UNIVERSITÀ DI PISA

Department of Computer Science
Master in Data Science and Business Informatics

DM1 – Data Mining Foundations

Submitted to:

Prof. Dino Pedreschi
Prof. Riccardo Guidotti

Submitted by:

Nimra Nawaz
Hafiz Muhammad Umer

Academic Year 2022/23

CONTENT

Chapter 1

1. Data Understanding

The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) contains audio of 24 professional actors (12 female, 12 male), vocalizing two statements in a neutral North American accent. Speech includes calm, happy, sad, angry, fearful, surprise, and disgust expressions, and song contains calm, happy, sad, angry, and fearful emotions. Each expression is produced at two levels of emotional intensity (normal, strong), with an additional neutral expression.

1.1. Data Semantics

The RAVDESS dataset is composed of 2453 records (rows) for a total of 38 independent variables (columns). Of the Thirtyeight variables, 9 of them are categorical types, 6 are numeric variables Int64 type, while the remaining 23 are of Float64 type. The following table describe each variable with an example.

Table 1.1 Data understanding and variables introduction.

Name	Description	Type	Example
modality	represent type	categorical	audio-only
vocal_channel	represent vocal type	categorical	song, speech
emotion	vocal emotion	categorical	sad, happy, calm
emotional_intensity	intensity of emotion	categorical	normal, strong
statement	two different statements	categorical	“dogs are sitting by the door”
repetition	repetition of the statement	categorical	1st, 2nd
actor	even numbers female, odd number man	categorical	1, 2, 5, 12
sex	gender of the actor	categorical	M, F
channels	audio channel	categorical	1, 2
sample_width	number of bytes for sample	categorical	2, 4, 8
frame_rate	frequency of samples used (in hertz)	int64	48000
frame_width	number of bytes for each frame	int64	2
length_ms	audio file length (in milliseconds)	int64	4001, 6712

frame_count	number of frames from the sample	float64	174575 , 180981
intensity	loudness in dbfs (decibel relative to the full scale)	float64	-31.21450288, -49.09504189
zero_crossings_sum	the rate at which a signal shift from positive to zero to negative or from negative to zero to positive	int64	11617, 15293
'mean', 'std', 'min', 'max', 'kur', 'skew'	statistics of the original audio signal	float64	-755.22, 171.929
mfcc_ 'mean', 'std', 'min', 'max'	statistics of the mel-frequency cepstral coefficients	float64	3328.43, 423.133
sc_ 'mean', 'std', 'min', 'max', 'kur', 'skew'	statistics of the spectral centroid	float64	7262.183, 293.722
stft_ 'mean', 'std', 'min', 'max', 'kur', 'skew'	statistics of the stft chromagram	float64	-372.452, 293.123

1.2. Distribution of the variables and statistics

In this section we will understand the variables more in depth using some count plots, histograms, and scatter plots. The first plot (Figure 1.1) represents the number of each emotion that has been represented by the female or male actor. From the plot we can inferred that both female and male actors have almost same amount of emotions representations but in some cases male actors have represented slightly more emotions than female actors. Furthermore, we can also find that the emotions fearful, angry, happy, calm, and sad have the same number of emotions whereas the emotions surprised, neutral, and disgust are equal in numbers. In second plot (Figure 1.2) we have compared two important variables intensity and length_ms based on emotions heatmap {'angry': 0, 'calm': 1, 'disgust': 2, 'fearful': 3, 'happy': 4, 'neutral': 5, 'sad': 6, 'surprised': 7}. It represents almost all of the audio's length lay between 3000 milliseconds to 5000 milliseconds. As comparing to the intensity, we can see that the intensity does not depend on length of the vocal, we can clearly see it is

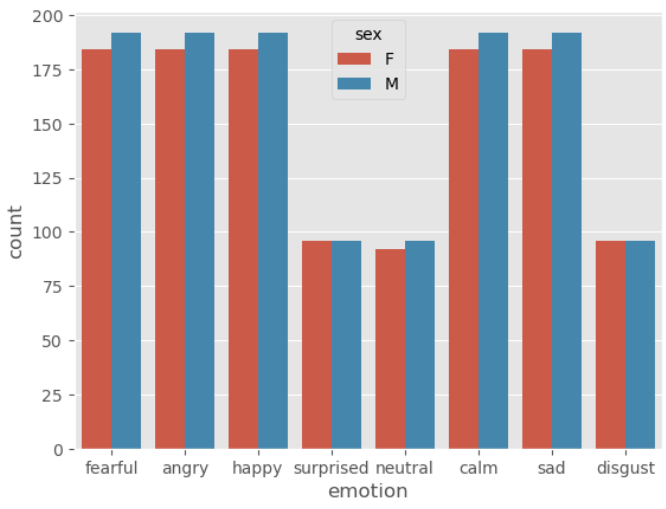


Figure 1.1 Emotions count based on sex.

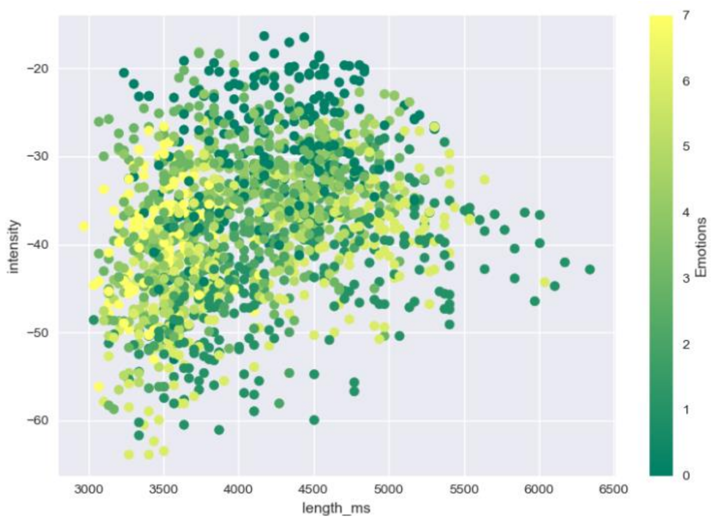


Figure 1.2 Intensity vs length_ms based on emotions.

scattered throughout the length of the vocals. The graph also represents that the emotion 0 (angry) is the most scattered emotion throughout the graph whereas the emotion 7 (surprised) is mainly lay between 3000 to 4000 length of vocals and -50 to -30 intensity level. In the third figure (Figure 1.3), we have different basic counting plots such as male and female actor count plot represents there are almost same number of female and male actors which are about 1200. The plot depicts the counts for each actor. It represents that the actor 24 has the most audios in the given dataset while other actor has different number of audios. The next graph represents the emotional intensity counts which shows that there is a slight fluctuation between both types but they both are almost equal in the dataset. The last graph represents the number of each emotion has in the dataset and we can see the emotions fearful, angry, happy, calm, and sad have the same number of rows in the dataset and the other emotions have the same number of rows.

