Academic year 2023-2024, Data Science and Business Informatics, University of Pisa

DATA MINING PROJECT REPORT

Borges Legrottaglie Javier Alejandro (536532),
Pastorelli Nicola (656431),
Sanna Marco (660521).

Contents

1. Introduction

The Spotify dataset used for this project contains information about audio tracks present in the mobile app at the time of its creation. It contains songs from 20 different genres, each row within the dataset displays a song and information about its features including *loudness, instrumentalness, energy, danceability, valence, acousticness,* and so on.

2. Data understanding & preparation

2.1. Data Semantics

After a brief examination of the dataset, the first thing that pops up are the names of each attribute, which contain information about each song, as well as their total number, which is 24, the overall shape of the dataset is 15000x24. Below is a summary of the features:

2.1.1. Column description

Name	Name of the track.
Duration_ms	The track length, measured in milliseconds.
Explicit	Whether the track has explicit lyrics (true = yes it does; false = no it does not OR unknown).
Popularity	The popularity of a track is a value between 0 and 100, with 100 being the most popular.
Artists	The artists' names who performed the track. If there is more than one artist, they are separated by a *;*.
Album_name	The album name in which the track appears.
Danceability	Danceability describes how suitable a track is for dancing. A value of 0.0 is least danceable and 1.0 is most danceable.
Energy	Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity.
Key	The key the track is in. Integers map to pitches using standard Pitch Class notation.
Loudness	The overall loudness of a track in decibels (dB).
Mode	Mode indicates the modality (major or minor) of a track. Major is represented by 1 and minor is 0.
Speechiness	Speechiness detects the presence of spoken words in a track.

Acousticness	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
Instrumentalness	Predicts whether a track contains no vocals. The closer the instrumentalness value is to 1.0.
Liveness	Detects the presence of an audience in the recording.
Valence	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track.
Tempo	The overall estimated tempo of a track in beats per minute (BPM).
Features_duration_ms	The duration of the track in milliseconds
Time_signature	An estimated time signature that corresponds to the relationship between the number of beats divided by the number of bars.
N_beats	The total number of time intervals of beats throughout the track.
N_bars	The total number of time intervals of the bars throughout the track.
Popularity_confidence	The confidence, from 0.0 to 1.0, of the popularity of the song.
Processing	A series of random numbers which correspond to the values present in the 'key' attribute.
Genre	The genre in which the track belongs.

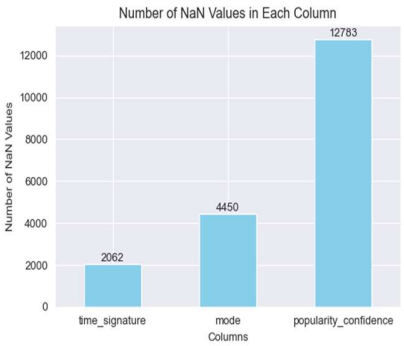
2.2 Data quality

To assess the quality of our data, we looked at how meaningful our features were through visualization, as well as checking for missing values, variables that provide little to no information and the highly correlated ones to better understand the dataset.

2.2.1. Missing values

After checking for which attributes had missing values, we found out that the only ones that contain them are *mode*, *popularity_confidence*, *time_signature* (*figure 1*).

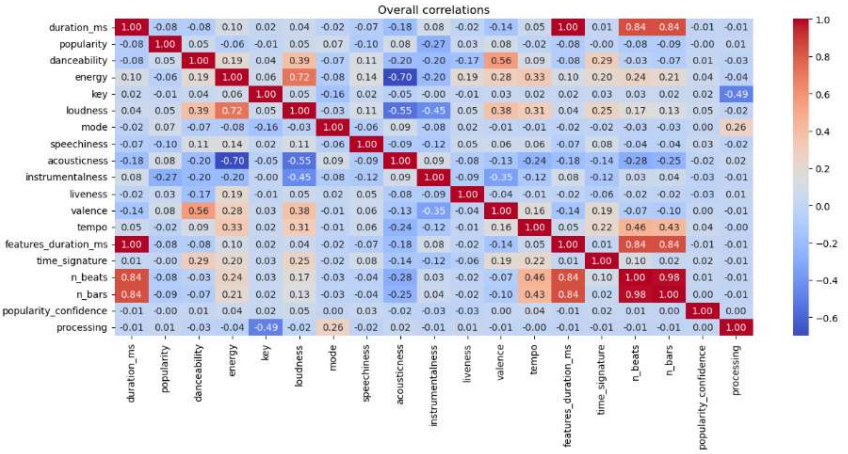
The *popularity_confidence* column contains a series of continuous variables of which there are 12783 missing values. On the other hand, the *mode* column, which contains categorical values, has 4450 missing values. Finally, the categorical column *time_signature*, contains the least amount of missing values at 2062, we decided to take a look at the correlations and distributions of the variables, as well as to do some research on music theory to figure out the proper way of filling those values.



(Figure 1)

2.2.2 Analysis of uninformative variables

Looking at the correlation heatmap (*figure 2*), it becomes apparent that *duration_ms* and *features_duration_ms* are perfectly correlated (1.00) given the fact that they contain almost the same values, reason for which we decided to drop the latter and only keep the *duration_ms* column for our analysis moving forward. Another couple of correlated variables that provide little information are *n_beats* and *n_bars* at 0.98, which seems obvious given that, according to music theory, each bar contains a set number of beats for a track.



(Figure 2)

While these variables might be useful for figuring out the NaN values in the *time_signature* column, they provide little to no other information by themselves.

2.3 Data cleaning

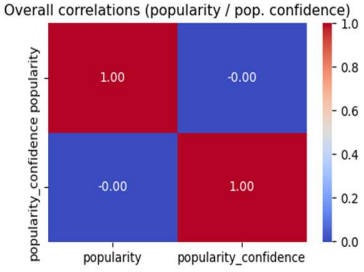
2.3.1 Handling of missing values

After looking into music theory, we realized that we could fill the missing values in the *time_signature* column by dividing the number of beats by the number of bars and using the *round* function to approximate the value of *time_signature* for each song as can be seen in *figure 3*.

With regards to the *mode* attribute, it was decided to drop the entire column as it was impossible to replace the missing values using estimations based on other attributes, not only that, but the *mode* attribute itself does not provide any significant information for our analysis.

	time_signature	n_beats/n_bars
0	4.0	4.0
1	4.0	4.0
2	4.0	4.0
3	4.0	4.0
4	4.0	4.0
...	...	...
14995	4.0	4.0
14996	3.0	3.0
14997	4.0	4.0
14998	4.0	4.0
14999	4.0	4.0

(Figure 3)



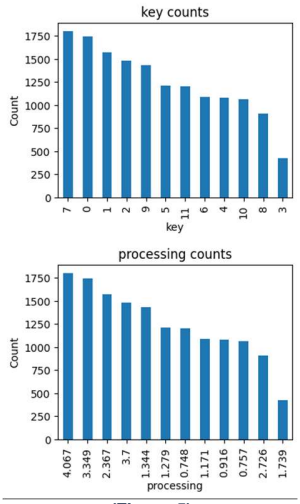
(Figure 4)

Finally, after trying to figure out a way to fill the NaN values present in the *popularity_confidence* column by looking at the correlation between the 2217 rows with actual values in them and their respective *popularity* (see *figure 4*), we ultimately decided to drop it.

1.3.2 Dimensionality reduction

For this phase, we decided to drop *n_beats* and *n_bars* since they became redundant after aggregating their values in the *time_signature* column. We also decided to drop *features_duration_ms* given it contains the same information as *duration_ms*.

Finally, we decided to drop the *processing* column given that it provides the same information as the *key* attribute as can be seen in *figure 5*.



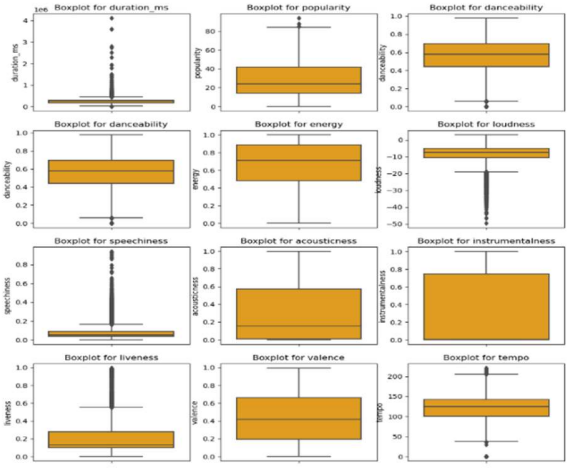
(Figure 5)

2.3.3 Handling of outliers

After having dealt with the NaN values and done dimensionality reduction, we visualized the remaining data through a series of boxplots (*figure 6*) which include the continuous attributes of our dataset and the number of values that deviate from most of the distribution.

