UNIVERSITY OF
GOTHENBURG

Project: Danceability classifier

Statistical Methods for Data Science

Group 6

Kinga Jenei

Jonna Marie Matthiesen

Fall semester 2021

Contents

1 Project Description

The goal of this project is to build a classifier for the danceability of songs. Based on 12 different song features, such as acousticness, energy and liveness, we want to classify a song as danceable or not-danceable in order to be able to select the best songs for a party.

End-use cases
There are several end-use cases for this classifier. For example, as mentioned in the previous paragraph, a party organizer can use our classifier to create the perfect dance playlist.

Another end-use case would be the production and creation of new songs. Current artists and producers can use the classifier to predict the party suitability of the song they are producing. Many attributes of the classifier are song features that can be derived directly from the track [e.g., duration, loudness, time signature, and mode]. Other features, like energy, speechiness, and valence, can be manually determined by the artist/ producers by applying their domain knowledge and expertise.

1.1 Content

In the first part of this report, we will explore our Spotify dataset in great detail.

Data specification
We will first explore every feature by describing what each column of our tabular data holds. In this section, it is also described how we collected our data and how we decided on the danceability of a track (ground truth labels).

Descriptive Analysis
After describing the data in general, we will explore our dataset by visualizing different features and especially by calculating the correlation of the danceability feature with all other features. Based on the correlation coefficient we will put a greater emphasis on those features which are highly correlated with the danceability of a track.
We will first visualize the dataset by using histograms for numerical data and bar plots + pie charts for categorical data. Secondly, we will analyse the dependence of some features in connection with the danceability of tracks.

Probability Distribution
In the third part of this report, we will analyse three features in greater detail: loudness, energy, and valence. We will use probability distributions to describe these selected variables and find theoretical distributions that fit the variables best.

Hypothesis testing
Based on the extensive visualization of our variables in the previous parts we will derive some interesting hypotheses and test those hypotheses in this part.

Predictive Analysis
After exploring the dataset we are ready to apply two predictive machine learning models to solve our problem of finding the best songs for a party. We will use the models k-Nearest Neighbors and decision tree. We will evaluate the performance of both classifiers and finally decide on one classifier.

2 Data specification

2.1 Data collection

We have collected the data for this project ourselves by using the Spotify API. *"Spotify is a digital music, podcast, and video service that gives [you] access to millions of songs and other content from creators all over the world."* [1]. Descriptions of the API itself and the derived song features can be found at [2].

We have chosen the songs of our dataset in the following way:
Since our goal is to create a classifier for the danceability of songs we have picked songs from dance / party playlists on Spotify (by choosing playlist from the Spotify category "party") and gave those songs the ground truth label 1 (danceable). For non-danceable songs, we picked a collection of playlists from various Spotify categories and gave those songs the ground truth label 0 (not-danceable). When a song appeared in two different playlist categories ("party" and "non-party") we classified this song as danceable.

This labelling method has some flaws since we don't check every song individually. In the last part of this report (Conclusion) we will critically evaluate how we collected our data and how this process could be improved.

It is to mention that due to the always-changing preference regarding party- and dance-songs over the decades, the ground truth labels of our dataset are not valid forever. Instead, the following analysis of the dataset in connection with the danceability of a track represents only the *current* preferences given the chosen playlists.

2.2 Data description

Target: danceability

integer, number in $\{0, 1\}$; Categorical - Nominal data

Index which indicates if a track is danceable (1) or not (0). A track is classified as danceable if it appears in one of the selected Spotify party-playlists (see data collection).

danceability value	description
0	track is unlikely to be danceable → track is not suitable for a party
1	track is likely to be danceable → track is probably suitable for a party

Table 1: Description of the danceability feature

Input features

track_id

string; Categorical - Nominal data

Unique id of the song / track.

artist_name

string; Categorical - Nominal data

The name of the artist.

track_name

string; Categorical - Nominal data

The name of the track.

duration_ms

integer; Numerical - Discrete (interval) data

Duration of the track in milliseconds (ms).

acousticnes

float, number in [0.0, 1.0]; Numerical - Continuous data

A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.

Example of an acoustic song:

To listen to the song: `https://open.spotify.com/track/6lz4vzv0JlD73qeKW5qsUT`

Artist: Benjamin Gustafsson

Track: I Heard You From Afar

acousticness: 0.99400

Example of a not acoustic song:

To listen to the song: `https://open.spotify.com/track/636kRUoh1D1ZJifHip4RtS`

Artist: Love Regenerator

Track: Rollercoaster

acousticness: 0.00004

energy

float, number in $[0.0, 1.0]$; Numerical - Continuous data

Energy represents a perceptual measure (e.g. a quantity measurement which relates to human perception and interpretation) of intensity and activity. This includes perceptual features like dynamic range, perceived loudness, timbre (perceived sound quality), ... Typically, energetic tracks feel fast, loud, and noisy.

high energy $\sim$ high intensity and activity

instrumentalness

float, number in $[0.0, 1.0]$; Numerical - Continuous data

The closer the value 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.

E.g. rap or spoken word tracks are clearly "vocal".

key

integer, number in $[1, 11]$; Categorical - Ordinal data

The key the track is in. See the following table for the mapping. The mapping follows standard pitch class notation.

key	pitch
-1	no key detected
0	C
1	C#/Db
2	D
3	D#/Eb
4	E
5	F
6	F#/Gb
7	G
8	G#/Ab
9	A
10	A#/Bb
11	B

Table 2: Key mapping

liveness

float, number in $[0.0, 1.0]$; Numerical - Continuous data

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 a strong likelihood that the track is live.

loudness

float; Numerical - Continuous data

The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track. The values typically range between -60 and 0 dB.

Strongly simplified: the closer the value is to 0, the louder the track.

Examples of different noise levels (not averaged): [3]

decibels (dB)	Noise source
120 - 129 dB	Sports crowd, rock concert, loud symphony
106 - 115 dB	chain saw
90 - 95 dB	Subway, shouted conversation
75 dB	vacuum cleaner
60 dB	normal conversation
40 dB	average home noise

Table 3: Decibel mapping

Example of a "silent" song:

To listen to the song: `https://open.spotify.com/track/6lz4vzv0JlD73qeKW5qsUT`

Artist: Benjamin Gustafsson

Track: I Heard You From Afar

loudness: -33.33100

Example of a loud song:

To listen to the song: `https://open.spotify.com/track/5YPMEOJ58kfl56VHxTgwx3`

Artist: David Guetta

Track: Play Hard (feat. Ne-Yo & Akon)

loudness: -1.70200

mode

integer; Categorical - Ordinal data

Indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. The mode feature is an additional feature of the key feature: A track can have a major key or minor key.

value	mode
0	minor
1	major

Table 4: mode mapping

speechiness

float; Numerical - Continuous data

Speechiness detects the presence of spoken words in a track. The value of speechiness ranges from 0.0 to 1.0.

speechiness value x	description
$0.66 < x$	tracks that are probably made entirely of spoken words (e.g., podcasts, audiobooks, poetry)
$0.33 < x \leq 0.66$	tracks that may contain both music and speech, either in sections or layered (e.g., rap music)
$x \leq 0.33$	most likely music and other non-speech-like tracks

Table 5: Description of the speechiness feature

tempo

float; Numerical - Continuous data

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.

tempo value x	description
$x \leq 85$	slow song
$85 < x < 120$	song with moderate speed
$120 \leq x$	fast song

Table 6: Description of the tempo feature

time_signature

integer, number ≥ 1 ; Numerical - Discrete data

An estimated time signature. The time signature usually ranges from 3 to 7 indicating time signatures of "3/4" to "7/4".

Example: A time signature of 3 indicates that there are three quarter notes (crotchets) per measure (bar).

valence

float, number in $[0.0, 1.0]$; Numerical - Continuous data

Describes the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).

2.3 Train-Test split

We are doing an 80-20 train-test split, meaning 80% of the data will be used for training, and 20% of the data will be used for the final testing. Our whole dataset consists of in total 1359 songs and is split the following way:

- training: 1087 tracks [869 for training and 218 for validation]
- testing: 272 tracks

3 Descriptive Analysis

We will start by describing the data using sample mean, sample standard deviation, and some quantiles for each feature. We will refer to the following table in the upcoming descriptive analysis.

Our train dataset consists of 1087 tracks. Since the danceability feature has a mean of nearly exactly 0.5 (0.49) we conclude that half of our training set consists of danceable tracks (tracks with danceability = 1).

	duration_ms	acousticness	energy	instrumentalness	key
count	1359.000000	1359.000000	1359.000000	1359.000000	1359.000000
mean	199450.822664	0.260244	0.647494	0.090653	5.573216
std	41270.148444	0.309641	0.232382	0.236409	3.623730
min	106880.000000	0.000032	0.015500	0.000000	0.000000
25%	172937.500000	0.025300	0.532000	0.000000	2.000000
50%	193929.000000	0.116000	0.708000	0.000006	6.000000
75%	219545.000000	0.398500	0.819000	0.002920	9.000000
max	578200.000000	0.994000	0.999000	0.971000	11.000000

	liveness	loudness	mode	speechiness	tempo
count	1359.000000	1359.000000	1359.000000	1359.000000	1359.000000
mean	0.168305	-7.245618	0.621045	0.081645	119.577702
std	0.121281	4.317450	0.485306	0.088906	23.730465
min	0.024400	-33.331000	0.000000	0.022500	54.727000
25%	0.095250	-8.047000	0.000000	0.036100	103.964500
50%	0.121000	-5.974000	1.000000	0.047100	123.056000
75%	0.196000	-4.644500	1.000000	0.080300	128.037000
max	0.954000	-1.431000	1.000000	0.723000	203.803000

	time_signature	valence	danceability
count	1359.000000	1359.000000	1359.00000
mean	3.970567	0.481322	0.48933
std	0.252852	0.228826	0.50007
min	1.000000	0.034800	0.00000
25%	4.000000	0.305000	0.00000
50%	4.000000	0.479000	0.00000
75%	4.000000	0.657500	1.00000
max	5.000000	0.972000	1.00000

Table 7: Descriptive statistics

3.1 Correlation

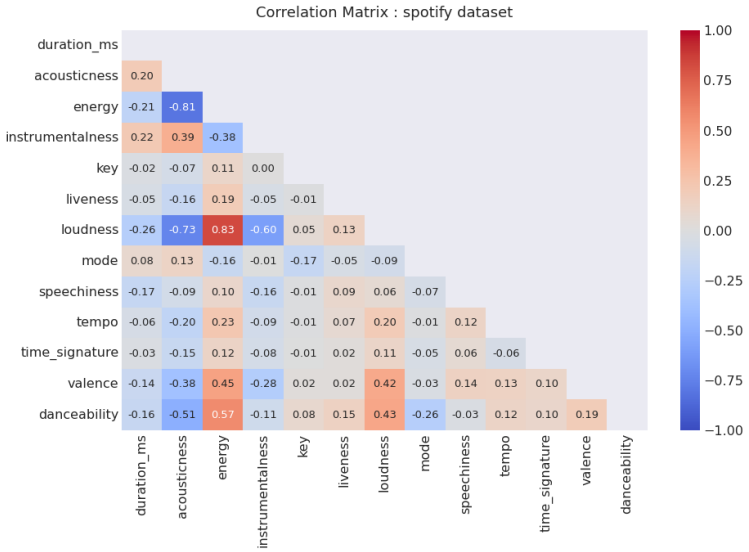

Figure 1: Correlation matrix of features

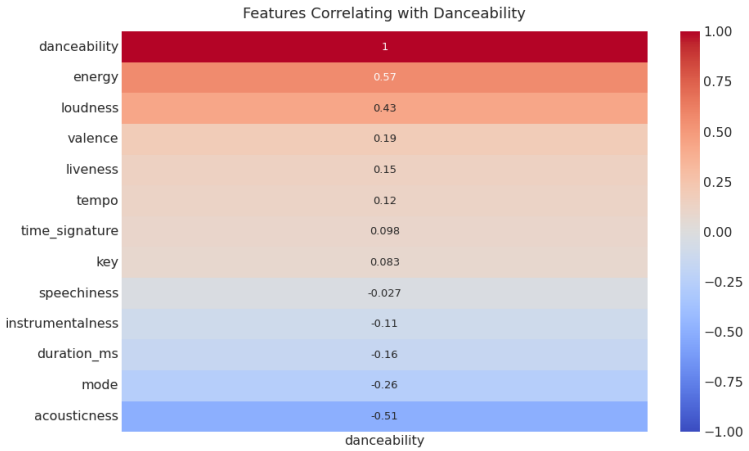

Figure 2: Correlation of features with danceability

The above correlation measurement shows that danceability and the features energy, loudness, and acousticness are highly correlated. This reflects our general interpretation / intuition of party songs: We suspect energetic songs (high energy value) to be danceable. Furthermore, since the noise level at parties is generally quite high, we suspect party songs / danceable songs to be louder than the average. These two intuitions are reflected in the positive correlations.

We also observe from the above plot a negative correlation between danceability and acousticness. *As a reminder:* Acousticness is a confidence measure from 0.0 to 1.0 of whether the track is acoustic - 1.0 represents high confidence the track is acoustic. Intuitively, the negative correlation of danceability and acousticness makes sense - out of experience most party songs appear to be not acoustic / have a low acousticness value.

Given the correlation with danceability, we can distinguish three groups of features. We decided on the

following groups based on the given correlations with the aim of having a similar number of features in each group:

Description	Correlation	Features
High correlation with danceability	$0.40 \leq \mid corr \mid \leq 1$	energy, loudness, acousticness
Medium correlation with danceability	$0.15 \leq \mid corr \mid < 0.40$	valence, mode, liveness, duration_ms
Low correlation with danceability	$0 \leq \mid corr \mid < 0.15$	tempo, key, time_signature, speechiness, instrumentalness

Table 8: Correlation Groups : Danceability

3.2 Histograms

We will now show the histogram of some selected variables and describe our conclusions from the plots. We will have a look at the following song features: energy, loudness, acousticness since those features are highly correlated with the danceability of tracks. Additionally, we will show the histogram of the features valence (medium correlation), and speechiness (low correlation) in order to plot at least one feature per group.