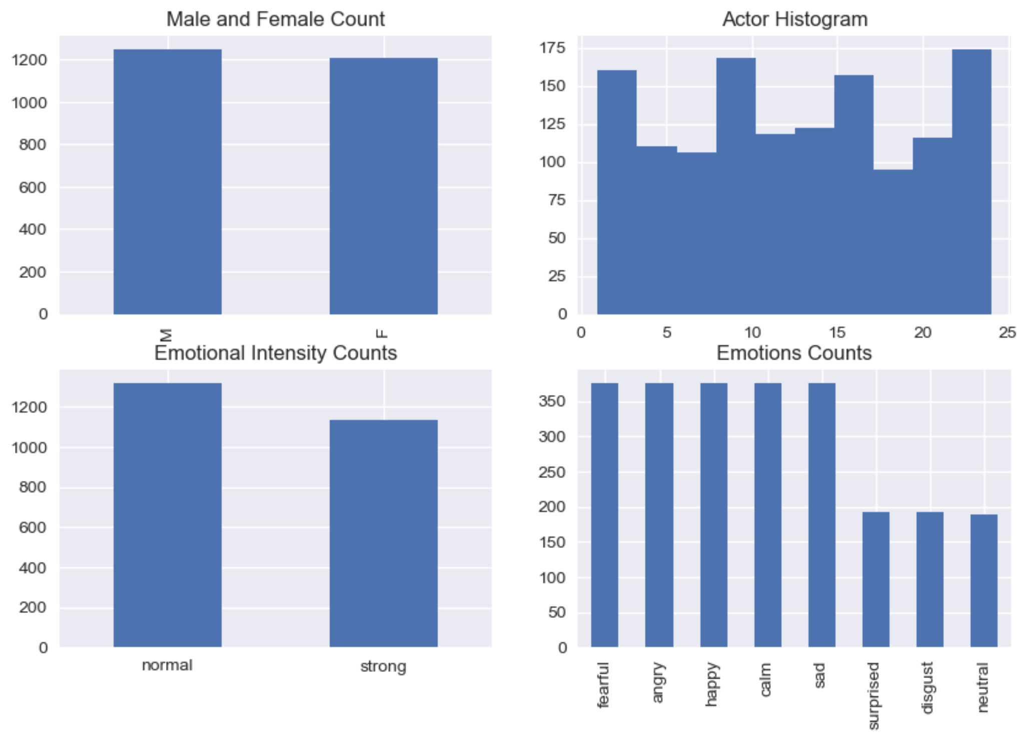


Figure 1.3 Basic counting plots.

1.3. Assessing Data Quality

It is crucial to have "quality" data for a successful data analysis. The search for the null values of the various characteristics and outliers, an outlier is a value that appears to be abnormal for a specific feature, is a step that must be taken in order to prepare the data for processing.

1.3.1. Handling Missing Values

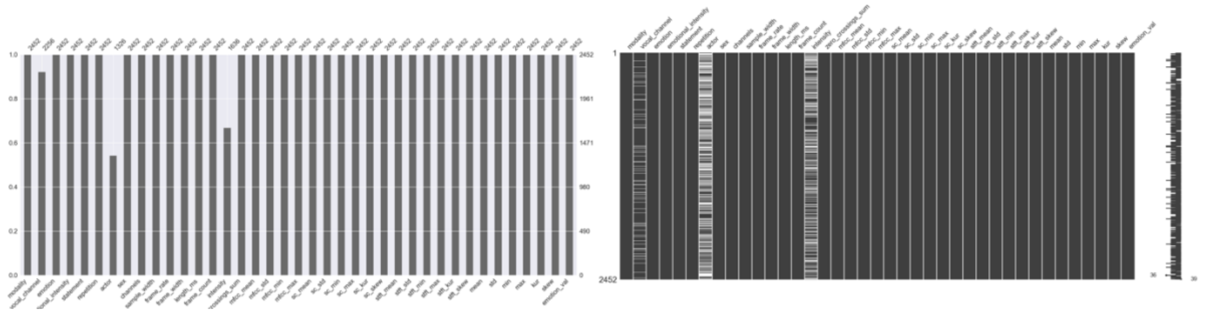


Figure 1.4 Missing values bar graph and matrix.

The above bar graph and matrix (Figure 1.4) shows missing values for some attributes in the dataset and we can clearly see that actor has the most missing values whereas vocal_channel has few missing values but intensity has many missing values.

To fill missing values, we have used different approach for each type of variable. For example, for vocal_channel has a ratio of null values around 10% so we have used the average value based on length_ms for instance the average length_ms for speech is 3700.42 and for song is 4650.36 so we decided for length_ms less than 4100 assign vocal_channel as speech otherwise song. The feature actor has around 64% of null value so to fill actor's missing values we used a similar approach based on the sex variable if the sex value is 'F' we chosen a random even value from 2 to 24 and if the sex value is 'M' we chosen a random odd value from 1 to 23. For intensity we have used interpolate method to fill null values. Interpolate fill missing values based on the estimation on the range of a discrete set of known data points. Figure 1.5 shows all the attributes are now filled with data and there are no missing values anymore in the dataset. The vertical left axis of the graph represents the percentage value of an attribute that has filled values and the right vertical axis shows the total number of rows in the dataset. The bottom horizontal axis of the graph displays the name of each attribute and the top horizontal axis of the graph shows total number of rows in an attribute.

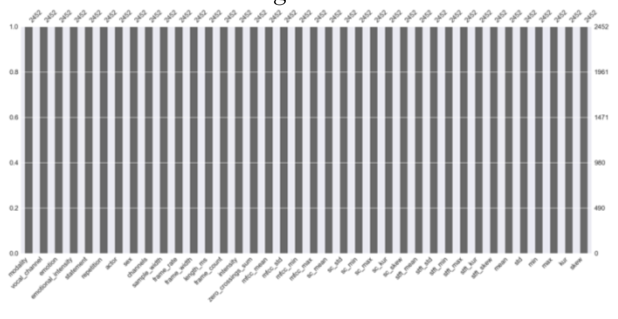


Figure 1.5 Graph filled values

1.3.2. Handling Outliers

By describing our dataset, we have found that there are some outliers existed in frame_count variable because the minimum value in frame_count is -1 which is not possible as frame_count represents the number of frames from the sample and each sample would have at least one frame or even if an audio is muted there is zero frame but negative frames are not possible. The given graph (Figure 1.6) is a boxplot implementation of the frame_count variable which clearly depicts that there are some outliers exists in the dataset and upon counting outliers by using count function on the variable we found that there are only 35 rows where frame_count is -1. To remove outliers, we used a simple method to drop out the 35 rows which has frame_count equal to -1. Our dataset has significant rows and only 35 rows could not affect our results that is why we removed such rows to clear our dataset from outliers. Figure 1.7 represents frame_count variable after removing outliers or simply deleting the rows. Now can inferred that the frame_count values range from 160000 to 220000 which in fact make sense with the length_ms variable.

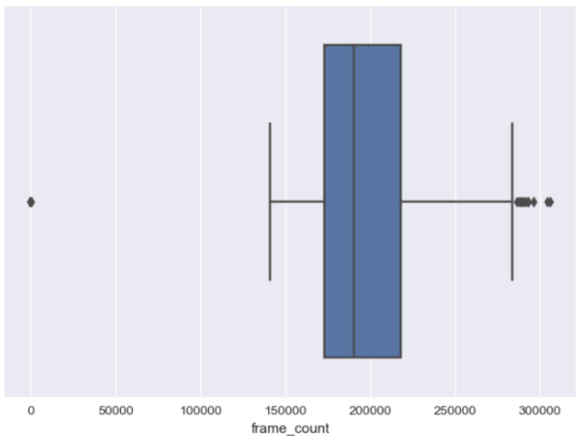


Figure 1.6 frame_count boxplot before

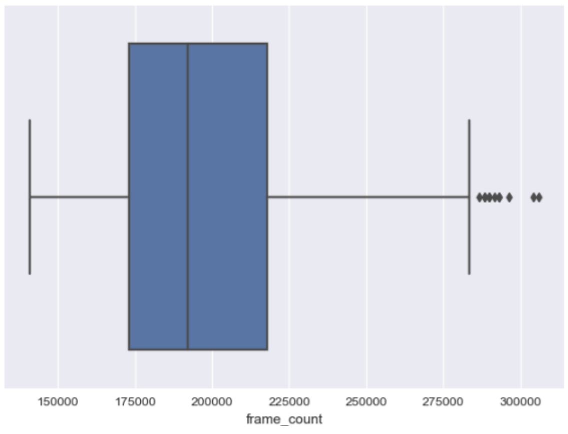


Figure 1.7 frame_count boxplot after

1.4. Variable Transformations

Features transformations or variable transformation is used to scale down long undistributed values under a similar range of distributed values. For example, scaling a continuous variable in the range of 0 and 1. There are different type of transformations such as log transformation, mean normalization, min-max scaling, square root transformation, etc. In our dataset, there are some discrete variables which needs label encoding such as vocal_channel, emotion, statement, etc. For example, we have transformed emotion as {'angry': 0, 'calm': 1, 'disgust': 2, 'fearful': 3, 'happy': 4, 'neutral': 5, 'sad': 6, 'surprised': 7}.