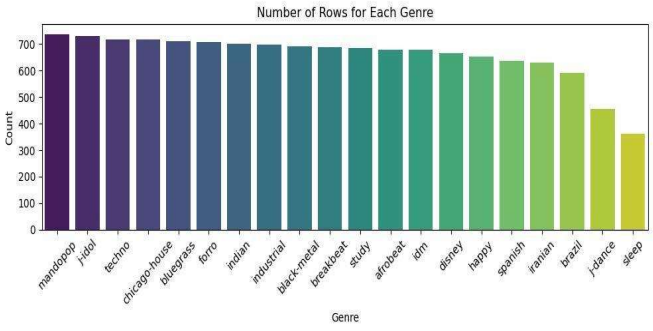
To treat these outliers, we used the interquartile range (IQR) method, deleting the rows that contain numerical values greater (for at least one feature) than the third quartile or lesser than the first quartiles.

After having dealt with the outliers, a description of the dataset was made to see how the values of the data had changed. This showed that the mean and the standard deviation of each attribute did not change significantly.



(Figure 6)

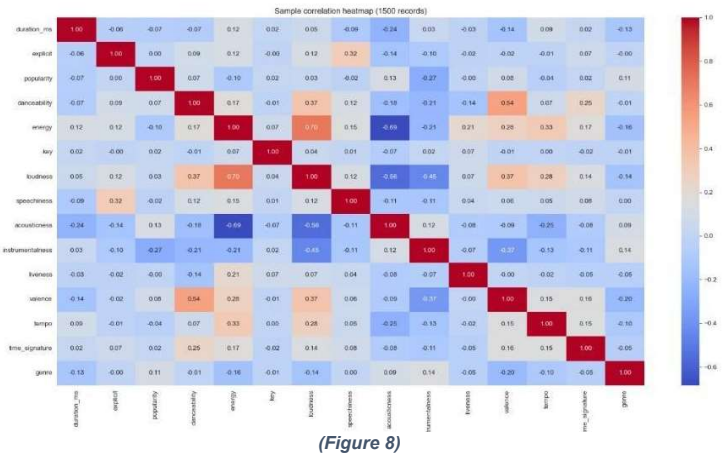
Furthermore, a genre count was done to see how the removal of outliers had affected our dataset. This inspection revealed that genres such as *sleep* and *j-dance* lost 388 and 293 records respectively (see figure 7) which indicates that the characteristics our calculations have deemed as 'extreme outliers' are actually important, or at least common, for such genres.



(Figure 7)

A new correlation heatmap matrix (figure 8) was done to check for any significant changes, none of which were identified as the correlations varied by 4-8 points at the most.

As a result, it was decided to keep the outliers in our dataset so as to have more complete information on each genre for our analysis moving forward.



(Figure 8)

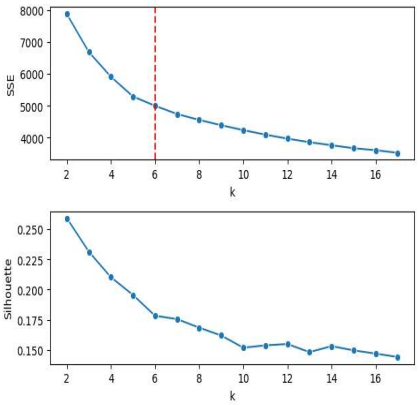
3- Clustering

In the beginning of this section, a decision was made to use the entire dataset with outliers included, since removing them resulted in unbalanced classes, which would, in turn, significantly impact the quality of our analysis. When choosing which scalers to use in the normalization phase, both Standard and Min-Max scalers were used to see which one yielded the better results.

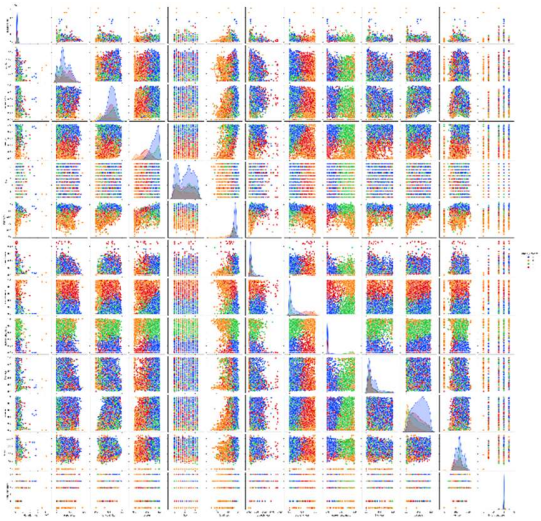
In the end, the Min-Max scaler proved to be the right fit for our analysis since it transforms each attribute that contains numerical values into a scale from 0 to 1, this makes it more suitable for our dataset since it works well with non-normally distributed variables unlike the Standard scaler.

3.1. K-means

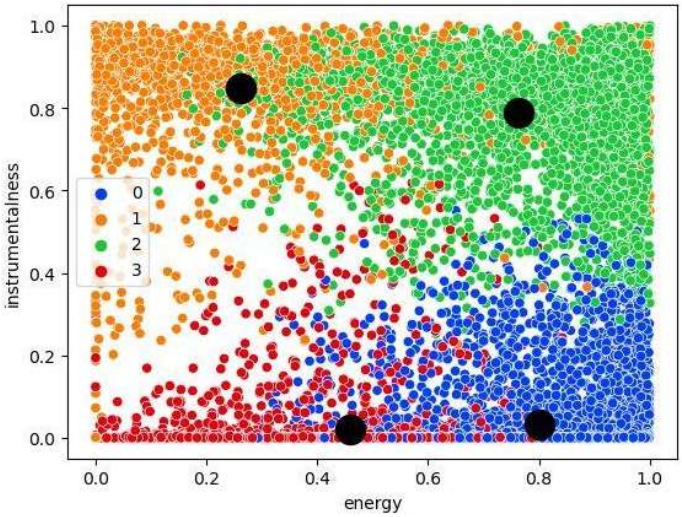
To choose the right number of k, we performed the calculations for the sum of squared errors (SSE) and silhouette (see figure 9), as the figure shows, the higher the value of k, the lower the value of the SSE got, the same result was also obtained with regards to the silhouette score, as this tended to go down the higher k got. According to the line chart the elbow point would be 6, due to the fact that it begins to show a lower difference in the value of the SSE from that point on, while the silhouette score indicated that the most optimal value for k would be 2 as the score is the highest at that point thus, when the time came to choose the right number of clusters for k-means, a compromise was made between the two resulting in 4 being the value chosen.



(Figure 9)



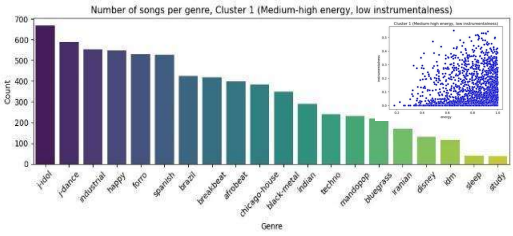
(Figure 10)



(Figure 11)

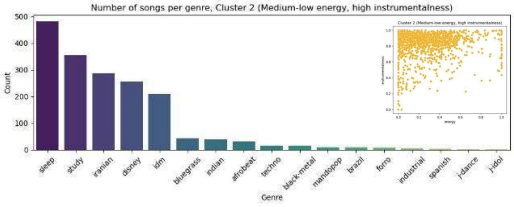
After applying the k-means algorithm to produce four clusters, a pair plot was made to show the clusters with respect to each pair of variables (see figure 10). A quick inspection of the pair plot revealed that the variables in which the clusters show a better split were *energy* and *instrumentalness*, in fact, as can been seen in figure 11, four well-defined clusters has been obtained.

In figure 12, it is possible to see that one cluster is formed by medium-high energy and medium-low instrumentalness, which features j-*idol*, j-*dance*, *industrial*, *happy*, *forro*, and *spanish* as predominant genres since they have the highest number of songs per genre. Considering the number of records, this turns out to be the biggest cluster among the ones obtained.



(Figure 12)

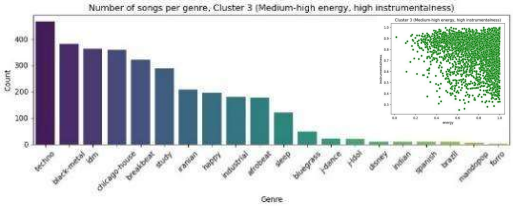
A second cluster (figure 13) features records with medium-high instrumentalness and medium-low energy, which contains the genres *sleep*, *study*, *iranian*, *disney* and *idm*.



