DM1 Project

Marco Del Pistoia, Francesco Longobardi, Andrea Mazzoni

AY 2023/24 Fall Semester

Contents

Chapter 1

Data understanding and preparation

1.1 Data semantics

The dataset on which we have worked and which we will present during this paper is the result of an extraction of 20,000 audio tracks, divided between training sets and test sets. The first is made up of 15,000 traces, while the test set is made up of the remaining 5,000. Both the training and test corpus share the same features.

The total number of features we were able to work on, contained in the starting dataset, is 23. This information covers a wide range of musical aspects and song-related metadata. The dataset includes key information such as the name of the track, its duration in milliseconds and whether it contains explicit lyrics or not. Track popularity is rated on a scale of 0 to 100, giving an indication of how popular it is. The artists involved in the making of the song are listed, allowing for musical collaborations to be displayed.

Variables	**Meaning**	**Type**
Name	The title of the track	String
Duration ms	The length of the track in milliseconds	Integer
Explicit	Indicates whether the track contains explicit lyrics (true = yes; false = no or unknown)	Boolean
Popularity	A value ranging from 0 to 100, representing the track's popularity, with 100 being the most popular	Integer
Artists	The names of the artists who performed the track. If there are multiple artists, their names are separated by semicolons	String
Album name	The name of the album in which the track appears	String
Danceability	Describes how suitable the track is for dancing, with a value of 0.0 being the least danceable and 1.0 being the most danceable	Float
Energy	A measure from 0.0 to 1.0 that represents the intensity and activity of the track	Float
Key	Indicates the musical key of the track, with integers mapping to pitches using standard Pitch Class notation	Integer
Loudness	The overall loudness of the track in decibels (dB)	Float
Mode	Indicates the modality of the track, with 1 representing major and 0 representing minor	Integer
Speechiness	Detects the presence of spoken words in the track	Float
Acousticness	A confidence measure from 0.0 to 1.0 of whether the track is acoustic, with 1.0 indicating high confidence that the track is acoustic	Float
Instrumentalness	Predicts whether the track contains no vocals, with a value closer to 1.0 suggesting instrumental music	Float
Liveness	Detects the presence of an audience in the recording	Float
Valence	A measure from 0.0 to 1.0 describing the positivity conveyed by the track's music	Float
Tempo	The estimated tempo of the track in beats per minute (BPM)	Float
Features duration ms	The duration of the track in milliseconds	Integer
Time signature	An estimated time signature for the track	Integer
N beats	The total number of time intervals of beats throughout the track	Integer
N bars	The total number of time intervals of bars throughout the track	Integer
Popularity confidence	The confidence, ranging from 0.0 to 1.0, in the track's popularity	Float
Genre	The genre to which the track belongs	String

Table 1.1: Below are all the columns of the dataset with a brief explanation of their content and the type of the variable is also reported.

Details about the album in which the track appears are included, along with musical feature information. The latter cover aspects such as danceability, energy, tonality, sound volume and musical modality (major or minor). Other aspects such as the presence of spoken words, acoustics, the instrumentality index and the detection of an audience presence during recording further enrich the understanding of the tracks.

Features relating to the tempo, number of beats and number of measures in the song offer details on the rhythmic structure. Confidence in song popularity and musical genre complement this rich data set, offering rich insights for analysis and musical purposes.

The variables of the dataset can be divided in two macro-categories, including respectively all the fundamental attributes of each track and audio-derived attributes (e.g. energy, danceability).

By splitting the dataset as above, we will have a first part that includes text-based information, such as the name of the track, the names of the artists involved and the name of the album in which the song appears. This data provides essential details about the identity and ownership of music tracks. The variability of track and artist names allows you to explore the diversity of musical works, while the album name links the tracks to the context of a larger project.

The second part of the dataset includes features of different types, including numeric, Boolean and float data. These features cover musical and technical aspects, such as track duration, popularity, danceability, energy, tone, modality, presence of spoken words, acoustics and many others. This data offers a deep understanding of the sonic, rhythmic and emotional characteristics of musical tracks, as well as technical details such as tempo, number of beats and measures. This part of the dataset is fundamental for detailed analyzes and musical classifications.

The dataset includes 20 musical genres equally distributed within the dataset, making it uniform and covering almost all musical genres, thus obtaining a well-distributed dataset.

Number	Musical genre
0	afrobeat
1	black-metal
2	bluegrass
3	brazil
4	breakbeat
5	chicago-house
6	disney
7	forro
8	happy
9	idm
10	indian
11	industrial
12	iranian
13	j-dance
14	j-idol
15	mandopop
16	sleep
17	spanish
18	study
19	techno

Table 1.2: Subdivision of the dataset according to musical genre

1.2 Distribution of the variables and statistics

Since the dataset contains 19 numerical variables, this report analyzes two of the most important variables for analyzing songs: their duration in milliseconds (*durations_ms*) and their popularity (*popularity*). The observations made for these two variables can be made for all the other variables: *danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, features_duration_ms, time_signature, n_beats, n_bars, popularity_confidence* and *processing*.

The figure below on the left shows the main statistics for the two variables considered, while the one on the right represents the frequency distributions of *duration_ms* and *popularity*. The two distributions have, respectively, on the x-axis the duration of the songs in milliseconds and the popularity score, while on the y-axis the number of songs with the duration and popularity score indicated by the values of the corresponding x-axis.

	duration_ms	popularity
count	1.500000e+04	15000.000000
mean	2.468075e+05	27.423667
std	1.279941e+05	18.587534
min	8.586000e+03	0.000000
25%	1.800000e+05	14.000000
50%	2.278260e+05	24.000000
75%	2.889030e+05	42.000000
max	4.120258e+06	94.000000

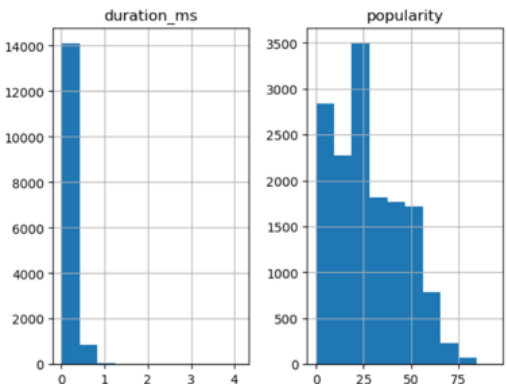


Figure 1.1: Main statistics and frequency distributions of *duration_ms* and *popularity*

The songs in the dataset have an average duration of approximately 4.11 minutes (2.468075e+05 ms). The longest song is "Ocean Waves Sounds" with a duration of approximately 69 minutes (4.120258e+06 ms), while the shortest is "The Exorsism Begins..." with a duration of approximately 9 seconds (8.586000e+03 ms) . 25% of the songs last less than 3 minutes (1.800000e+05 ms), half last less than approximately 4 minutes (2.2728260e+05 ms) while 75% last less than approximately 5 minutes (2.889030e+05 ms). The observations deviate, on average, by 2.13 minutes (1.279941e+05 ms) from the average.

The "popularity" variable, which takes values between 0 and 100, has an average of 27.4. The most popular song is "Clean White Noise – Loopable with no fade" with a value of 94. It could be said that most of the songs in the dataset (75%) are not popular, as they take values lower than 42. The observations deviate, on average, by 18.59 from the average. It is also interesting to divide the dataset based on musical genres. From this subdivision we first notice that there are 20 different genres, each of which has exactly 750 songs (20 x 750 = 15000 = number of total observations of the dataset). The most popular genre (highest average *popularity* value) is "Indian", while the one whose songs last the longest (highest average *duration_ms* value) is "chicago-house".

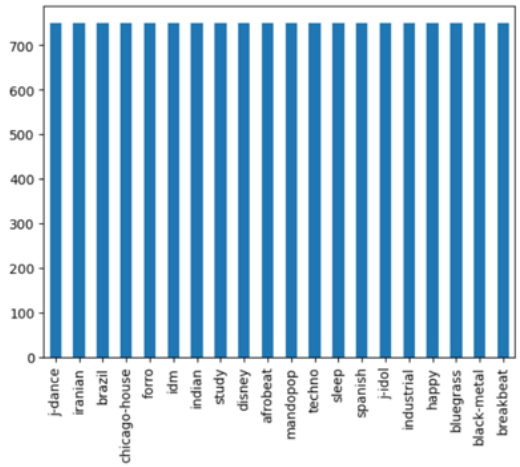


Figure 1.2: N° songs for each genre

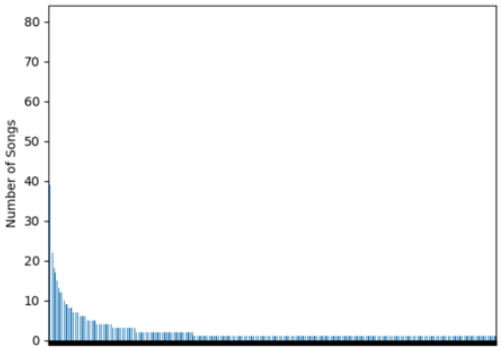


Figure 1.3: N° songs each artist made

Furthermore, multiple songs were made by the same artist, as can be seen in the figure 1.3. In fact, the dataset includes 15,000 songs produced by 6,257 different artists. For greater clarity of display, the artist names in the x-axis have been removed.
The artist with the most songs in the dataset is Vybz Franco with 80. Followed by Germaine Franco with 75; Scooter with 74; Sarah, the Illstrumentalist, with 69; Jay Chou with 67 and so on.