1.5. Pairwise Correlations and Eventual Elimination of Variables

These days, databases frequently contain hundreds or even thousands of features. This could initially appear to be a beneficial thing because more characteristics provide more details about each sample. But the majority of the time, these extra features do not bring any value and add pointless complication. Pairwise correlation is a powerful technique that enables us to identify the relationship between features and choose whether they are valuable enough to keep. There are some variables which do not provide any useful information and just bring complexity for our model because they have only one value or the one value repeats most of the time so we decided to drop those features before implementing a pairwise correlations. Those features are 'modality', 'sample width', 'frame_rate', 'stft_max', 'frame width' and 'channels'.

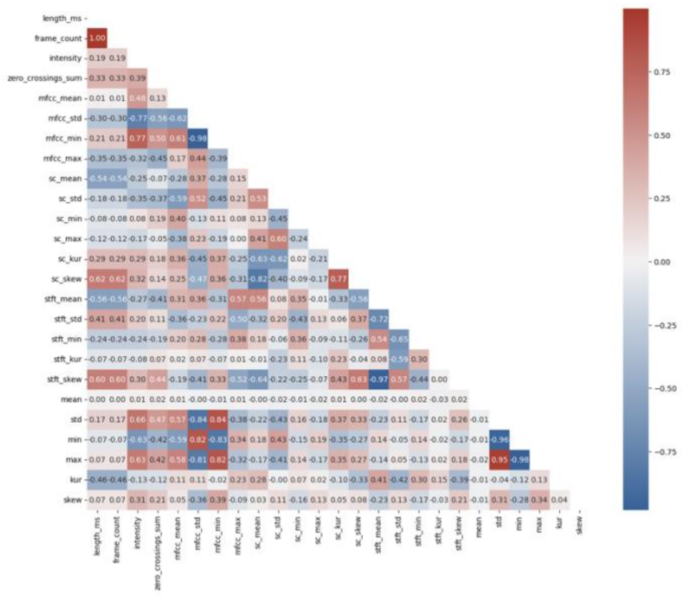


Figure 1.8 Correlation heatmap.

The above heatmap (Figure 1.8) displays the correlation between each variable by distracting the extra information because the correlation between A and B is the same as the correlation between B and A. Also, the diagonal consists the correlation of features with themselves which is always 1. So, what does this correlation represents? Let's imagine there is a 0.85 correlation coefficient between two features. This indicates that you can accurately predict feature 2 85% of the time by utilizing the values of feature 1. In other words, feature 2 won't add much new information if feature 1 already exists in your dataset. Due to the fact that feature 2 merely increases model training complexity, it is useless to keep it. Next, we set threshold of 0.80 for both positive and negative to drop out the features. Keeping in mind the threshold, we decided to drop the following features 'frame_count', 'mfcc_min', 'mfcc_std', 'sc_skew', 'std', 'min', 'max', 'stft_skew'.

Chapter 2

2. Clustering

In the following section, we are going to apply different clustering algorithm on our dataset such as k-means, density base (DBScan) and hierarchical clustering.

2.1. Analysis by Centroid-based Method – K-means

In the following section we have applied K-means clustering on all continuous variables and selected variables. Firstly, we have decided to get an optimal value for k.

2.1.1. Optimal K value

The selection of this value is crucial because the outcomes can be significantly altered depending on the value of the specified K. The trend of the SSE is taken into account while determining the k value. The decision is determined by looking at the graph's "elbow of the curve." Based on the silhouette value we decided to use 3 as the k value initially for all continuous features.

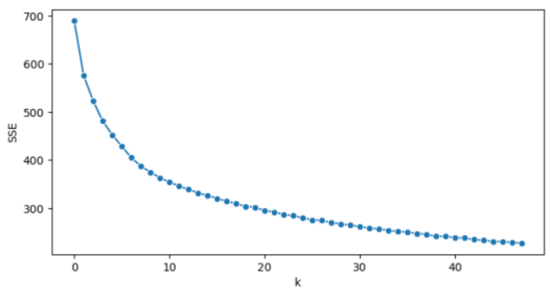


Figure 2.1 Sum of Square Error (SSE)

2.1.2. K-Means with all Continuous Features

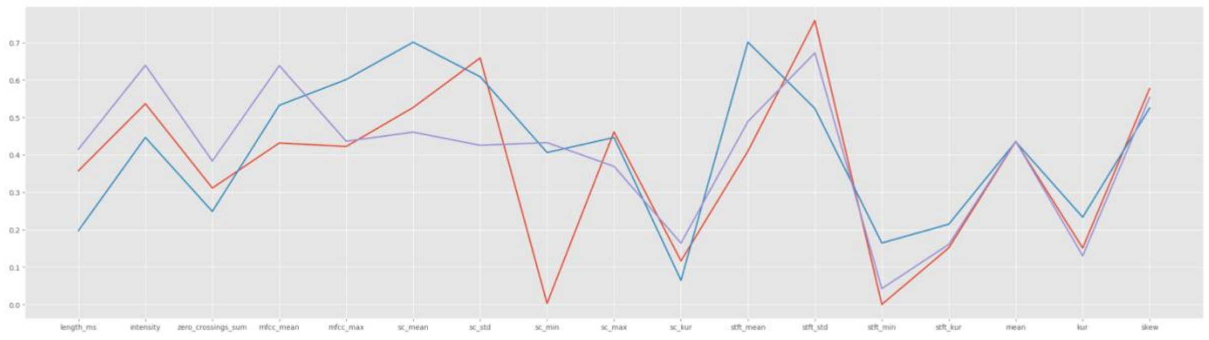


Figure 2.2 K-means with k=3 with continuous features

The results we received with k=3 are not satisfying. Silhouette was also quite low and the SSE was high. Additionally, K was not the optimal value. First, we attempted to obtain the ideal K. The elbow approach reveals that the ideal value of k is between 5 and 10. We attempted to train a model with k=6 in the following phase. The outcomes, though, remained poor. We noticed that the measurements are really high.

2.1.3. K-Means with Selected Features

In this stage, we used a heuristic approach to try and minimize the dimensions. We just took into account the three variables "zero crossing sum," "length_ms," and "intensity." We attempted to create a model for this one using k=6. The outcomes were adequate but not very optimistic. As a result, we attempted to test the ideal k using the elbow approach. The findings indicate that k is best at 3. We trained the three variables and k=3 for our final model. The clustering derived from the three previously mentioned features is depicted in the graph (Figure 2.3). The figure demonstrates that there are three distinct clusters. This appears to be a really elegant solution for a three-dimensional data set. The elbow method was used to select k=3, and the graphs support this decision. The SSE rate was 94.556, and the Silouhette score we obtained is very acceptable at 0.301.