3.2.1 Energy

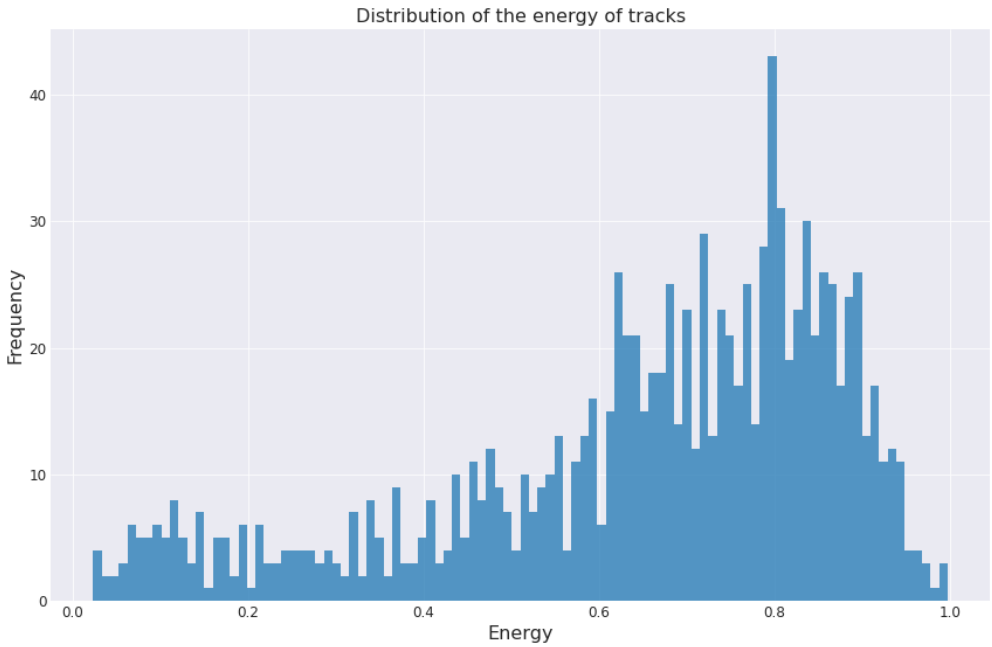

Figure 3: Distribution of the energy of tracks

In the above histogram of the energy feature, we can determine two modes: One mode around 0.8 and another around 0.1. This observation reflects our chosen playlists from Spotify. We collected songs from dance - playlists and also from chill - playlist. By intuition, one would suspect chill songs to have a low(er) energy. Also, as we have seen before, the features energy and danceability are strongly positively correlated which might result in a higher frequency of energetic songs in our data since half of our data are danceable tracks.

3.2.2 Loudness

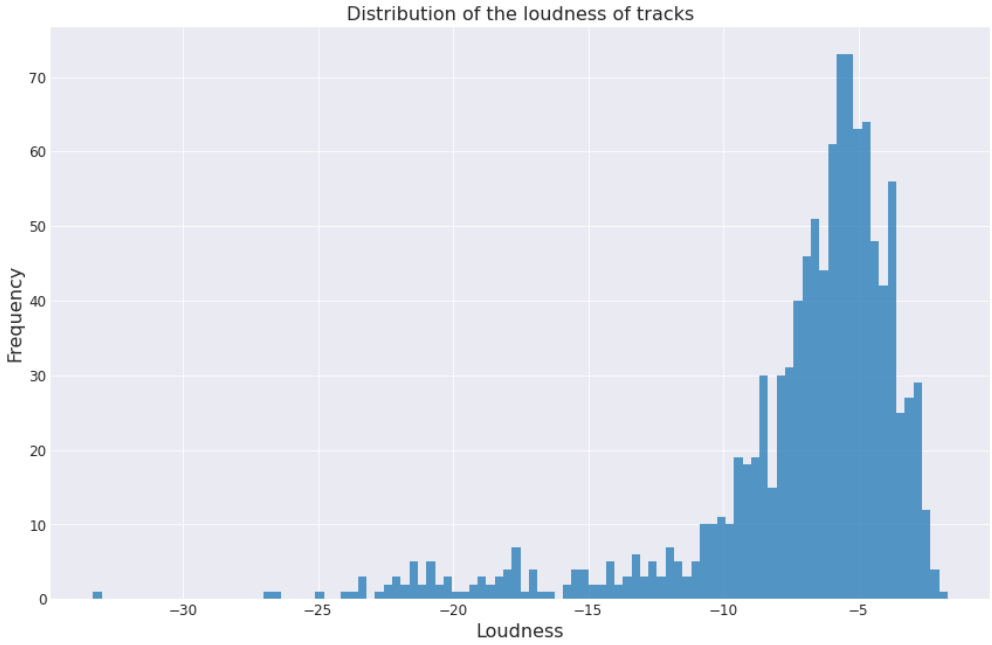

Figure 4: Distribution of the loudness of tracks

The above plot of the distribution of the loudness of tracks shows a unimodal distribution with most values centred around -7.15 and a standard deviation of about 4.2. Since our plot is negatively skewed most of our selected data can be classified as loud. *Reminder: the closer the value is to 0, the louder the track [simplified definition of the loudness feature].* This distribution makes sense since we are dealing with songs and songs are in general louder than for example audiobooks or podcasts - the range of the loudness feature of tracks includes for example podcasts and other spoken audio which can be classified as 'silent'.

3.2.3 Speechiness

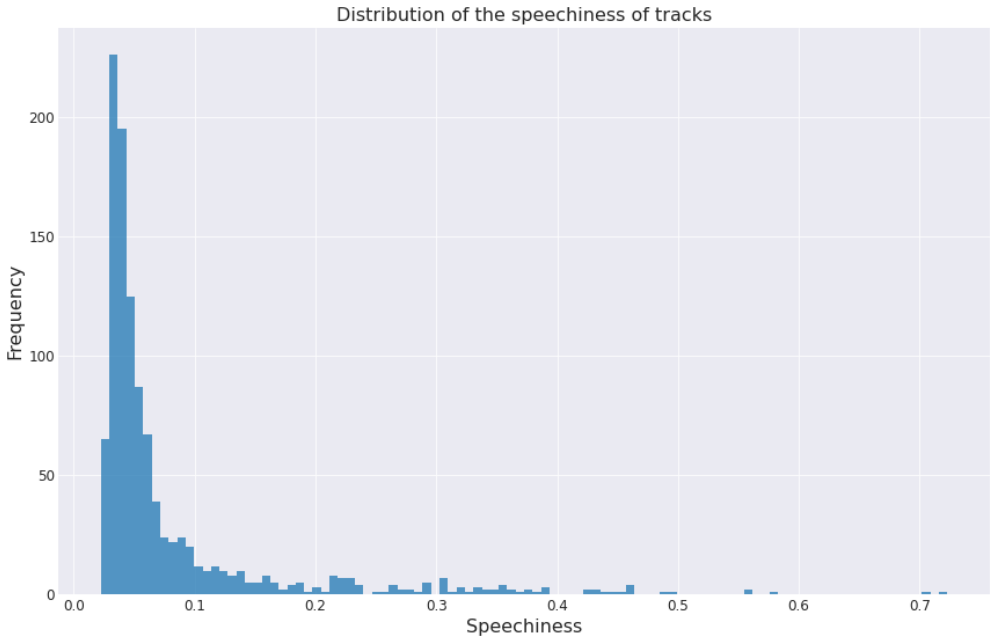

Figure 5: Distribution of the speechiness of tracks

The above plot shows a unimodal distribution of the speechiness feature with most of the values centered around 0.08 (mean). As expected from the definition of the speechiness column we mainly have tracks with a low speechiness value (close to 0), i.e. few spoken words.

Definition of speechiness: $x \leq 0.33$ - most likely music and other non-speech-like tracks.

Even though we are only dealing with music, our data set includes some tracks with a speechiness value of above 0.5. We will now have a closer look at the outlier on the far right-hand side with a speechiness value of above 0.7:

> To listen to the song: `https://open.spotify.com/track/0W7brFokN6QtGRNp32RCQP`
>
> Artist: Arz
>
> Track: Alone With You
>
> speechiness: 0.72300

The given song "Alone With You" from Arz is a Hip-Hop/Rap - song with many clearly spoken words.

3.2.4 Acousticness

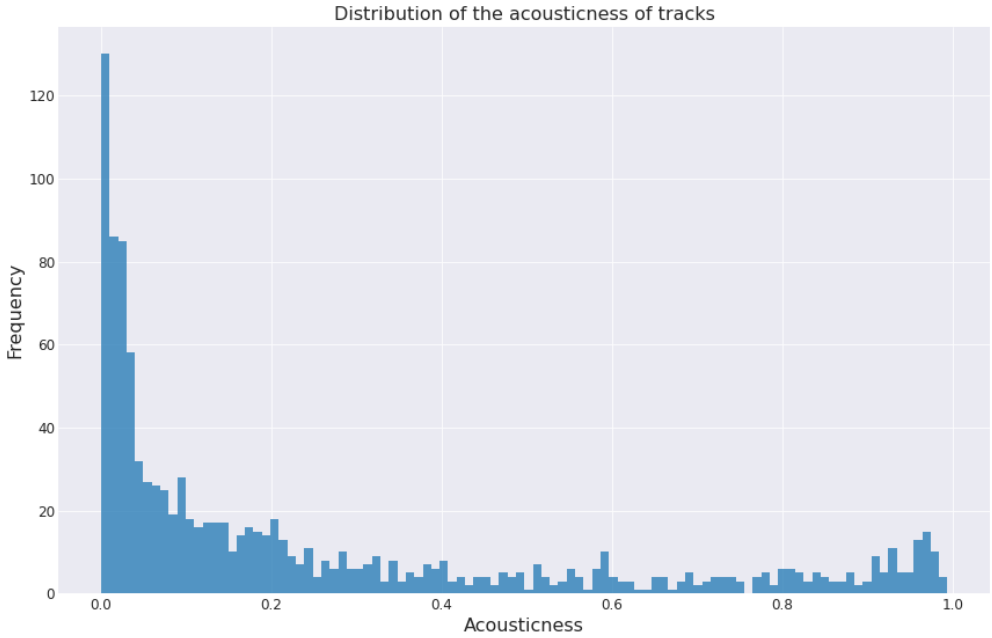

Figure 6: Distribution of the acousticness of tracks

The acousticness of our data ranges from 0.0 to 1.0 and covers the whole range of the acousticness feature. This observation is a result of our data selection which covers tracks from all kinds of genres.

1.0 : very high confidence the track is acoustic.
0.0 : very low confidence the track is acoustic.

We can identify two modes: one mode close to 1 and the other close to 0. As described before, we assume danceable tracks not to be acoustic / to have a low acousticness measure. Since half of our data consists of tracks classified as danceable we expect a high frequency of tracks with low acousticness.

Later on, we will analyze this observation in more detail and especially check, if the mode around 0 is due to the fact that half of our data is danceable.

3.2.5 Valence

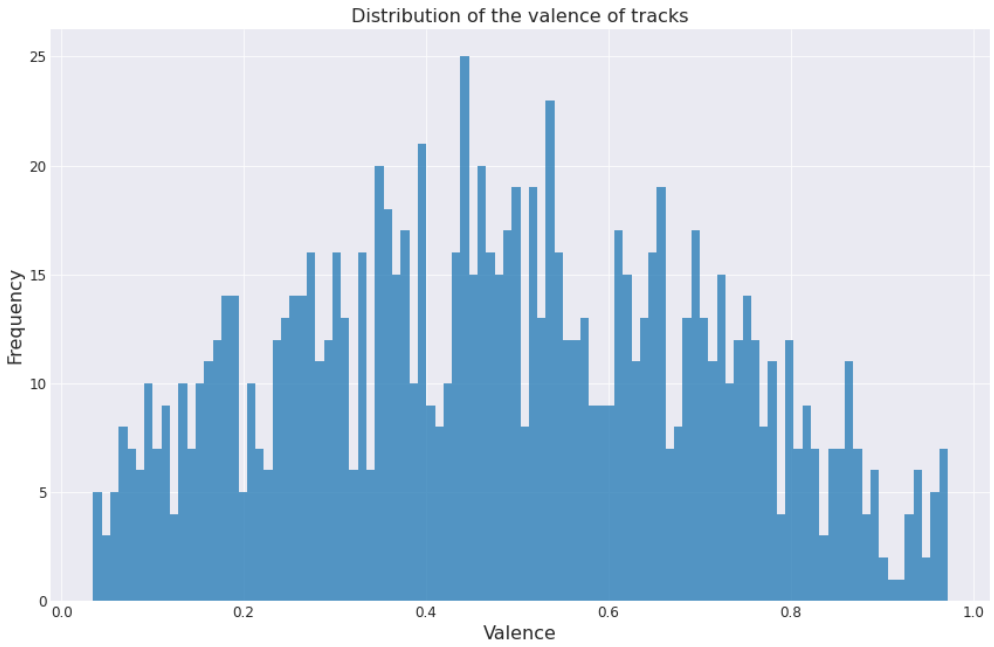

Figure 7: Distribution of the valence of tracks

The histogram of the valence feature shows a unimodal, symmetric distribution, centred around 0.48 (mean) and with a standard deviation of 0.23. This distribution shows that we have collected a variety of positive, negative and neutral tracks, with similar amounts of positive and negative tracks.