In addition to what has generally been observed so far, it is possible to distinguish the two values that the binary variable *explicit* can take on: True, if the track contains explicit lyrics; False, if the track does not contain them. As the following pie chart shows, 93.6% of the tracks (14034) doesn't contain explicit lyrics, while the remaining 6.4% (966) does.

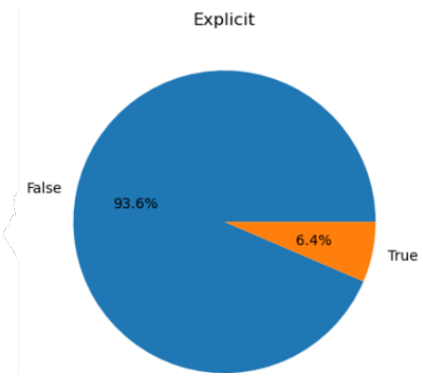


Figure 1.4: Distribution of *explicit*

	duration_ms	popularity
count	9.660000e+02	966.000000
mean	2.230383e+05	26.495859
std	9.042020e+04	19.303882
min	5.402600e+04	0.000000
25%	1.725838e+05	15.250000
50%	2.027760e+05	23.000000
75%	2.509382e+05	40.000000
max	1.482242e+06	88.000000

Figure 1.5: Statistics of *duration_ms* and *popularity* for *explicit* = True

Considering Explicit = False, the main statistics and distributions of *duration_ms* and *popularity* undergo

very small changes, given that the size of this new dataset is almost like the original (14034 tracks vs 15000).

It is not possible to say the same thing in the case in which Explicit = True, given that the dataset is very small (966 tracks vs 15000). As can be seen from the following figure, the average of *duration_ms* decreases by 23769.2 milliseconds, while the average of "popularity" decreases by approximately 0.93 points. Furthermore, while the standard deviation of the *popularity* variable increases by 0.72, that of the *duration_ms* variable decreases by 37573.9. This result is clearly predictable: by "ignoring" the remaining 14034 observations, the variability between the observations considered is decreased, the smaller the deviations from the mean, the smaller the sum of squares and, therefore, the smaller the standard deviation.

1.3 Pairwise correlation and eventual elimination of variables

A bigger number of variables doesn't necessarily lead to more information; that's because some variables can be highly correlated to each other. The Pearson's correlation matrix is a useful instrument to catch this and it has been chosen for its properties - invariant to scaling and translation - that aren't simultaneously satisfied by other proximity measures like *Minkowski distance* and *cosine similarity*.

corr(*duration_ms*, *n_beats*)	**Pearson's correlation**	**Minkowski distance**
Raw data	0.8393	33982558
Standardized data	0.8393	69.428

Table 1.3: Example of the invariance to scaling property

The previous table shows how the standardization of the data doesn't affect the Pearson's correlation - that is invariant to scaling - but has a huge impact on the Minkowski distance; since there's no need to transform the data, the first measure is preferable.

	duration_ms	explicit	popularity	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	features_duration_ms	time_signature	n_beats	n_bars	popularity_confidence	processing
duration_ms	1.000000	-0.048723	-0.076202	-0.080932	0.102996	0.016712	0.039403	-0.023310	-0.074774	-0.184902	0.079685	-0.021184	-0.141938	0.048765	0.999918	0.009033	0.839313	0.838041	-0.005830	-0.010400
explicit	-0.048723	1.000000	-0.013096	0.056992	0.124405	-0.000991	0.131777	-0.042846	0.294287	-0.113975	-0.125101	0.006995	0.008631	0.016972	-0.048699	0.051260	-0.039830	-0.039435	0.029119	-0.005878
popularity	-0.076202	-0.013096	1.000000	0.051046	-0.056547	-0.008992	0.046703	0.073043	-0.096382	0.078424	-0.266843	0.027105	0.077652	-0.024162	-0.076228	-0.003160	-0.084774	-0.089134	-0.003665	0.008733
danceability	-0.080932	0.056992	0.051046	1.000000	0.193940	0.035311	0.385244	-0.072422	0.111454	-0.199082	-0.202316	-0.174193	0.559578	0.092267	-0.080891	0.292167	-0.033604	-0.069332	0.011602	-0.026273
energy	0.102996	0.124405	-0.056547	0.193940	1.000000	0.057815	0.720907	-0.079969	0.143454	-0.699454	-0.203111	0.191024	0.284069	0.331538	0.102985	0.204646	0.243586	0.208615	0.040022	-0.037847
key	0.016712	-0.000991	-0.008992	0.035311	0.057815	1.000000	0.047718	-0.157515	0.018157	-0.049867	-0.001532	-0.007420	0.028221	0.022414	0.016786	0.028388	0.027133	0.022042	0.019790	-0.490366
loudness	0.039403	0.131777	0.046703	0.385244	0.720907	0.047718	1.000000	-0.033544	0.109793	-0.552519	-0.447106	0.047502	0.377743	0.308203	0.039415	0.250568	0.165565	0.131064	0.049174	-0.020563
mode	-0.023310	-0.042846	0.073043	-0.072422	-0.079969	-0.157515	-0.033544	1.000000	-0.064289	0.085246	-0.080035	0.015738	-0.006907	-0.013001	-0.023409	-0.019378	-0.030685	-0.027687	0.003698	0.261129
speechiness	-0.074774	0.294287	-0.096382	0.111454	0.143454	0.018157	0.109793	-0.064289	1.000000	-0.087997	-0.115803	0.050034	0.056068	0.062161	-0.074722	0.080739	-0.043217	-0.040820	0.026399	-0.022909
acousticness	-0.184902	-0.113975	0.078424	-0.199082	-0.699454	-0.049867	-0.552519	0.085246	-0.087997	1.000000	0.094256	-0.075924	-0.131961	-0.244244	-0.184846	-0.139706	-0.283324	-0.248164	-0.021155	0.022684
instrumentalness	0.079685	-0.125101	-0.266843	-0.202316	-0.203111	-0.001532	-0.447106	-0.080035	-0.115803	0.094256	1.000000	-0.088589	-0.348955	-0.117466	0.079757	-0.123982	0.034830	0.044498	-0.025716	-0.012960
liveness	-0.021184	0.006995	0.027105	-0.174193	0.191024	-0.007420	0.047502	0.015738	0.050034	-0.075924	-0.088589	1.000000	-0.035305	-0.008141	-0.021158	-0.063152	-0.015254	-0.015667	-0.025104	0.014917
valence	-0.141938	0.008631	0.077652	0.559578	0.284069	0.028221	0.377743	-0.006907	0.056068	-0.131961	-0.348955	-0.035305	1.000000	0.157773	-0.141968	0.193726	-0.072477	-0.095825	0.004771	-0.005609
tempo	0.048765	0.016972	-0.024162	0.092267	0.331538	0.022414	0.308203	-0.013001	0.062161	-0.244244	-0.117466	-0.008141	0.157773	1.000000	0.048738	0.215747	0.455310	0.434540	0.038471	-0.003981
features_duration_ms	0.999918	-0.048699	-0.076228	-0.080891	0.102985	0.016786	0.039415	-0.023409	-0.074722	-0.184846	0.079757	-0.021158	-0.141968	0.048738	1.000000	0.008997	0.839172	0.837915	-0.005828	-0.010569
time_signature	0.009033	0.051260	-0.003160	0.292167	0.204646	0.028388	0.250568	-0.019378	0.080739	-0.139706	-0.123982	-0.063152	0.193726	0.215747	0.008997	1.000000	0.100644	0.018492	0.023644	-0.007339
n_beats	0.839313	-0.039830	-0.084774	-0.033604	0.243586	0.027133	0.165565	-0.030685	-0.043217	-0.283324	0.034830	-0.015254	-0.072477	0.455310	0.839172	0.100644	1.000000	0.983696	0.008429	-0.011199
n_bars	0.838041	-0.039435	-0.089134	-0.069332	0.208615	0.022042	0.131064	-0.027687	-0.040820	-0.248164	0.044498	-0.015667	-0.095825	0.434540	0.837915	0.018492	0.983696	1.000000	0.002789	-0.009546
popularity_confidence	-0.005830	0.029119	-0.003665	0.011602	0.040022	0.019790	0.049174	0.003698	0.026399	-0.021155	-0.025716	-0.025104	0.004771	0.038471	-0.005828	0.023644	0.008429	0.002789	1.000000	0.000027
processing	-0.010400	-0.005878	0.008733	-0.026273	-0.037847	-0.490366	-0.020563	0.261129	-0.022909	0.022684	-0.012960	0.014917	-0.005609	-0.003981	-0.010569	-0.007339	-0.011199	-0.009546	0.000027	1.000000

Figure 1.6: Correlation matrix of the quantitative variables of the dataset

As we can see from the previous matrix some correlations stand out; *duration_ms* and *features_duration_ms* gives almost the same information (correlation $>$ 99%) as well as *n_beats* and *n_bars* that are also highly correlated with the first two variables mentioned. The significant correlation present justify the elimination of *features_duration_ms, n_beats* and *n_bars* from the dataset. The correlation between *energy* and *loudness* is relevant too (>72%) ; to choose which one is better to be removed we can look at other correlations that involve these variables: for example *energy* is more correlated to *acousticness* than *loudness* (69% vs 55%) so it makes more sense to remove the first one from the dataset.
Every correlation coefficient used to justify the eliminations aforementioned passed a T-Test so it's going to be valid for inference too. The assumption of normally distributed data is verified by the central limit theorem.
The feature *processing* doesn't have any sense, so it's removed.

1.4 Assessing data quality

1.4.1 Missing values

In this part the quality of the data is evaluated through the research and handling of missing values, outliers and eventual presence of semantic inconsistencies.

