UNIVERSITY OF PISA

DEPARTMENT OF INFORMATICS

Master Degree in Data Science and Business Informatics

Data Mining 1

Matteo Rofrano - Gabriele Gori - Francesco Lanci Lanci

Academic Year 2022/2023

Contents

Chapter 1

Data understanding and preparation

In this chapter will be introduced the meaning and the characteristics of the variables of the dataset used for the following analysis. The full list of variables with their data type[1] and their relative attributes are presented in the table (1). But first, let's get a basic understanding of the audio format and structure.

Variable	***Type***	***Attributes***
modality	Categorical	audio only
vocal_channel	Categorical	speech, song
emotion	Categorical	neutral, calm, happy, sad, angry, fearful, disgust, surprised
emotional intensity	Categorical	normal, strong
statement	Categorical	"Kids are talking by the door", "Dogs are sitting by the door"
repetition	Categorical	1st repetition, 2nd repetition
actor	Categorical	from 01 to 24
sex	Categorical	male, female
channels	Categorical	1 (mono audio), 2 (stereo audio)
sample_width	Categorical	1 (8-bit), 2 (16-bit)
frame_rate	Numerical	frequency of sample in Hertz
frame_width	Numerical	number of bytes for each frame
lenght_ms	Numerical	audio file lenght in milliseconds
frame_count	Numerical	number of frames from the sample
intensity	Numerical	loudness in dBFS
zero_crossing_sum	Numerical	sum of the zero-crossing rate
'mean', 'std', 'min', 'max', 'kur', 'skew'	Numerical	statistics of the original audio signal
'mfcc_mean', 'mfcc_std', 'mfcc_min', 'mfcc_max'	Numerical	statistics of the Mel-frequency Cepstral Coefficient
'sc_mean', 'sc_std', 'sc_min', 'sc_max', 'sc_kur', 'sc_skew'	Numerical	statistics of the spectral centroid
'stft_mean', 'stft_std', 'stft_min', 'stft_max', 'stft_kur', 'stft_skew'	Numerical	statistics of the stft chromagram

Table 1.1: Ravdess dataset variables

[1]In the Type column is presented the data type of the variables after the correction of errors presented in paragraph(1.3.1)

1.1 Data semantics

At the most basic level, audio is represented by a stream of samples (each sound sample is stored as binary data), each specifying the amplitude of the audio waveform as measured for a given slice of the overall waveform of the audio signal. There are several formats used for the individual samples within an audio file, in this study are used only 8-bit and 16-bit audio records which define the size of an individual sample called in this study sample width. The position of each audio source within the audio signal is called a channel. Each channel contains a sample indicating the amplitude of the audio being produced by that source at a given moment in time. For instance, in stereo sound, there are two audio sources: one speaker on the left, and one on the right. Each of these is represented by one channel, while in mono audio there is only one source of audio. The channels are then assembled into a series of audio frames, each consisting of one sample for each of the audio's channels. The number of frames that comprise a single second of audio varies depending on the sample rate used when recording the sound. Since the sample rate corresponds to the number of "slices" a sound wave is divided into for each second of time, it is sometimes thought of as a frequency and the samples per second measurement therefore uses the Hertz as its unit.
It is important to learn more about the most technical variables of the dataset, in particular we will discuss about the following ones:
The **sample_width** or bit depth is the number of bits available for each sample. The higher the bit depth, the higher the quality of the audio.
The **frame_rate** or sampling frequency is about how many samples, or measurements, of the sound are taken each second. The more samples that are taken, the more detail about where the waves rise and fall is recorded and the higher the quality of the audio. Also, the shape of the sound wave is captured more accurately.
The **frame_width** is calculated by multiplying the sample size in bytes by the number of channels.
The **intensity** is a measure of loudness which use decibels relative to full scale (dBFS) as unit of measure. It is a unit of measurement for amplitude levels in digital systems, which have a defined maximum peak. The level of 0 dBFS is assigned to the maximum possible digital level. For example, a signal that reaches 50% of the maximum level has a level of -6 dBFS, which is 6 dB below full scale.
The **zero_crossing_sum** is a variable that contain sum of the zero crossing rate. The Zero-Crossing Rate (ZCR) of an audio frame is the rate of sign-changes of the signal during the frame. In other words, it is the number of times the signal changes value, from positive to negative and vice versa, divided by the length of the frame. ZCR can be interpreted as a measure of the noisiness of a signal. For example, it usually exhibits higher values in the case of noisy signals. It has been widely used in both speech and music recognition.
The statistics of **Mel-frequency Cepstral Coefficient** or MFCC are some key variables of the dataset because the MFCC is a measure similar to the voice system of a human, so it can efficiently be used to characterize speakers, for instance. In fact, they are mainly used as features in speech recognition and music information retrieval.
The statistics of **spectral centroid** could be useful in the following study because a higher value of SC corresponds to more energy of the signal being concentrated within higher frequencies. Basically, it measures the spectral shape and position of the spectrum[2]. It can be observed that

[2]The audio frequency spectrum represents the range of frequencies that the human ear can interpret. Sound frequency is measured in Hertz (Hz) unit. This audible frequency range, in the average person at birth, is from 20Hz to 20000Hz, or 20 kHz.

higher values correspond to brighter sounds. Because the spectral centroid is a good predictor of the "brightness" of a sound, it is widely used in digital audio and music processing as an automatic measure of musical timbre.
The statistics of the **STFT Chroma** describe the Chroma value of an audio that, basically, represent the intensity of the twelve distinctive pitch classes that are used to study music. So, they can be employed in the differentiation of the pitch class profiles between audio signals. One main property of chroma features is that they capture harmonic and melodic characteristics of music, while being robust to changes in timbre and instrumentation.

1.2 Distribution of the variables and statistics

In order to avoid an excessive use of space we omit the plots of the single variable distribution. The important things that can be observed from the distribution of the single numerical variables is that "lenght_ms" and "frame_count" have a bimodal distribution. There are several features with a skewed distribution. The variables with the steepest skewed distribution are "skew" and "kur" relative to the statistics of the "stft", "sc" and the original audio signal. The statistics of the audio signals show how the audio waves are done. In fact we can see a higher boundary and a lower boundary since the maximum and the minimum values are ranging between -1 and +1 with a mean that is more or less always at 0 for every sample.

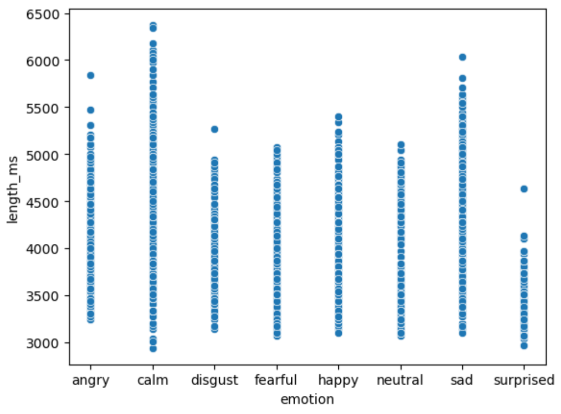


Figure 1.1: Length_ms for different emotions

Can be seen from the plot of the length_ms for the different emotion that the longer audio signals are those relative to the calm, sad and angry emotions. While the shortest audio signals are those where we have the surprised emotion.

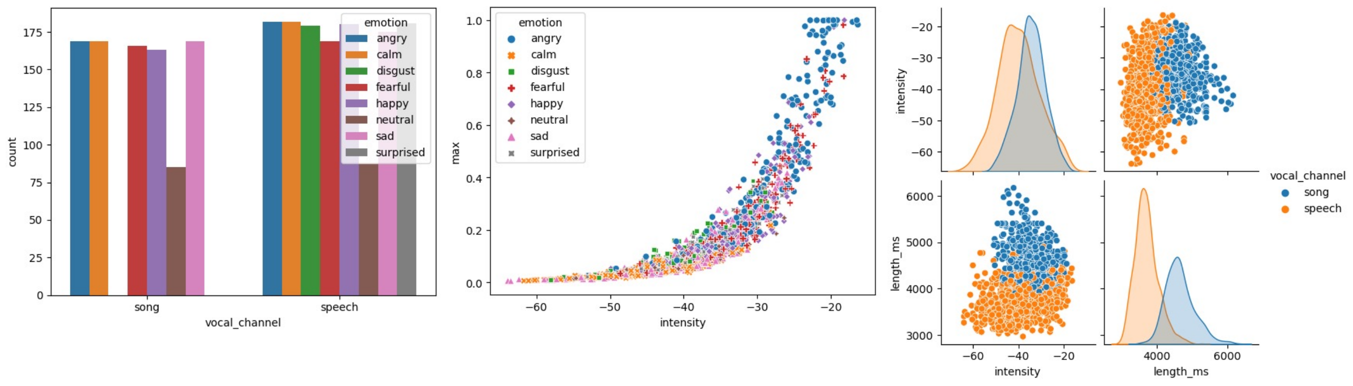


Figure 1.2: Exploring variables

The most interesting things that we have discovered by exploring singles and multiple variables are the following presented in figure(1.2). It can be seen by assessing the "song" observation and the "speech" observation that the first ones does not have some emotion that the second ones have. In particular, the "song" observation does not have the disgust and surprised emotions. "Intensity" has a non linear correlation with "max" variable and can be observed that for the higher level of the intensity and max of the audio signals the most dominant emotions are angry and fearful. Another interesting thing is that by assessing the length_ms distribution for song and speech it can be seen that songs audio signals have typically a longer duration than the speeches audio signals and moreover the songs tend to have an higher intensity than the speeches.