Reminder: Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).

3.3 Bar plots and Pie charts

We will now have a closer look at categorical features by plotting their distributions using bar plots and pie charts. We have selected the following features from two correlation - groups:

- medium correlation: mode
- low correlation: key, time signature

3.3.1 Key

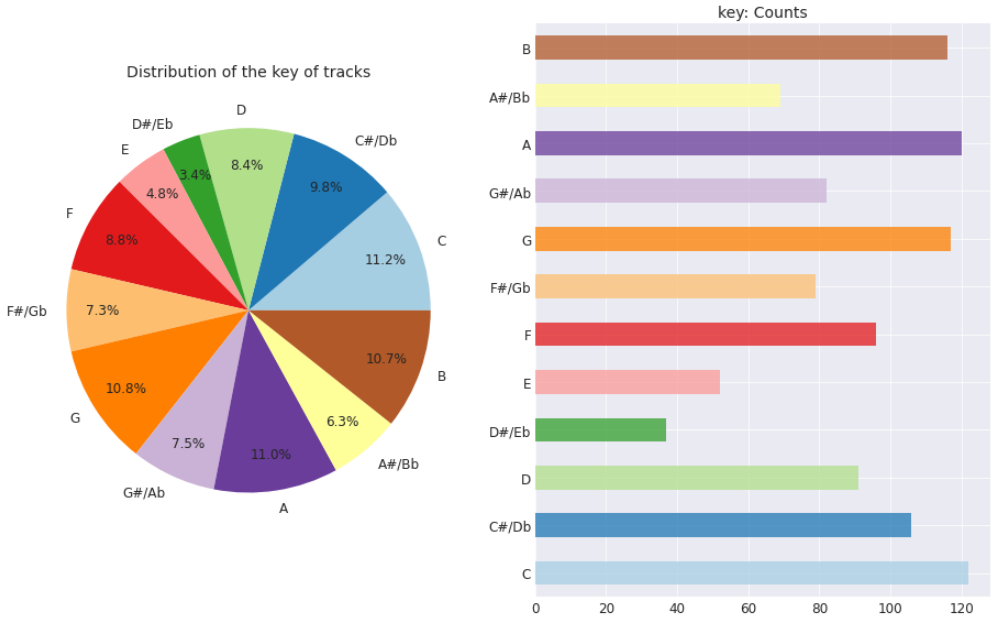

Figure 8: Distribution of the key of tracks

The above plots show that counts of the key of tracks range from 37 to 122 with a mean of about 90. The keys B, A, G, and C occur most frequently (above 10% each) while the keys E, and D#/Eb occur the least (below 5% each). Since the distributions of the keys are more or less evenly distributed we can not determine a specific trend for this feature.

3.3.2 Time signature

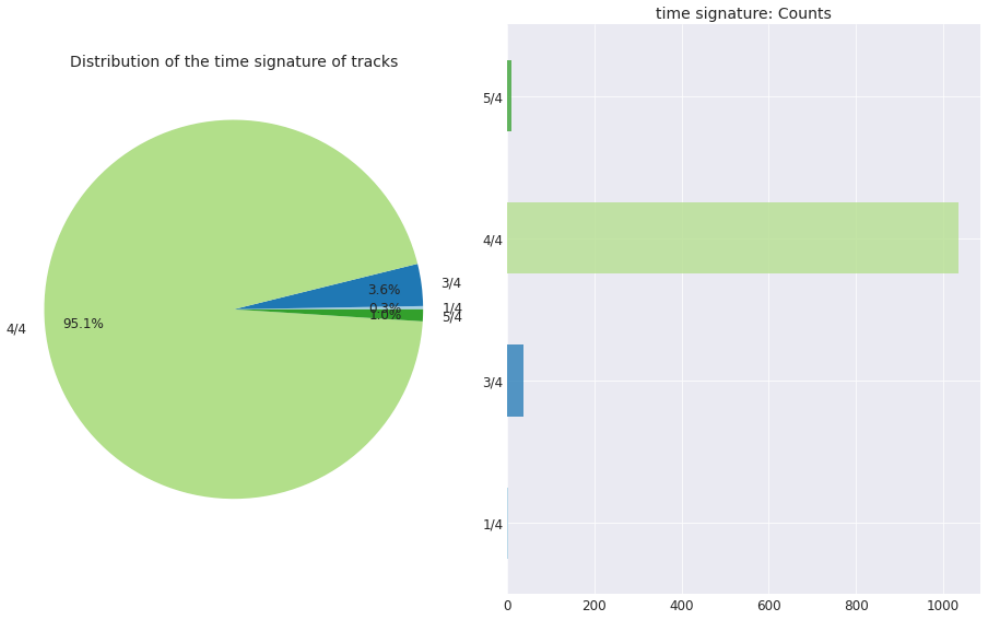

Figure 9: Distribution of the time signature of tracks

The above plots show that most tracks have an estimated time signature of 4/4 (above 95% of the tracks) and just a few tracks deviate from this trend with an estimated time signature of 5/4, 3/4 or 1/4. These plots reflect the popularity of the 4/4 time signature for songs. As a fact, most western music is written in 4/4 which is a regular and simple beat pattern. [4]

3.3.3 Mode

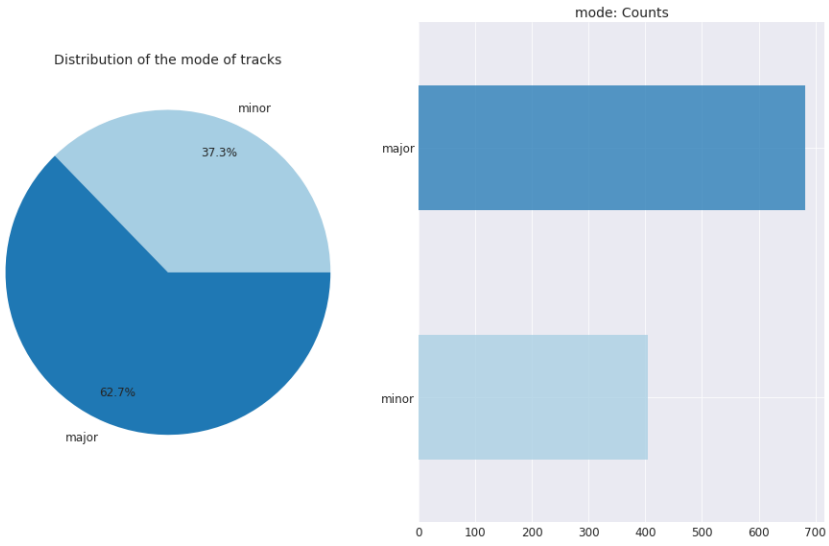

Figure 10: Distribution of the mode of tracks

The mode of tracks is either major or minor. We can derive from the above plots that most of the tracks have a major key (above 60%). We will shortly analyse the dependence between the mode of a track and the danceability in more detail.

3.4 Dependence

In the following, we will analyze the dependence of the features acousticness, energy, loudness, mode, and danceability since those features strongly correlate with danceability.

3.4.1 Acousticness vs Energy

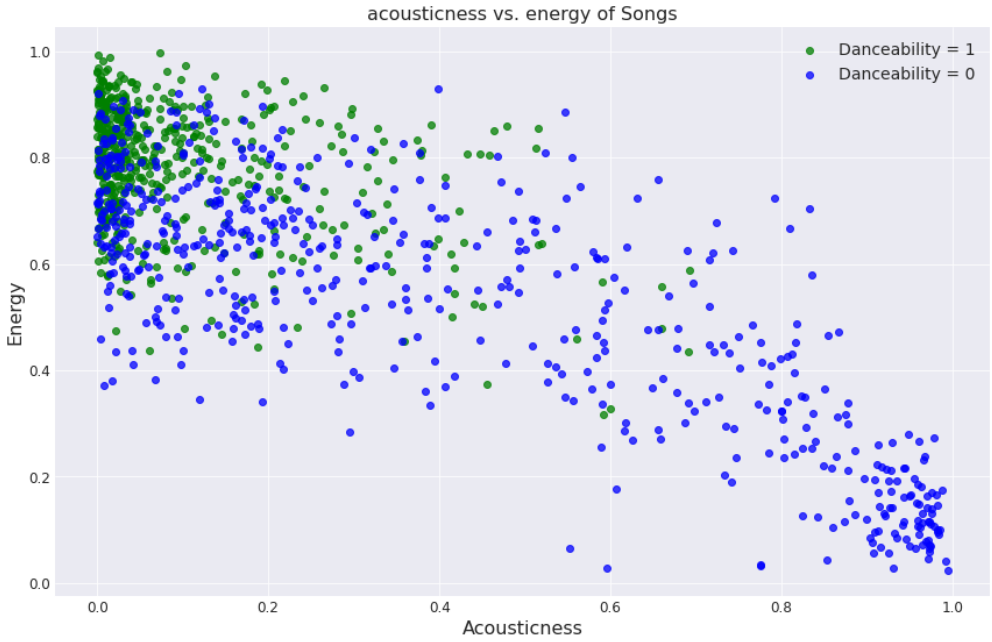

Figure 11: Acousticness vs energy of songs

The above plot shows the dependence of the three features energy, acousticness, and danceability. We can derive from the plot that most danceable songs in our dataset have high energy and a low acousticness value [most of the green dots which represent danceable tracks are scattered in the top left corner]. On the other hand, non-danceable tracks seem to appear in all ranges of acousticness and energy. They appear most frequently for high energy + low acousticness and vice versa (i.e. low energy + high acousticness). Again, out of intuition and experience, we assume energetic songs to be less acoustic and vice versa, which might be a justification for the given distribution - the negative relationship of energy and acousticness.

In connection to the histograms of the features energy and acousticness we assumed the high frequency of energetic songs / non-acoustic songs to be due to the data selection, i.e. the fact that half of our data is danceable. The given scatter plot strengthens this assumption as described before.

3.4.2 Acousticness vs Loudness

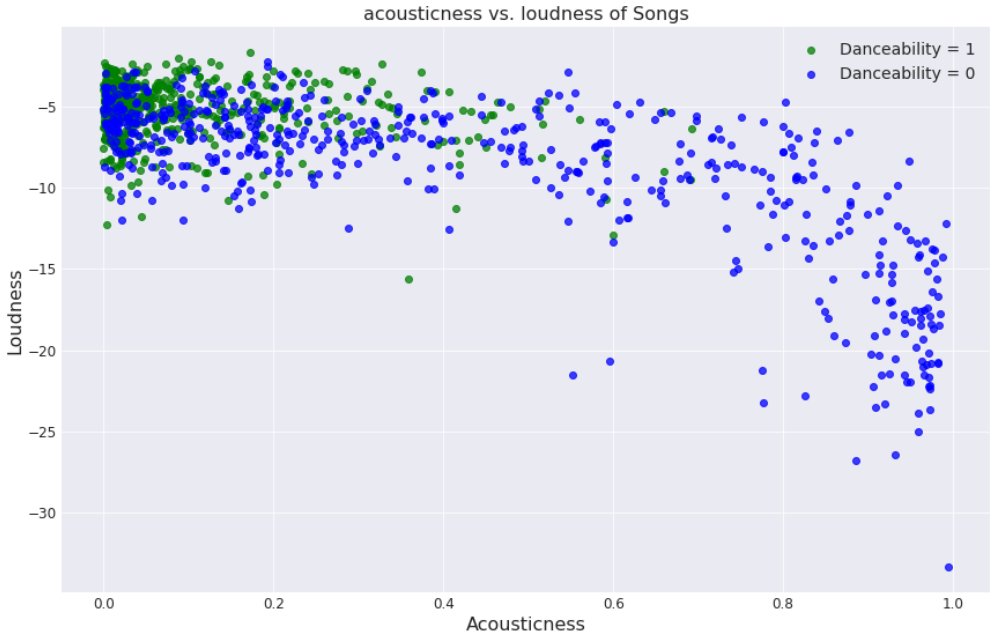

Figure 12: Acousticness vs loudness of songs

The given scatter plot shows the dependence of the features loudness, acousticness, and danceability of tracks. Once again, we can determine a trend of danceable tracks: most danceable tracks are scattered in the top-left corner of the plot (high loudness and low acousticness). Non-danceable songs are scattered more widely and cover a greater range of the acousticness and loudness values. For non-danceable tracks, we can also spot a cluster in the top-left corner.

The variance of the loudness of danceable tracks seems much smaller than the variance of the acousticness of danceable tracks. To check this assumption we will take a look at the reduced data set consisting only of danceable songs:

danceability	loudness	acousticness
0	4.974427	0.341573
1	1.824767	0.131030

Table 9: Standard deviation of the features loudness and acousticness

As we can see in the table above, the standard deviation (square root of the variance) of the loudness of danceable tracks is more than 10 times bigger than the standard deviation of the acousticness of danceable tracks.

As mentioned in the section "Histograms", we assume that more people dance to non-acoustic songs. This assumption is underlined by the two scatter plots above. We want to test this hypothesis using a t-test in the upcoming section "Hypothesis Testing".