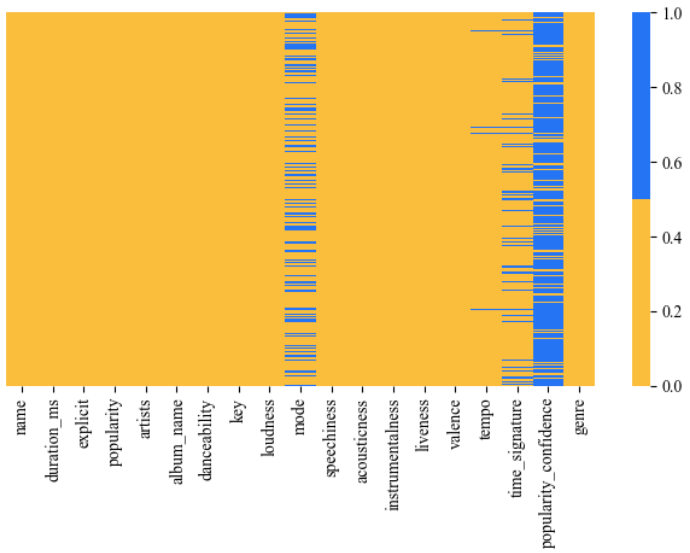


Figure 1.7: Heatmap of missing values

The heatmap shows whether some values are missing and it's evident that the features *mode, tempo, time_signature* and *popularity_confidence* have *None* values, respectively 4450, 90, 2154 and 12783.
In order to keep the feature distributions as close as possible to the original ones the following approach consists in using the probability distribution to randomly extract the "filling" values.

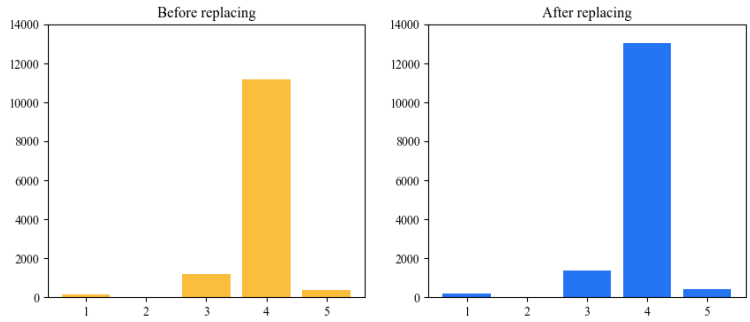


Figure 1.8: Barplots of the *time_signature* variable before and after the filling of missing values

The charts take as an example the *time_signature* variable to show the proportionality between the distributions before and after the missing values replacement. Furthermore in this particular case, as well as with *tempo*, "0" values have been considered *None* values due to the very nature of the variables (in music a time signature or a tempo of 0 have no sense).
The distribution of the variable *tempo* is close to a Normal, so the value have been extracted from a Normal distribution with the mean and standard deviation of the variable itself. The same approach has been used for the *mode* variable, however, we opted for the elimination of the feature *popularity_confidence* due to the large number of missing values (over 85% of the total values).

1.4.2 Outliers

The variables for which is necessary to look for the possible presence of outliers are *duration_ms, loudness* and *tempo*. A reconstruction-based approach has been chosen for the purpose, specifically the multi-layer neural network known as autoencoder, that works in two main steps: encoding, where the dataset is transformed into a low-dimension representation, and decoding, where the previous representation is used to reconstruct the original dataset; the difference between the reconstructed dataset and the original one is used to find the outliers.

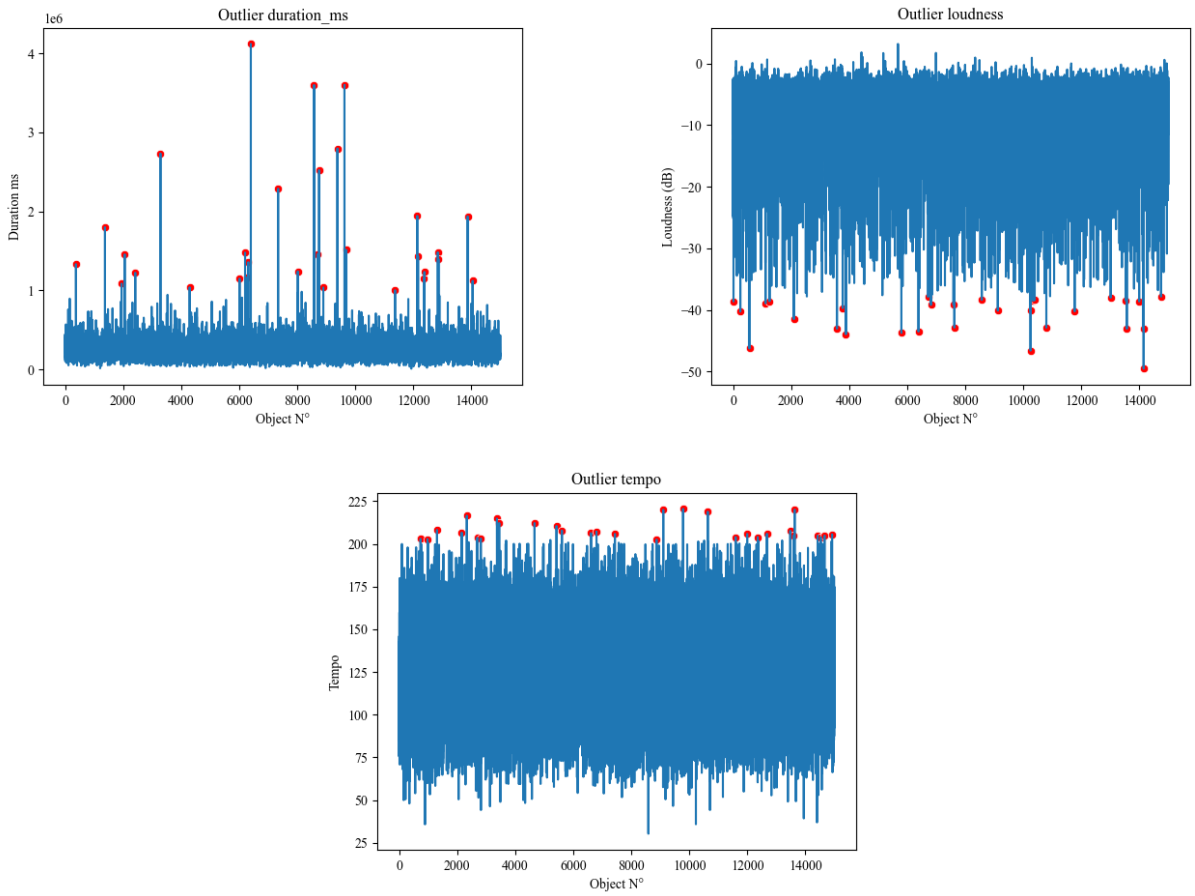


Figure 1.9: Outlier detected by the autoencoder

Given the small amount of outliers (< 100) the choice leans towards the elimination from the dataset.

1.5 Variables transformations

Some transformations can be useful for a better use of the data. The variable *popularity* has been scaled from the original range 0-100 to the range 0.0-1.0, used by the most of the other variables. The genres are represented by 20 different string that has been sorted in alphabetical order, then converted to integer numbers in the range 1-20 to be better interpreted in classification processes. We decided to change the values of the feature *explicit*, *True* and *False*, into the numerical ones 1 and 0 for the same reason of the feature *genre*. The feature *duration_ms* has been standardized, due to its large variance and mean that can lead to problems in further analysis as well as the feature *tempo*.

Chapter 2

Clustering

2.1 Analysis by centroid-based methods

2.1.1 K-Means

After the removal of the outliers and the transformation of the variables, the dataset is almost ready to perform clustering. Clustering algorithms require continuous variables, so only *duration_ms, popularity, danceability, loudness, speechiness, acousticness, instrumentalness, liveness, valence* and *tempo* are considered.
The initial cluster centroids for the K-Means algorithm are chosen using sampling based on an empirical probability distribution of the points' contribution to the overall inertia.
Before running the algorithm it's important to look for the best number of clusters k. We chose to use the Sum of Squared Error (SSE) curve together with the silhouette Coefficient curve.

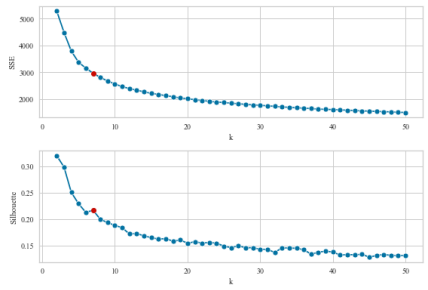


Figure 2.1: SSE and Silhouette curves

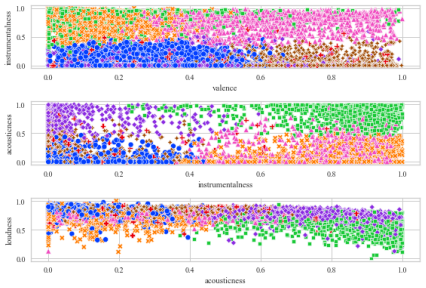


Figure 2.2: Clustering examples using K-Means

The red point highlight that the best number of k is 7, so this number will be used to compute the algorithm that leads to the results shown in the figure 2.2.
The figure 2.3 helps us in the analysis of the centroids. The clusters are close to each other with reference to the variables *duration_ms, popularity* as well as *speechiness*; the cluster 2 contains objects with higher value of the variables *acousticness* and *instrumentalness* while the third has the lowest number of elements and includes objects with high values of *liveness*. These informations are going to be useful in future steps like classification.