(Figure 13)

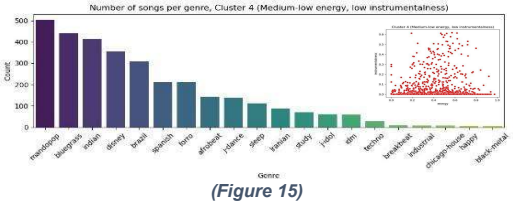
The third cluster (figure 14) contains records with medium-high instrumentalness and energy, this one features *techno*, *black-metal*, *idm*, *chicago-house* and *breakbeat* as its main genres.

The last cluster (figure 15) shows records with medium-low energy and medium-low instrumentalness, and contains *mandopop*, *bluegrass*, *indian*, *disney* and *brazil* as its top 5 genres.



(Figure 14)

Applying the algorithm with k = 4 resulted in a silhouette score of 0.21 which, even if it is a low value, was acceptable when considering the high dimensionality of the dataset, despite this, it can be said that from the point of view of visualization, K-means is an optimal algorithm for a dataset of this kind.

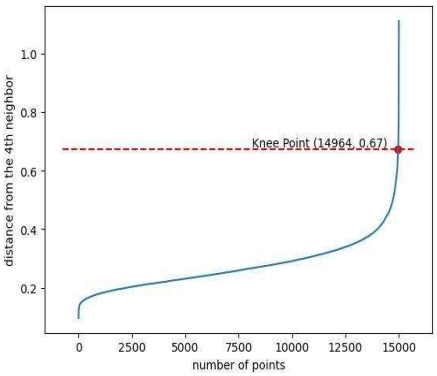


(Figure 15)

3.2. DBSCAN

When approaching the DBSCAN clustering method, we started by plotting the distance of each point from the 4th nearest neighbor using the Euclidean distance metric (see figure 16), and by utilizing the Kneed library, we were able to find the knee point of a curve (or elbow of a curve) is a point where the curve visibly bends, specifically from high slope to low slope (flat or close to flat), or in the other direction.

As can be seen in the figure, when the radius is 0.67, the DBSCAN algorithm is able to group 14964 into density-based clusters while the rest are identified as noise points.



(Figure 16)

After this, a pair plot was made to see the result of DBSCAN on each pair of variables (see figure 17).

As the figure shows, applying DBSCAN to our dataset produced a single density-based cluster while isolating a few noise points.

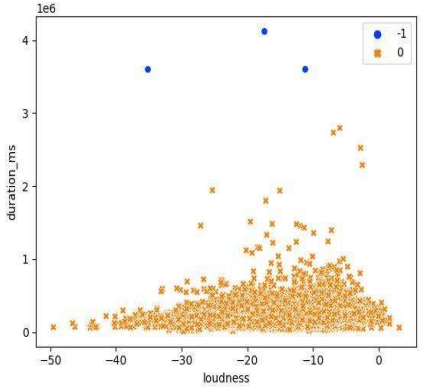
Afterwards we made a count of the noise points by genre which revealed that 2/3 of the noise points found were of the *sleep* genre, this can be seen well in figure 18, in which we can see a scatter plot of the *loudness* and *duration* attributes that shows that the noise points have a high duration, just like most tracks of the *sleep* genre.


(Figure 17)

Finally, we calculated the silhouette score, which is 0.35, even though this score can be considered 'good', it isn't reflected in the actual results, since its application yielded a single massive cluster.

Before drawing a conclusion, we decided to apply DBSCAN using different metrics, such as Manhattan and Jaccard, but unfortunately, they both produced similar results.

After having done a comparison, we concluded that density-based clustering does not produce any usable results due to the curse of dimensionality.

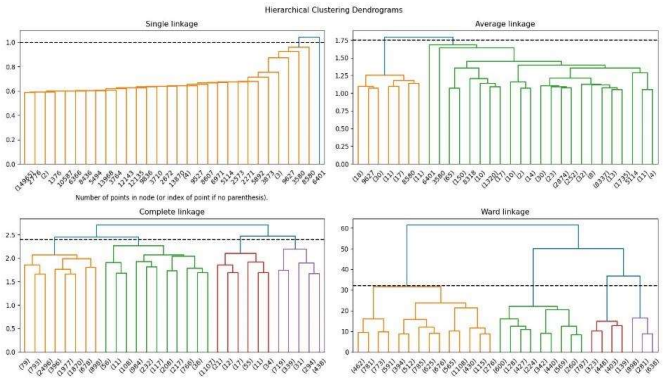


(Figure 18)

3.3. Hierarchical

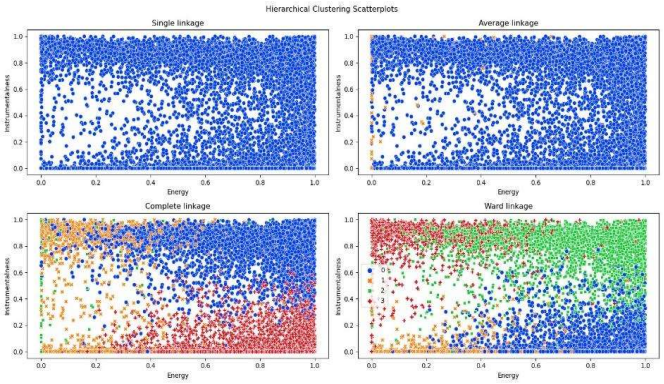
The hierarchical cluster is a useful method for representing the nested structure of a dataset and finding the optimal number of clusters. The first step of the agglomerative clustering was the choice of the best kind of linkage parameter between the *single*, *complete*, *average* and *ward* configurations.

The dendrograms using the different distances were plotted to visualize the clustering processes, (see figure 19)



(Figure 19)

As can be seen in figure 20, the hierarchical clustering using the *single* and *average* linkage parameters produced results like those obtained from DBSCAN, this is due to the fact that *single* linkage uses the minimum distance to generate the clusters and given that the *average* distance between the clusters in a dataset as large as ours is close to the minimum, they end up producing similar results.

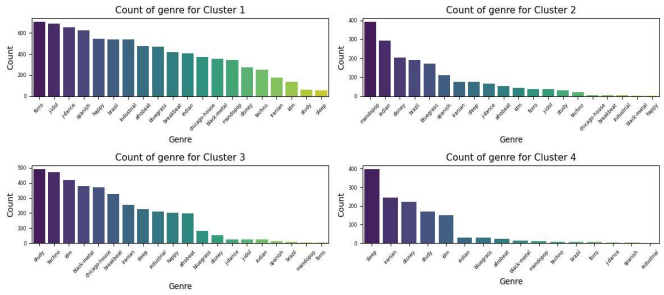


(Figure 20)

The hierarchical clustering using the *complete* and *ward* linkages produced better-defined clusters.

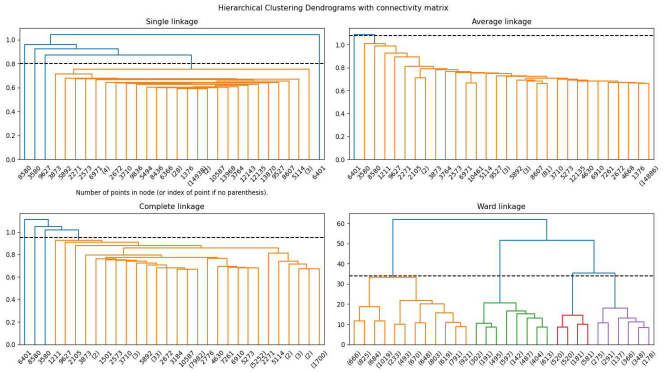
This can be explained by the fact that the *complete* linkage uses the maximum distance between points to create the groups, while *ward* utilizes the sum of squared errors to do the same, which makes its behavior like that of K-means.

Therefore, after comparing both methods, *ward*, - with a silhouette score of 1.77, - produces the better results.