1.3 Data quality

1.3.1 Errors

First of all, assessing the different data types contained in the dataset we have noticed that "actor","channels" and "sample_width" variables have been stored as float data types so we have fixed these errors by transforming those variable, as we have done for the other categorical variables too, in the proper "category" data type which can be useful to save memory space and to use if necessary the logical order instead of the lexical one.

1.3.2 Missing values

In this dataset are present missing values for 3 particular variables. In fact are missing 196 values for "vocal_channel", 1126 values for "actor" and 816 values for "intensity". For each of this features we have used different techniques to fill the missing values. In particular for the "**vocal_channel**" feature we filled the values based on the distribution of the actual values within the 2 different classes. So we have kept the proportion between song and speech that are 60% of the total observation for speech and 40% for song.
For "**actor**" instead, we used the Nearest-Neighbour classification technique, because the accuracy by replacing the missing values according to the distribution is low.
First of all, for a better accuracy we decided to split the dataset in two: one with "sex" equal to female and the other equal to male, in this way the model cannot predict a female actor when the "sex" is male, and vice versa. Then since for the models we have used different variables we have scaled them in a range of (0,1) where 0 is the minimum value of the variable and 1 is the maximum one. Then we split each part in train set and test set and used the k-fold cross validation on the train sets, obtaining the average accuracy for each k of the KNN.
The best mean accuracy is obtained with k=3 for the set with "sex" equal to female and k=7 for the other set, so we trained the models on the entire train sets with these parameters and evaluated them on the test sets, obtaining the following results presented in table (1.2).
The accuracy are both good considering the number of classes, then the other evaluation measures are good for some classes and bad for others. In the end we used all the data to train the models, so we replaced the missing values.

Table 1.2: Actor classification report

Class M	Precision	Recall	F-score	Class F	Precision	Recall	F-score
1	0.53	0.59	0.56	2	0.79	0.32	0.46
3	0.50	0.32	0.39	4	0.57	0.42	0.48
5	0.68	0.52	0.59	6	0.67	0.71	0.69
7	0.29	0.31	0.30	8	0.88	0.75	0.81
9	0.18	1.00	0.30	10	0.71	0.81	0.76
11	0.73	0.48	0.58	12	0.21	0.35	0.27
13	0.69	0.48	0.56	14	0.57	0.36	0.44
15	0.50	0.47	0.48	16	0.32	0.75	0.44
17	0.29	0.71	0.42	18	0.58	0.88	0.70
19	0.21	0.33	0.26	20	0.48	0.71	0.57
21	0.83	0.79	0.81	22	0.67	0.50	0.57
23	0.45	0.45	0.45	24	0.38	0.56	0.45
accuracy			0.50				0.55
macro avg	0.49	0.54	0.48		0.57	0.59	0.55
weighted avg	0.56	0.50	0.51		0.61	0.55	0.55

In order to fill the missing values of "**intensity**" we used the linear interpolation method that allows us to estimate unknown data points between two known data points. First of all we noticed that intensity is strongly correlated with the "std" feature so in order to keep this correlation we have sorted all the values according to the std values and then we applied linear interpolation to "intensity". In this way the mean of "intensity" before and after the interpolation is approximately the same (-37.6) and even the correlation with "std" is approximately the same (0.845 vs 0.848).

1.4 Outliers and variables transformation

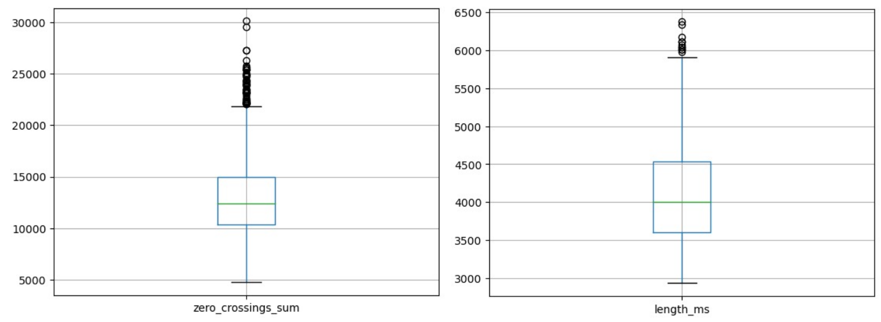


Figure 1.3: box-plot for zero_crossings_sum and length_ms

Since there are several variables that present a skewed distribution and for this reason they present even some outliers we have preferred to apply to these variable the log-transformation in order to deal with a more normalized distribution and treating in the same time the outliers caused by the skewed distribution. We have done this logarithmic transformation for the "zero_crossings_sum" and "lenght_ms" variables. Then for the variables relative to the statistics of the "mffc","stft" and "sc" we have noticed that the most meaningful for further clustering and classification modules are those relative to the mean and the standard deviation (that presents only few outliers) so only for those we have chosen to replace those outliers with the median since the mean is strongly affected by outliers.
Others interesting cases are those relative to "frame_count" and "frame_width". In fact, are presents 32 observations which present a frame_count=-1 and these are clearly errors since a sample cannot have a negative number of frames and for this reason these observations

have been replaced with the median. Since "frame_width" is the result of the multiplication between "channels" and "sample_width" and considering that there are only 6 observation with "channels" equal to 2 so there are only 6 observations for "frame_width"that have values equal to 4 while the rest are all 2. This 6 values represent only the 0.16% of the dataset so it leads to poor information and for this we have decided to drop this 6 observations.