3.4.3 Energy vs Loudness

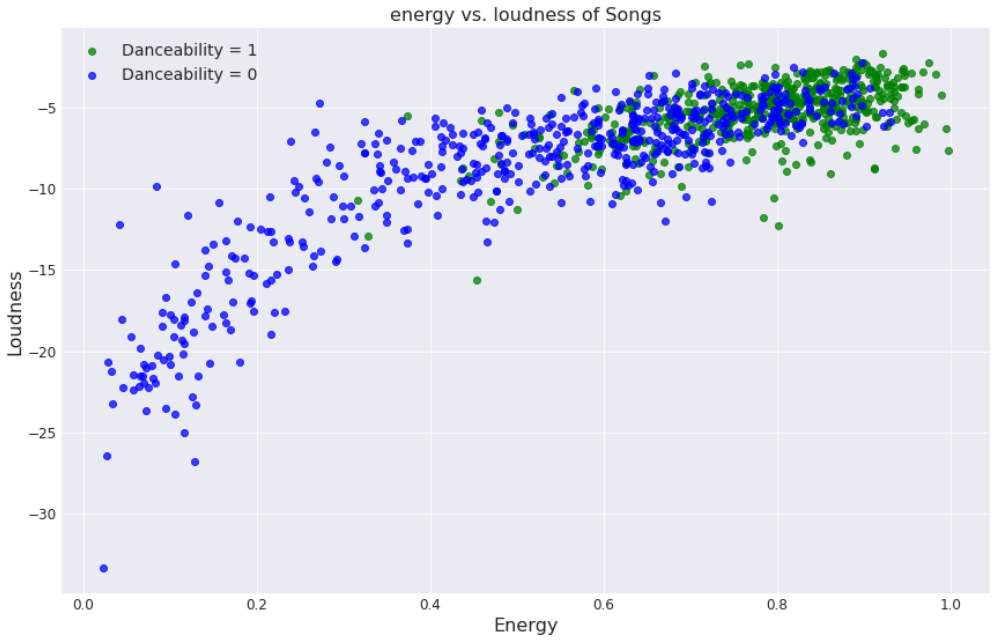

Figure 13: Energy vs loudness of songs

The scatterplot above reflects the positive correlation of energy and loudness with the danceability feature which was observed before (see figure "Features Correlating with Danceability"). The plot shows, that most songs that were classified as danceable have high energy and high loudness while songs that were not classified as danceable appear in all ranges of the loudness and energy feature. The scatterplot also reflects the very high correlation of energy and loudness of a track which was first seen in the figure "Correlation Matrix : Spotify dataset".

danceability	loudness	energy
0	4.974427	0.236122
1	1.824767	0.117732

Table 10: Standard deviation of the features loudness and energy

Similar to the previous observation from the dependence of loudness vs. acousticness of tracks we can see again from the above table: The standard deviation (square root of the variance) of the loudness is more than 15 times bigger than the standard deviation of the energy of danceable tracks.

3.4.4 Mode vs danceability

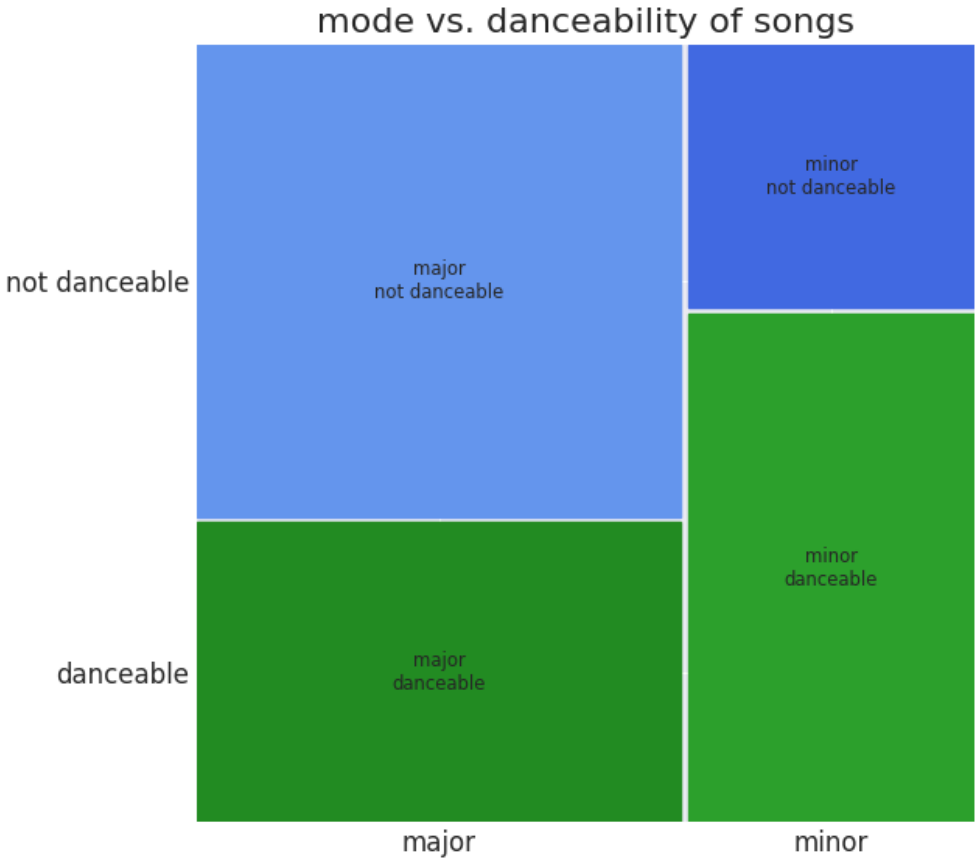

Figure 14: Mode vs danceability of songs

The given mosaic plot shows the dependence of the features mode and danceability of tracks.

From the above plot, we develop the hypothesis that the features mode and danceability are stochastically dependent since the rectangles do not have similar sizes. We want to test our hypothesis using the Chi-Square test of independence. The execution of this test will happen in the upcoming Section "Hypothesis Testing".

4 Probability Distribution

We will now analyze the features energy, loudness (high correlation with danceability), and valence (medium correlation with danceability) in more detail by finding theoretical probability distributions which fit the given distributions of the features best.

4.1 Loudness

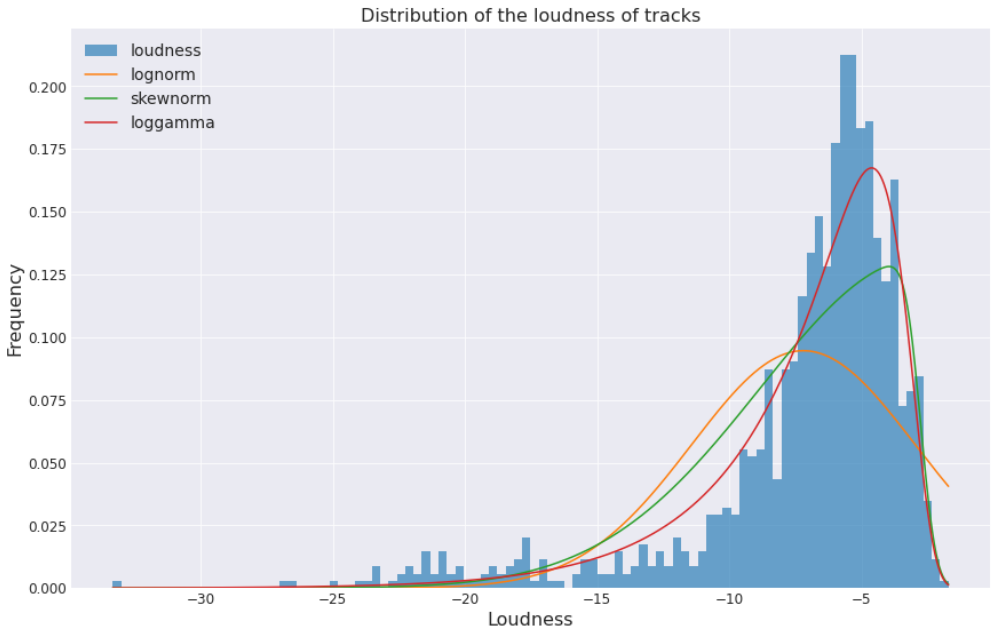

Figure 15: Distribution of the loudness of tracks

Given the histogram, we assume the factor loudness to follow a log-gamma or skew-normal distribution. To check this hypothesis we will plot a Q-Q plot for each theoretical distribution and analyse the result.

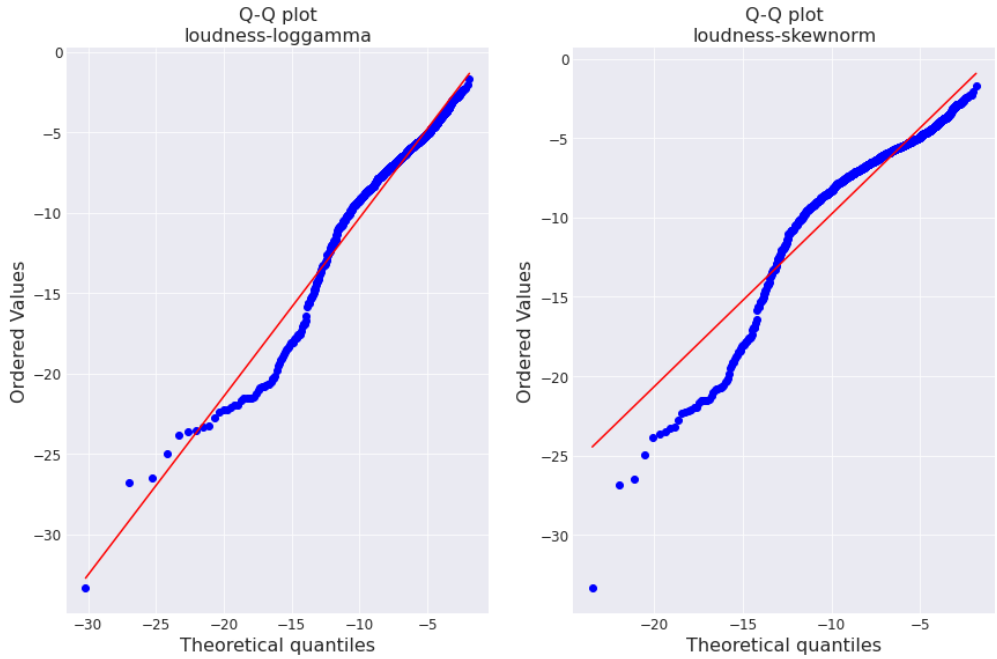

Figure 16: QQ plot of the loudness of tracks

The Q-Q plots of the quantiles of the loudness feature and quantiles from the log-gamma / skew-normal distribution show that the loudness feature is more likely to follow a log-gamma distribution since the given scatter plot follows the 45° line closer than the Q-Q plot of the loudness feature with the skew-normal distribution.

Parameter Estimation

The probability density function for the log-gamma distribution (standardized form):

$$f(x, c) = \frac{\exp(\beta x - \exp(x))}{\Gamma(\beta)},$$

for all x, $\beta > 0$ and Γ describes the gamma function. The log-gamma distribution belongs to the location-scale family and has three parameters: β (shape parameter), α (scale parameter), and μ (location parameter). We can estimate those parameters using maximum likelihood estimation MLE.

Estimated parameters given data: beta = 0.257, loc = -3.353, and scale = 0.925.

4.2 Energy

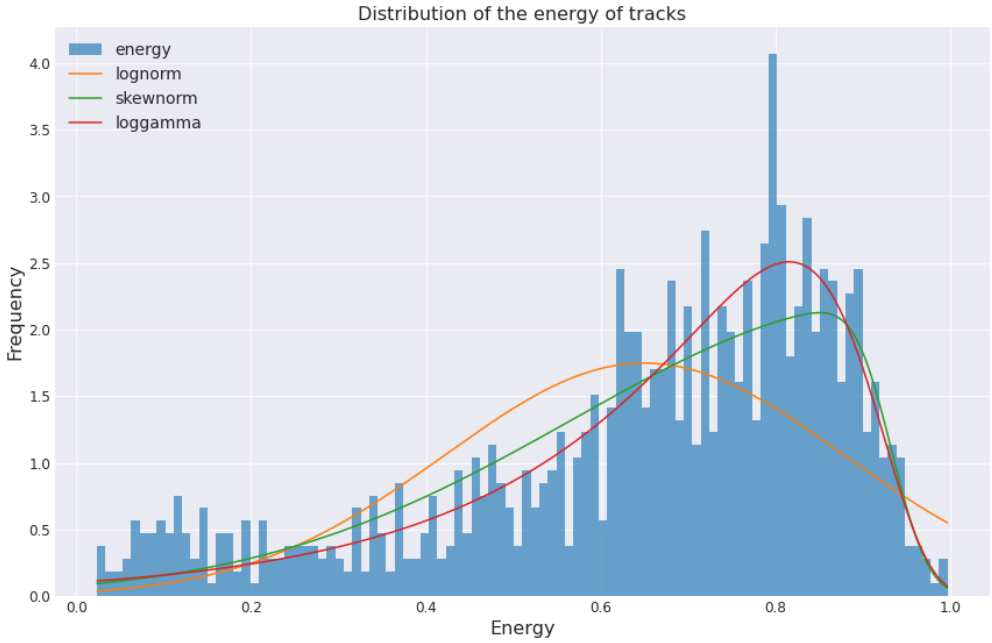

Figure 17: Distribution of the energy of tracks

Similar to the histogram of the loudness feature, the histogram of the energy of tracks is negatively skewed. Given the histogram, we assume the factor energy to follow a log-gamma or skew-normal distribution.