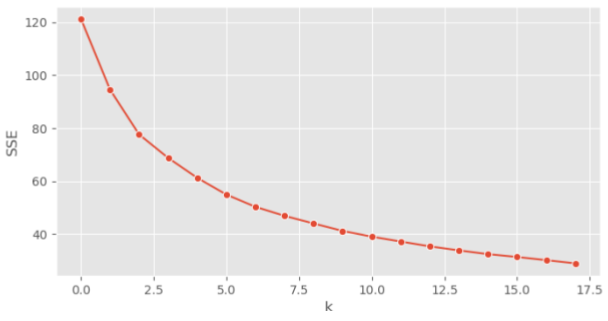


Figure 2.3 Sum of Square Error (SSE) for selected features

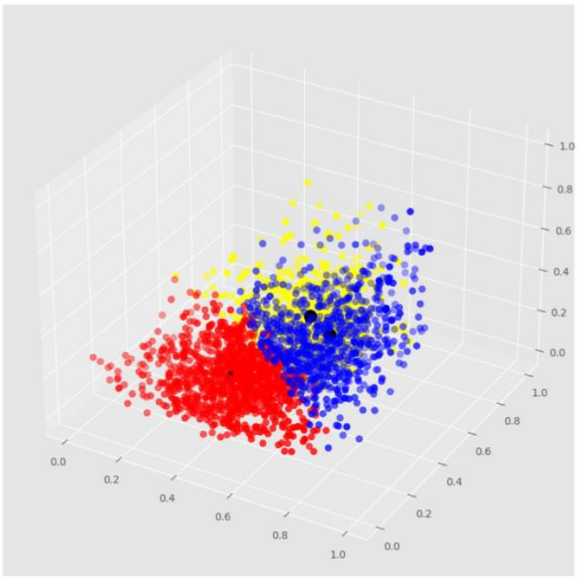


Figure 2.4 3D plot of clusters.

2.2. Analysis by Density-based Clustering – DBScan

For analysis with DBScan we have used the same MinMaxScaler which we have used in K-means algorithm.

2.2.1. DBScan for all Continuous Variables

We must use the "Elbow Method" to get the suitable epsilon level before applying the DBScan clustering algorithm. In graph (Firgure 2.5) epsilon appears to be best with a value of about 0.4. Finally, since our data has seventeen primary components, we will set our minimum points need to 17. Only one cluster, a Silhouette score of 0.258, and over 129 data points that are regarded as outliers or noise were produced when the epsilon was set to 0.4 and the min samples to 17 respectively. The Silhouette score was acceptable keeping in mind the dimensionality of the dataset but the results were not appealing. So, we tried increasing the min_samples to 34 but still we received similar results with Silhouette score of 0.244, one cluster, and 187 noise points. To improve our results, we decided to use an iterative approach to fine-tuning our DBSCAN model. We iterated through a range of epsilon 0.4 to 0.8 at 0.1 intervals and minimum point values ranging from 10–17. We found that with epsilon value 0.7 and min samples point 17 we got Silhouette score 0.510 and 1 cluster.

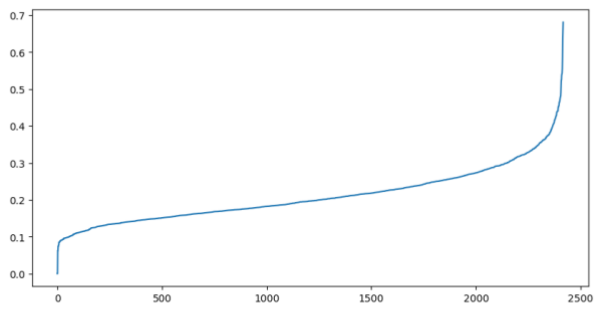


Figure 2.5 Elbow graph for DBScan

2.2.2. DBScan with Selected Variables

We have tried DBScan with selected features as well same as the selected features of K-means such as "zero crossing sum," "length_ms," and "intensity." Again, we determined the epsilon value using elbow method and we found the optimum value for epsilon is 0.04. We are going to use min sample points 6 keeping in mind the dimensionality of the dataset. As a result, we got Silhouette score -0.422, up to 926 points as outliers, and 18 clusters which was not a good result according to our dataset. So, again, we tried the iterative method to achieve best score and this time we iterated through a range of epsilon value from 0.04 to 0.17 at 0.01 intervals and minimum sample points ranging from 3-7. We found quite impressive results at epsilon 0.14 and the minimum sample points at 3 where we got Silhouette score of 0.490, only 2 points were resulted as outliers and only one cluster. Figure 2.6 displays pairplot of selected features which shows the outliers are detected very well and the cluster is very well visible.

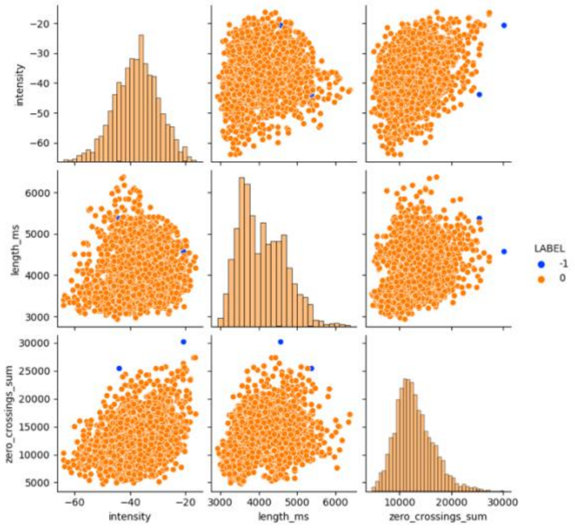


Figure 2.6 DBScan pairplot of selected variables

2.3. Analysis by Hierarchical Clustering

For analysis by hierarchical clustering, we have used the MinMaxScaler same as we have used in our previous clustering algorithms. Agglomerative (Bottom to Top) technique has been used to form clusters. Euclidean has been used to calculate distance between data points.

2.3.1. Hierarchical Clustering with all Continuous Variables

Initially we decided to set number of clusters to 4 and check estimated score of Silhouette and we got Silhouette for 4 clusters 0.131 which is not a satisfactory score. We decided to use the iterative approach to find out the best number of clusters to achieve a good or acceptable score of Silhouette. In iterative method we have used a range of clusters from 2 to 20. After iteration, we found that the best Silhouette value was 0.142 which is obtained through 2 cluster which was not quite satisfactory. So, we decided to do some research to improve our results especially the Silhouette score. We decided to change our linkage method to “ward” and we got a good Silhouette score of 0.209 with 2 clusters. It looks like a good score considering the dimensions of our dataset.