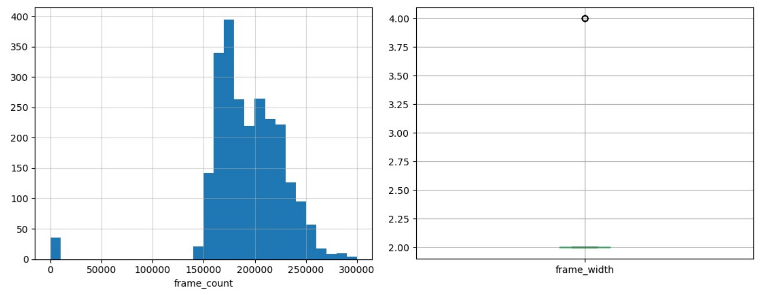


Figure 1.4: outliers for frame_count and frame_width

1.5 Correlations and elimination of variables

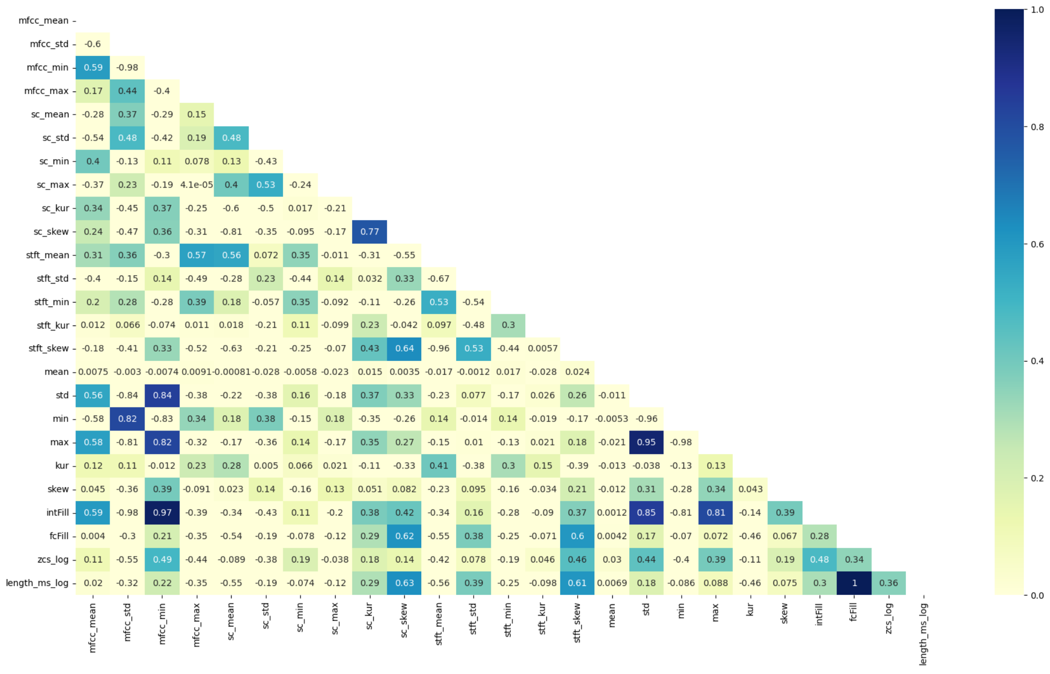


Figure 1.5: correlation matrix

In order to reduce the dimensionality and take only the more informative variables we have dropped all that variables with a single value ('modality', 'channels', 'frame_width', 'sample_width', 'frame_rate', 'stft_max'). Then we have dropped all that variables that are near to be perfect collinear[3] so we dropped fcFill[4], max e mfcc_min.

[3]We have used as threshold correlation ≥ 0.95

[4]It is frame_count after we have filled the missing values

Chapter 2: Clustering

The cluster analysis has been performed only on the numerical variables while the categorical ones has been used to check the composition of each cluster we have found. Since the numerical variables of this dataset have different scale of measure, in order to avoid that distances are affected by this differences we have performed the scalarization of the variables and the following normalization in order to have the same scales of measure for all the different variables and to make them ranging from 0 to 1. We performed cluster analysis on the entire dataframe (without getting any particular result) and then we performed it on a subset dataframe (that we will call clustering subset) in order to get more informative clusters, in fact we kept the variables that have a correlation higher than 50% with the most of the other variables and we dropped these ones and moreover we dropped those variables with a steep skewed distribution since they show a large amount of noise. So now our **clustering subset** is composed by "mfcc_mean","sc_mean","stft_mean" and "lenght_ms_log". In the end, since we found that one of the best pair of variables to plot for clustering analysis is given by "stft_mean" and "sc_mean" (that are measure used for song information retrieval), we also decided to apply clustering algorithms on song observations, creating in this way the **song subset** (exploiting the same attributes). All the details about the single algorithms we used will be discussed below.

2.1 Kmeans algorithm and variations

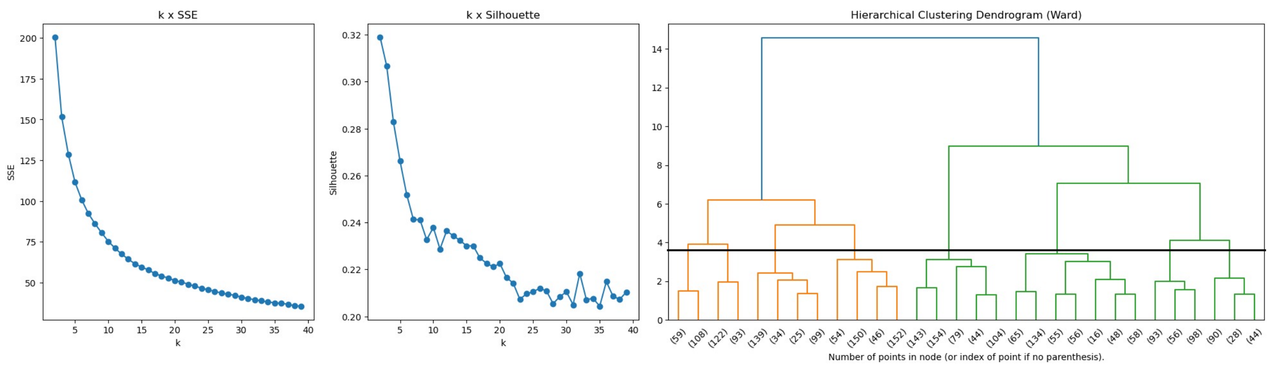


Figure 2.1: Measures to choose the best k

In order to choose the best k for the K-means of the **clustering subset** we have computed the SSE and the Silhouette for each k. The Silhouette results shows that the best k to use are k=8 and k=10. We exploited dendogram with Ward's linkage because it can be used to initialise K-means. From the dendrogram we can see that the lowest possible interesting cut is k=8 so we go for this. It is also possible to notice that the algorithm identifies two main clusters. We plotted with respect to "stft_mean" and "sc_mean" the results of the Kmeans algorithm using k=8. We obtained an SSE=90.4176 and a Silhouette=0.24196. Almost all the clusters contain a majority of female observations or male observations. We also notice that K-means provides a really good division between speech and song clusters. It is interesting to see that in almost all clusters with the majority of female observations (cluster 0, 2, 4 and 5) we have also a majority of speech, while if we have a majority of male observations we have also a majority of song observations[1]. On the other hand, K.means doesn't provide a good distinction between emotions, since the clusters are heterogeneous with respect to this variable.

[1]Vocal_val=0 is for song observation while vocal_val=1 is for speech observations.

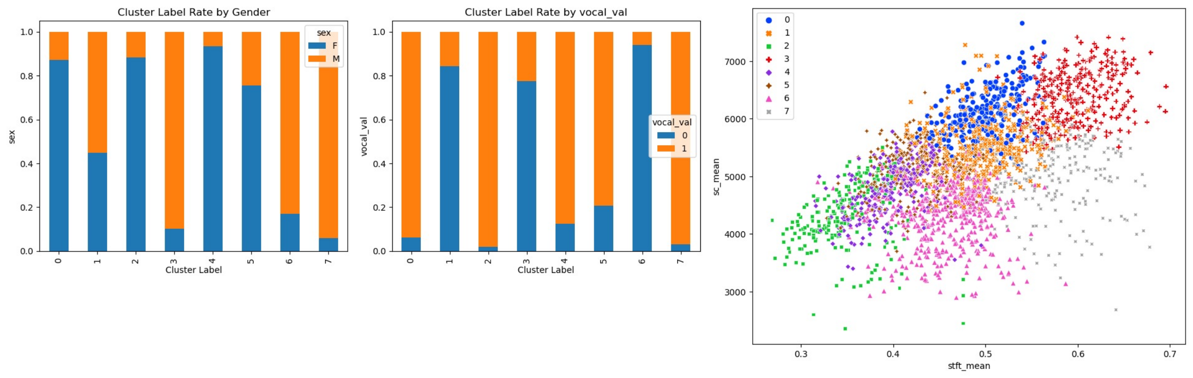


Figure 2.2: Clustering subset K-means clustering results

for the **song subset** we performed K-means using the same way as the one above to choose the best k. Even for this subset we came to the conclusion that the best k is 8. Exploiting K-means using the euclidean distance, the results show again a pattern where each cluster is composed primarily by female or male observations and song or speech observation. This time we have a Silhouette=0.2434 and an SSE=30.0417.