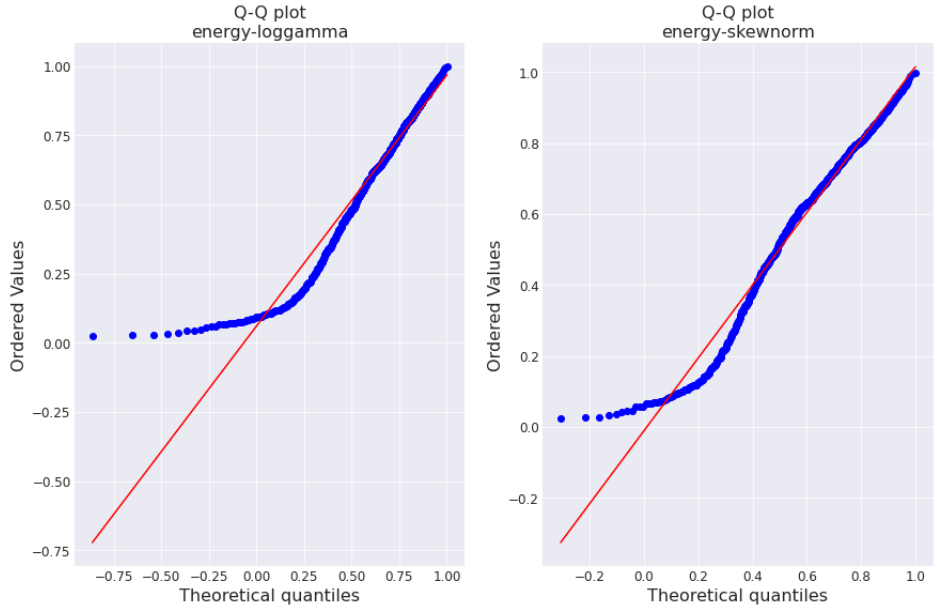

Figure 18: QQ plot of the energy of tracks

According to the given Q-Q plots we expect the energy feature to follow a skew-normal distribution.

Parameter Estimation

The probability density function for the skew-normal distribution (standardized form):

$$f(x, \alpha) = 2\frac{1}{\sqrt{2\pi}} \exp(-\frac{x^2}{2})\frac{1}{2}[1 + \text{erf}(\frac{\alpha x}{\sqrt{2}})],$$

for all $x,\ \alpha \in \mathbb{R}$ and erf describes the error function. The skew-normal distribution belongs to the location-scale family and has three parameters: α (shape parameter), ω (scale parameter), and ξ (location parameter). We can estimate those parameters using maximum likelihood estimation MLE.

Estimated parameters given data: alpha = -10.762, loc = 0.933, and scale = 0.363.

4.3 Valence

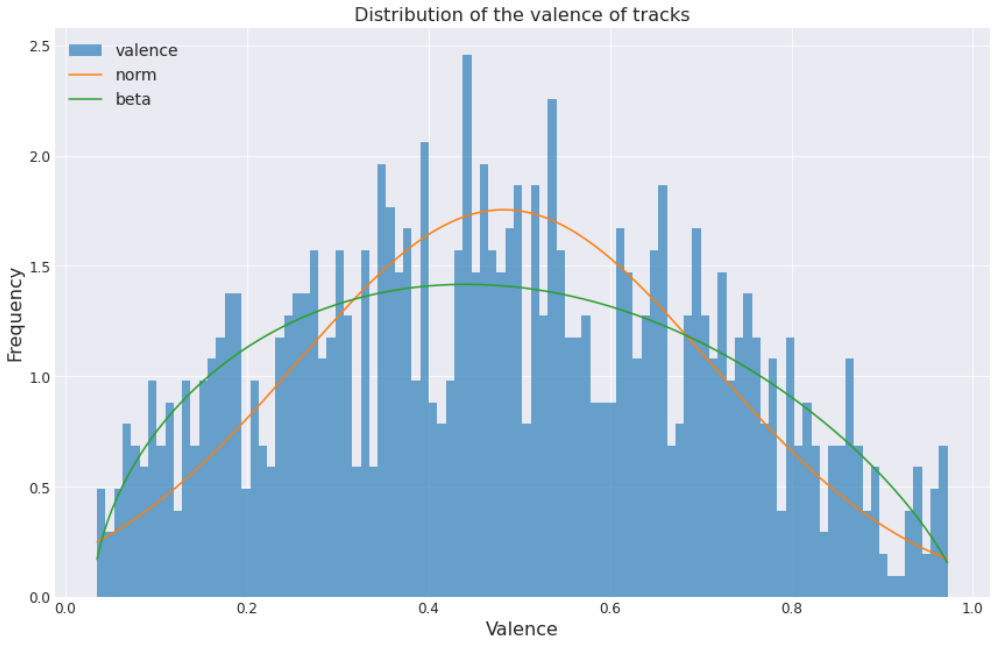

Figure 19: Distribution of the valence of tracks

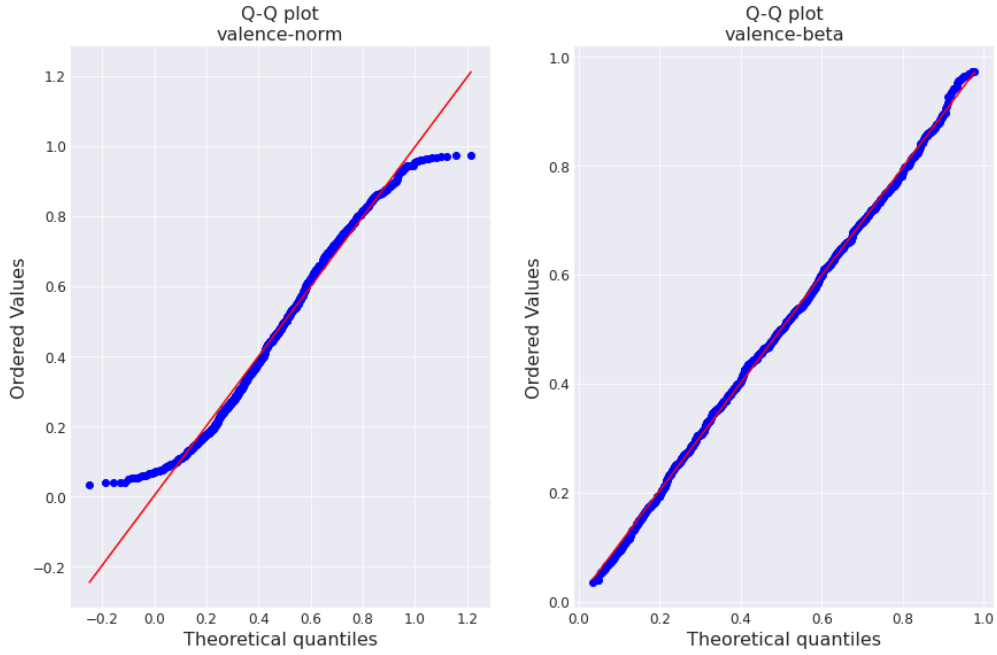

Figure 20: QQ plot of the valence of tracks

In this case, the Q-Q plot of the valence feature of tracks and the theoretical beta distribution follows almost exactly the 45° line; we expect the feature valence to follow a beta distribution.

Parameter Estimation

The probability density function for the beta distribution (standardized form):

$$f(x, \alpha, \beta) = \frac{\Gamma(\alpha + \beta) x^{\alpha-1}(1-x)^{\beta-1}}{\Gamma(\alpha)\Gamma(\beta)},$$

for all $x \in [0, 1]$, $\alpha > 0$, $\beta > 0$ and Γ describes the Gamma function. The beta distribution belongs to the location-scale family and has four parameters: α, β (shape parameters), scale parameter, and location parameter. We can estimate those parameters using maximum likelihood estimation MLE.

Estimated parameters given data: alpha = 1.565, beta = 1.749, loc = 0.030, and scale = 0.957.

5 Hypothesis testing

5.1 Mode and Danceability are independent

Chi-Square Test of Independence

When we previously took a look at the dependence of a track's mode and danceability, we formulated the following hypothesis: *"The features mode and danceability are stochastically dependent."*

We can apply the Chi-Square Test since our variables mode and danceability are categorical. In general, the purpose of the Chi-Square Test is to test if two variables are independent. In our specific case we have the following null hypothesis:

H_0: *"mode and danceability are independent."*
H_A: *"mode and danceability are stochastically dependent."*

Given the data $(x_1, y_1), \ldots, (x_n, y_n)$ corresponding to danceability (x_i) and mode (y_i) we assume (x_i, y_i) to be i.i.d. From our Spotify dataset, we have the following contingency table:

mode danceability	major	minor	All
0	417	138	555
1	265	267	532
All	682	405	1087

Table 11: Contingency table

For the variables mode and danceability (d) to be independent, the following must hold:

- $\mathbb{P}(d = 0, major) = \mathbb{P}(d = 0)\mathbb{P}(major)$
- $\mathbb{P}(d = 0, minor) = \mathbb{P}(d = 0)\mathbb{P}(minor)$
- $\mathbb{P}(d = 1, major) = \mathbb{P}(d = 1)\mathbb{P}(major)$
- $\mathbb{P}(d = 1, minor) = \mathbb{P}(d = 1)\mathbb{P}(minor)$

The left-hand side of the above equations are our parameters of interest which can be estimated using the above contingency table:

Estimate (LHS) $= \dfrac{\text{All}(d)\text{All}(mode)}{n}$, with $n = 1084$.

Our test statistic converges to a chi-square distribution under the assumption, the null hypothesis is correct. Expression of our test statistic:

$$s = \sum \frac{(O_i - E_i)^2}{E_i},$$

where: c = degrees of freedom, O = observed values, E = expected values.

Expression for the null distribution:

$$\chi_c^2 = \frac{1}{2^{c/2}\Gamma(k/2)} x^{c/2-1} \exp(-x/2).$$

Since mode and danceability have 2 classes each we have $c = 1$.

We choose the significance level $\alpha = 0.05$, i.e. an error level of 5%.

Test statistic s = 74.512
p-value p = 0.0000

The p-value is < 0.00001. Since $0.0 \approx p \ll \alpha = 0.05$ we clearly reject the null hypothesis H_0 : *"mode and danceability are independent"*.

5.2 More people dance to non-acoustic music

Two-sample t-test

When we previously took a look at the histogram of the acousticness value and later on when analyzing the dependence of a track's acousticness with its energy, loudness; and danceability, we developed the following hypothesis: *"More people dance to non-acoustic music on average."*

We can apply the two-sample t-test since we have two samples of independent test subjects (acousticness given danceability = 0 [*sample X*] and acousticness given danceability = 1 [*sample Y*]), where the two samples are independent from one another.

Our data:

- sample X : $x_1, \ldots, x_{N_X}$; $X_1, \ldots, X_{N_X}$ i.i.d.
- sample Y : $y_1, \ldots, y_{N_Y}$; $Y_1, \ldots, Y_{N_Y}$ i.i.d.

$N_X = 555$ and $N_Y = 532$.

We are interested in the mean of sample X and Y: μ_X, μ_Y. We can estimate these parameters using the sample mean $\bar{x}$ and $\bar{y}$.

For both samples; the sample size is quite large. By applying the central limit theorem we can assume that the sample mean $\bar{x}$ and $\bar{y}$ approximately follow a Gaussian distribution.

With those preparations, we develop the following null hypothesis:

H_0 : $\mu_X - \mu_Y = 0$
H_A : $\mu_X - \mu_Y \neq 0$

Our test statistic can be described as follows:

$$t_0 = \frac{\bar{x} - \bar{y}}{\sqrt{\frac{s_X^2}{N_X} + \frac{s_Y^2}{N_Y}}},$$

where s_X and s_Y describe the sample standard deviation.

The null distribution follows a Student's-t distribution with degrees of freedom df:

$$df = \frac{(s_X^2/N_X + s_Y^2/N_Y)^2}{(\frac{s_X^2}{N_X})^2/(N_X - 1) + (\frac{s_Y^2}{N_Y})^2/(N_Y - 1)}.$$

We choose the significance level $\alpha = 0.05$, i.e. an error level of 5%.

Test statistic t = 19.609
Two-tailed p-value p = 0.0000

The p-value is < 0.00001. Since $0.0 \approx p \ll \alpha = 0.05$ we clearly reject the null hypothesis H_0. This test shows, that there is sufficient evidence to support the alternative hypothesis: *"More people dance to non-acoustic music on average"*.

6 Predictive Analysis

In this section, we will apply two predictive machine learning models [5] to solve our objective of finding the best songs for a party. We will apply the following models / algorithms:

- K-Nearest Neighbors (KNN)
- Decision tree

For each of the algorithms, we will train a number of different models in order to maximize the accuracy / f1-score. In the end, we will compare the best model of each algorithm with each other and decide on one algorithm for our problem.

Mathematical Modeling for binary classification

$$y = g(x;\ \Theta \mid h)$$

$y \in \{0 :$ 'not danceable', $1 :$ 'danceable'$\}$: Categorical - Nominal data
x : numerical (with encoded categorical values)
g : classification model - KNN or Decision Tree
Θ : parameters
h : hyperparameters

6.1 Train-Validation split

We are doing an 80-20 train-validation split.

- Number of train data: 869
- Number of validation data: 218

Distribution

We now want to check if the training set and the validation set follow the same distribution by using Q-Q plots to show their relations.
Since our input vector is quite large (12 different features) we will only check on the features loudness, energy, and acousticness. Those features have the highest correlation with the danceability of tracks and, therefore, might have a big impact on the classification of a track.