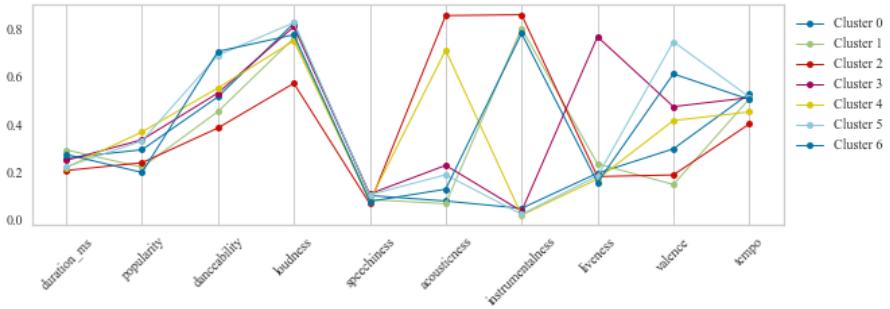


Figure 2.3: Centroid analysis

Labels	N° elements
0	3035
1	1741
2	1582
3	840
4	2482
5	3645
6	1555

Table 2.1: Elements in each clusters

2.1.2 Bisecting K-Means

K-Means needs k points to use as initial centroids and this is a weakness, because different starting points lead to different results that can be sub-optimal. A solution to this problem consists in using the Bisecting K-Means algorithm to find centroids to be used as initial ones in the K-Means algorithm. It's useful to compare the results between this two methods.

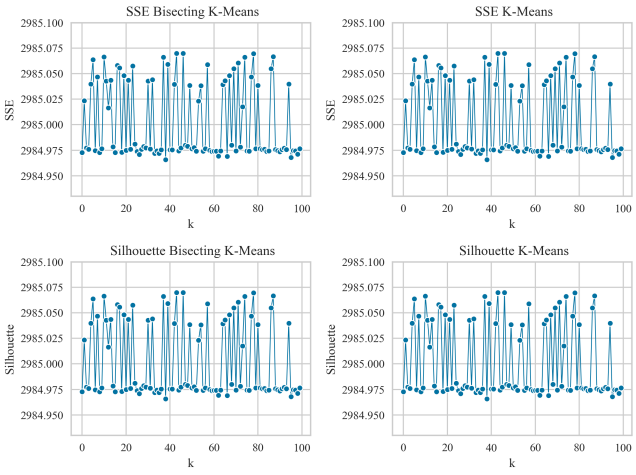


Figure 2.4: K-Means / Bisecting K-Means comparison

In the figure 2.4 we can see the comparison between the classic K-Means algorithm and the one described before. To have solid and valid results the algorithms has been run 100 times and it's clear that using the Bisecting K-Means doesn't lead to better performance in terms of SSE or Silhouette.

2.2 Analysis by density-based clustering

2.2.1 DBSCAN

To use DBSCAN it is necessary to first identify the eps and MinPts values. To find the value of eps we plot the knn graph, given by the average of the distances of each point to its k nearest neighbours. Consider k = 4 (i.e., MinPts = 4).

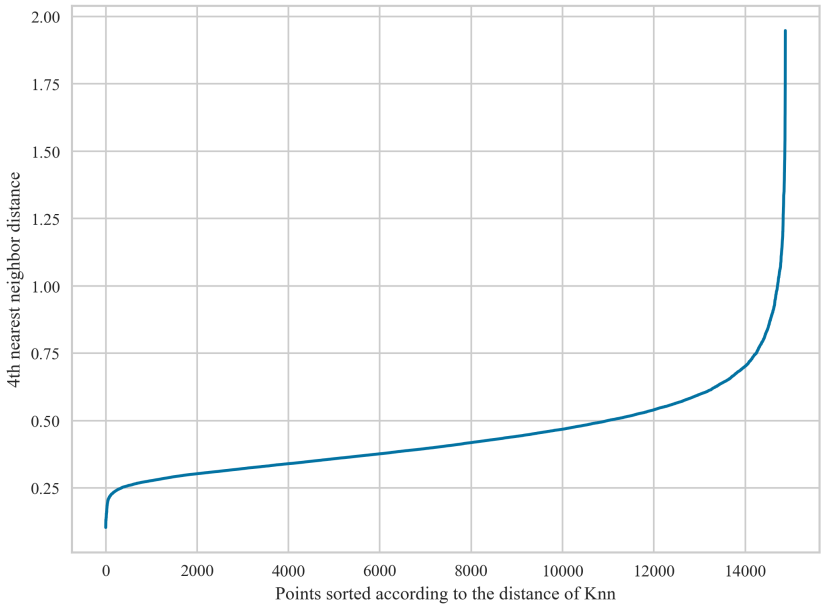


Figure 2.5: Knn graph

The optimal eps value is the one that corresponds to the "elbow" of the curve; in this case consider eps=0.7. With eps = 0.7 and MinPts = 4, the dataset is divided into 4 different clusters and contains 421 outliers.

The silhouette coefficient of 0.4307 seems good, but looking at the number of elements almost all of them are into the first cluster (table 2.2). Even trying multiple combinations of eps and MinPts the results are analogues, and this can be due to high-dimentional data or the heterogeneous density of the dataset that are both weak point of this algorithm.

	Outlier	**Cluster 0**	**Cluster 1**	**Cluster 2**	**Cluster 3**
N° elements	421	14445	7	3	4

Table 2.2: Elements in each clusters using the DBSCAN algorithm

Because of these bad results we decided to not consider this algorithm in the latter evaluation.
The following scatterplot between the variables "duration ms" and "tempo" explains the observations made.

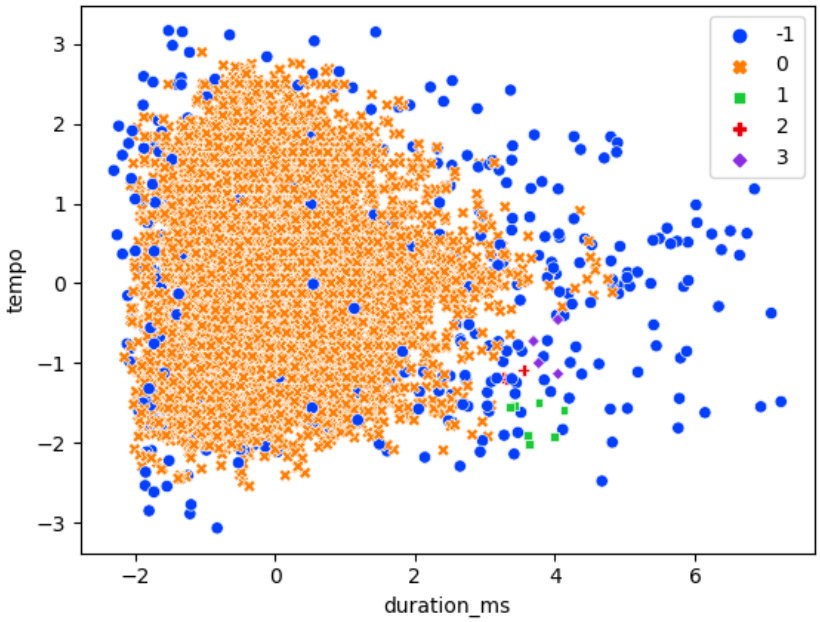


Figure 2.6: Scatterplot between *duration_ms* and *tempo*

2.3 Analysis by hierarchical clustering

In this section we have reported the results of the analysis through the histograms of the dataset at our disposal.

In order to select the most suitable number of clusters to divide the dataset we used different methods including ward with Euclidean metric, average with Manhattan metric, single-link and complete linkage, finding that the method that best divided the dataset was ward with Euclidean metric and the use of 6 clutsers.

As regards the choice of the number of clusters, we reiterated the simulations on a scale of values from 2 to 50, and we were able to observe how 6 was the most suitable number to carry out this task.

In the image below you can observe the dendrogram that reflects the hierarchical clustering according to the ward method. In fact, the latter turned out to be much more precise in separating the data compared to the model based on the minimum distance and the others.

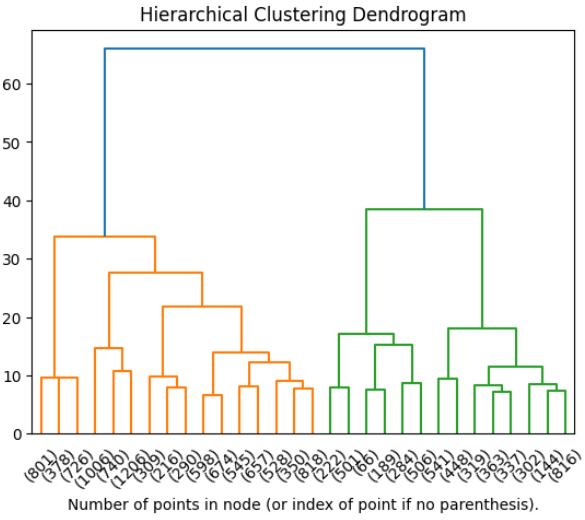


Figure 2.7: Hierarchical Clustering Dendrogram

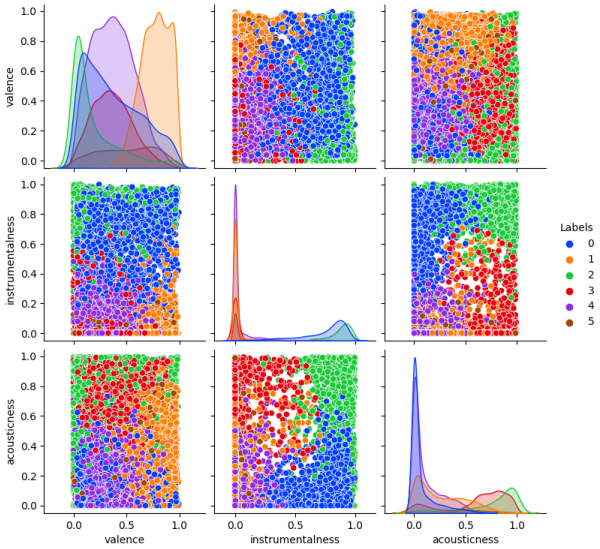


Figure 2.8: Hierarchical Clustering Scatter Plot

Once the ward method with Euclidean metric was identified and selected as the best method for cluster subdivision, we carried out and analyzed all possible combinations. Wanting to go into more detail, we studied the arrangement of the data distributed within the clusters, observing enough homogeneity in the distribution of the data in the various clusters.