(Figure 21)

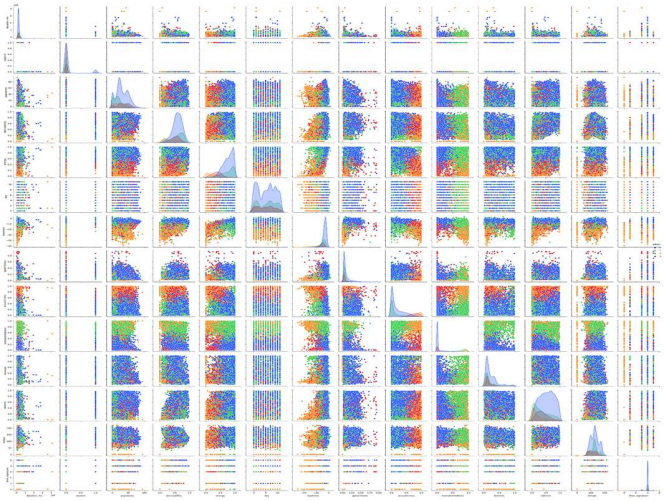
Looking at figure 21, the distribution of genres per cluster that has been found is not the same as the one obtained with K-means, that's why a connectivity matrix, - with 5 as the number of neighbors, - has been imposed by using the *connectivity* parameter in the agglomerative clustering. As can be seen in figures 22 and 23, single, average, and complete linkage did not provide an



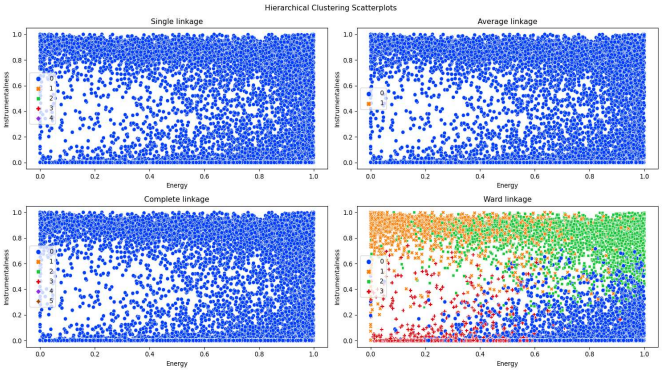
(Figure 22)

efficient clustering, this is why, quoting the scikit-learn documentation, “when using a connectivity matrix, single, average and complete linkage are unstable and tend to create a few clusters that grow very quickly”.
The ward linkage, with a cut at the 34, has represented a dendrogram similar to the previous one, but with a more balanced distribution of the clusters, with a distribution of the genres per cluster, with reference to the energy - instrumentalness scatterplot at figure 23 that is almost equal to the one obtained with k-means and with a higher silhouette score, closer to 1.9.

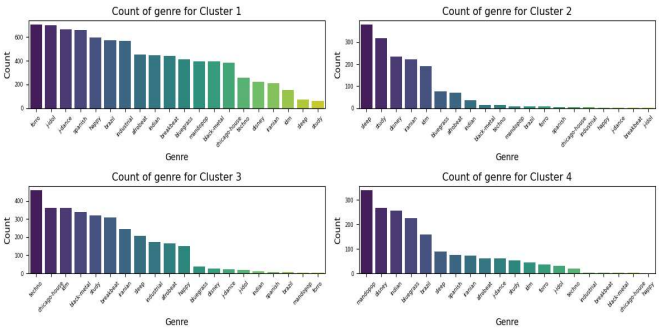
Therefore, it has been decided to choose the agglomerative clustering with ward linkage and connectivity matrix, which pairplot has been plotted in figure 25



(Figure 25)



(Figure 23)



(Figure 24)

CLUSTERING ALGORITHM	SILHOUETTE SCORE
K-means	0.21
DBSCAN	0.35
Hierarchical (Ward + connectivity)	0.189

(Table 1)

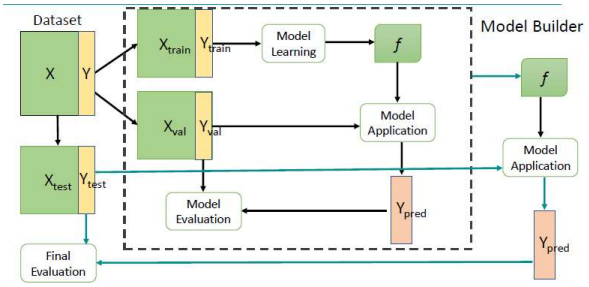
3.4. Conclusion

Drawing the conclusion, we analyzed all the three algorithms and we found that we can divide in this dataset the genres in 4 major groups, considering their mean value of energy and instrumentalness, starting from the less energetic and more instrumental genres, like *study* and *sleep*, to the most energetic and least instrumental, like *techno* and *black metal*.

Considering their representative performance, their behavior with reference to our dataset and the silhouette score, we found that k-means has the best trade-off between visual representation, distribution of the genres and silhouette score.

4- Classification

Moving into classification, many supervised classification algorithms were applied and compared to find out whether it was possible to get an efficient classification for the categorical attributes present in our dataset. We mainly focused on attempting to classify each of the songs into the correct genres.



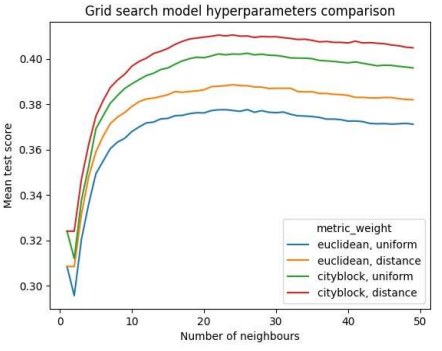
(Figure 26)

When it came to the preparation of our classification models, we decided to do a stratified split of our training dataset, which contains 15000 records, with 2/3 of the records for the training of the model leaving the remaining 1/3 for validation, while leaving our test dataset for the final evaluation. This decision was made to abide by the Model Evaluation Process (see figure 26).

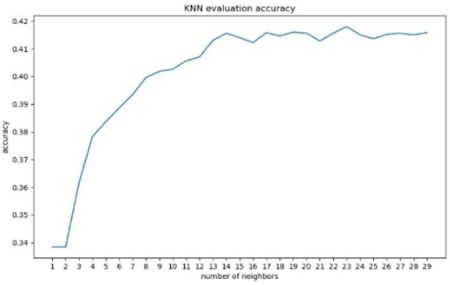
During the application of each model, a grid search was applied to fine-tune the classification algorithms by trying different parameters with a stratified cross validation (RepeatedStratifiedKFold from sk-learn).

4.1.- KNN

When applying the KNN classification algorithm, the first step was to choose the right hyperparameters to use as well as choosing the number of neighbors based on the changes in accuracy for the different metrics and weights (see figure 27). After examining the results of the grid search, it turned out that the model with the highest level of accuracy is the model which has the number of neighbors equal to 24, when using the Manhattan metric and applying the *distance* weights, ultimately yielding an accuracy of 0.415 in its classification.
This result was later confirmed by showing how the accuracy score changes across different numbers of neighbors using the parameters previously mentioned (see figures 28).



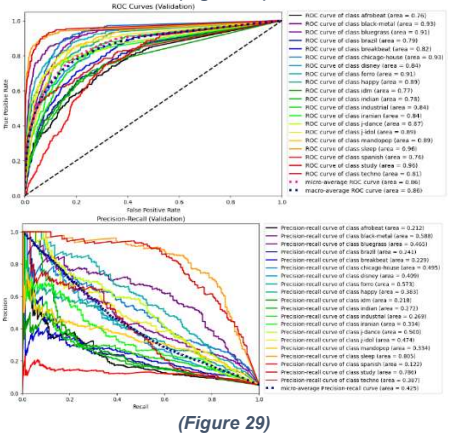
(Figure 27)



(Figure 28)

The next step before concluding the validation process was to look at the receiver operating characteristic (ROC) curve of the performance of our model. As can be seen in figure 29, our model was able to perform well despite the high number of classes, yielding micro and macro average AUC scores of 0.86, indicating that the model was able to classify the classes as "true positives" and seeing as none of the classes fell under the 45-degree line.

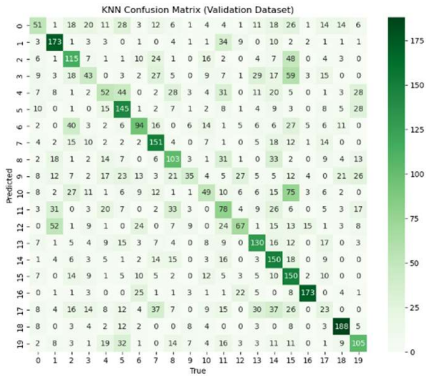
However, before moving on to the final evaluation process, a plot of the Precision-Recall curve was made to see how accurately our model was able to classify each class, our model was able to classify about half of the genres above the area under the curve (AUC) with a micro-average score of 0.425, with the better-performing ones being the *sleep* and *study*



(Figure 29)

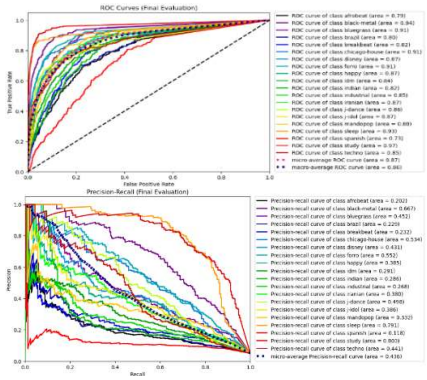
genres, while the worst-performing ones where the *afrobeat* and *spanish* genres, this can ultimately be down to the fact that those genres don't have musical characteristics, at least ones that can be estimated with numbers, that are so distinct from other genres present in our dataset.