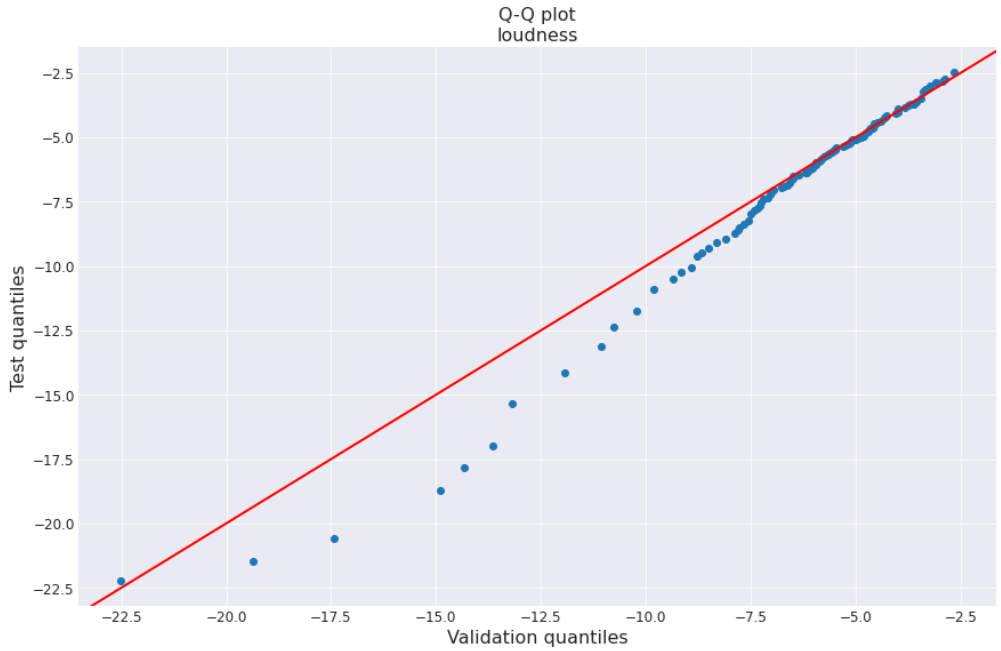

Figure 21: QQ plot for loudness

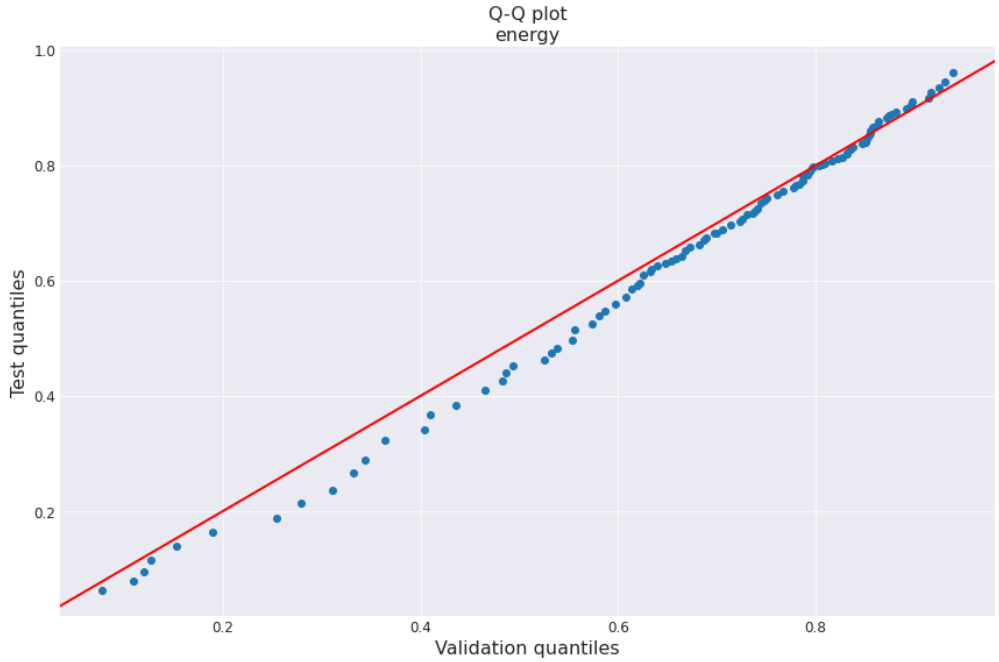

Figure 22: QQ plot for energy

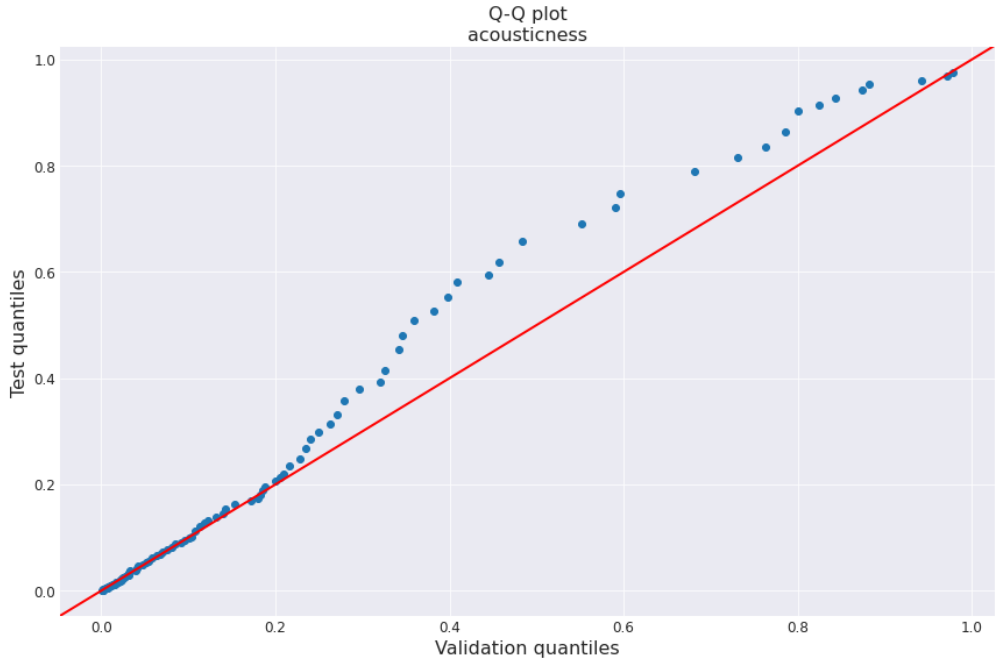

Figure 23: QQ plot for acousticness

Sample mean of the danceability feature:

- train set : 0.4822
- validation set: 0.5183

Taking the above plots into consideration, we conclude that the sample distributions of the features energy and acousticness given the validation and training set follow a similar distribution. The quantiles of the energy and acousticness feature for the train and validation set follow the 45° line quite well. On the other hand, the loudness feature quantiles do not follow the 45° line that well. Especially for lower quantiles, the validation set seems to differ quite a bit from the test set given the feature loudness.

However, looking at the danceability feature, about half of our data is danceable in both sets - the sample mean is close to 0.5 for each. Even though the Q-Q plot of the loudness factor does not follow the 45° line as precise, we still conclude that our validation dataset and our test dataset follow the same distribution, since the other features considered seem to follow similar distributions.

6.2 Models

6.2.1 K - Nearest Neighbors

We will use the k-Nearest Neighbors algorithm for classification. In this algorithm, classification is computed from a simple majority vote of the nearest neighbors of each point: a point is assigned danceable if most (nearest) neighbors of the point have the label "danceable" (danceability = 1).

Description

Name
k-Nearest Neighbors

Mathematical expression
Given a training set $x_0, \ldots, x_N$ with labels $y_0, \ldots, y_N$ the label y of a new data point (represented by the d-dimensional feature vector $x = [x^0, \ldots, x^{d-1}]$) can be calculated the following way:

Given the set $D_x = \{\tilde{x}_0, \ldots, \tilde{x}_{K-1}\}$ of the K-nearest neighbors of x [see *parameter estimation* on how to compute those] and the corresponding set of labels $\tilde{y}_0, \ldots, \tilde{y}_{K-1}$ it is

$$y = \underset{s \in \{0,1\}}{\arg\max} \sum_{i=0}^{K-1} w_i I_s(y_i),$$

where I_s for $s \in \{0, 1\}$ describes the so-called indicator function

$$I_s : \{0,1\} \to \{0,1\},\ I_s(\iota) = \begin{cases} 0, & \iota \notin \{s\} \\ 1, & \iota \in \{s\} \end{cases}.$$

Further, w_i describes the weight for each neighbor x_i. We will see an explanation of the weighting in the upcoming part.

Hyperparameters

- n_neighbors - the number of neighbors
- weights ∈ { 'uniform', 'distance' } or callable - weight function
- metric: str or callable - metric function

Parameters
table of actual nearest neighbors

For the *weight function* we have the following options:

- 'uniform' : uniform weights. All points in each neighborhood are weighted equally: $w_i = 1$
- 'distance' : weight points by the inverse of their distance → closer neighbors of a point have a greater influence on the classification of that point: $w_i = 1 - \text{dist}(x, \tilde{x}_i)$
- callable: a user-defined function which accepts an array of distances, and returns an array of the same shape containing the weights.

In the following we will always choose both weights, 'uniform' and 'distance', and compare the result for each model.

As *metric* we are choosing the euclidean distance/ minkowski metric since this metric seems the most fitting for our kind of data as we want to compute the absolute distance of different data points (given their feature values).

Parameter estimation
We can estimate the table of nearest neighbors by minimizing the following loss function:

$$L(\hat{x}_0, \ldots, \hat{x}_K) = \sum_{i=0}^{K-1} \text{dist}(\hat{x}_i, x),$$

where $\hat{x}_i \in \{x_0, \ldots, x_N\}$ and $\hat{x}_i \neq \hat{x}_j$ for all $i, j \in \{0, \ldots, K-1\},\ i \neq j$. Finally, the set of the K-nearest neighbors can be described as

$$D_x = \{\tilde{x}_0, \ldots, \tilde{x}_{K-1}\} = \underset{\hat{x}_0,\ldots,\hat{x}_K}{\arg\min} L = \underset{\hat{x}_0,\ldots,\hat{x}_K}{\arg\min} \sum_{i=0}^{K-1} \text{dist}(\hat{x}_i, x).$$

When computing a KNN classifier we will not set the value for K (number of neighbors) manually but instead, choose the best value of K by calculating accuracy and f1-score and choosing the K value for which the model has the best scores.

All features

We will start with KNN classifiers that only differ in the chosen weight function. In the following, we will fit 30 models for each, 'distance' and 'uniform' weights. Since we are dealing with a binary classification problem we will only choose an odd number of neighbors to prevent draws of neighboring labels.

For each weight label, we decide on the best model/ the best K by choosing the K value for which the accuracy/ f1-score is the highest.

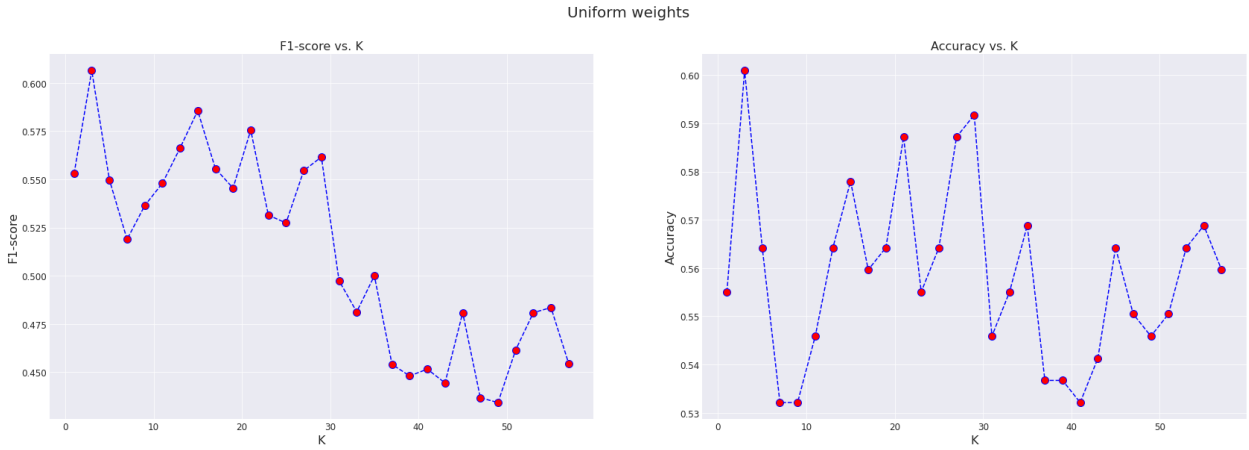

Figure 24: Performance of KNN models with uniform weights

- Best model scores at K = 3
- F1-score = 0.6063

- Accuracy = 0.6009

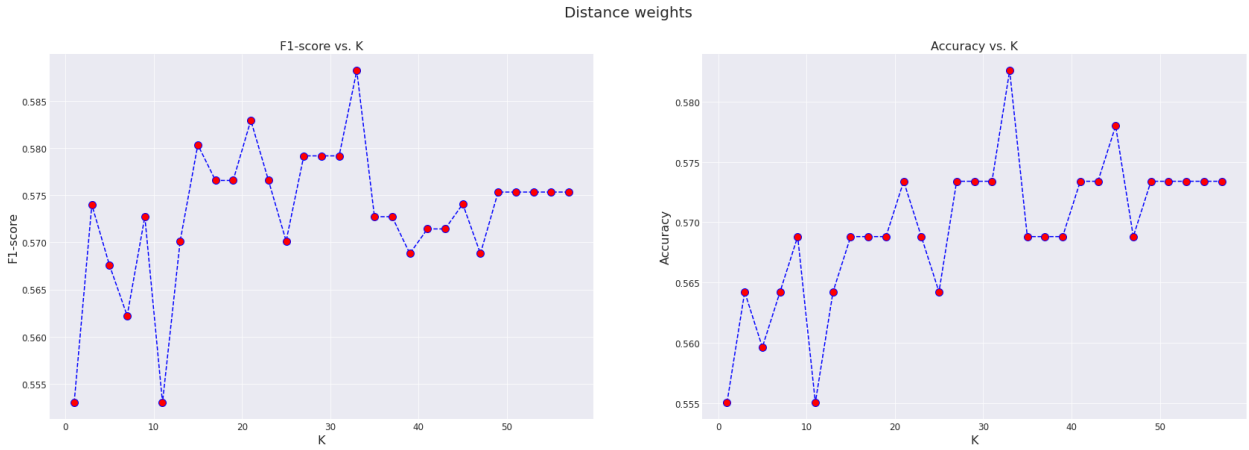

Figure 25: Performance of KNN models with distance weights

- Best model scores at K = 33
- F1-score = 0.5882
- Accuracy = 0.5826

When training KNN models on all features we only get maximum accuracy and f1-score of below 60% for both, distance and uniform weights. These low scores might be due to the fact that only the K nearest neighbors decide on the classification of a sample point and not all points. It is not guaranteed that the K neighbors are the best points to decide on the danceability of the current sample point.
When considering all 12 features we put our sample points into a 12-dimensional space. Some of those dimensions, e.g. features that have a low correlation with danceability, might not have a big impact on the model and might only make the model more complicated, or worse, be contra-productive and lead to a wrong classification.