In the table 2.3 we have reported the labels and their relative content in terms of quantity of elements contained within them.

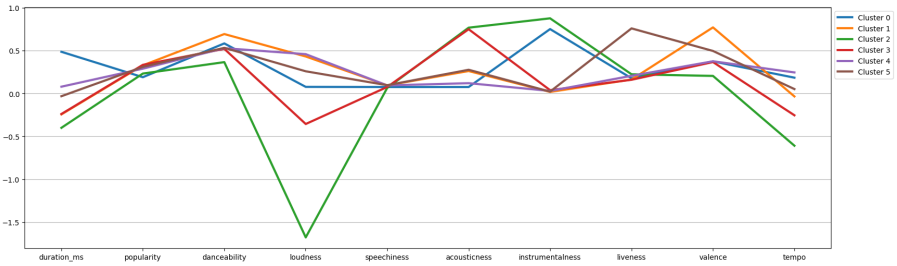


Figure 2.9: Centroid analysis

Labels	N° elements
0	3270
1	2952
2	1768
3	1905
4	4170
5	815

Table 2.3: Elements in each clusters

Looking at the Silhouette coefficient for each algorithm we analysed before we can see that, for the centroid-based methods, the K-Means is the one that performs better, while for the hierarchical ones using the *ward* linkage criterion is the best choice. The difference between these two methods, based upon the silhouette score, is not so tight, but the K-Means looks better.

Algorithm	Silhouette coefficient
K-Means	0.2173
Bisecting K-Means	0.1943
Hierarchical ward	0.1934
Hierarchical min	0.0922

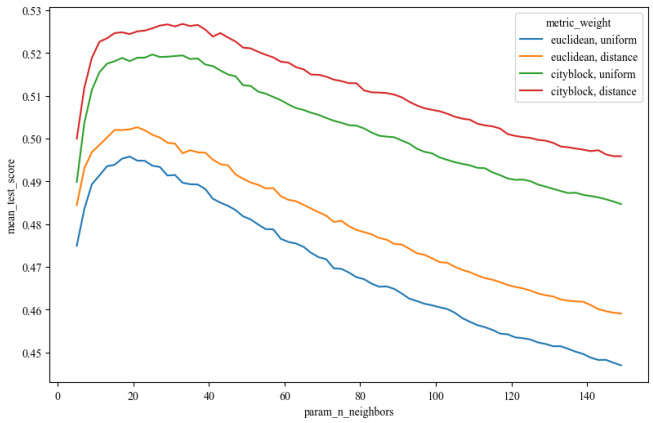
Table 2.4: Silhouette score for each algorithm

Chapter 3

Classification

3.1 KNN

The K-nearest-neighbors is an instance-based classifier which allows to classify instances based on the majority class of their neighbours and its use require three main parameters: number of k, distance metric and weight handler (that's not mandatory). To find the best ones it's good practice to evaluate all the different combinations. In this case a Grid Search method has been used together with a Repeated Stratified K-fold cross validator, that allows multiple fits using different splits that ensure each class is represented with approximately equal proportions in both subsets, training and validation. In the figure 3.1 we can see the results of the training process, comparing two distance metrics - *euclidean* and *cityblock* (aka *manhattan*) - and two weights handler - *uniform* and *distance* - for each k in the range 5 to 150. This range has been chosen to see the behaviour of the model starting from a very small value up to one that's slightly larger than $\sqrt{N} = 121, N = 14880$, the value usually used in this model.



The distance-weighted model performs better than the uniform-weighted ones, so the best model is the one with $k = 33$, *cityblock* as distance metric and distance-weighted.

Figure 3.1: KNN GridSearch cross validation with Repeated Stratified K-fold

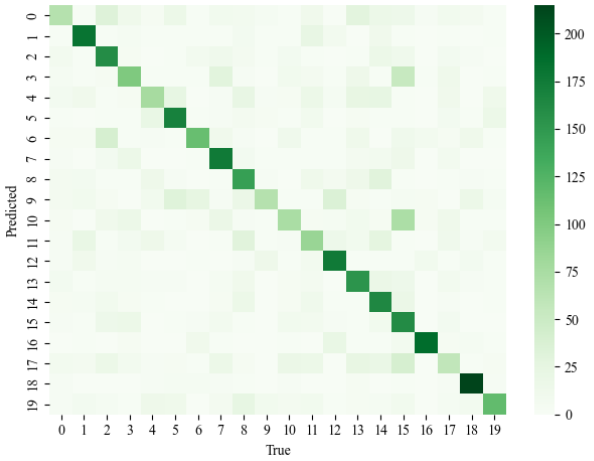


Figure 3.2: Confusion matrix of class *genre*

Parameters	
k	33
Distance metrics	*euclidean*
Weight	*distance*
Metrics	
Accuracy	0.5298
Precision	0.5362
Recall	0.5298
F1-score	0.5197

Table 3.1: Model parameters and evaluation metrics

The confusion matrix in figure 3.2 seems showing good results, but looking at the evaluation metrics in table 3.1 we see an accuracy of ≈ 0.5 as well as the other metrics, so the prediction capacity is not the best, probably due to overfitting problem, given that the accuracy computed on the training set is $= 1$ for each genre. This information will be useful to compare the different models at the end of the chapter.

3.2 Naïve Bayes

Also for the classification with the Naïve Bayes classifier the variable "genre" was used as the target class. This classifier can correctly predict 2057 observations out of 5000 total (size of the test dataset), thus reporting an accuracy score of 0.41.

Metrics	
Accuracy	0.4114
Precision	0.4313
Recall	0.4114
F1-score	0.3781

For example, the first instance of the test dataset is classified (wrongly because it should belong to the "j-dance" genre) as the "j-idol" genre as the probability of belonging to it is 70.1%, greater than the probability of belonging to the other genres. The first observation to be correctly predicted (the seventh of the test dataset), however, belongs to the "idm" genre with a probability of approximately 78%.

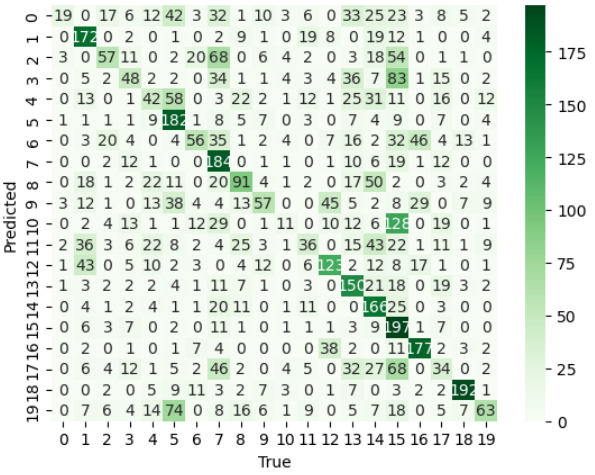


Figure 3.3: Confusion matrix of class *genre*

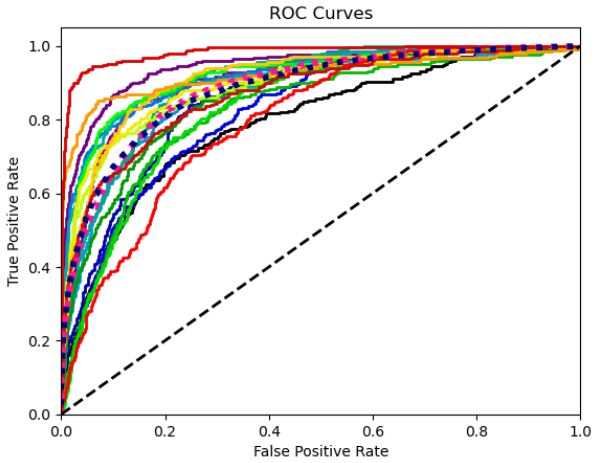


Figure 3.4: ROC curves of each *genre*

Figure 3.3 shows the confusion matrix while figure 3.4 shows the ROC curves for each genre, with the AUC areas between 0.79 ("afrobeat") and 0.98 ("study").

3.3 Decision Tree

For the classification part, using Decision Trees, we studied various ensemble methods, including Random Forest and AdaBoost, until we determined that the most accurate for our research could be the first. This in fact recorded the best parameters of accuracy, precision, recall and f1-score among the various ensemble methods.