After having trained our model and validating using the stratified split subset, we moved on to running the "test" dataset through our classification model created using the whole dataset of 15000 records and yielding the following results (see figure 30). By comparing them to those in the matrix in figure 31, we can see that the final model produced similar results, which means that our model is able to perform in a consistent manner when classifying previously unseen data.



(Figure 30)

(Figure 31)



(Figure 32)

Drawing from the results of the final evaluation, the model was unable to classify the afrobeat, *brazil*, *disney*, *idm*, *indian*, *industrial*, *iranian*, and *spanish* genres accurately. This can be explained by the fact that those genres lack any dominant features when compared to the others, this is also compounded by the fact that the model is attempting to classify them across 20 different classes making it unable to perfectly tell them apart.

After looking at these underwhelming results, we moved on to classifying the 4 groups obtained from the K-means clustering previously done section 3.1 to see if it was able to classify them correctly.

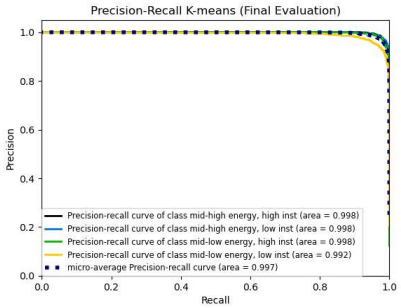
	precision	recall	f1-score	support
afrobeat	0.27	0.16	0.20	250
black-metal	0.56	0.67	0.61	250
bluegrass	0.40	0.48	0.44	250
brazil	0.28	0.20	0.23	250
breakbeat	0.26	0.20	0.22	250
chicago-house	0.42	0.59	0.49	250
disney	0.50	0.35	0.41	250
forro	0.47	0.60	0.53	250
happy	0.38	0.38	0.38	250
idm	0.49	0.18	0.27	250
indian	0.33	0.19	0.24	250
industrial	0.30	0.33	0.32	250
iranian	0.51	0.34	0.41	250
j-dance	0.46	0.53	0.49	250
j-idol	0.35	0.53	0.42	250
mandopop	0.28	0.61	0.39	250
sleep	0.81	0.70	0.75	250
spanish	0.13	0.09	0.11	250
study	0.69	0.80	0.74	250
techno	0.47	0.44	0.45	250
accuracy			0.42	5000
macro avg	0.42	0.42	0.40	5000
weighted avg	0.42	0.42	0.40	5000

(Table 2)

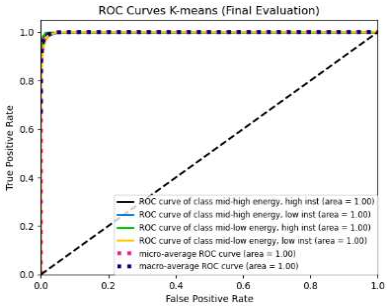
For the selection of the model, the same hyperparameters from the one previously used to produce these results (see figures 33, 34, 35 and table 3). As can be seen in the confusion matrix, the model was able to do a much better job at assigning the record to the right labels with an accuracy of 0.97.

	precision	recall	f1-score	support
0	0.96	0.95	0.96	1027
1	0.97	0.99	0.98	2298
2	0.99	0.97	0.98	618
3	0.99	0.97	0.98	1057
accuracy			0.97	5000
macro avg	0.98	0.97	0.97	5000
weighted avg	0.97	0.97	0.97	5000

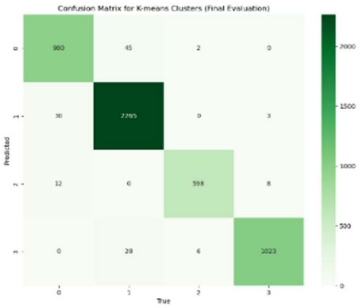
(Table 3)



(Figure 33)



(Figure 34)



(Figure 35)

Moving on to the ROC curve, an AUC score of 1.00 was obtained for each of the classes while the Precision-Recall curve yielded a micro-average AUC score of 0.997, meaning that the model was able to assign almost every record to the right label.

4.2 Naïve Bayes

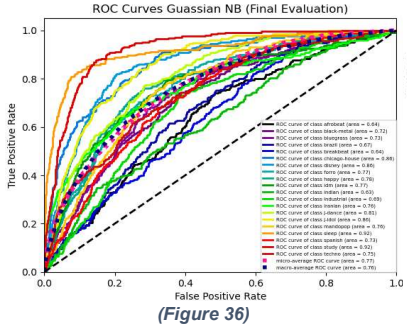
When applying the Naïve Bayes algorithm, we repeated the same step of the Model Evaluation Process to prepare our classification model for each music genre, which yielded the following results after the final evaluation (see figure 37).

By looking at the confusion matrix, it becomes apparent that the Naïve Bayes algorithm is unable to assign the records to the correct genre. A possible reason for this is that this algorithm assumes that the variables are conditionally independent from each other, which may not hold for some attributes.

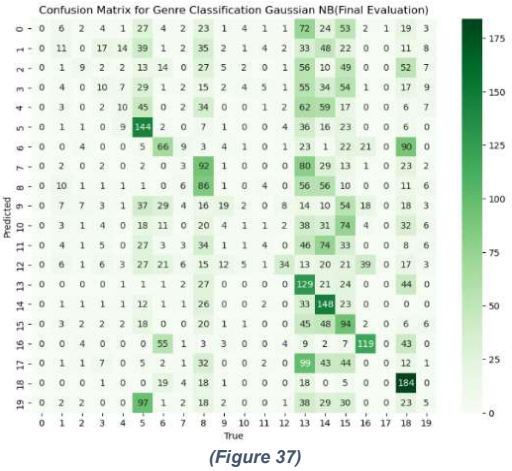
Also the ROC curves has been analyzed for this algorithm and while it confirms that the model is able to assign the records of the *study*, *sleep*, *j-idol*, *j-dance* and *chicago-house* correctly, it is only able to do so a little more than half of the time, while misassigning the rest of the record to different classes with an overall accuracy of 0.2152, ultimately showing that it heavily underperformed when compared to the KNN as can be seen on the table below.

	precision	recall	f1-score	support
afrobeat	0.00	0.00	0.00	250
black-metal	0.17	0.04	0.07	250
bluegrass	0.24	0.04	0.06	250
brazil	0.15	0.04	0.06	250
breakbeat	0.19	0.04	0.07	250
chicago-house	0.26	0.58	0.36	250
disney	0.29	0.26	0.27	250
forro	0.06	0.01	0.02	250
happy	0.16	0.34	0.21	250
idm	0.32	0.08	0.12	250
indian	0.05	0.00	0.01	250
industrial	0.16	0.02	0.03	250
iranian	0.56	0.14	0.22	250
j-dance	0.14	0.52	0.21	250
j-idol	0.21	0.59	0.31	250
mandopop	0.14	0.38	0.20	250
sleep	0.57	0.48	0.52	250
spanish	0.00	0.00	0.00	250
study	0.30	0.74	0.42	250
techno	0.07	0.02	0.03	250
accuracy			0.22	5000
macro avg	0.20	0.22	0.16	5000
weighted avg	0.20	0.22	0.16	5000

(Table 4)



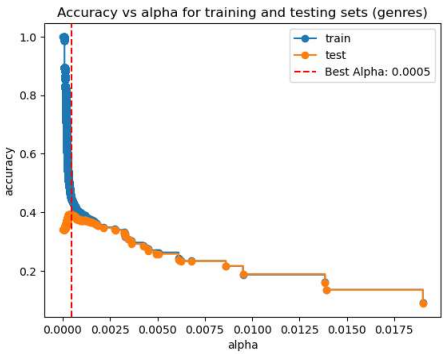
(Figure 36)



(Figure 37)

4.3 Decision Tree

When applying the Decision Tree classification algorithm, we first decided to use it to classify each genre, just like we did for both KNN and Naïve Bayes. The first step when doing so was to do the pre-pruning process, which was done through a Randomized Search Cross Validation together with the Repeated Stratified K-Fold to find the parameters that yielded the best results (max_depth = 12, min_samples_leaf = 20, min_samples_split = 50) when applying the classification on our dataset.