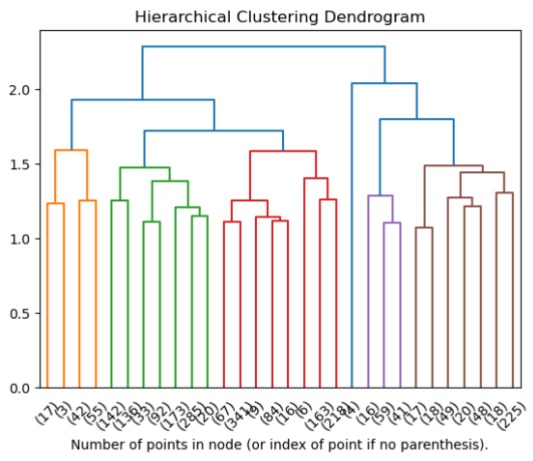


Figure 2.7 Dendrogram for all continuous variables

2.3.2. Hierarchical Clustering with Selected Variables

We have decided to use the same selected variables that we are using so far such as "zero crossing sum," "length_ms," and "intensity." Firstly, we have decided to use the complete linkage method to check the quality of the clusters. Figure 2.8 displays clustering dendrogram for 2 clusters using complete linkage method and the Silhouette score for 2 clusters was 0.310 which was a good score in terms of dimensions. Next, we tried the iterative approach to check whether we can get much better score than 0.310 or not. We iterate through a range of 2-50 of clusters but we found that only 0.310 was the best score with 2 clusters. We also decided to change our linkage method to “ward” to check if we can get better results but unfortunately this time ward method did not perform well and we got the Silhouette score of 0.272 for 2 clusters.

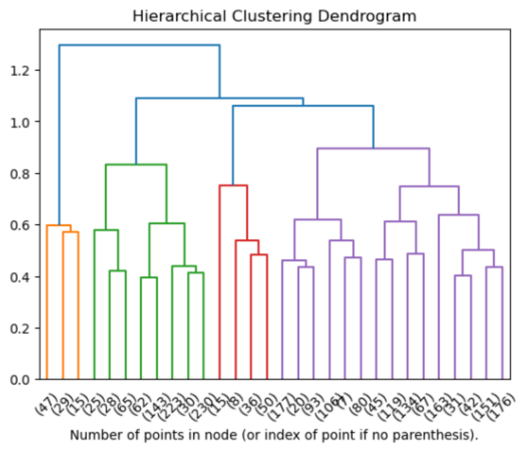


Figure 2.8 Dendrogram for selected variables.

2.4. Conclusion

If we conclude for all continuous variables, we have implemented three different clustering algorithms and evaluated results using relevant matrixes i.e. the Silhouette score. We found that the best algorithm for all continuous was DBScan which has obtained a quite satisfactory Silhouette score of 0.510. Therefore, the best algorithm for our dataset with all possible variables was DBScan. For selected variables such as "zero crossing sum," "length_ms," and "intensity" we have implemented different clustering algorithms and found that DBScan did very well with selected features as well. The best Silhouette score with selected features using DBScan we got was 0.490. For selected features, we will mark DBScan as winner and the best algorithm to work on our dataset.

Chapter 3

3. Classification

In this section of the report, three different types of classification have been performed such Decision Trees, KNN (K-nearest neighbors), and Naive Bayes. At the end of the chapter, we have discussed about the best algorithm based on relevant performance matrixes.

3.1. Data Splitting

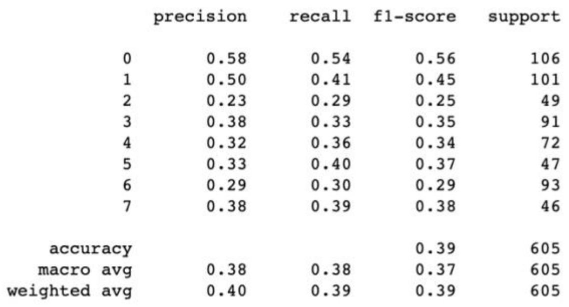
To determine the performance of a classifier, it is important to keep some data hidden from the model during the training process. This process of dividing data in train and test part is known as splitting of dataset. The test data is used to evaluate the model on unknown input. For this purpose, we divided our dataset into 75 and 25 ratio which means 75% of the dataset will be used for training purpose and 25% of the dataset will be used for testing purpose.

3.2. Decision Trees

In the following section, we did multiclass and binary class classification using decision tree classifier considering different variables as target variables each time.

3.2.1. Multiclass Classification

For multiclass classification we have used emotion as our target variable which classifies into 8 different classes as {'angry': 0, 'calm': 1, 'disgust': 2, 'fearful': 3, 'happy': 4, 'neutral': 5, 'sad': 6, 'surprised': 7}. Initially, we simply trained our model without any hyper-parameter tuning and all the 37 variables were used to train the model. The training has been performed on a total of 1812 tuples but the results were not appealing although the accuracy for the training set was 1.0 but for the test set the accuracy dropped down to 0.39 or 39%. The following Figure 3.1 depicts the classification report of the predictions on the test set. We can interpret that the numbers are not good even for the single class the score is not good. The highest precision we got is 0.58 for the first class, the highest recall is also for the first class which is 0.54, and likewise the f1-score 0.56 which is highest for the first class. Figure 3.2 displays the confusion matrix for each class for the test set. We can see only the first class has the highest numbers of true positive predictions. The figure 3.3 on the right shows ROC curves for multiclass and the Area Under the Curve score for multi class classification is 0.64.



```
              precision    recall  f1-score   support

           0       0.58      0.54      0.56       106
           1       0.50      0.41      0.45       101
           2       0.23      0.29      0.25        49
           3       0.38      0.33      0.35        91
           4       0.32      0.36      0.34        72
           5       0.33      0.40      0.37        47
           6       0.29      0.30      0.29        93
           7       0.38      0.39      0.38        46

    accuracy                           0.39       605
   macro avg       0.38      0.38      0.37       605
weighted avg       0.40      0.39      0.39       605
```

Figure 3.1 Classification Report for multi-class.

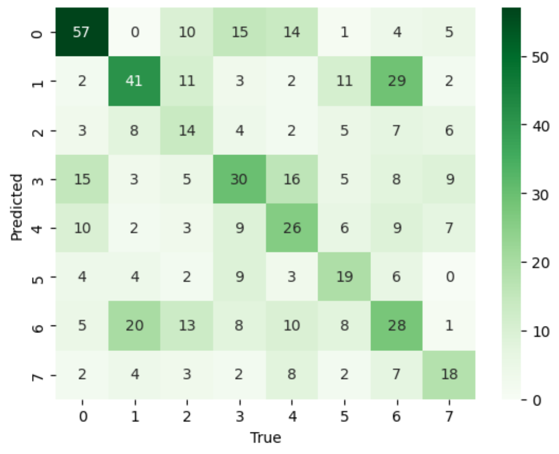


Figure 3.2 Confusion matrix for multi-class

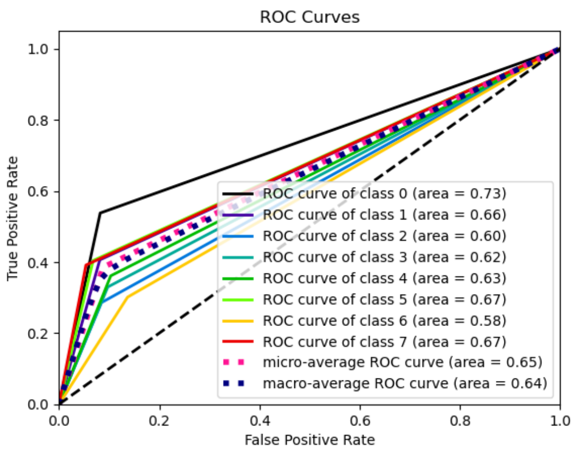


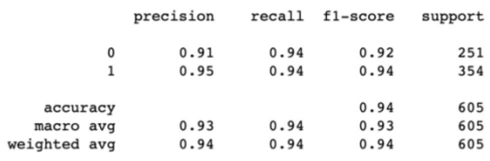
Figure 3.3 ROC Curve for multi-class

3.2.2. Binary Class Classification

There are multiple variables which are binary by nature such as vocal_channel, emotional_intensity, statement, repetition, and sex. We have decided to experiment with each variable one by one and evaluate the model for each variable using different matrixes. First of all, we decided to use channel as our target variable.