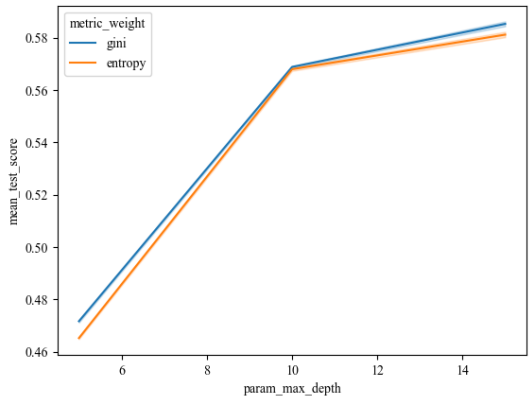


Figure 3.5: Grid search with ensemble method Random Forest

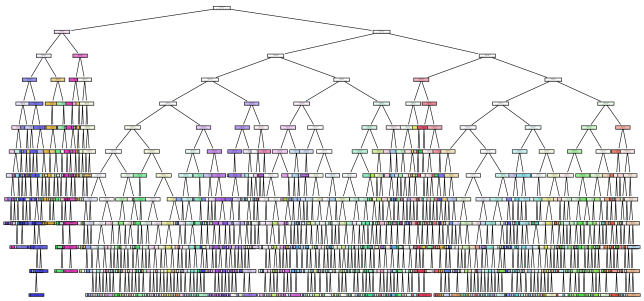


Figure 3.6: Plot tree with ensemble method Random Forest

To establish the best parameters with which to train the model, a grid search was also used, testing both *Gini* and *entropy* as criteria, establishing that the former was the most suitable to carry out the classification task for the Decision Tree. In the confusion matrix below we have reported the performances recorded by the Random forest method. This method, compared to the others used, proved to be more precise in terms of accuracy. The precision, recall and F1-score parameters also proved to be better than the classic method and AdaBoost, which however achieved good results.

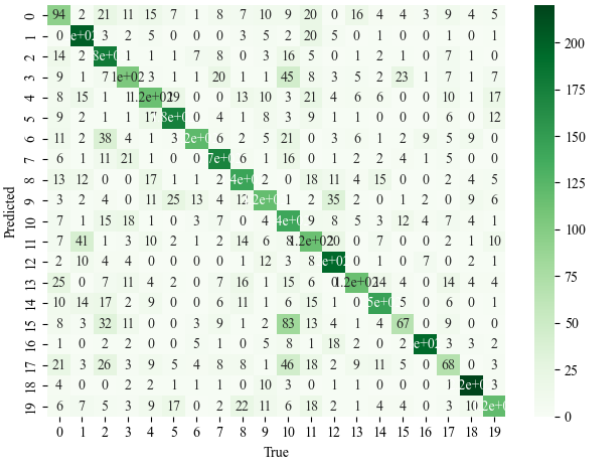


Figure 3.7: Confusion matrix of class *genre*

Parameters	
Criterion	*Gini*
max_depth	*none*
min_samples_leaf	*1*
min_samples_split	*2*
n_estimators	*300*
Metrics	
Accuracy	0.5626
Precision	0.4115
Recall	0.4094
F1-score	0.4074

Figure 3.8: Model parameters and evaluation metrics

In this case the accuracy is 56.26%. This value alone might seem acceptable, but it does not provide a complete view. Overall, the results show that the Decision Tree model does not have excellent performance. Precision and recall are both around 40%, indicating that the model has difficulty accurately distinguishing classes or correctly identifying positive instances.

3.4 Models comparison

Looking at the barchart in figure 1.3 it's clearly visible that the genres are equally distributed in the dataset, so there's any difference in using the micro or the macro-average to build the ROC curves, so we used the Micro-averaged one. In figure 3.9 the curves for the three models are visible, and we can see that the decision tree is the one that performed the best w.r.t. the AUC, slighly better than KNN (0.94 vs 0.92). Looking at the

metrics in the previous sections, in particular at the F1-score, KNN seems the best so is the one we've decided to choose.

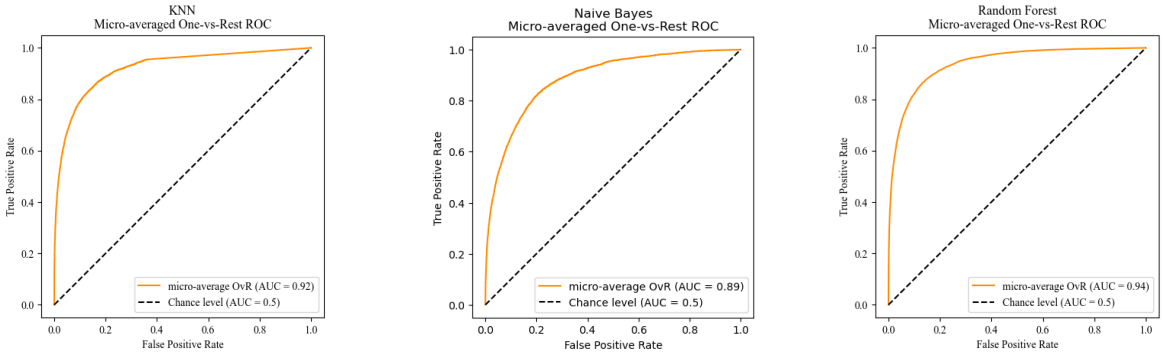


Figure 3.9: Micro-averaged One-vs-Rest ROC for the three models

Chapter 4

Pattern Mining and Regression

In this chapter two algorithms will be used to extrapolate association rules from the data we have and will be evaluated from a quantitative and qualitative point of view. This process happens in two steps: firstly the frequent itemsets are generated, then the rules directly from the latter.
Before the implementation of the algorithms a brief preprocessing on the data is mandatory to have the best results; all the continuous attributes has been discretized by a subdivision in three equally distributed classes. The table 4.1 shows the classes.

Features(breviter)	**Classes**		
Duration ms (dur)	[-2.2996, -0.4654)	[-0.4654, 0.2063)	[0.2063, 7.2303]
Popularity (pop)	[0, 0.18)	[0.18, 0.35)	[0.35, 0.98]
Danceability (dance)	[0, 0.497)	[0.497, 0.657)	[0.657, 0.98]
Loudness (loud)	[-6.8012, -0.0540)	[-0.0540, 0.5163)	[0.5163, 2.0124]
Speechiness (speech)	[0, 0.0411)	[0.0411, 0.0698)	[0.0698, 0.939]
Acousticness (acoustic)	[0, 0.0316)	[0.0316, 0.414)	[0.414, 0.996]
Instrumentalness (instrum)	[0, 1.25e-5)	[1.25e-5, 0.374)	[0.374, 1]
Liveness (live)	[0, 0.107)	[0.107, 0.209)	[0.209, 0.994]
Valence (val)	[0, 0.27)	[0.27, 0.579)	[0.579, 0.995]
Tempo (tempo)	[-3.0659, -0.4560)	[-0.4560, 0.364)	[0.364, 3.1661]

Table 4.1: Features subdivision in equally distributed classes

4.1 Apriori algorithm

For the Apriori algorithm it is necessary to identify two parameters: the minimum number of elements that must be present in a set of frequent elements (zmin) and a measure of the frequency with which a pattern is present in the dataset (support). The figure 4.1 represents the number of frequent itemsets for each combination of zmin (from 0 to 7) and support (from 1 to 10). The figure 4.2 shows how the number of frequent closed and maximal itemsets changes as support varies, considering zmin = 2.

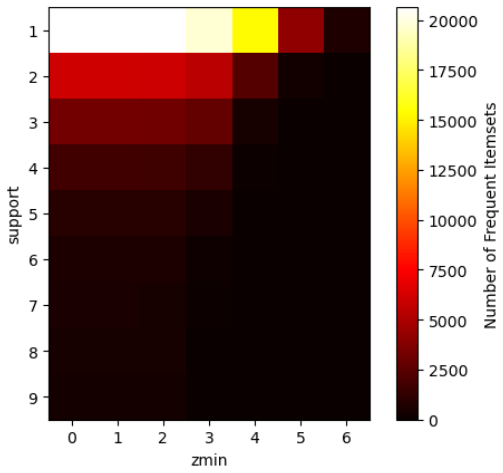


Figure 4.1: Number of frequent itemsets with different values of support and zmin

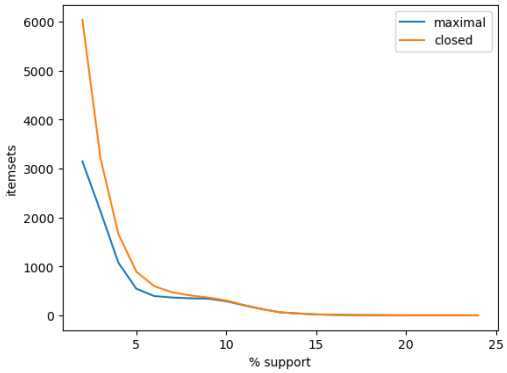


Figure 4.2: Number of closed and maximal frequent itemsets w.r.t. the support

With zmin = 2, the algorithm is able to find frequent subsets up to a support of 20. Considering zmin = 2 and support = 20, the most frequent itemset is composed by "Valence", with a value belonging to interval (0, 0.27], and "Danceability", with a value belonging to the interval (0, 0.497]. By lowering support to 18, we obtain 4 frequent itemsets: the first is composed by "Acousticness", ((0.414 , 0.006]), and "Loudness" ((-6.8012, -0.0540]); the second one by "Instrumentalness", ((0.374, 1]), and "Loudness", ((-6.8012, -0.0540]); the third one by "Instrumentalness", ((0.374, 1]), and "Valence" ((0, 0.27]); the last one by "Valence", ((0, 0.27]), and "Danceability", ((0, 0.497]), as in case where support is equal to 20. The number of rules that can be extracted from frequent itemsets changes depending on the confidence level considered. The figure 4.3 shows how this number changes for different combinations of support (from 10 to 20) and confidence (from 60 to 80). As the confidence level increases, given a fixed value of support, the number of rules decreases.