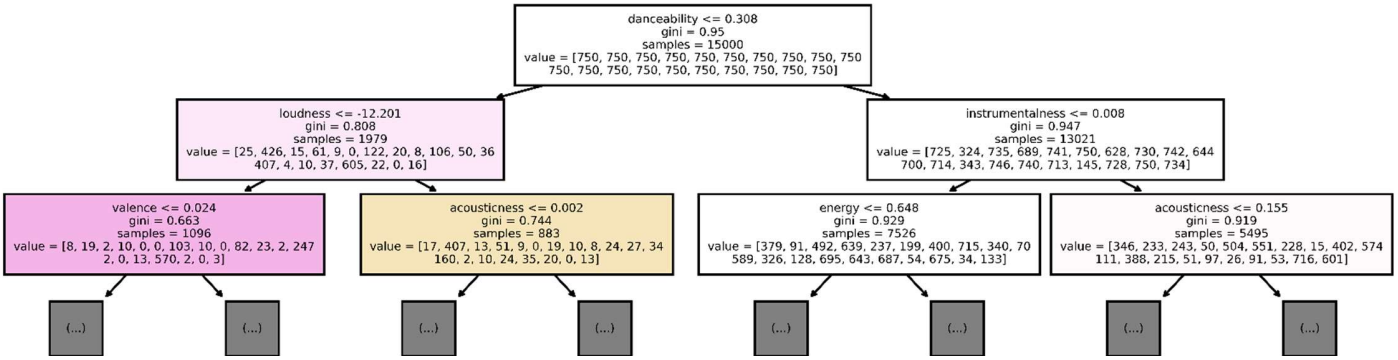
(Figure 38)

After this process, the following parameters were found (see figure 38), and to prevent overfitting, we moved on to the post-pruning phase in which we removed the weakest links, by calculating the right value for the Cost Complexity Pruning-Alpha (ccp_alpha).

This was done by comparing the accuracy scores for the training and validation subsets as the value of alpha increases and finding the point where the accuracy scores meet (see figure 38), giving us the best value for *ccp_alpha* (0.0005) with an accuracy of 0.38, which is the point where the model is able to generalize and, in turn, perform well when dealing with previously unseen data.

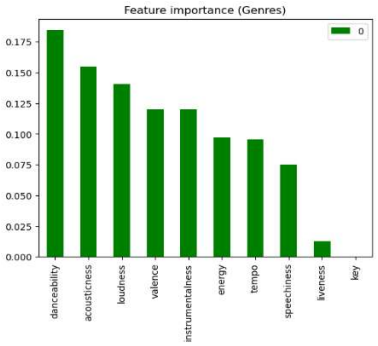
Moving forward in our analysis, we generated a decision tree from the whole "train" dataset applying all the parameters obtained during pruning process. As can be seen in figure 39, the first split was done between the record with high and low danceability, showing that, even in early stages, the model was able to identify genres that are not associated with dancing such as *black-metal*, *sleep* and *iranian*. This group is subsequently divided in terms of loudness, effectively separating the *black-metal* and *sleep* genres.

Meanwhile on the other branches of the second split, the model divides the remaining records according to their instrumentalness values, and even though there are still many records present, the *techno*, *sleep*, *idm*, *chicago-house* and breakbeat genres are the most prevalent.



(Figure 39)

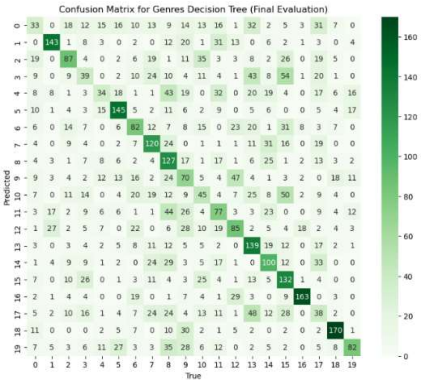
Finally, we looked at the feature importance table to see which of them were the most useful for the model, and while there is not much difference between them, it can be said that the most relevant were danceability and acousticness overall, while the key attribute remained as the least informative one.



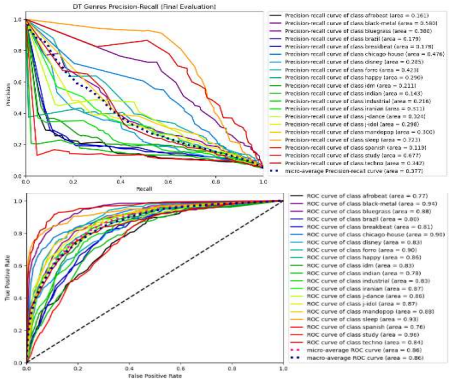
(Figure 40)

After looking at the confusion matrix, we can confirm the insights obtained from the first splits of our tree, i.e. that our model is able to classify the *black-metal*, *chicago-house*, *forro*, *sleep* and *study* genres quite well, since they have more defining characteristics compared to the other genres.

And finally, after examining the ROC and the Precision-Recall curves, we can see that the Decision Tree classification algorithm yielded similar results as those produced by KNN.

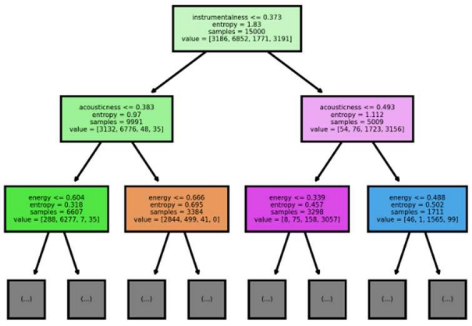


(Figure 41)



(Figure 42)

To conclude the application of the Decision Tree classification algorithm, we applied it to see how it performed in classifying the records into the 4 groups obtained from K-means, just like we did with KNN.



(Figure 43)

We can observe that in this case, the model decided to split the record according to their instrumentalness value, being able to created two groups where in one we have first and second clusters, while leaving the others on the right branch. Subsequently, the algorithm separates the resulting nodes in terms of acousticness, effectively separating the clusters into 4 distinct groups for each label. This behaviour suggests that a good grouping for the records in our dataset could be formed based on these two attributes.

Ultimately this model was able to classify almost all of the previoulsy unseen records correctly with an accuracy rate of 0.967.

4.4 Conclusion

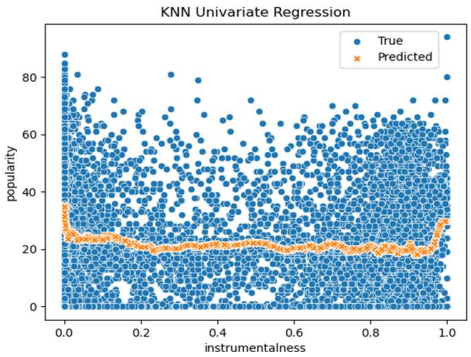
By comparing the results, we can conclude that the KNN algorithm is the most accurate of the three classification methods when applied to both 20 different genres as well as labels obtained from the K-means clustering, while the Naïve Bayes underperfoms when compared to the other two.

	Genres	K-means labels
KNN	0.418	0.973
Naïve Bayes	0.22	0.574
Decision Tree	0.382	0.967

(Table 5)

5- Regression

5.1 Univariate regression



(Figure 44)

Starting with regression, we set off to predict the continuous variable *popularity* using the *instrumentalness* attribute, the choice of predicting it through this attribute was made since it was the one with the highest correlation with our dependent variable.

A random search was also done in the application of each method to find the best parameters for each of them during the validation process, these same parameters were later applied during our final evaluation.

For the linear regression, we applied three methods from the scikit-learn library to predict our target variable, these were LinearRegression, Ridge and Lasso.

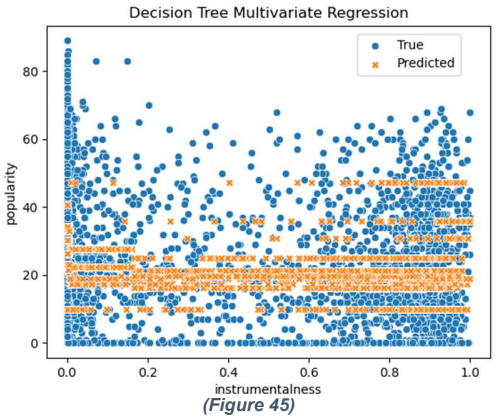
Moving on to the nonlinear regression, the K-Nearest Neighbors, and the Decision Tree regressors were applied.

As can be seen on the table, overall, the results were quite underwhelming, with the KNN regressor yielding the best results, since it was able to explain a over 12% of the variance in our dependent variable while having a lower amount of error compared to the other methods, the results of which can be seen in table 6.