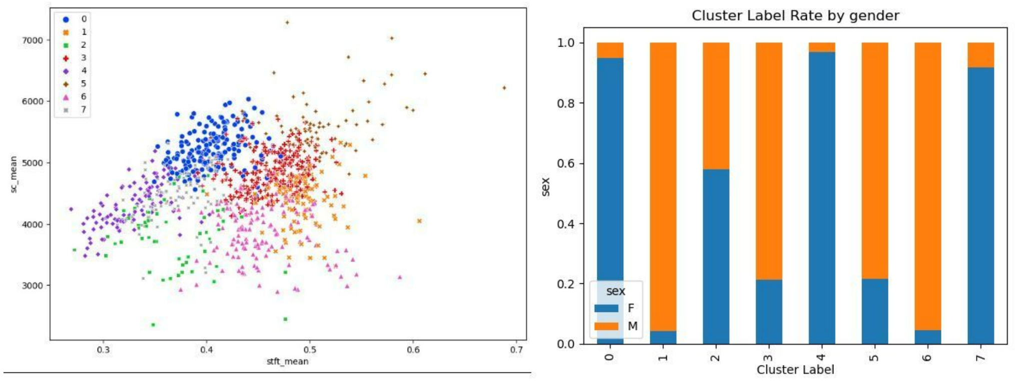


Figure 2.3: K-means for song subset

2.1.1 Bisecting K-means algorithm

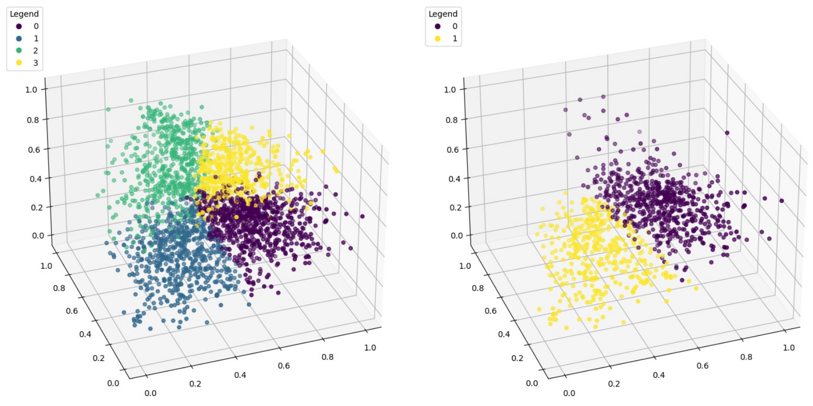


Figure 2.4: 3D Bisecting K-means results for clustering subset (on the left) and song subset

We initially performed the Bisecting K-means on the **clustering subset** using k=8 (to compare the different algorithms), the euclidean distance and using as bisecting strategy the "biggest_inertia" (it gave us better results than largest_cluster in terms of silhouette score).

The results shows again that some clusters contain almost all female/male observations or speech/song observations. For this algorithm we have a Silhouette=0.2001 and an SSE 96.5678. To get better results we used the information we get from the dendogram 2.5 (we can get even 4, 3 or 2 clusters) and we tried to merge the closest clusters. The best result comes when we merge each pair of 2 closest clusters, in fact we get an SSE=133.9476 and a Silhouette=0.2481. In this way we get a good separation of male, female, song and speech observations, as it results from the bar plots. For the **song subset** we exploited the same method we used above. This time the better result comes when we merge the closest cluster until we end up with only 2 cluster (using the bar plots can be seen that these 2 clusters identify male and female). We get an SSE=65.9728 and a Silhouette=0.34054.

2.2 Dbscan algorithm

Since Dbscan is a clustering algorithm that struggles with clusters of similar density, it does not give interesting results for this kind of dataset, in fact, we have not found pairwise of variables that presents different density areas. We performed the Dbscan algorithm using the min_sample=$2D$ (preferred when there is a lot of noise) and min_sample=$D+1$ where D is the dimensionality. So we tried min_sample equal to 8 and 5 while for the eps parameter we choose the best one using the plot of the distance of the points from the k_{th} neighbour and the result is that the optimal eps is 0.17 for min_sample=8 and 0.13 for min_sample=5. We used the euclidean distance because it brings to the higher Silhouette among the other kind of distances. The results of this algorithm, plotting as usual "stft_mean" and "sc_mean", brings to one big cluster and some sparse noise points. We used the same procedure with **song subset** and the results does not change, as expected.

2.3 Hierarchical algorithm

We performed the Hierarchical clustering algorithm on the **clustering subset** trying different kinds of distances and linkages. The best agglomerative clustering algorithm is the one that computes the euclidean distance and use the Ward linkage instead of the average or min linkages because it brings to the best silhouette among the other different possible results. From the dendogram (figure 2.1) we can see that the lowest cut we can do produces 8 clusters while with the higher cut we can produce 2 clusters. The Hierarchical algorithm with 2 clusters gives us a Silhouette of 0.3115 while the Hierarchical algorithm with 8 clusters gives a lower Silhouette of 0.1988. Nevertheless, results with K=8 on the left are plotted with mapped colours: we used the same colour as K-means for similar clusters to compare the results (since they show a similar composition).

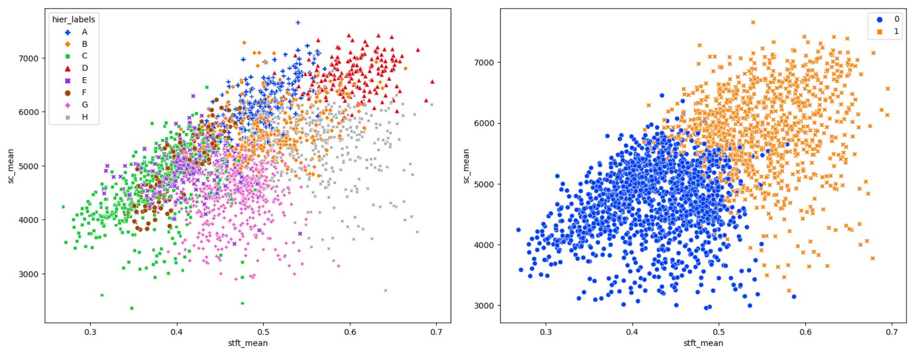


Figure 2.5: Agglomerative algorithm result of the clustering subset with 8 clusters and 2 clusters

For the **song subset** we used the same procedures described above and we get almost the same results. In this subset we have a slightly better Silhouette for the algorithm with 8 clusters in fact it is improved to 0.2101 in respect to the reduced subset, while the algorithm with 2 cluster get worse performance than the one of the reduced subset, in fact we have a silhouette 0.2960 but now we get a better visual separation of the 2 clusters. In the figure below can be observed that now the algorithm (left plot) tends to make a good separation of male observation and female observation (right plot). We come to a similar separation even with the bisecting algorithm when we merge closest clusters till we end up to have just 2 clusters (see figure 2.4)

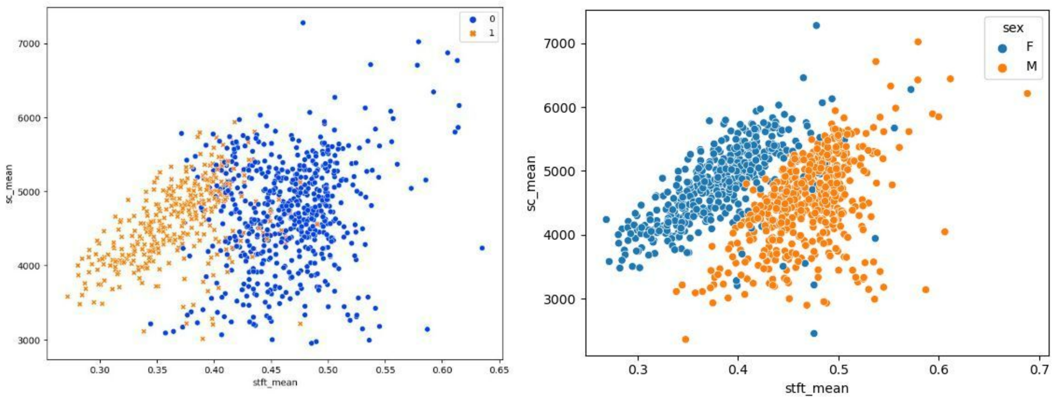


Figure 2.6: Agglomerative algorithm of song subset with 2 clusters compared to M/F division

2.4 Conclusion

After the clustering analysis we can say that the Dbscan on this kind of dataset and in particular with this set of features we chose seems to work very bad both on the clustering subset and on the song subset.
For the **clustering subset** we can conclude that centroid-based methods can provide a good distinction between male/female and song/speech. The algorithm that produces the higher Silhouette is the Agglomerative one using a number of cluster equal to 2, but from the plot 2.5 we can see that the algorithm leads to a bad separation of the clusters.
For the **song subset** it can be observed that both the Agglomerative algorithm and the Bisecting K-means algorithm for these features give good results (as we expected since the features we have chosen are used a lot for music information retrieval), but the Bisecting leads to a better Silhouette so it is the best algorithm for the song subset to use.
Below is presented a table that contains the resume of the results of the clustering analysis[2].

Table 2.1: Results of clustering analysis

Measures	Clustering subset		Song subset	
	Silhouette	SSE	Silhouette	SSE
K-means	0.2419	90.4176	0.2434	30.0417
Bisecting K-means	0.2481	133.9476	0.34054	65.9728
Hierarchical	0.3115		0.2960	

[2]Note that Dbscan results are omitted because it leads to one big cluster so it is poorly informative

Chapter 3: Classification and regression

For the classification we have utilized 2 different dependent variables: "emotion" and "vocal_val". We have used different algorithms, in particular: the Decision Tree algorithm, the KNN and the Naive Bayes classifier, in the end, we have discussed the results using the reports.