Vocal Channel

Vocal channel variable has two classes: speech and song. For binary class classification the decision tree performed very well with an average accuracy of up to 0.94. Figure 3.4 represents the precision, recall, and f1-score matric for the evaluation of the model as each class and average. The result for binary classification were good and the model did very well on the test dataset. Figure 3.5 represents the total numbers of each class in test set and we can see



```
              precision    recall  f1-score   support

           0       0.91      0.94      0.92       251
           1       0.95      0.94      0.94       354

    accuracy                           0.94       605
   macro avg       0.93      0.94      0.93       605
weighted avg       0.94      0.94      0.94       605
```

Figure 3.4 Classification Report for vocal channel

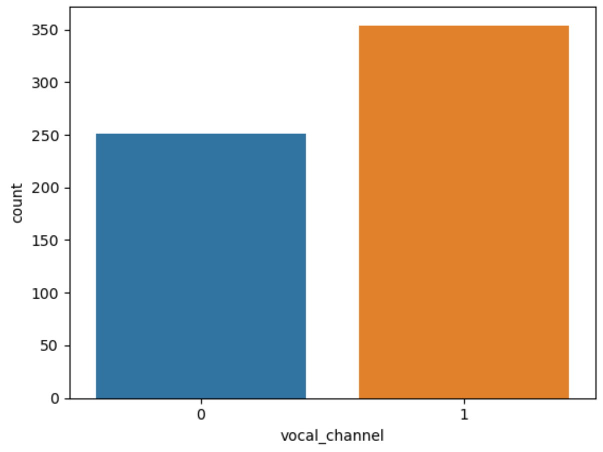


Figure 3.5 Count plot of test set.

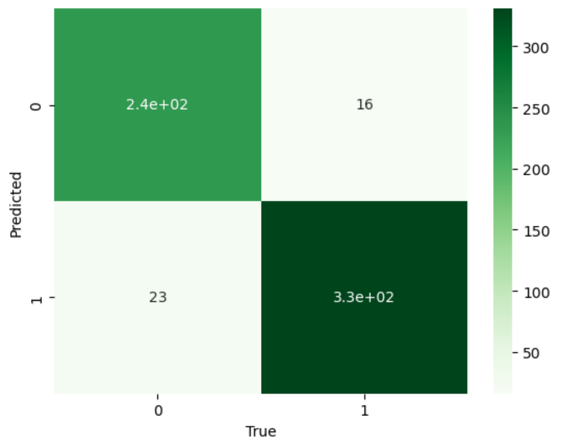


Figure 3.6 Confusion matrix for binary classification

that there are around 250 speeches and almost 350 songs which is a good distribution for a test set. On the right side the figure 3.6 represents the confusion matrix for the test set on the left.

From the matrix we can interpret that out of 250 speeches input the model correctly predicted 240 inputs and out of 350 songs input the correctly predicted 330 inputs.

Other Binary Variables Classification

The same Decision Tree algorithm was trained for other binary variables. For instance, the same model used for the vocal channel has been trained keeping emotional intensity as the target variable. The outcomes were acceptable but not good. The model yielded 1 percent of train accuracy but 73 percent accuracy and 0.73 F1-Score.

Likewise, the same model has been trained for the statement variable. The results obtained was not so promising as the training accuracy was 1 but the test accuracy and f1-score dropped to 0.57 each.

3.2.3. Cross Validation

After training our model on different target variables without any hyperparameter tunning, we decided to cross validate our model with different parameters using randomized search and grid search validation.

3.2.3.1. Randomized Search Cross Validation

For randomized search cross validation, we have selected emotional intensity as our target variable. Emotional intensity is a binary variable and we have used the following hyperparameters tunning for our classifier: min_samples_split [0.002, 0.01, 0.05, 0.1, 0.2], min_samples_leaf [0.001, 0.01, 0.05, 0.1, 0.2], max_depth [2, 4, 6, 8, 10, 12, 16, None], criterion ["gini", "entropy"]. The best parameter obtained by the Randomized Search were criterion='entropy', max_depth=8, min_samples_leaf=0.01, and min_samples_split=0.05 with the score of 0.75.

The results obtained from randomized search cross validation are slightly better than the model we trained without any hyperparameter tunning. The results varied slightly because they were random, but we didn't see any significant changes.

3.2.3.2. Grid Search Cross Validation

For grid search cross validation, we have decided to use the same parameters as we used in the randomized search to check the difference between the two methods. We are going to keep the emotional intensity as our target variable. During grid search we realized that it was taking more time than randomization and it is normal as because grid search tries all the possible combination of the parameters to get results as compared to randomization which only picks random parameters to run the tests on. The optimal values for the parameters were criterion='entropy', min samples split=0.05, min samples leaf=0.01, and max depth=8. This combination didn't significantly raise the score; the outcomes were same as the randomization search.

3.3. KNN

In the following section we have implemented k-nearest neighbors algorithm for multi-class and binary class classification. For KNN, we decided to use StandardScaler to normalize our dataset instead of MinMaxScaler. Furthermore, we have used the same splitting strategy of dividing our dataset in 75% training set and 25% testing.

3.3.1. KNN for Multiclass Classification

For multiclass classification, we have used Euclidean distance as metric and n_neighbors equal to 37 keeping in mind the dimensionality of the dataset. We used emotion variable as our target variable same as the decision tree classifier. The result we got was not satisfactory because the accuracy we got is 0.17 which is really bad. So, we decided to improve our model to find out the optimal value for k neighbors. Figure 3.7 depicts the error rate in comparison between different k values and we can see that the smallest error we got is 0.69 at K=34. So, after finding an optimal value, we used k=34 for our classifier to find out the improvements in our accuracy. After training and testing our model with the above said parameters we found that the maximum test accuracy we got is 0.30 with 34 k neighbors. We found some improvements in our results as 0.30 is way better than 0.17. The results are acceptable if we check the dimensionality of the dataset.

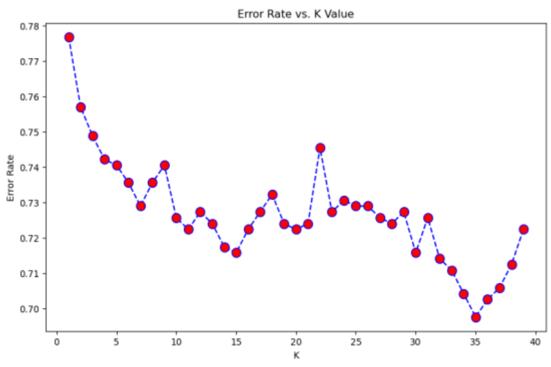


Figure 3.7 Error rate vs k neighbors for multiclass KNN

3.3.2. KNN for Binary Class Classification

For binary class classification, initially we have used the same parameters as we did for the multiclass classification such as Euclidean distance as metric and n_neighbors equal to 37. The results we got was not much satisfactory but much better than the multiclass classification. The accuracy for the model was 65% which is good without any special hyperparameter tunning. Then we decided to use an iterative approach to find an optimal value of k. Figure 3.8 displays the iterative tests we have performed with the k value ranging from 1 to 50. The maximum accuracy we got is 66% with k equal to 44 which is only 1% better than the previous experiment without any special tunning. Finally, we decided to perform a grid search cross validation.

