted not only globally but also for each region. To do that we decide to consider the Random Forest model as the best model. This analysis helps to understand how the music tastes of different regions are shaped by audio features. The most important features for each region are shown below. The mean values of these features are also given to help understand the music tastes of each region.

Feature Importance for Each Region Extracted From Random Forest Model

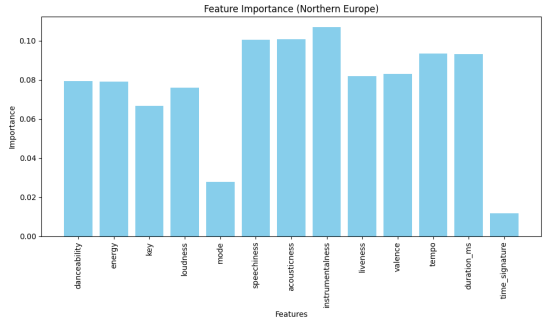


Figure 6.2: Feature Importance in North Europe

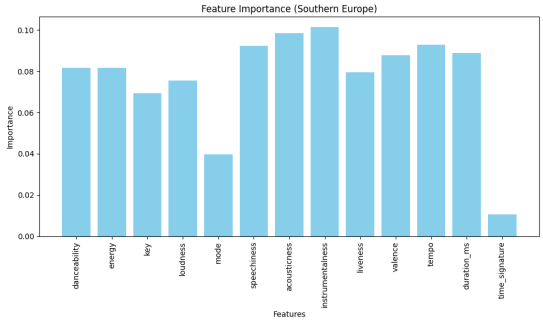


Figure 6.3: Feature Importance in South Europe

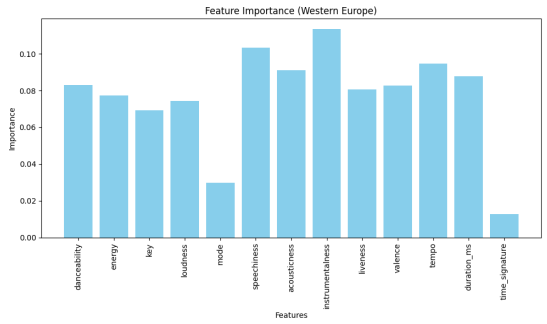


Figure 6.4: Feature Importance in Western Europe

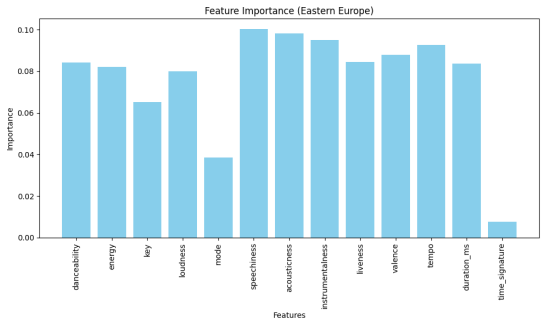


Figure 6.5: Feature Importance in Eastern Europe

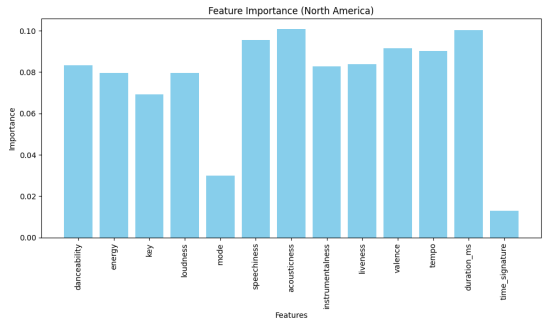


Figure 6.6: Feature Importance in North America

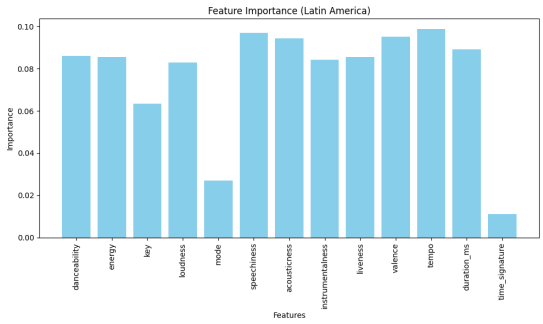


Figure 6.7: Feature Importance in Latin America

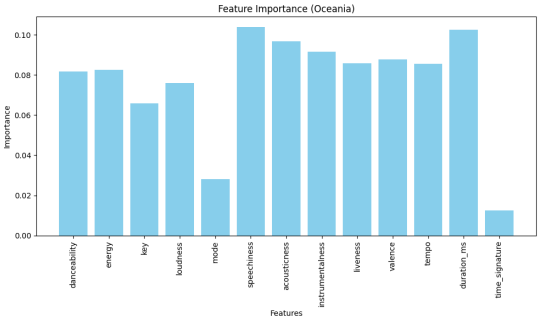


Figure 6.8: Feature Importance in Oceania

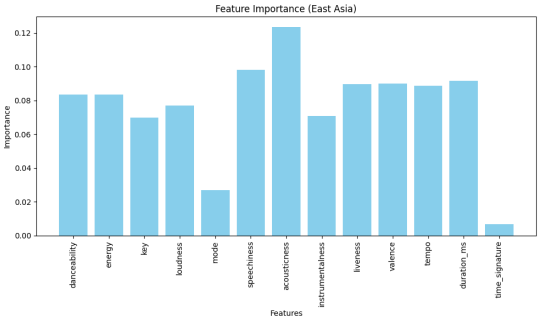


Figure 6.9: Feature Importance in East Asia

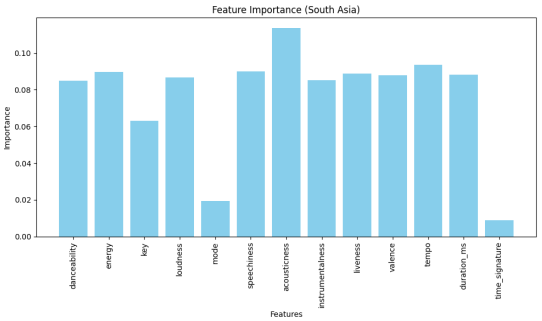


Figure 6.10: Feature Importance in South Asia