3.1 Emotion classification

To classify emotion we first used the label encoder function of Scikit learn on the emotion variable and in this way our dependent variable assumes value 0 if the emotion is "angry", value 1 for "calm", 2 for "disgust", 3 for "fearful", 4 for "happy", 5 for "neutral", 6 for "sad", 7 for "surprised". After this, we transformed all the categorical variables into dummies. In the end, we have split our dataset into train set and test set with the stratified method.

3.1.1 Decision Tree classifier

To get the best possible results from the Decision Tree algorithm and to avoid overfitting we have decided to tune the hyperparameters. So to get the best hyperparameter tuning we have first of all, used the cross-validation technique to plot the changes of accuracy for different values of our hyperparameters then we used this information to construct our Randomized cross-validation search. It turns out that the best hyperparameter configuration for the Decision Tree is the following: ccp_alpha = 0.002, criterion = "gini", max_depth = 21, min_impurity_decrease = 0.0024, min_sample_leaf = 0.0019, min_sample_split = 0.003 and in this way we got around the 40% of accuracy on the train set and 37% on the test set. We have not reached an higher accuracy but in this way we are not overfitting since the difference between the accuracy of the train set and the test set is close. Since the resulting decision tree shows a lot of leaves we have decided to omit it.

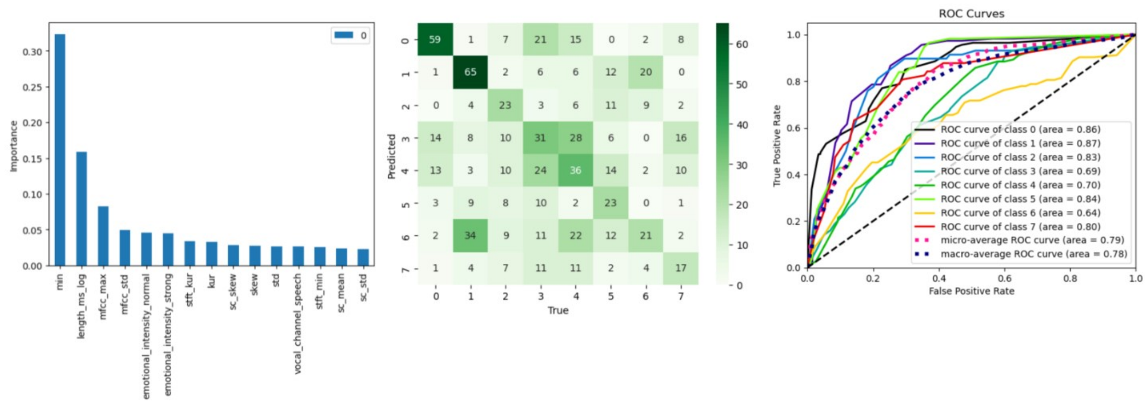


Figure 3.1: Importance of the features, confusion matrix and ROC curve

From the first figure can be seen the main features responsible for the splits, then from the confusion matrix and the ROC curve can be seen that the emotion where we have the best results in terms of true positive are the first 2 (calm and angry), but a classification error that happens often is classifying an observation that is "sad" when in reality it is "calm" (34 errors). Then, from the ROC curve it is clear that for values of false positive rate less than 0.1 the best performance of the decision tree is on angry class but for values between 0.1 and 0.2 the best one is calm.

3.1.2 Knn classifier

First, we have standardized test and train separately, then through the gridsearch and the cross-validation we have obtained the best hyperparameters: metric = "cityblock", n_neighbors = 96, weights = "distance". With these settings, we have achieved 37% accuracy on the validation test and 36% on the test set. However, from the ROC, curve we obtained little improvements in the area below the classes for almost every class against the decision tree ROC curve. The most interesting thing to observe is that the best performances are obtained from angry and calm. In particular, from values of false positive rate less than 0.4 the best one is angry, while for values greater than 0.4 the best one is calm.

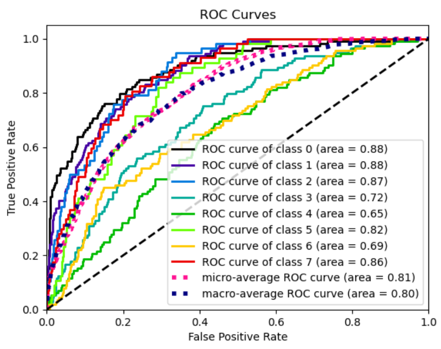


Figure 3.2: ROC curve for emotions with Knn

3.1.3 Naive Bayes classifier

We have applied the Naive Bayes classifier first on the non-categorical variables and then on the categorical ones and it turns out that this algorithm had a better performance on the non-categorical variables. However with the Naive Bayes classifier, we got the worst results compared to the other two algorithms, in fact, the average accuracy is 32% and even from the ROC curve can be seen that for almost each level of false positive rate others algorithm had a better performance. This is an expected results since the Naive Bayes classifier assumes conditionally independence among input variables and this is not true in reality, in fact we could have strong correlation among them.

3.1.4 Conclusion

Below we have the reports of the three algorithms used to classify emotions.

Table 3.1: Emotion classification reports

	Decision Tree			Knn			Naive Bayes		
Class	Precision	Recall	F-score	Precision	Recall	F-score	Precision	Recall	F-score
0	0.63	0.52	0.57	0.58	0.58	0.58	0.70	0.58	0.63
1	0.51	0.58	0.54	0.44	0.54	0.49	0.66	0.22	0.33
2	0.30	0.40	0.34	0.48	0.28	0.35	0.28	0.34	0.31
3	0.26	0.27	0.27	0.29	0.14	0.19	0.41	0.15	0.22
4	0.29	0.32	0.30	0.25	0.37	0.29	0.25	0.24	0.25
5	0.29	0.41	0.34	0.26	0.21	0.24	0.17	0.84	0.29
6	0.36	0.19	0.25	0.31	0.24	0.27	0.42	0.10	0.16
7	0.30	0.30	0.30	0.30	0.53	0.38	0.30	0.46	0.36
accuracy			0.37			0.36			0.32
macro avg	0.37	0.37	0.36	0.37	0.36	0.35	0.40	0.37	0.32
weighted avg	0.38	0.37	0.37	0.37	0.36	0.35	0.43	0.32	0.32

From the reports we can see that the Decision Tree and Knn have generally similar evaluation measures results, the first one has slightly better results. The Naive Bayes on the other hand has lower average accuracy and weighted average recall than the other two models, but has higher weighted average precision, this means that in this case the Naive Bayes classifier produces less false positive in particular for class 0.
In general, we can say that with this kind of dataset we get the best result by predicting the angry emotion and this is true for all the 3 algorithms.

3.2 Vocal_val classification

We have used the same procedure followed for the classification of the emotions. So, we used the vocal_val variable as target variable, where we have 1 if the audio signal refers to a speech and 0 if it is a song, then we have transformed all the categorical variables in dummies and after this, we have partitioned the data set into train set (70% of the dataset) and test set (30% of the dataset) using the stratified sample method to keep the same proportion of speech and song audio signals both in the train and in the test set (in particular we have more or less the 40% of song and the 60% speech).

3.2.1 Decision Tree classifier

We have used the cross validation with 7 different folds to test single values of the hyperparameter due to get an idea of the best distribution of values that we could use in the Randomized cross-validation method. In the end, we got our best hyperparameters which lead us to a 90% of mean accuracy on the train set and an accuracy of the 89% on the test set.

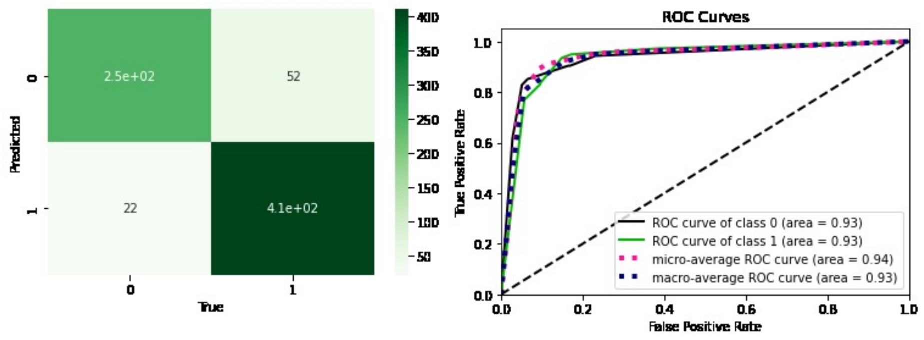


Figure 3.3: Confusion matrix and ROC curve for song and speech

From the confusion matrix can be seen that there are more than twice as many false negatives as false positives. Then with the ROC curve can be seen that the model is a good predictor for both song and speech at every level of false positive rate despite the model tends to perform slightly better for speech for false positive rate greater than 0.2 more or less.