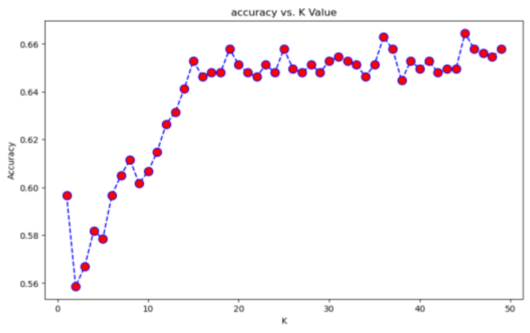


Figure 3.8 Accuracy vs k value for binary class KNN

3.3.2.1. Grid Search Cross Validation

For grid search cross validation for KNN, we have used different hyperparameters such as "weights": ["uniform", "distance"], "metric": ["euclidean", "cityblock"], n_neighbors range from 1 to 50. Performing the Grid Search, the following optimum parameters for the "emotional intensity" variable were discovered: metric: cityblock, n neighbors:22, and weights: distance. The obtained accuracy was 75%, and the F1-Score was 0.74. The figure 3.9 displays the result of different parameters of the grid search cross validation and we can clearly see that the best accuracy has been found when number of neighbors are 22. The results have been improved in grid search validation than without any special hyperparameter tunning in the previous section. Likewise, we have performed the grid search cross validation for vocal channel and statement and interestingly we got 96% of accuracy and 0.95 f1-score for vocal channel whereas for statement we got 59% accuracy and 0.59 f1-score.

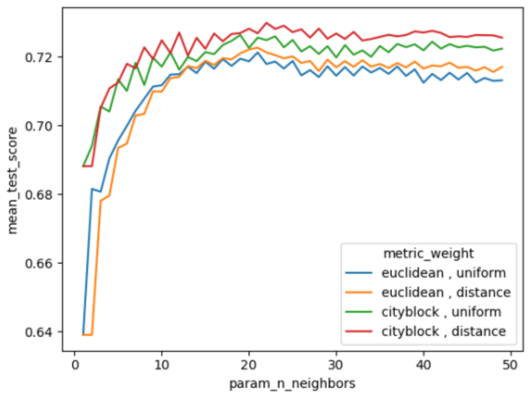


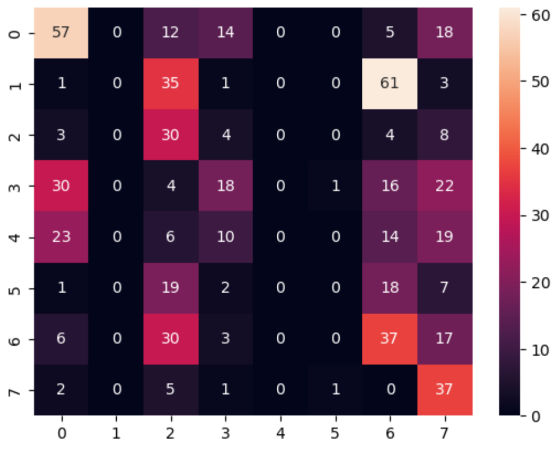
Figure 3.9 Accuracy vs k value for Grid Search KNN

3.4. Naïve Bayes

In the following section, we have applied Naïve Bayes algorithm on our dataset using Gaussian Naïve Bayes on multiclass classification and Categorical Naïve Bayes on Binary Class Classification. We have used the stander scaler same as we used in the previous algorithm and the splitting strategy was 75 ratio 25.

3.4.1. Naïve Bayes for Multiclass Classification

For multiclass classification, the results were not really good as the overall accuracy of the model was only 30% and shockingly, the probability of some classes such as 1, 4, and 5 was zero which means no tuple has been classified in 1, 4, and 5th class. Figure 3.10 displays the confusion matrix for multiclass Naïve Bayes where we can see that true positive value for 1, 4, and 5th class is 0. The highest probability for multiclass classification is for 0 class which is up to 57. The overall results according to the dimensionality of the dataset is acceptable but not good for use in real life application. Next, we tried Naïve Bayes algorithm for binary class classification considering vocal_channel as our target class.

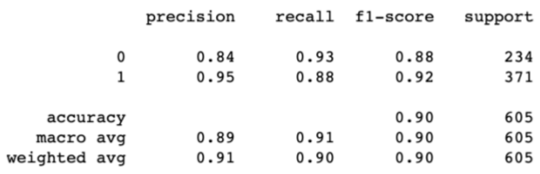


	0	1	2	3	4	5	6	7
0	57	0	12	14	0	0	5	18
1	1	0	35	1	0	0	61	3
2	3	0	30	4	0	0	4	8
3	30	0	4	18	0	1	16	22
4	23	0	6	10	0	0	14	19
5	1	0	19	2	0	0	18	7
6	6	0	30	3	0	0	37	17
7	2	0	5	1	0	1	0	37

Figure 3.10 Confusion Matrix for multiclass Naive Bayes.

3.4.2. Naïve Bayes for Binary Classification

Naïve Bayes performed very well for binary class classification, as you can see in the figure 3.11, we achieved an accuracy of 90% for binary classes 0 and 1. Furthermore, we have implemented naïve bayes algorithm considering other binary variables such as emotional_intensity and statement and the accuracy we achieved is 47% and 55% respectively.



```
              precision    recall  f1-score   support

           0       0.84      0.93      0.88       234
           1       0.95      0.88      0.92       371

    accuracy                           0.90       605
   macro avg       0.89      0.91      0.90       605
weighted avg       0.91      0.90      0.90       605
```

Figure 3.11 Classification Report for Naive Bayes.

3.5. Conclusion

Three algorithms such as Decision Trees, K-NN, and Naive Bayes were used in this section to classify the various target variables. The models were trained first without parameter tunning, and then they were tuned via Grid and Randomized Search. The dataset was split into two portions for training and testing, 75% and 25% respectively. The accuracy, Precision, Recall, F1 Score, ROC Curve, and Confusion Matrix were then used to assess the results. Both the multi-class classification of the "emotional" variable and several binary variables like "emotional intensity," "vocal channel," and "statement" were trained into the algorithms. For multiclass classification, the highest accuracy we achieved was 39% with decision tree classification. For Binary class classification, the highest accuracy we achieved before parameters tunning was 94% with decision tree classification. In conclusion, the decision tree algorithm performed very well on overall classification of the dataset.

Chapter 4

4. Pattern Mining

In the following section, we will focus on pattern mining using apriori and fp growth algorithm for frequent pattern extraction, closed itemset extraction, maximal itemset extraction, and rule extraction. Firstly, the dataset has been preprocessed before diving in. At the end the most intresting rules are use to predict a general target variable.

4.1. Data Preprocessing

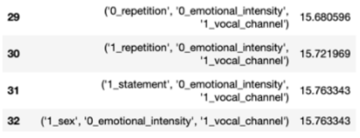
It was necessary to manage the data in order to perform a proper association analysis. To do this, we have dropped the useless variables which do not provide any useful information to the data such as modality, channels, sample_width, frame_rate, stft_max, and frame_width. Furthermore, we transformed continuous variables into categories. We used bining technique and converted continuous variables into bins of different intervals.

4.2. Frequent Pattern Extraction

Frequent pattern extraction has been performed on the dataset using apriori and fp growth algorithm.

4.2.1. Apriori Algorithm

Since there were 31 separate variables, we kept the minimum number of items in an itemset at 3 and the minimum support level at 15%. The apriori algorithm mined 32 Frequent Itemsets in total as shown in figure 4.1.