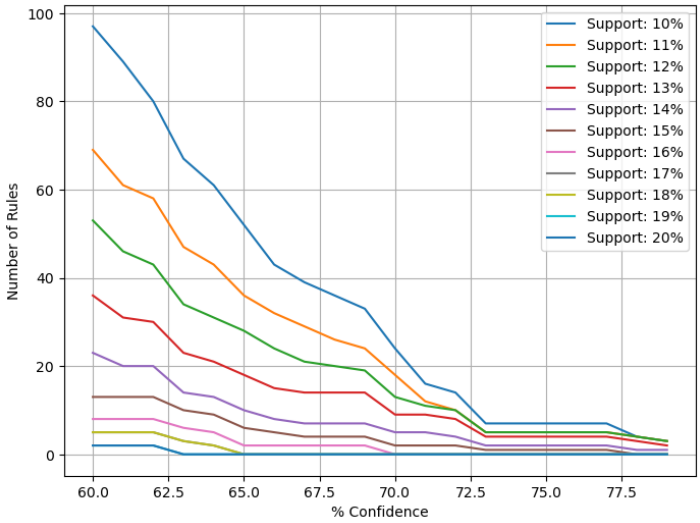


Figure 4.3: Number of rules for *support-confidence* pairs

Let's consider, for example, support equal to 15. The figure 4.4 shows that for confidence intervals between 50% and 60% it is possible to extract 55 rules; 13 rules with confidence between 60% and 70%; 2 rules with confidence between 60% and 70%; 0 rules for intervals ranging from 80% to 100% (the highest confidence level found corresponds to 77.7%).

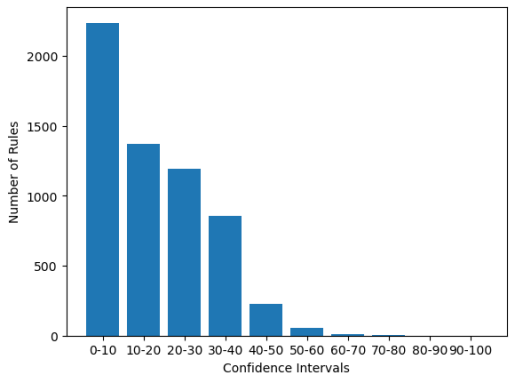


Figure 4.4: Number of frequent itemsets with different values of support and zmin

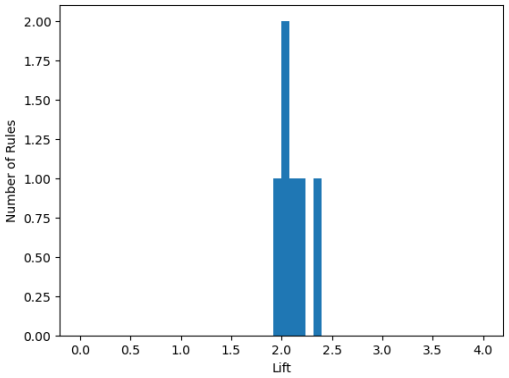


Figure 4.5: Number of closed and maximal frequent itemsets w.r.t. the support

Let us now consider a confidence level of 65. The algorithm is able to find 19 frequent itemsets and extract 6 rules, shown in the following table, ordered in decreasing order according to the values of "lift".

Consequent	**Antecedent**	**Confidence**	**Lift**
Valence [0, 0.27)	(loudness [-6.8012, -0.0540), danceability [0, 0.497))	0.777	2.329
Acousticness [0.414, 0.996]	(loudness [-6.8012, -0.0540), danceability [0, 0.497))	0.723	2.170
Instrumentalness [0.374, 1]	(loudness [-6.8012, -0.0540), Valence [0, 0.27])	0.699	2.099
Danceability [0, 0.497]	(loudness [-6.8012, -0.0540), Valence [0, 0.27])	0.694	2.076
Loudness [-6.8012, -0.0540)	(Acousticness [0.414, 0.996], Tempo [-3.0659, -0.4560])	0.667	2.000
Loudness [-6.8012, -0.0540)	(Acousticness [0.414, 0.996], duration ms [-2.2996, -0.4654))	0.659	1.977

Table 4.2: Rules sorted by lift value

From the table we can see that low loudness and danceability values lead to a low valence value and to a high acousticness value; low loudness and valence values lead to a high instrumentality value and a low danceability value; a high acousticness value and a low tempo value lead to a low loudness value; a high acousticness value and a low duration value in ms lead to a low loudness value. We use the first rule from the table 4.2 (highest lift value) to predict the variable valence of the test dataset: if loudness has a value in the range [-6.8012, -0.0540) and danceability in the range [0, 0.497), then valence has a value in the interval [0, 0.27). With this rule, out of 750 observations that have a valence value included in the range considered, 573 were correctly predicted (76.4%). Considering all the observations of the test dataset, however, 3698 observations out of 5000 were correctly predicted (3125 with a valence value greater than 0.27 and 573 with the value between 0 and 0.27), thus reporting an accuracy equal to 73.96%.

4.2 FP-Tree algorithm

The FP-Tree algorithm is faster than the Apriori one thanks to its compactness; it reduces irrelevant information removing the items with a frequency lower than a given bound. The figure 4.6 is helpful to show how the percentage of closed and maximal frequent itemsets vary with reference to the support; in particuar we can see that it decrease exponentially and the number of itemsets is reasonably low for support $> 10\%$.
The figure 4.7 gives a hint about how the number of rules change with reference to support and confidence. In order to increase the readability of the heatmap small ranges of support and confidence have been chosen, but wide ones have been used, showing that the number of rules increase exponentially - up to millions - for lower confidence and support values.
After some tests, we opted for a minimum support of 10%, a minimum number of elements of 2 and a minimum confidence of 70% , obtaining the quantities visible in table 4.3.

Frequent itemsets	304
Maximal itemsets	290
Rules	24

Table 4.3: Number of itemsets/rules obtained

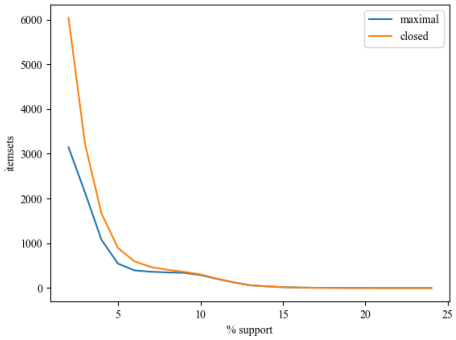


Figure 4.6: Trend of #closed and maximal frequent itemsets w.r.t. the support

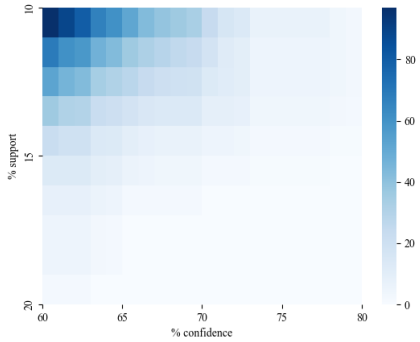


Figure 4.7: Number of rules for *support-confidence* pairs

To ease comprehension and readability, we'll refer to the feature with an abbreviation, as pointed in the table 4.1 and to their different classes as low, medium and high.
We computed the frequent and maximal itemsets and then we sorted them by their support value in descending order; the first five are visible in the table 4.4. val low and loud low are present two times in this table and 22 times each in all the frequent itemsets.

Frequent itemsets	**Maximal itemsets**
(val low, dance low)	(dance high, val high)
(acoustic high, loud low)	(instrum high, pop low)
(instrum high, loud low)	(dance high, tempo medium)
(instrum high, val low)	(pop high, instrum low)
(dance high, val high)	(loud high, acoustic low)

Table 4.4: Frequent and maximal itemsets sorted by support value

The same method has been used to build the rules table 4.5. Looking at the first rule as an example, it's visible that a high level of acousticness and instrumentalness leads to a low level of loudness; thinking about it and looking at the figure 1.6, even from a mere logical point of view, this rule makes sense; furthermore, a lift of 2.627 and a confidence of 0.876 give a big support to it.

Consequent	**Antecedent**	**Confidence**	**Lift**
loud low	(acoustic high, instrum high)	0.876	2.627
loud low	(acoustic high, val low)	0.82	2.459
val low	(instrum high, dance low)	0.813	2.436
loud low	(acoustic high, dance low)	0.788	2.363
val low	(loud low, dance low)	0.777	2.329

Table 4.5: Rules sorted by lift value

Give the values of the metrics *confidence* and *lift*, as well as the computational efficiency, we choose to use the FP-Tree algorithm.

4.3 Regression

In this section we will illustrate the results obtained during the experimentation on various regression models used. The models used are: Linear Regression, Ridge Regression, Lasso Regression, Decision Tree Regression and KNN Regression. The models were used to predict the features present in the test dataset once trained on

the train dataset. We conducted several experiments, the first phase was the prediction of the values of each single feature, in the second part the prediction was made on pairs of features.

4.3.1 Univariate Regression

In this first part we used regression models to predict a single feature at a time. Each regressor was fitted on the train dataset from which the features to be predicted were removed each time and transformed into the target of the function. the same procedure was therefore done with the test dataset from which the same features were removed and predicted by each regressor. In the end we used the mean squared error, the mean absolute error and r2 score as metrics to study the discrepancy between the predicted and target values and therefore study the errors committed by the various regressors.