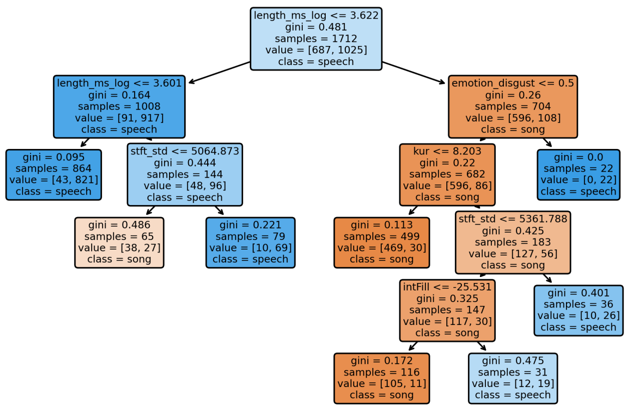


Figure 3.4: Decision tree for song and speech

Since we got 6 as the best value for the max_depth hyperparameter the resulting Decision Tree plot is easy to interpret. In fact, for example, we could easily say that if the value of "length_ms" is less than 3.62 and less than 3.6 our observation refers to a speech otherwise if it is less than 3.62 but greater than 3.6 we have to look at the value of "stft_std", if it is less or equal to 5064 we have a song observation otherwise a speech observation. An expected result is the split via "emotion_disgust", if this is greater than 0.5 (thus equal to 1, given the nature of the variable) the record is classified as speech, because as seen in Figure(1.2) song observations do not have the emotion disgust.

3.2.2 Knn classifier

As before we have standardized test and train separately, then through the gridsearch and the cross-validation we have obtained the best hyperparameters: metric = "cityblock", n_neighbors = 25, weights = "distance". With these settings, we have achieved 89% accuracy on the validation test and 90% on the test set, basically the same as the Decision Tree. In the figure below we can see how through the different combinations of parameters the "cityblock" metric has an accuracy about 2% higher than that obtained through the Euclidean distance.

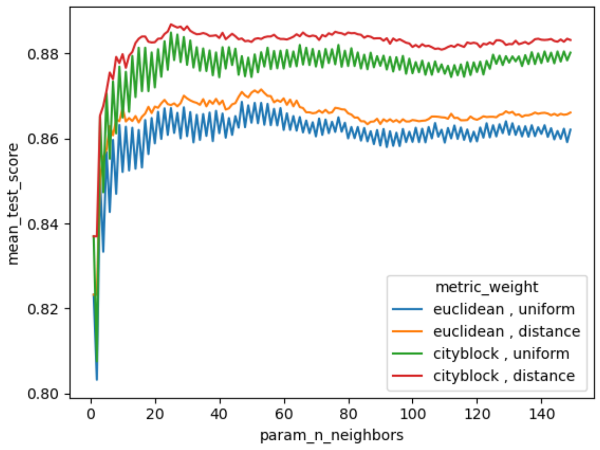


Figure 3.5: Knn parameter combination

3.2.3 Naive Bayes classifier

Even in the song and speech classification we have obtained that this algorithm has better performance with non-categorical attributes, in fact by applying the algorithm to the categorical attributes we can see from the ROC curve that we got almost the same results that we could obtain with a random prediction. Compared with the other 2 algorithm we still got the worst results using the Naive Bayes classifier, in fact the average accuracy is 84%.

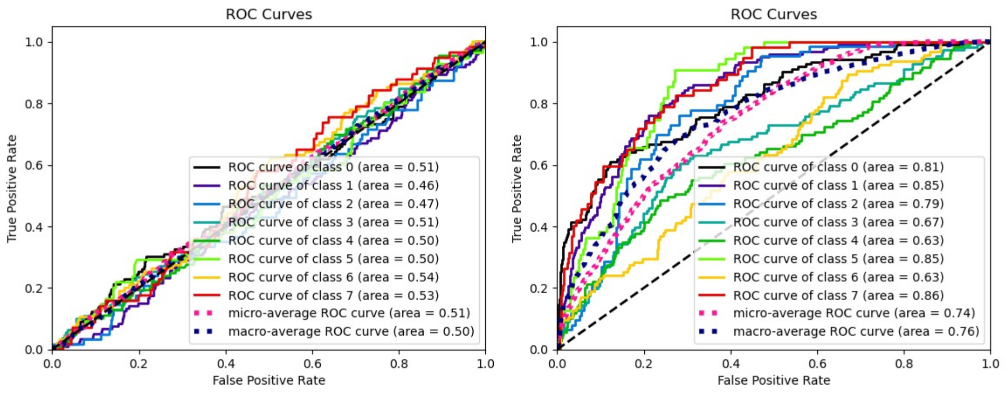


Figure 3.6: Categorical Naive ROC curve vs numerical naive ROC curve

3.2.4 Conclusion

Below we have the reports of the three algorithms used to classify vocal_val.

Table 3.2: Vocal_val classification reports

	Decision Tree			Knn			Naive Bayes		
Class	Precision	Recall	F-score	Precision	Recall	F-score	Precision	Recall	F-score
0	0.91	0.83	0.87	0.83	0.93	0.88	0.73	0.96	0.83
1	0.89	0.94	0.91	0.95	0.87	0.91	0.96	0.76	0.85
accuracy			0.90			0.90			0.84
macro avg	0.90	0.89	0.89	0.89	0.90	0.89	0.85	0.86	0.84
weighted avg	0.90	0.90	0.90	0.90	0.90	0.90	0.87	0.84	0.84

From the reports, we can see that the Naive Bayes has the worst average results, while the Decision Tree and Knn have more or less the same average evaluation measures results, in particular, the DT has higher precision for the song and a better recall for the speech observations, while the Knn has the opposite findings. About the F-score, the Knn is better on song classification, while for speech Knn and DT are equivalent.

3.3 Regression

In this section we have analysed relationships between different variables using both univariate regression and multivariate regression. We used linear regression to asses how "min" variable is affected by "intfill" variable. For the multivariate regression we have chosen as third variable "zcs_log". As usual we have split the dataset in train set, that holds the 70% of observation, and test set, that holds the 30% of observation.

3.3.1 Univariate regression

First of all we have applied the linear regression using the least square method to find the intercept and coefficient that minimize the SSE. We got as result a coefficient of -0.017 for "Intfill" and an intercept of -0.800 both this results are statistically significant since the resulting p-value of the coefficient test is less than the usual significance levels. The coefficient of "intFill" can be interpreted as follow: an unitary increase of the intFill value leads to a decrease of 0.017 on the "min" value on average. We got an R^2 of 0.677 which means that the 67.7% of the "min" variance is explained by "Intfill". Using a regularization term we can see that the Lasso regularization brings to worse results and this can be explained by the tendency of Lasso regularization to penalize coefficient close to 0 while we cannot see changes by applying Ridge regularization and this occurs because we do not have problem of multicollinearity since we are using a single independent variable.

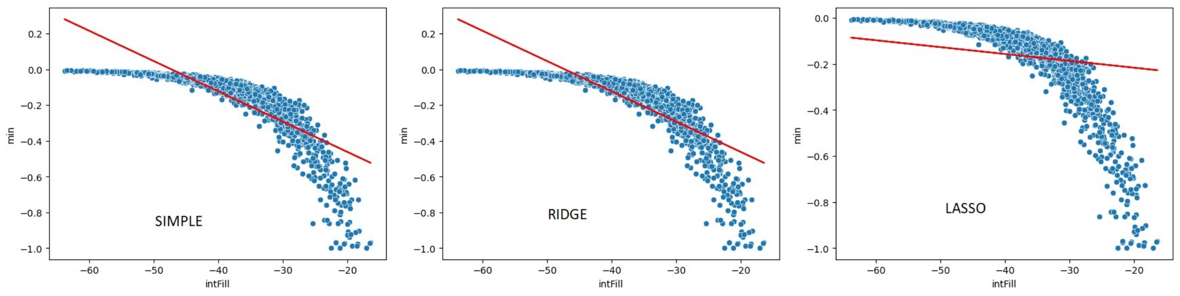


Figure 3.7: "min" linear regression

We have then tried to add "mfcc_std" as independent variable. In this way now we have a multiple linear regression that produce the results showed in table (4.1) presented below. From the result we can see that now "intFill" have less effect on "min" since now a unitary increase leads to a decrease of 0.0067 of the "min" value when all other variables are constant. This new model gives us a poor improvement of the R^2.