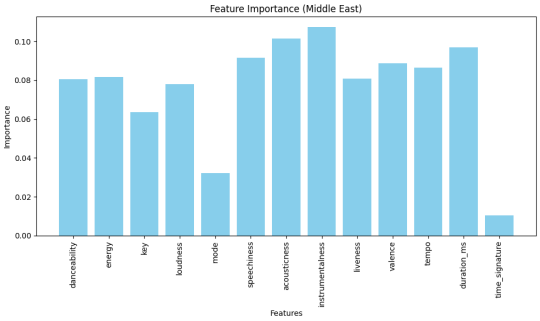


Figure 6.11: Feature Importance in Middle East

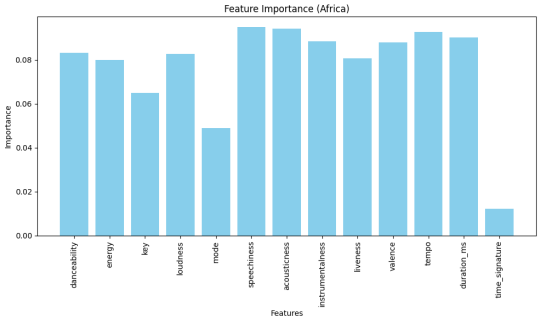


Figure 6.12: Feature Importance in Africa

Important Features for Each Region and Their Mean Values

The different regions tend to have a general coherence in the importance of the features, but there are still differences in the music tastes of each region.

Northern Europe

1. **Instrumentalness**: Importance: 0.1073, Mean Value: 0.04
2. **Speechiness**: Importance: 0.1005, Mean Value: 0.11
3. **Acousticness**: Importance: 0.0994, Mean Value: 0.23
4. **Duration_ms**: Importance: 0.0935, Mean Value: 203514.75
5. **Tempo**: Importance: 0.0918, Mean Value: 121.29

Southern Europe

1. **Instrumentalness**: Importance: 0.1007, Mean Value: 0.04
2. **Acousticness**: Importance: 0.0980, Mean Value: 0.27
3. **Speechiness**: Importance: 0.0933, Mean Value: 0.12
4. **Tempo**: Importance: 0.0929, Mean Value: 121.24
5. **Duration_ms**: Importance: 0.0886, Mean Value: 210862.34

Western Europe

1. **Instrumentalness**: Importance: 0.1137, Mean Value: 0.06
2. **Speechiness**: Importance: 0.1029, Mean Value: 0.15
3. **Tempo**: Importance: 0.0946, Mean Value: 120.90
4. **Acousticness**: Importance: 0.0909, Mean Value: 0.25
5. **Duration_ms**: Importance: 0.0873, Mean Value: 205864.40