To test this **hypothesis**: *Not all features are needed for classification / fewer features lead to a more accurate model*, we will train several KNN models on fewer features [in total 10 different test-cases for each weight function]. We decide on the features by sorting those by their absolute correlation with danceability. We then train models with different numbers of features with decreasing correlation. By this method, we hope to find a few specific features for which the KNN classifier works best.

Sorted feature list: ['energy', 'acousticness', 'loudness', 'mode', 'valence', 'duration_ms', 'liveness', 'tempo', 'instrumentalness', 'time_signature', 'key', 'speechiness'].

Again, for each number of features, we will train models for 30 different K values and pick the one model with the highest accuracy / f1-score. Out of those 10 best models we pick that number of features that maximizes the accuracy / f1-score for each weight function.

We will call the resulting best model *"distance model"* or *"uniform model"*, depending on the applied weight function.

Selected features

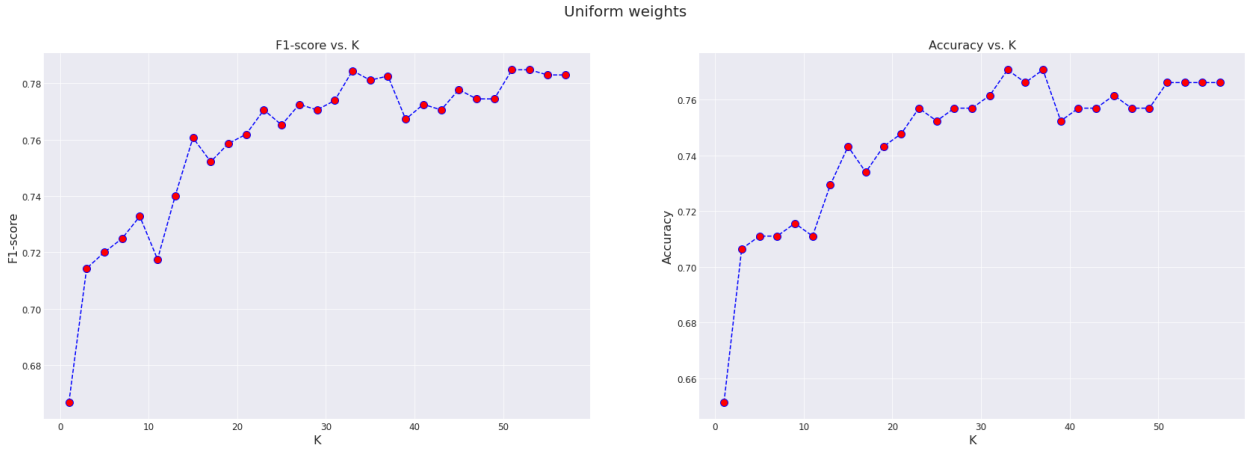

Figure 26: Performance of KNN models - two features - with uniform weights

- Best model scores for 2 features
- Best model scores at K = 33
- F1-score = 0.7845
- Accuracy = 0.7706

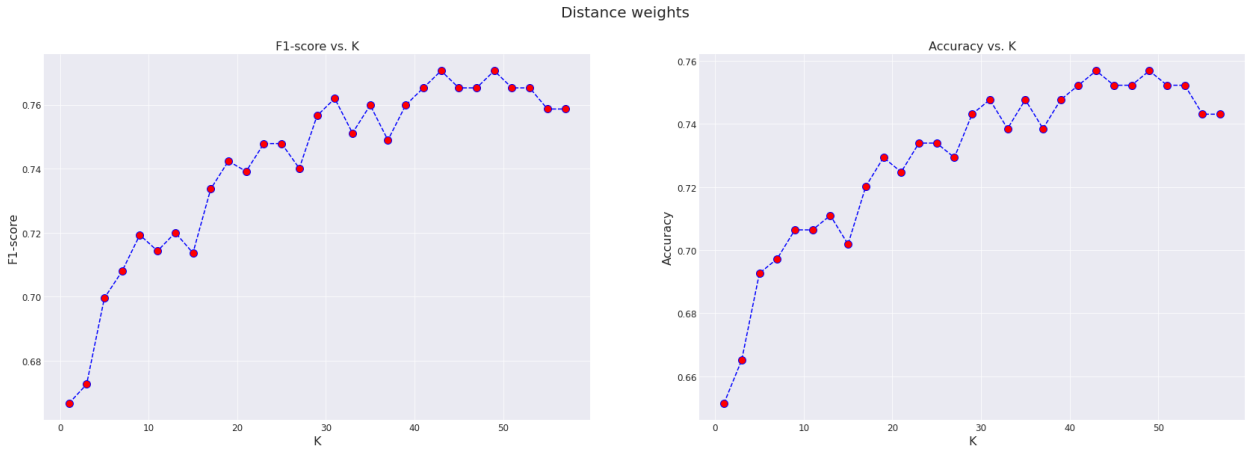

Figure 27: Performance of KNN models - two features - with distance weights

- Best model scores for 2 features
- Best model scores at K = 43
- F1-score = 0.7706
- Accuracy = 0.7569

As we can see above, for both cases (uniform and distance weights) we get a much better model using only 2 features: energy and acousticness *(based on the increased accuracy and f1 score)*. For uniform weights, the KNeighborsClassifier has the highest accuracy (+ f1 score) for 2 features.

Overall, these results support our hypothesis: *"Not all features are needed for classification / fewer features lead to a more accurate model"*. We can therefore drop the previous 'best' models trained on all features and choose the following model as the final model using the KNN algorithm:

- metric = euclidean metric
- features = ['energy', 'acousticness']
- weights = 'uniform'
- K = 33

We have decided on 'uniform' and not on 'distance' weights for two reasons:

1. Both accuracy and f1-score are about 1-3% higher using the uniform model
2. According to Occam's razor philosophy, simpler models are preferred: the uniform model has both, a lower K value and a simpler calculation of the weights : $w_i = 1$, which makes this model simpler than the distance model.

The following figure shows the confusion matrix, including accuracy and f1-score for the best model using the KNN algorithm.

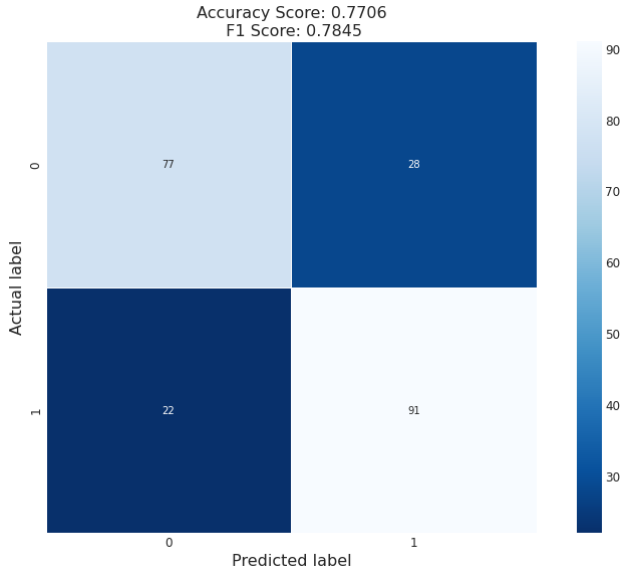

Figure 28: Confusion matrix for the KNN algorithm

The confusion matrix above shows that the final KNN model misclassifies 50 out of 218 songs, with similar amounts of type 1 and type 2 errors.

6.2.2 Decision Tree

Description

Name
Decision tree

Mathematical expression
Given a training set $x_0, \ldots, x_N$ with labels $y_0, \ldots, y_N$ the label y of a new data point (represented by the d-dimensionl feature vector $x = [x^0, \ldots, x^{d-1}]$) can be calculated the following way: For a tree with M many decision regions R_m (leaf nodes) it is

$$y = \sum_{m=1}^{M} c_m I_{R_m}(x) \in \{0, 1\},$$

where I_{R_m} for $m \in \{1, \ldots, M\}$ describes the so-called indicator function

$$I_m : R_m \to \{0, 1\},\ I_m(\iota) = \begin{cases} 0, & \iota \notin R_m \\ 1, & \iota \in R_m \end{cases},$$

and $c_m \in \{0, 1\}$ is the class index for the given decision region (danceable or not).

To be noted: $\forall x\ \exists!\ m \in \{1, \ldots, M\} : x \in R_m$.

Hyperparameters

- criterion - function to measure the quality of a split: gini (Gini impurity), entropy (information gain)
- splitter - strategy used to choose the split at each node
- max_depth - maximum depth of the tree
- min_samples_split - minimum number of samples required to split an internal node
- min_samples_leaf - minimum number of samples required to be at a leaf node
- min_weight_fraction_leaf - minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node
- max_features - number of features to consider when looking for the best split: integer, float, sqrt, log2
- random_state - controls the randomness of the estimator
- max_leaf_nodes - maximum number of leaf nodes
- min_impurity_decrease - a node will be split if this split induces a decrease of the impurity greater than or equal to this value
- class_weight - weights associated with classes
- ccp_alpha - complexity parameter used for Minimal Cost-Complexity Pruning: the subtree with the largest cost complexity that is smaller than ccp_alpha will be chosen

Parameters

- All non-leaf nodes

- Decision regions R_m (leaf nodes) for $m \in \{1, \ldots, M\}$
- A class index c_m for each region R_m

Parameter estimation

- Iterative Dichotomiser 3 (ID3): creates a multiway tree, finding for each node the categorical feature that will yield the largest information gain for categorical targets
- C4.5: converts the trained trees (i.e. the output of the ID3 algorithm) into sets of if-then rules, the accuracy of each rule is then evaluated to determine the order in which they should be applied
- C5.0: It uses less memory and builds smaller rulesets than C4.5 while being more accurate
- Classification and regression tree - CART: it supports numerical target variables (regression) and does not compute rule sets, CART constructs binary trees using the feature and threshold that yield the largest information gain at each node

Algorithm	ID3	C4.5	C5.0	CART
Variable type	Categorical	Continuous and categorical	Continuous and categorical	Continuous and categorical
Target type	Categorical	Categorical	Categorical	Continuous and categorical
Splitting score	Information gain	Information gain	Information gain	Gini impurity
Computational efficiency	Low	Okay	High	Good

Table 12: Comparison of parameter estimation algorithms
Source: lecture slide 11

First we are going to create models for different tree depth values without setting any other hyperparameters. We will train the models for a maximum depth of 2 to 20 and pick the model with the highest accuracy / f1-score.

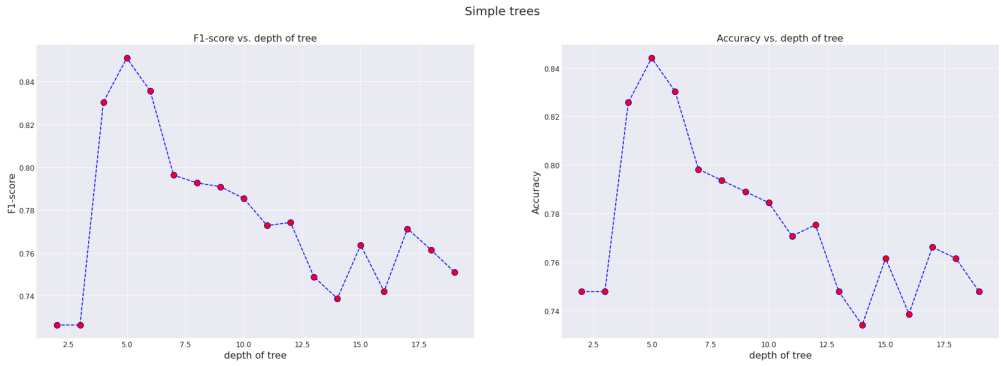

Figure 29: Performance of Decision tree models

The above plots show the f1-score and accuracy of decision tree models for different tree depths. All of the models have an f1-score and accuracy above 70% and the best result is obtained at a tree depth of 5 with

both metrics above 84%. With a depth of 5, the model is not that complex and there are less splits compared to a deeper tree.

In the next part we are going to tune the hyperparameters of the decision tree model to see if we can improve the model and get a better f1-score / accuracy.

We are trying out different values for the min_samples_leaf and min_samples_split parameters since these stopping criteria are some of the most common ones that are tuned when building a Decision Tree.

By building models with different combinations of values for these two hyperparameters, we obtained the best metrics with min_samples_leaf=5 and min_samples_split=25 at a maximum tree depth of 5.

Next, we are going to plot the f1-score and accuracy for different tree depths to see how the models perform.