Table 3.3: Coefficients test results for simple linear model and multiple regression model

	Dependent variable:	
	min	min
	(simple)	(multiple)
mfcc_std		0.0043***
intFill	−0.0169***	−0.0067***
intercept	−0.8005***	−1.0068***
R^2	0.66	0.67
Note:	*p<0.1; **p<0.05; ***p<0.01	

From the plots can be seen that we need a non-linear model in order to have a proper fit of the data. Thus, we have used the Decision tree regressor and due to avoid overfitting we have explored how the R^2 changes while we make "ccp_alpha" and "min_samples_leaf" assume different values (we have made this using cross-validation). We take the value of this hyperparameters that maximize our R^2, then we applied this model on the test set.
From the figure (3.3.1) below, we can see that our model makes predictions that have the same value for given ranges of "intFill" values. In particular we can observe that it tends to have the best performance for value of "intFill" less than -45 more or less, while for greater value tends to make prediction that can differ among themselves even if we have a little change of our independent variable, for example for values of "intFill" greater than -20 it will predict a minimum of the audio signals equal to -0.9 more or less but if our value decreases a little it will predict a "min" value of -0.7. However, we got a clearly better model than the linear one, in fact now we got an $R^2 = 0.90$, an $MSE = 0.003$ and a $MAE = 0.034$.

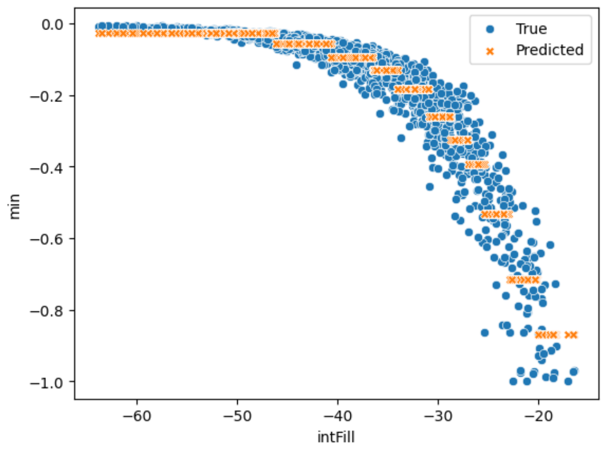


Figure 3.8: "min" non-linear regression with decision tree regressor

3.3.2 Multivariate regression

For the multivariate we have chosen as third variable "zcs_log". Our linear model gave us the following results when we tried to predict values of "min": now our constant is -0.774 and it is still statistically significant since we have a p-value less than the lowest traditional significance level[1]; but now we have a coefficient level of the "zcs_log" which is not statistically significant because now our p-value is 0.731 so we can widely accept the null hypothesis that tells us that the coefficient of "zcs_log" is statistically equal to 0; the coefficient of the "intFill" variable is still -0.0169 and has the same meaning of the univariate case because "zcs_log" is not different from 0 so it is the same situation of the univariate case where we have only one explanatory dependent variable, in fact, even the R^2 has not changed and it is equal to 0.66. Below are presented the plot where we keep first the "intFill" variable on the x-axis and then we keep the "zcs_log" variable on the y-axis.
In support of our analysis it can be observed that the plot that we got putting "intFill" on the x-axis and "min" on the y-axis produce the same regression line that we got in our univariate linear model.

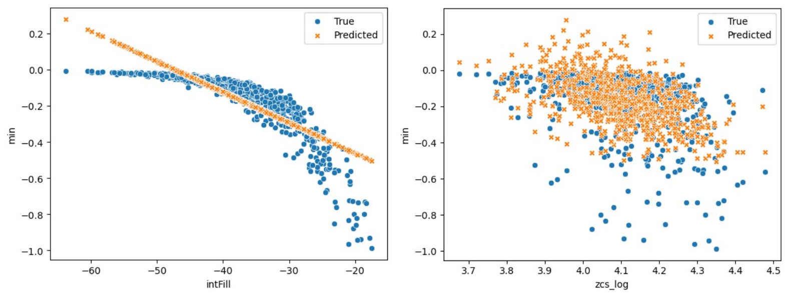


Figure 3.9: Multivariate regression visualization changing the x-axis

[1]The lowest traditional significance level is 0.01.

Chapter 4: Pattern mining

We have used all the attributes in the clean dataset from Chapter 1 to find interesting patterns. We exploited the qcut function to divide each non-categorical attribute range into 4 bins. Then, we did some minor fixes just for readability's sake.

4.1 Frequent patterns extraction

The first thing we want to check is how the number of frequent patterns varies w.r.t. the minimum support value (min_sup) and the minimum length (zmin).

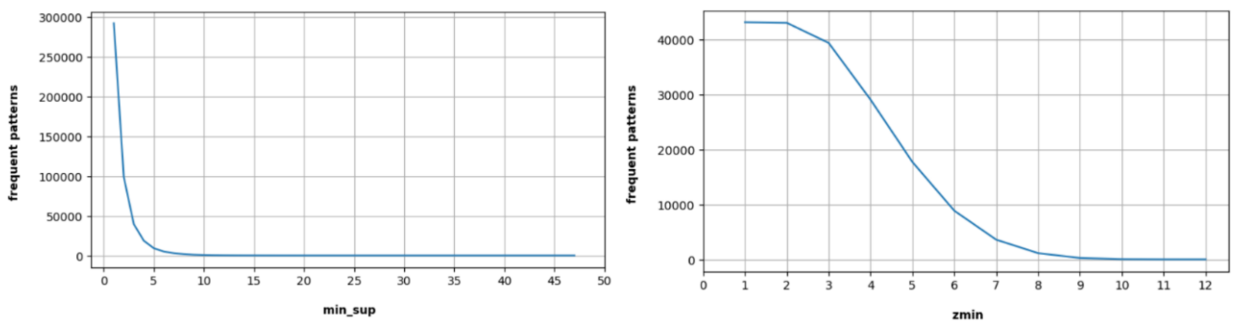
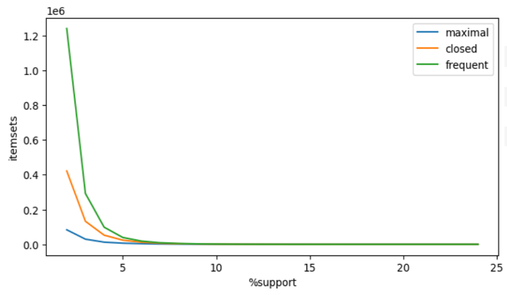
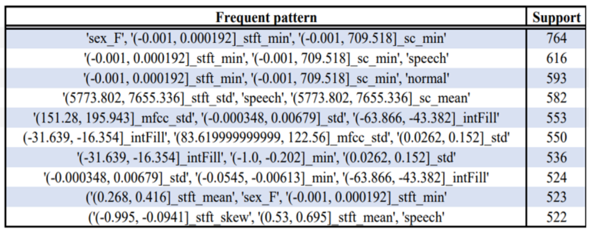


Figure 4.1: Frequent patterns with respect to min_sup and zmin

In the left-hand plot we can notice that there is a knee at 4 or 5. The number of patterns changes enormously decreasing "min_sup" in the range (2,4). With a minimum support equal to 22 there are just 6 patterns left. After some attempts, we chose 5 to prevent the algorithm from generating too many patterns, but preserving the interesting patterns involving emotions, as we will discuss later. Regarding the minimum number of items per set, we chose 3.





Frequent pattern	Support
'sex_F', '(-0.001, 0.000192]_stft_min', '(-0.001, 709.518]_sc_min'	764
'(-0.001, 0.000192]_stft_min', '(-0.001, 709.518]_sc_min', 'speech'	616
'(-0.001, 0.000192]_stft_min', '(-0.001, 709.518]_sc_min', 'normal'	593
'(5773.802, 7655.336]_stft_std', 'speech', '(5773.802, 7655.336]_sc_mean'	582
'(151.28, 195.943]_mfcc_std', '(-0.000348, 0.00679]_std', '(-63.866, -43.382]_intFill'	553
(-31.639, -16.354]_intFill', '(83.619999999999, 122.56]_mfcc_std', '(0.0262, 0.152]_std'	550
'(-31.639, -16.354]_intFill', '(-1.0, -0.202]_min', '(0.0262, 0.152]_std'	536
'(-0.000348, 0.00679]_std', '(-0.0545, -0.00613]_min', '(-63.866, -43.382]_intFill'	524
('(0.268, 0.416]_stft_mean', 'sex_F', '(-0.001, 0.000192]_stft_min'	523
('(-0.995, -0.0941]_stft_skew', '(0.53, 0.695]_stft_mean', 'speech'	522

Figure 4.2: Frequent, closed and maximal itemsets at different support thresholds, and some frequent patterns

With a low support threshold, the number of frequent itemsets is significantly higher than maximal and closed. These values tend to converge for min_sup values in the interval (15:20). It is possible to note that the most frequent patterns involves the same bins for stft_min and sc_min. We exploited bot Apriori algorithm and **FP-growth**. The best performances were achieved with FP-growth, which is significantly faster.