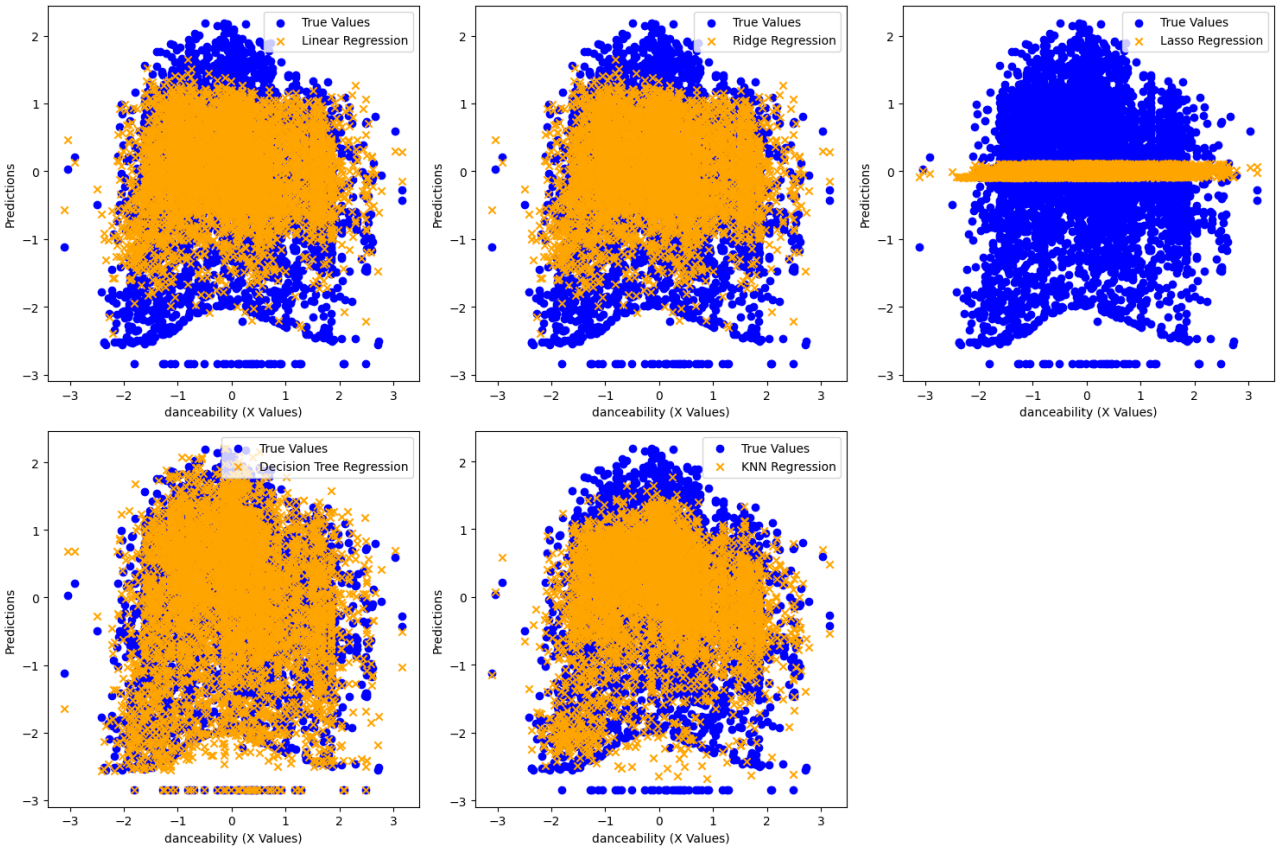


Figure 4.8: Prediction of danceability

	Linear	Ridge	Lasso	Decision Tree	KNN
MSE	0.59656	0.59656	0.93837	0.77516	0.46039
MAE	0.61802	0.61802	0.77100	0.66594	0.51814
R2	0.40344	0.40344	0.06163	0.22484	0.53961

Table 4.6: Errors and metrics of different univariate regression methods

Looking at the results obtained in the example above:

- Linear Regression and Ridge Regression show virtually identical performance in terms of MSE, MAE and R2. Both models have an R2 score around 0.4, suggesting that they can explain only a modest part of the variance in the data.
- Lasso Regression appears to have slightly poorer performance, with a higher MSE and MAE than linear models. Furthermore, the R2 is very low (0.06), indicating that this model explains only a small portion of the variance in the data.

- Decision Tree Regression shows a higher MSE and MAE than linear models and an R2 around 0.22, indicating limited ability to predict test data.
- KNN Regression has a lower MSE and MAE than the other models, along with an R2 above 0.5, suggesting that it is able to explain a more significant portion of the variance in the data than the others.

Based on these results, it seems that KNN Regression is the regressor that has proven to be the most accurate for this specific dataset, followed by Linear Regression and Ridge Regression which obtained similar results. Lasso Regression and Decision Tree Regression appear to show lower performance than the other models in terms of prediction ability for this specific dataset.

4.3.2 Multivariate Regression

In this second phase we used regression models to predict two features of the dataset at a time. In this second part the models struggled more in terms of precision, proving to be less suited to this task.

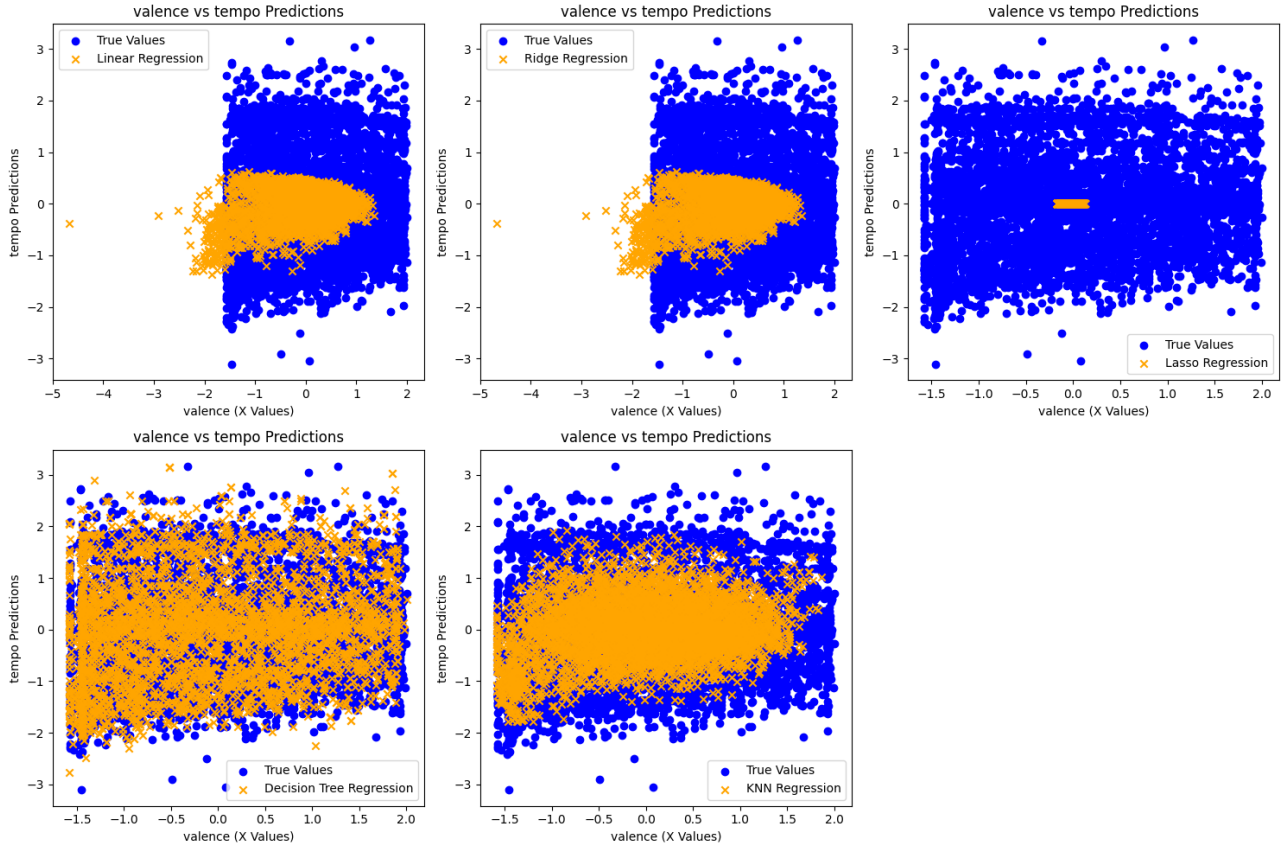


Figure 4.9: Prediction of valence and tempo

	Linear	**Ridge**	**Lasso**	**Decision Tree**	**KNN**
MSE	0.76598	0.76598	0.96919	1.42610	0.79749
MAE	0.70263	0.70263	0.81692	0.89942	0.68763
R2	0.23402	0.23402	0.03081	-0.42610	0.20251

Table 4.7: Errors and metrics of different multivariate regression methods

Looking at the results obtained in the example above:

- Linear Regression and Ridge Regression have similar MSE and MAE, with relatively low R2 values (around 0.23). They indicate moderate but not excellent prediction ability for the dataset. It appears that linear models are unable to capture the complexity of data accurately.

- Lasso Regression shows a higher MSE and MAE than linear models, with an even lower R2 (0.03). This indicates a poor ability to predict the data, suggesting that the Lasso regularization may have shrunk the model too much.
- Decision Tree Regression has a higher MSE and MAE, with a negative R2 (-0.42), indicating that the model is much worse than a simple average. This may indicate overfitting to the training data.
- KNN Regression has a similar MSE and MAE to Linear Regression and Ridge Regression, with a similar R2 (around 0.20). This model appears to have moderate but not exceptional predictive ability.

Based on these results, no regressor proved to be particularly accurate. Linear models, together with KNN, seem to perform better than the others. However, none of them can significantly predict the dataset.