	R2	MSE	MAE	Parameters
Linear Reg.	0.090	316.436	14.763	-
Ridge	0.090	316.453	14.763	alpha = 5.176
Lasso	0.089	316.453	14.786	alpha = 0.110
KNN Reg.	0.126	304.143	14.293	algorithm = kd_tree n° of neighbors = 260 p = 2 weights = uniform
Decision Tree	0.121	305.95	14.336	ccp_alpha = 0.375 max_depth = 29 max_features = log2 min_samples_leaf = 43 min_samples_split = 9

(Table 6)

5.2 Multivariate regression



(Figure 45)

In this section, we made a second attempt at predicting the *popularity* attribute, using the remaining continuous musical attributes present in our dataset (i.e. *instrumentalness*, *speechiness*, *loudness*, *duration_ms*, *valence*, *liveness*, *acousticness*, *danceability* and *energy*)

Following the same steps as in section 5.1, we applied the five previously used methods for both linear and nonlinear regression.

As can be seen on the table to the right, the use of multiple variables did improve the score of the determination coefficient for every method except the KNN regressor, which somehow yielded worse results after introducing multiple independent variables.

This drastic change in performance could be explained by the increase in dimensionality, although further analysis is needed to pinpoint the reason behind it.

By looking at our results, the best method for multivariate regression for our dataset is Decision Tree, since, according to its determination coefficient, it was able to explain close to 20% of the variance of the dependent variable. These results lead us to conclude that the Decision Tree regressor is the best performer of the bunch, since it remained consistent in both univariate and multivariate regression.

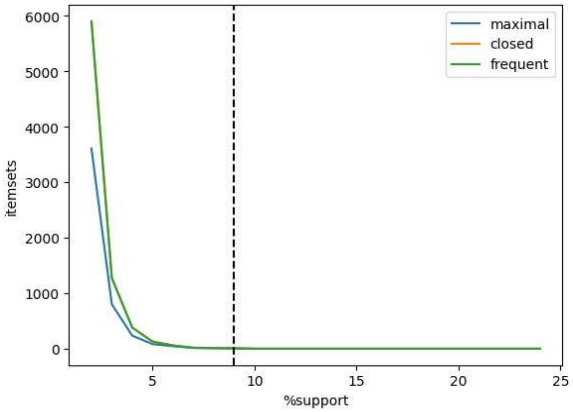
	R2	MSE	MAE	Parameters
Linear Reg.	0.118	306.885	14.374	-
Ridge	0.118	306.890	14.375	alpha = 1.301
Lasso	0.109	310.145	14.518	alpha = 0.110
KNN Reg.	0.022	340.224	15.379	algorithm = kd_tree n° of neighbors = 646 p = 1 weights = uniform
Decision Tree	0.198	278.926	13.333	ccp_alpha = 0.593 max_depth = 11 max_features = None min_samples_leaf = 17 min_samples_split = 39

Table 7

6- Pattern Mining

This part is about finding out the frequent sets of items present in our dataset and drawing association rules from them. We particularly focused on discovering patterns that could be associated with a particular range of popularity values and see if it is possible to determine which features are commonly linked with highly popular tracks.

During this part, the original dataset was revisited to try and handle the missing values by using the association rules found during this process.



(Figure 46)

6.1 Item sets

When determining the itemsets, the first step was to remove the categorical attributes from our dataset, this was followed by a discretization of the continuous variables into groups of three (low, medium, and high), and subsequently applied both the APRIORI and FP Growth algorithms to draw the patterns for our data.

To find the best value for the minimum support, a line chart (figure 46) of the cardinality of each list of itemsets as the value of support increased from 2 to 25 was made with a 'zmin' of 4 to ensure relevant results were obtained. A support value of 9 was then chosen to extract only the most frequent patterns that were composed of 4 attributes or more.

After having run the algorithm with the previously mentioned parameters, it was possible to identify the best itemsets for each type (*maximal*, *all*, *closed*).

	frequent_itemset	support
0	HIGH_ACOU (0.415, 0.996], LOW_TEMP (-0.001, 109.646], LOW_LOUD (-49.532, -9.214], LOW_ENER (-0.001, 0.562]	9.373333
1	HIGH_ACOU (0.415, 0.996], LOW_DURA (8585.999, 196231.667], LOW_LOUD (-49.532, -9.214], LOW_ENER (-0.001, 0.562]	9.193333
2	HIGH_ACOU (0.415, 0.996], HIGH_INST (0.377, 1.0], LOW_LOUD (-49.532, -9.214], LOW_ENER (-0.001, 0.562]	9.400000
3	HIGH_ACOU (0.415, 0.996], LOW_LOUD (-49.532, -9.214], LOW_DANC (-0.001, 0.496], LOW_ENER (-0.001, 0.562]	9.800000
4	HIGH_ACOU (0.415, 0.996], LOW_LOUD (-49.532, -9.214], LOW_VALE (-0.001, 0.269], LOW_ENER (-0.001, 0.562]	9.966667
5	LOW_LOUD (-49.532, -9.214], LOW_DANC (-0.001, 0.496], LOW_VALE (-0.001, 0.269], LOW_ENER (-0.001, 0.562]	9.420000

(Figure 47)

As can be seen in figure 47, the resulting itemsets were a group of highly correlated variables, particularly in the lower end of our range, which seem to indicate that roughly ten percent of the record in our dataset are ones that are characterized by having low values for energy, valence, danceability, loudness, tempo, together with a high acousticness.

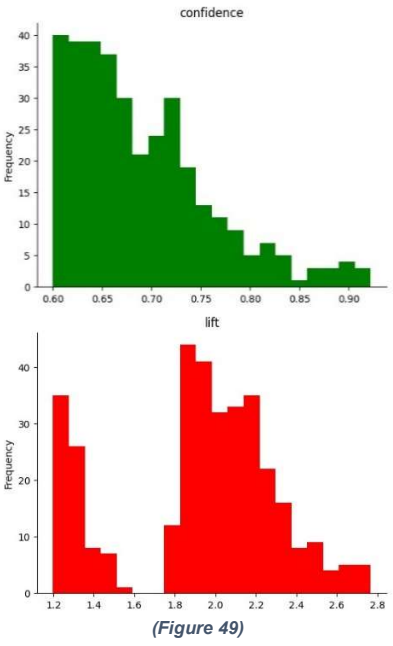
6.2 Association Rules

The first step taken to mine the association rules was to create a matrix (figure 48) that would show the number of rules that would come up for different values for both support and confidence.

index	50	60	70	80	90
10	830	286	126	28	4
15	102	36	19	3	0
20	12	4	2	0	0
25	0	0	0	0	0
30	0	0	0	0	0

(Figure 48)

After looking at the results, a rerun of the algorithms was done after having set the parameters for support and confidence to 10 and 50 respectively. This yielded a total of 830 association rules, the distribution of which can be seen in the bar charts in figure 49.



(Figure 49)

As can be seen in the figures, all these rules have a confidence value above 0.60, which indicates that over 60 percent of the consequents of each rule are preceded by the antecedents found by our algorithm. These rules are further cemented by the fact that a large majority of them have a lift value above 1.8, indicating a very positive relationship between the items and signaling that they are way more likely to be found together than one would expect by chance.

A table of the 4 rules with the highest lift values can be seen in figure 50, indicating the strong relationship between loudness, instrumentalness, energy and acousticness. This strong relationship between the attributes explains why classification algorithms, such as Decision Tree, choose to split the records based on these variables first as seen in section 4.3.

	consequent	antecedent	abs_support	%_support	confidence	lift
1	LOW_LOUD (-49.532, -9.214]	HIGH_ACOU (0.415, 0.996], HIGH_INST (0.377, 1.0], LOW_ENER (-0.001, 0.562]	1410	9.400000	0.922171	2.766514
0	LOW_ENER (-0.001, 0.562]	HIGH_ACOU (0.415, 0.996], LOW_TEMP (-0.001, 109.646], LOW_LOUD (-49.532, -9.214]	1406	9.373333	0.913580	2.735816
3	LOW_ENER (-0.001, 0.562]	HIGH_ACOU (0.415, 0.996], LOW_LOUD (-49.532, -9.214], LOW_VALE (-0.001, 0.269]	1495	9.966667	0.911030	2.728179
2	LOW_ENER (-0.001, 0.562]	HIGH_ACOU (0.415, 0.996], LOW_LOUD (-49.532, -9.214], LOW_DANC (-0.001, 0.496]	1470	9.800000	0.900735	2.697351

(Figure 50)

The next step in the search for association rules was to see which patterns included the popularity attribute and see if there were more intricate connections between it and the rest of the variables. For this part, the value of the 'zmin' parameter was changed to 3, to allow for a broader range of results.

In order to obtain relevant results, a quantile-based split was done to divide the values of the popularity attribute into 2 bins of high and low popularity. After rerunning the algorithm with the values for support and confidence previously used, some patterns emerged.

As can be seen in figure 51, tracks that turn out to have a high popularity are mainly those with low instrumentalness since this attribute is present in the top five association rules while the opposite is also true for tracks with low popularity, given the fact that a sizable amount of them have a high value of instrumentalness.