Eastern Europe

1. **Speechiness**: Importance: 0.1009, Mean Value: 0.13
2. **Acousticness**: Importance: 0.0973, Mean Value: 0.24
3. **Instrumentalness**: Importance: 0.0967, Mean Value: 0.06
4. **Tempo**: Importance: 0.0928, Mean Value: 122.33
5. **Valence**: Importance: 0.0872, Mean Value: 0.47

North America

1. **Acousticness**: Importance: 0.1013, Mean Value: 0.24
2. **Duration_ms**: Importance: 0.0993, Mean Value: 206987.98
3. **Speechiness**: Importance: 0.0950, Mean Value: 0.13
4. **Valence**: Importance: 0.0914, Mean Value: 0.46
5. **Tempo**: Importance: 0.0898, Mean Value: 121.36

Latin America

1. **Tempo**: Importance: 0.0980, Mean Value: 122.61
2. **Speechiness**: Importance: 0.0971, Mean Value: 0.11
3. **Valence**: Importance: 0.0956, Mean Value: 0.58
4. **Acousticness**: Importance: 0.0953, Mean Value: 0.29
5. **Duration_ms**: Importance: 0.0886, Mean Value: 217127.50

Oceania

1. **Speechiness**: Importance: 0.1027, Mean Value: 0.12
2. **Duration_ms**: Importance: 0.1024, Mean Value: 213396.98
3. **Acousticness**: Importance: 0.0959, Mean Value: 0.23
4. **Instrumentalness**: Importance: 0.0923, Mean Value: 0.05
5. **Valence**: Importance: 0.0878, Mean Value: 0.47

East Asia

1. **Acousticness**: Importance: 0.1223, Mean Value: 0.30
2. **Speechiness**: Importance: 0.0982, Mean Value: 0.08
3. **Liveness**: Importance: 0.0910, Mean Value: 0.18
4. **Duration_ms**: Importance: 0.0902, Mean Value: 229571.73
5. **Valence**: Importance: 0.0896, Mean Value: 0.47

South Asia

1. **Acousticness**: Importance: 0.1135, Mean Value: 0.36
2. **Tempo**: Importance: 0.0928, Mean Value: 119.92
3. **Energy**: Importance: 0.0914, Mean Value: 0.58
4. **Speechiness**: Importance: 0.0902, Mean Value: 0.08
5. **Liveness**: Importance: 0.0885, Mean Value: 0.17

Middle East

1. **Instrumentalness**: Importance: 0.1062, Mean Value: 0.06
2. **Acousticness**: Importance: 0.1010, Mean Value: 0.30
3. **Duration_ms**: Importance: 0.0959, Mean Value: 213894.57
4. **Speechiness**: Importance: 0.0912, Mean Value: 0.10
5. **Valence**: Importance: 0.0894, Mean Value: 0.47

Africa

1. **Instrumentalness**: Importance: 0.1137, Mean Value: 0.06
2. **Speechiness**: Importance: 0.1029, Mean Value: 0.15
3. **Tempo**: Importance: 0.0946, Mean Value: 120.90
4. **Acousticness**: Importance: 0.0909, Mean Value: 0.25
5. **Duration_ms**: Importance: 0.0873, Mean Value: 205864.40

Interpretation of the Results

- **Global Coherence**

  Instrumentalness, *Speechiness*, *Acousticness*, and *Duration* emerge consistently as top features in nearly every region, though with different relative importance. These features often relate to how a song conveys mood and rhythm and might align with the universality of music as a form of emotional expression and storytelling. The wide influence of Western music on pop culture may have contributed to the popularity of these features, as they are core elements in pop, hip-hop, and electronic genres.

- **Region-Specific Preferences**

  In **Latin America** *Tempo* and *Valence* are highly important, reflecting Latin music's rhythmic vibrancy and high-energy, uplifting tones (e.g., salsa, reggaeton). Latin American music often emphasizes danceability, aligning with culturally rooted music traditions focused on rhythm and movement. In **East Asia** and **South Asia** Acousticness is prioritized, showing a preference for traditional, soft, or acoustic elements. This aligns with the longstanding cultural appreciation for subtle, melodic sounds in classical and popular Asian music. In the **Middle East** *Instrumentalness* and *Acousticness* are essential, reflecting the region's rich heritage of instrumental music and acoustic sounds (e.g., oud, ney). This preference is likely influenced by traditional musical forms that emphasize texture and melody over vocal prominence.