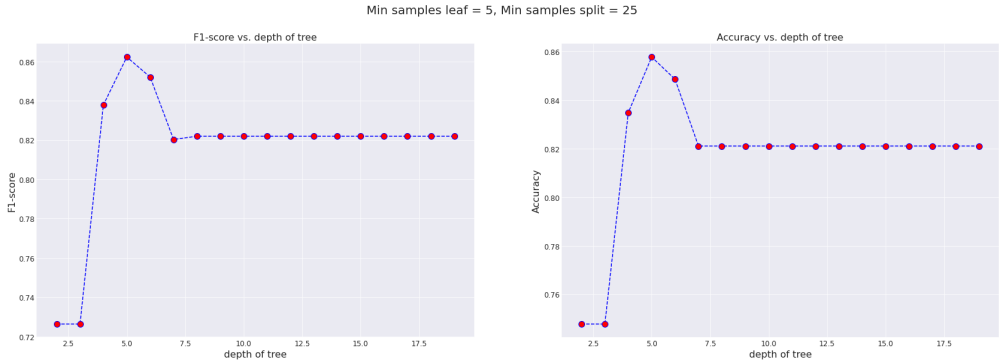

Figure 30: Performance of Decision tree models with set hyperparameters

As we can see, the best model has metrics above 85% and the models with a tree depth above 7 have the same f1-score and accuracy. This can be the result of setting the other hyperparameters: the minimum number of samples required to split an internal node (min_samples_split) and to be at a leaf node (min_samples_leaf).

The following figure shows the confusion matrix, including accuracy and f1-score for the best model using the decision tree algorithm. The model correctly classified most songs with a similar amount of type 1 and type 2 errors.

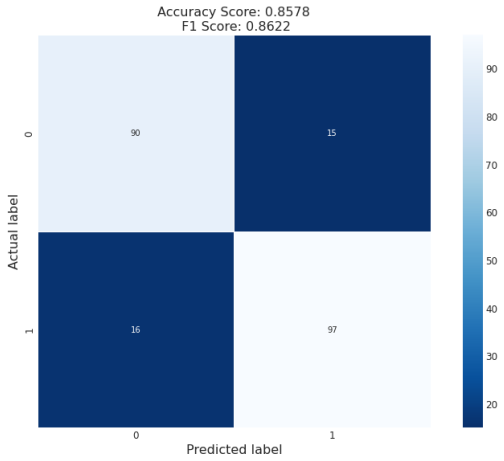

Figure 31: Confusion matrix of the Decision tree model

Decision tree visualization

The following figures show our final decision tree (Fig 32) with a tree depth of 5 and the decision path of one example song (Fig 33).

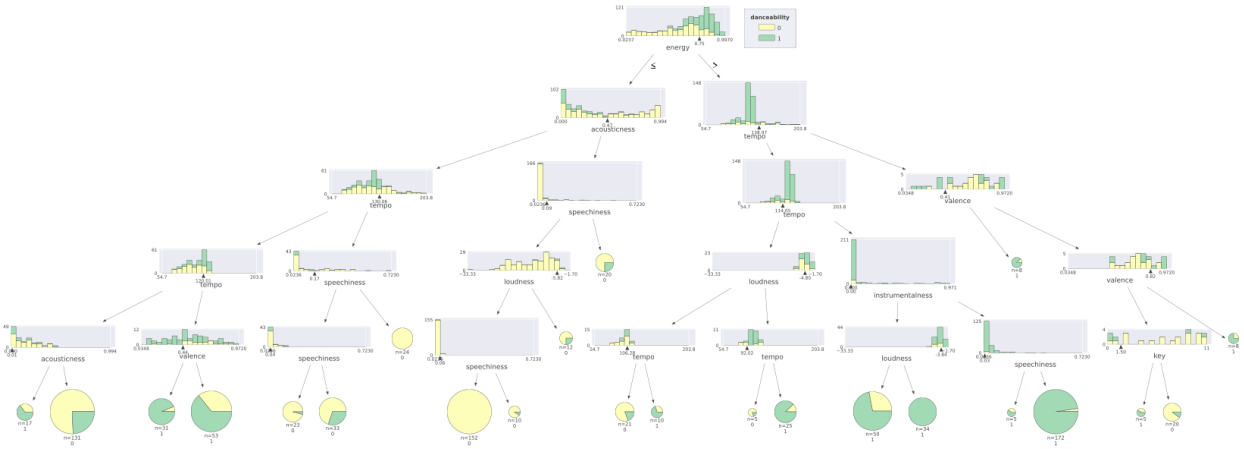

Figure 32: Decision tree

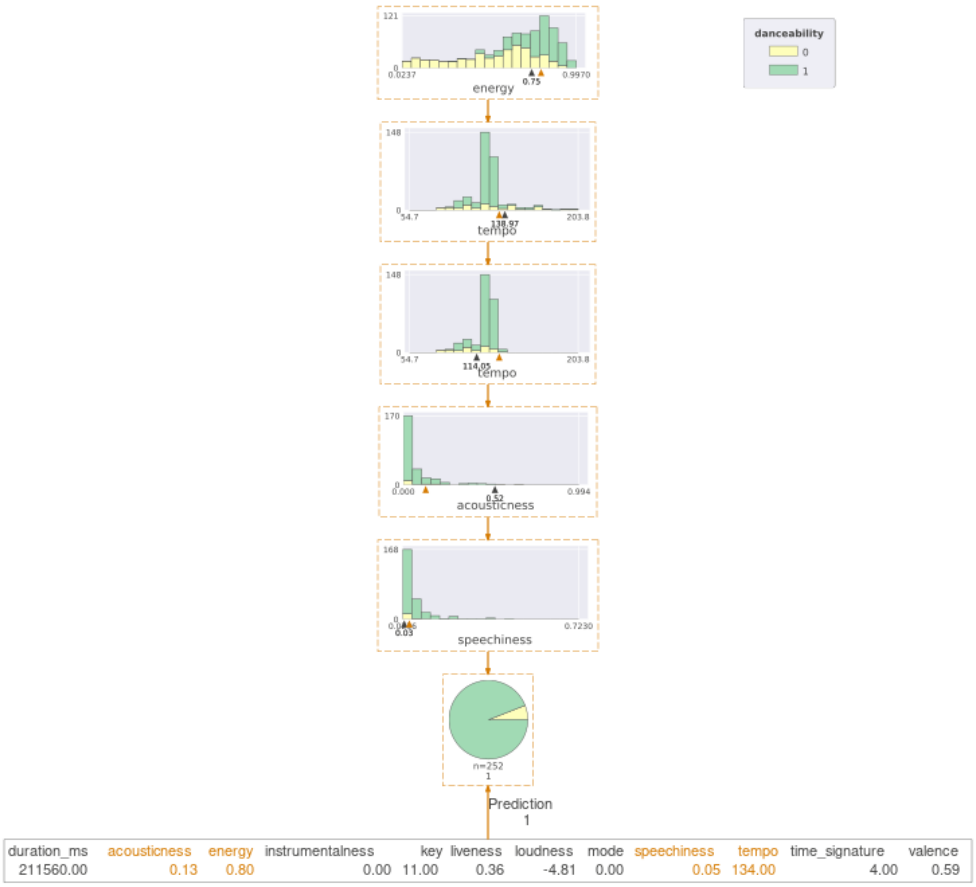

Figure 33: Decision tree of an example song

To listen to the song: `https://open.spotify.com/track/3Wrjm47oTz2sjIgck11l5e`

Artist: Måneskin

Track: Beggin'

6.3 Testing

We will now finalize our classification model in order to make predictions on new, unseen data. We have previously trained two final models given two different algorithms:

K-Nearest Neighbors

- metric = euclidean metric
- features = ['energy', 'acousticness']
- weights = 'uniform'
- K = 33

Accuracy : 0.7706 F1-Score : 0.7845

Decision Tree

- max_depth = 5
- min_samples_leaf = 5
- min_samples_split = 25

Accuracy : 0.8578 F1-Score : 0.8622

As our final model we are choosing the best model of the decision tree algorithm for the following two reasons:

1. Both accuracy and f1-score are about 7-9% higher for the decision tree model

Taking a step back and having a look at the results of the Descriptive Analysis part, we have seen, that many features are dependent / correlated to the danceability of a track. This observation leads us to the second reason why we choose the decision tree model over the KNN model:

2. The best KNN model only takes two features into account. Since we believe that most features are important for the classification of a song we prefer the decision tree model because all features are taken into consideration during training

In the following, we will test the final model, *Decision Tree*, using the test dataset and evaluate the performance.

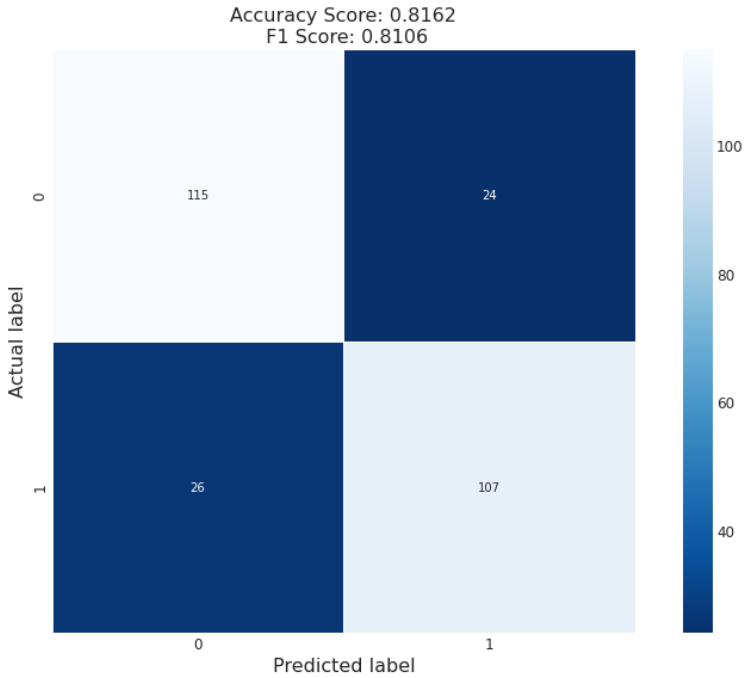

Figure 34: Confusion matrix for the test data

The chosen model using the decision tree algorithm performs well on unseen data / our test dataset. When applying the model to our test data we get accuracy and f1-score of above 80%. From the above scatterplot, we can derive that 49 out of 272 songs were misclassified, with similar amounts of type 1 and type 2 errors.

7 Conclusion

In this project, we have worked with Spotify data in order to find the best songs for a party. The collected Spotify data consists of in total 1359 songs:

- training: 1087 tracks [869 for training and 218 for validation]
- testing: 272 tracks

For each song we collected 12 different song features: 'energy', 'acousticness', 'loudness', 'mode', 'valence', 'duration_ms', 'liveness', 'tempo', 'instrumentalness', 'time_signature', 'key', 'speechiness', and decided on a ground truth label for the 'danceability' feature based on the playlist we derived the song from.

In order to train the best model given our objective and the available time, we extensively analyzed / visualized the dataset using different methods:

- histograms
- correlation matrices
- bar plots and pie charts
- dependence scatterplots
- dependence mosaic plots

During the visualization of the training dataset, we started to develop various hypotheses and tested a few of those using for example the Chi-Square test of independence (mode and danceability are independent) and the two-sample t-test (more people dance to non-acoustic music). In both cases, we failed to reject the null hypothesis given a significance level of 0.05.

To get a better grasp of some song features, i.e. loudness, energy, and valence, we tried to find a theoretical distribution that describes the data / feature best. In particular, for the valence feature, we found a theoretical distribution: the beta distribution, which fits the data very well.

In the last part of this report, we applied two predictive machine learning models, *K-Nearest Neighbors* and *Decision Trees*, to solve our problem of finding the best songs to dance to. Based on the performance of each model and our domain knowledge, we decided on one final model. The final model, using the decision tree algorithm, performed quite well on our test dataset / on unseen data with an accuracy of about 82%.

Due to the high accuracy score, it can be said, that our model, to classify a song as danceable or not, is reliable. Because of the great performance, we would recommend the model to party organizers and artists / producers as mentioned in the introduction.

7.1 Next steps

If we would have more time for the project, there are several ways to improve the two predictive machine learning models:

K-Nearest Neighbors:
Since we are dealing with mixed categorical and numerical values we could improve the KNN algorithm by creating our own distance function which takes into account that some values are categorical.

Decision Tree:
To improve the model using the decision tree algorithm, we could also tune different hyperparameters other than the ones we tried, for example, the criterion, splitter or max_features. We could also apply the ensemble method *random forest*. Compared to decision trees, random forest results in significantly lower variance with slightly increased bias and is, in general, less interpretable.

Not only the model training / parameter estimation can be improved but also the initial **data collection:** As mentioned before, our data collection and especially the creation of ground truth labels does show some flaws. Instead of deciding on a danceability label based on the playlist we derive a song from, we could decide on a ground truth label based on other metrics: For example, one could count the people on the dance floor of a party and decide on a danceability label for each song based on the number of people dancing. Another way could be, to gather survey data where participants can label a song as danceable or not-danceable.

The two described metrics above could also be used to **evaluate the performance** of our model, given the objective of finding the best songs for a party. If possible, we would like to collect feedback from people at the party by either building a simple app where people can react to each song with an emoji - thumbs up / down or observe, how many people are on the dance floor.

Further, an improvement could be, to simply enrich the **dataset**. We would like to collect more songs (10,000 and above) from various categories and genres to improve our classification model.

As said in the "Setup" part, our dataset is *not* valid forever and needs constant updates. Due to the ever-changing music, songs that are danceable now might not be classified as danceable forever. In order for this model to be valid for longer, one needs to constantly update the dataset and train the model with new, changed data.

References

[1] What is spotify? `https://support.spotify.com/us/article/what-is-spotify/`. (Last visited: January 5, 2022).

[2] Spotify for developers. `https://developer.spotify.com/`. (Last visited: January 5, 2022).

[3] Healthlinkbc. `https://www.healthlinkbc.ca/health-topics/tf4173`. (Last visited: January 5, 2022).

[4] Time signatures explained. `https://www.skoove.com/blog/time-signatures-explained/`. (Last visited: January 5, 2022).

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.