All of these association rules had a lift value above 1.0 (see figure 52), and while not so high, it does indicate that these patterns occur more frequently than they otherwise would by mere chance. Furthermore, this relationship confirms that instrumentalness is the best attribute for predicting the popularity value in a regression task as it was done in section 5.1.

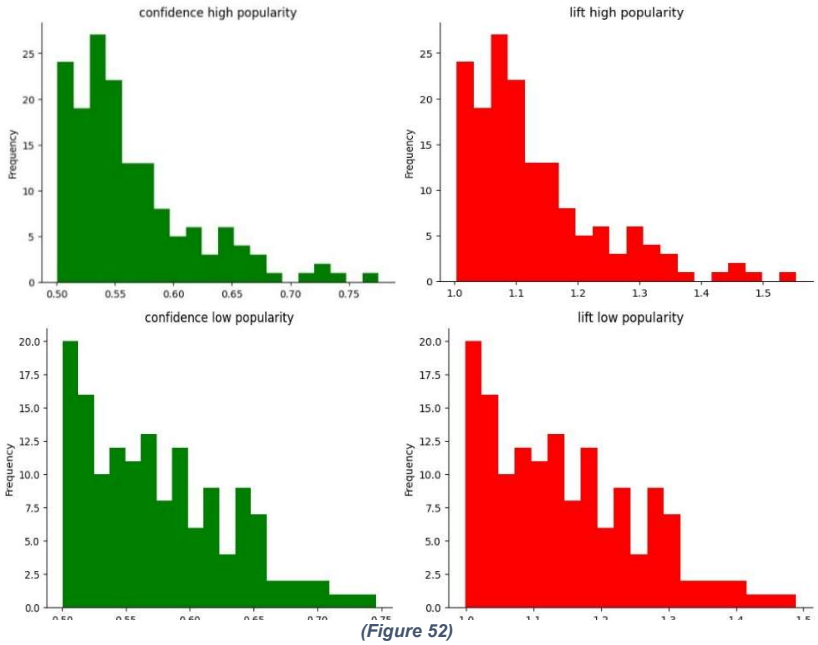
	consequent	antecedent	abs_support	%_support	confidence	lift
0	HIGH_POP (24.0, 94.0]	('HIGH_ACOU (0.415, 0.996]', 'LOW_INST (-0.001, 1.27e-05]')	1355	9.033333	0.732037	1.467204
1	HIGH_POP (24.0, 94.0]	('MID_LOUD (-9.214, -5.791]', 'LOW_INST (-0.001, 1.27e-05]')	1328	8.853333	0.715517	1.434094
2	HIGH_POP (24.0, 94.0]	('HIGH_TEMP (134.984, 220.525]', 'LOW_INST (-0.001, 1.27e-05]')	1303	8.686667	0.725097	1.453295
3	HIGH_POP (24.0, 94.0]	('LOW_INST (-0.001, 1.27e-05]', 'LOW_ENER (-0.001, 0.562]')	1125	7.500000	0.742574	1.488324
4	HIGH_POP (24.0, 94.0]	('LOW_INST (-0.001, 1.27e-05]', 'LOW_SPEE (-0.001, 0.0412]')	1505	10.033333	0.775374	1.554062
	consequent	antecedent	abs_support	%_support	confidence	lift
0	LOW_POP (-0.001, 24.0]	('HIGH_ENER (0.834, 1.0]', 'HIGH_INST (0.377, 1.0]')	1139	7.593333	0.723175	1.443270
1	LOW_POP (-0.001, 24.0]	('HIGH_ENER (0.834, 1.0]', 'HIGH_DURA (265257.667, 4120258.0]')	1297	8.646667	0.701081	1.399177
2	LOW_POP (-0.001, 24.0]	('HIGH_ENER (0.834, 1.0]', 'LOW_VALE (-0.001, 0.269]')	1059	7.060000	0.701325	1.399663
3	LOW_POP (-0.001, 24.0]	('HIGH_INST (0.377, 1.0]', 'HIGH_DURA (265257.667, 4120258.0]')	1596	10.640000	0.746143	1.489109
4	LOW_POP (-0.001, 24.0]	('HIGH_INST (0.377, 1.0]', 'LOW_ACOU (-0.001, 0.0316]')	1495	9.966667	0.721525	1.439978

(Figure 51)

Finally, an attempt to use the association rules to replace missing values in the *popularity_confidence* attribute was done using our original dataset.

To do this, the first step was to drop all the rows which had NaN values, leaving the dataset with only 2217 records.

The second step was to discretize the *popularity_confidence* attribute by splitting it into three bins using the pd.qcut() function. Our hypothesis was to substitute the missing values in our original dataset with a representative value of *popularity_confidence* which corresponds to the mean of the edges of each bin.



(Figure 52)

After this, a run through the algorithms was done using only the continuous attributes, except for *popularity*, and the following parameters: supp = 15, conf = 60, and zmin=3. However, the algorithm was not able to identify any association rules whatsoever, leading to a reduction in our parameters (supp = 10, conf = 40, zmin = 3) which yielded underwhelming results.

Considering the support, confidence and lift values for the association rules obtained (see figure 53), even though these rules have a lift value slightly above 1.2, low support (4-6%) together with confidence values of around 0.40, It was concluded that these association rules do not hold and would not be a proper way of dealing with the missing values.

	consequent	antecedent	abs_support	%_support	confidence	lift
0	HIGH_POP_CONF (0.656, 1.0]	('HIGH_ENER (0.829, 1.0]', 'HIGH_LIVE (0.204, 0.994]', 'HIGH_LOUD (-5.856, 0.643]')	99	4.465494	0.407407	1.222222
1	HIGH_POP_CONF (0.656, 1.0]	('HIGH_ENER (0.829, 1.0]', 'HIGH_LOUD (-5.856, 0.643]', 'HIGH_SPEE (0.0687, 0.901]')	114	5.142084	0.425373	1.276119
2	HIGH_POP_CONF (0.656, 1.0]	('HIGH_ENER (0.829, 1.0]', 'MID_DANC (0.504, 0.658]')	102	4.600812	0.408000	1.224000
3	HIGH_POP_CONF (0.656, 1.0]	('HIGH_LIVE (0.204, 0.994]', 'HIGH_SPEE (0.0687, 0.901]')	124	5.593144	0.423208	1.269625
4	HIGH_POP_CONF (0.656, 1.0]	('MID_INST (1.27e-05, 0.341]', 'HIGH_DURA (266593.667, 1149106.0]')	101	4.555706	0.413934	1.241803
5	HIGH_POP_CONF (0.656, 1.0]	('HIGH_LOUD (-5.856, 0.643]', 'HIGH_SPEE (0.0687, 0.901]')	140	6.314840	0.408163	1.224490
6	HIGH_POP_CONF (0.656, 1.0]	('MID_TEMP (109.759, 134.948]', 'LOW_INST (-0.001, 1.27e-05]')	96	4.330176	0.406780	1.220339

	consequent	antecedent	abs_support	%_support	confidence	lift
0	MID_POP_CONF (0.318, 0.656]	('HIGH_VALE (0.575, 0.995]', 'HIGH_TEMP (134.948, 220.525]')	104	4.691024	0.403101	1.209302

	consequent	antecedent	abs_support	%_support	confidence	lift
0	LOW_POP_CONF (-0.001, 0.318]	('HIGH_LIVE (0.204, 0.994]', 'LOW_TEMP (-0.001, 109.759]')	95	4.285070	0.400844	1.202532
1	LOW_POP_CONF (-0.001, 0.318]	('LOW_TEMP (-0.001, 109.759]', 'LOW_INST (-0.001, 1.27e-05]')	98	4.420388	0.404959	1.214876

(Figure 53)

7. Conclusion

Looking through our entire work, we found that the most representative features were *energy*, *loudness*, *instrumentalness*, *acousticness* when it came to the application of the different methods seen throughout the course.

To summarize, in the clustering section of the report, it was possible to identify four distinct clusters which we labeled based on their distribution with regards to the *energy* and *instrumentalness* attributes.

Moving on to the classification section, we were able to obtain satisfying results when attempting to classify previously unseen data to the correct cluster K-means' labels using both KNN and Decision Tree algorithms. Furthermore, it was possible to see the importance of these attributes when applying the Decision Tree classification method, as the first splits it does are based on *instrumentalness*, *acousticness*, and *energy*.

In the case of regression, we see the importance of *instrumentalness* once again, as it was possible to predict the potential values for popularity, even though we lacked more relevant information such as the number of times each particular song has been played as well as their release dates.

Finally, the importance of these features was again confirmed in the pattern mining section, since the most important association rules contained exactly these four features, not only that, but it was also seen how the rules for high and low popularity predominantly featured a combination of these attributes as antecedent.