4.2 Association rules

In this section we discuss the most interesting rules obtained with the algorithms. First of all, we need to set a confidence threshold. We generated rules with different confidence thresholds, we found optimal results with a confidence of 70%. We used the same parameters as before, a minimum support of 5 and a minimum length of 3 elements.

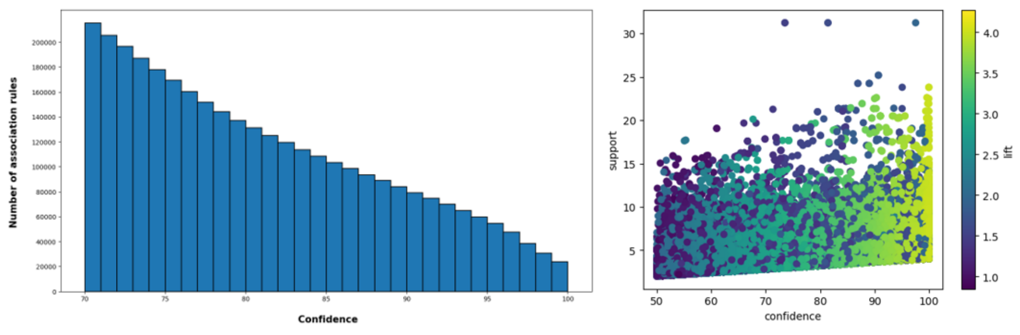


Figure 4.3: Rules w.r.t. confidence thresholds and a support-confidence-lift scatterplot

Figure 4.3 shows a detail of the trend of the number of rules from 70% to 100% of confidence. We noticed that the number of rules with a %_support higher than 25% is extremely low. They are probably the most interesting ones, so we want to filter results with a min_sup greater than 25.

Consequent	Antecedent	Support (%)	Confidence	Lift
(-0.001, 0.000192]_stft_min	sex_F, (-0.001, 709.518]_sc_min	31,234669	0.975734	1,951469
(-0.001, 709.518]_sc_min	(-0.001, 0.000192]_stft_min, speech	25,183974	0.907216	1,814433
(-0.001, 0.000192]_stft_min	(-0.001, 709.518]_sc_min, speech	25,183974	0.873759	1,747518
(-0.001, 709.518]_sc_min	sex_F, (-0.001, 0.000192]_stft_min	31,234669	0.814499	1,628998
sex_F	(-0.001, 709.518]_sc_min, (-0.001, 0.000192]_stft_min	31,234669	0.735322	1,497584

Figure 4.4: Most interesting extracted rules

It is interesting to notice that with low values of sc_min and stft_min we almost always obtain a female speaker (confidence is extremely high). Moreover, we can see that rules 1, 4 and 5 involve the same items with high levels of lift and confidence; it means that they are positively correlated. There are also two rules involving 'speech', with the same levels of sc_min and stft_min as before. This is interesting, we have already observed this speech/female pattern in figure (2.2).

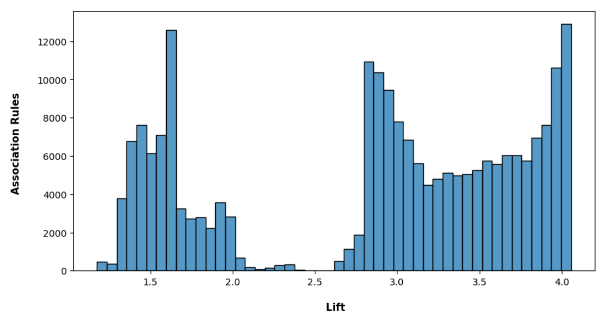


Figure 4.5: Generated rules number for each lift value

We can immediately notice two main peaks around lift 1,7 and lift 2,7. We can clearly see a surprising behaviour, there is also a peak around a lift value of 4.0. Results confirm what we have already seen with the scatterplot from figure (4.3): there is plenty of rules with an extremely high confidence and lift 4. Unfortunately, with such a confidence threshold there are no rules concerning emotions. We wanted to go deeper, as we can see from figure (4.6).

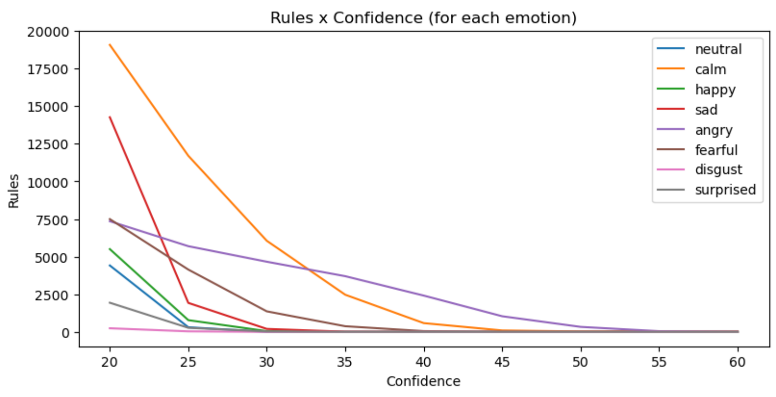
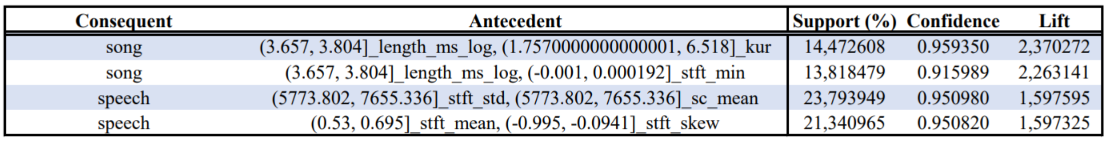


Figure 4.6: Rules number with emotions as consequence, with different confidence thresholds

With confidence values greater than 55, basically, there are no rules involving emotions as consequences. Some emotions, like ‘disgust’ or ‘surprised’, are less likely to be discovered by the algorithm; they pretty much disappear with a confidence equal to 30. ‘angry’ and ‘calm’ are the easiest to be discovered, they’re still present with a confidence level greater than 40. Sadness, on the other hand, is involved in a lot of rules at conf = 20, but it quickly decreases with a small change in the minimum confidence level. Results found with this plot seems to agree with what we had found with the confusion matrix for the decision tree in figure (3.1). ‘Angry’ and ‘calm’ were the easiest classes to be discovered, while ‘disgust’ and ‘neutral’ were the worst ones in terms of performances.

4.3 Target variable prediction

We wanted to exploit the extracted rules to predict vocal_channel. We used 2 rules with the highest lift and confidence to predict ’song’ and the same for ’speech’. If the algorithm could not find a class for the observation, it assumed it to be ’speech’ since it’s the most frequent.



Consequent	Antecedent	Support (%)	Confidence	Lift
song	(3.657, 3.804]_length_ms_log, (1.7570000000000001, 6.518]_kur	14,472608	0.959350	2,370272
song	(3.657, 3.804]_length_ms_log, (-0.001, 0.000192]_stft_min	13,818479	0.915989	2,263141
speech	(5773.802, 7655.336]_stft_std, (5773.802, 7655.336]_sc_mean	23,793949	0.950980	1,597595
speech	(0.53, 0.695]_stft_mean, (-0.995, -0.0941]_stft_skew	21,340965	0.950820	1,597325

Figure 4.7: Rules for target variable prediction

Then, we compared results with what we have found in Chapter 3:

Table 4.1: Vocal_channel confusion matrix

	Predicted class:	
	Song	Speech
Song	454	536
Speech	37	1419

F_measure (0.832) is quite good, but accuracy (0.766) is not as good as with decision tree classifier. We observe a really high recall of 0.975 just because if there’s no prediction, everything belongs to the ’speech’ class. In fact, we can observe a very low number of false negatives. In conclusion, since there are just four rules involved, we can say that this is not a bad model but neither a perfect one.