- **Shared Importance of Instrumentalness**

  Instrumentalness appears consistently across **Northern Europe**, **Southern Europe**, **Western Europe**, **Africa**, and the **Middle East**. This shared importance suggests a cross-regional appreciation for compositions where instrumental texture plays a major role, which could reflect the influence of classical music traditions that are still prominent in these regions. Additionally, this trend may point to a cultural appreciation for genres like electronic or ambient music, which often rely heavily on instrumentals.

- **Tempo as a Key Feature**

  Tempo is particularly prominent in **Latin America** and **Eastern Europe**, regions with strong dance traditions (e.g., cumbia, salsa, and Balkan beats). A faster or more rhythmic tempo may appeal to these audiences due to the cultural significance of danceable music. In other regions, such as **North America**, *Tempo* is present but slightly less emphasized, potentially due to the popularity of more varied genres, including slower-tempo hip-hop and R&B.

- **Duration as a Consistent Feature**

  Duration_ms holds similar importance across regions like **North America**, **Western Europe**, and **Oceania**, indicating a common preference for song length that aligns with global pop music standards. Shorter or moderate-length songs, commonly around 3 to 4 minutes, are more favorable in these regions, aligning with the mainstream pop structure popularized by global record labels and radio formats.

- **Valence and Mood Preferences**

  Valence, which measures musical positiveness, is a notable feature in **Latin America** and **North America**, suggesting a preference for more "positive" or "happy" songs. This could reflect cultural preferences for upbeat, feel-good music in these regions, where music often accompanies social gatherings. In **East Asia**, however, *Valence* is less prominent, potentially indicating a preference for more emotionally complex or even melancholic tones, which is often present in Asian pop and ballads.

- **Impact of Globalization**

  Regions with high influence from Western pop culture, such as **North America**, **Europe**, and **Oceania**, show a strong alignment in the importance of core features like *Acousticness* and *Speechiness*, likely due to the widespread dissemination of pop, hip-hop, and EDM. In contrast, regions with strong traditional music influences (e.g., **Middle East**, **East Asia**) retain unique feature preferences, such as a stronger emphasis on *Acousticness* and *Instrumentalness*, reflecting a blend of contemporary and traditional music tastes shaped by both global trends and local heritage.

- **Underrepresented Features**

  Energy, *Key*, and *Mode* are not among the top features for most regions, indicating these are less influential in determining a song's popularity. This suggests that, despite varying genres and tastes, there's less of a focus on tonal characteristics and more on rhythmic and acoustic qualities across different regions.

- **Evolution of Regional Music Preferences**

  Over time, these differences in preferences might evolve as global trends influence local tastes, or as regions increasingly revert to traditional forms in response to external influences. Monitoring how these values change over time could offer insights into the trajectory of regional music trends. These considerations highlight the diversity in music preferences across regions, while also emphasizing certain universal elements that transcend cultural boundaries. Understanding these nuances helps map out how cultural and social factors influence regional music trends.

Conclusion

Future Work

For further developments of the project, it could be interesting to take into consideration also the sentiment analysis. This can be done by studying the lyrics of the songs as it was done in the literature review. Finding out if a happy song is more keen to be popular than a sad song could be interesting.
Sentiment analysis could also be done on the reviews of the songs. This could be done for example by analyzing what people write about a specific song or a specific artist on social media. Also the most important music magazines could be analyzed to see if the reviews of the songs are positive or negative and how this affects the popularity of the song.
All this information can be used to improve the model.
Another interesting thing to do is to improve the User Interface of the application. As of now the application allows the user to insert the region and the different values of the audio features. It could be improved by adding more features, like the possibility to search for a specific song and see its popularity, or the possibility to see the most popular songs of a specific artist.

Conclusion

In general, finding what really makes a song popular and make correct predictions is a very difficult task as the popularity of a song is influenced by many factors.
This project has shown that it is possible to predict the popularity of a song with a good accuracy, which could help a lot in the music industry, but it could be improved.
The algorithms used in this project are proved to be more accurate than the ones used in the state of the art. In the state of art the more accurate algorithm is the Logistic Regression, which has an accuracy of 0.52, while in this project it is the Random Forests with an accuracy of 0.83.
According to the results of the scientific paper, danceability is the most important feature for a song to be popular. In our project, danceability is surely an important feature, but it is not the most important one.
Furthermore, we found out that some features are similar in different regions of the world, like Acousticness, which is important in all the regions, while some others are different, like Valence, which is important in some regions and not in others.
This shows that globalization has an influence on the music industry, but there are still some cultural differences that have to be taken into consideration.