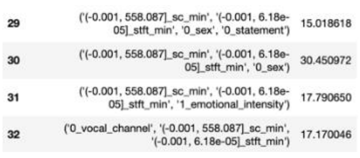


29	('0_repetition', '0_emotional_intensity', '1_vocal_channel')	15.680596
30	('1_repetition', '0_emotional_intensity', '1_vocal_channel')	15.721969
31	('1_statement', '0_emotional_intensity', '1_vocal_channel')	15.763343
32	('1_sex', '0_emotional_intensity', '1_vocal_channel')	15.763343

Figure 4.1 Apriori frequent itemset.

4.2.2. FP Growth

The FP Growth method was applied to the same dataset while maintaining the same parameters as apriori e.g. 3 minimal items and 15% support. The number of frequent item sets that were extracted using FP Growth was 32, the same as it was apriori as shown in Figure 4.2.



29	('(-0.001, 558.087]_sc_min', '(-0.001, 6.18e-05]_stft_min', '0_sex', '0_statement')	15.018618
30	('(-0.001, 558.087]_sc_min', '(-0.001, 6.18e-05]_stft_min', '0_sex')	30.450972
31	('(-0.001, 558.087]_sc_min', '(-0.001, 6.18e-05]_stft_min', '1_emotional_intensity')	17.790650
32	('0_vocal_channel', '(-0.001, 558.087]_sc_min', '(-0.001, 6.18e-05]_stft_min')	17.170046

Figure 4.2 FP Growth frequent itemset.

4.3. Closed Itemsets Extraction

The itemsets that are closed don't have the same support as their immediate supersets. So, the closed itemsets can be considered a subset of the overall frequent itemsets.

4.3.1. Apriori Algorithm

Interestingly, all the items that were included in the frequent itemset was in the closed itemset because of the support value we have chosen. We have kept the minimum support of 15% and the minimum items per set 3.

4.3.2. FP Growth

The results were different for fp growth algorithm keeping the same parameters as frequent items extraction such as minimum support 15% and minimum items per set 3. But the closed itemset were same in numbers i.e. 32.

4.4. Maximal Itemsets Extraction

Any frequent itemset for which none of its immediate supersets are frequent is said to be maximal itemset.

4.4.1. Apriori Algorithm

Apriori algorithm has been applied using the same parameters such as 15% minimum support and 3 minimum items in a set and a total of 13 itemsets were extracted.

4.4.2. FP Growth

By using the same parameters as of apriori fp growth algorithm has been applied to extract maximal itemsets and the results were same and the total of 13 itemsets were extracted.

4.5. Finding the Optimal Support

Initially, we decided to use a fix support for each type of extraction such as 15% to analyze the itemsets by different techniques. In order to find an optimal support, we decided to use the iterative method and visualization to check what is the best support for each type of technique. Figure 4.3 shows the number of itemsets upon support from 2% to 25% for apriori algorithm for different techniques. Here we choose the minimum items per set is 3. The plot shows the optimal minimum support would be 5% but such a low support would be not very good for such amount of data.

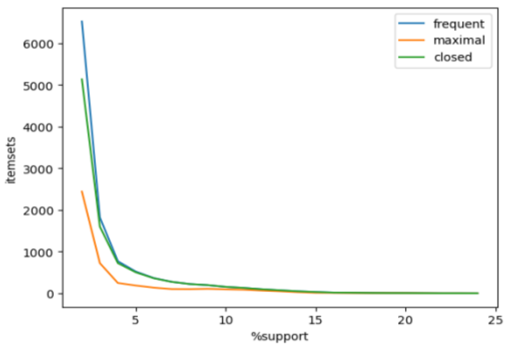


Figure 4.3 Apriori support vs itemsets plot

An effort has been made to examine the support to obtain itemsets with a particular item. Figure 4.4, for instance, demonstrates that 3% support is required to obtain more than 800 itemsets with the word "speech" in them, but 12% support is required to obtain itemsets with no songs.

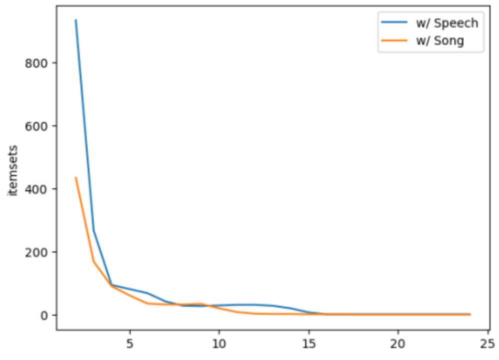


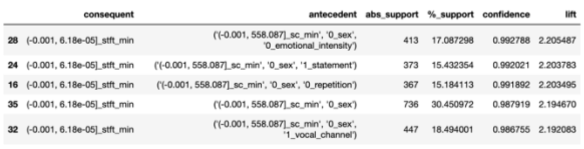
Figure 4.4 Vocal channel plot for support vs itemset

4.6. Rules Extraction

The most important component of pattern mining is the examination of association rules. Business users may easily make decisions based on the patterns in the dataset thanks to the if-then rules.

4.6.1. Rules Extraction using Apriori

The dataset was subjected to the Apriori method in order to discover the association rules. The 60 percent confidence level was maintained. Three minimum items were required, and 15% support was given. By using such thresholds with the apriori method, a total of 84 rules were extracted. Figure 4.5 shows the top rules based on the lift parameter.

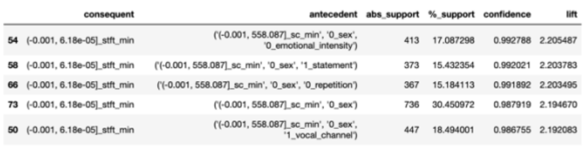


	consequent	antecedent	abs_support	%_support	confidence	lift
28	(-0.001, 6.18e-05]_stft_min	('(-0.001, 558.087]_sc_min', '0_sex', '0_emotional_intensity')	413	17.087298	0.992788	2.205487
24	(-0.001, 6.18e-05]_stft_min	('(-0.001, 558.087]_sc_min', '0_sex', '1_statement')	373	15.432354	0.992021	2.203783
16	(-0.001, 6.18e-05]_stft_min	('(-0.001, 558.087]_sc_min', '0_sex', '0_repetition')	367	15.184113	0.991892	2.203495
35	(-0.001, 6.18e-05]_stft_min	('(-0.001, 558.087]_sc_min', '0_sex')	736	30.450972	0.987919	2.194670
32	(-0.001, 6.18e-05]_stft_min	('(-0.001, 558.087]_sc_min', '0_sex', '1_vocal_channel')	447	18.494001	0.986755	2.192083

Figure 4.5 Rules extraction using apriori.

4.6.2. Rules Extraction using FP Growth

The same parameters settings were applied to the fp growth algorithm. Same number of rules were extracted i.e. 84 but the results were different. Figure 4.6 shows top rules based on the life parameter.



	consequent	antecedent	abs_support	%_support	confidence	lift
54	(-0.001, 6.18e-05]_stft_min	('(-0.001, 558.087]_sc_min', '0_sex', '0_emotional_intensity')	413	17.087298	0.992788	2.205487
58	(-0.001, 6.18e-05]_stft_min	('(-0.001, 558.087]_sc_min', '0_sex', '1_statement')	373	15.432354	0.992021	2.203783
66	(-0.001, 6.18e-05]_stft_min	('(-0.001, 558.087]_sc_min', '0_sex', '0_repetition')	367	15.184113	0.991892	2.203495
73	(-0.001, 6.18e-05]_stft_min	('(-0.001, 558.087]_sc_min', '0_sex')	736	30.450972	0.987919	2.194670
50	(-0.001, 6.18e-05]_stft_min	('(-0.001, 558.087]_sc_min', '0_sex', '1_vocal_channel')	447	18.494001	0.986755	2.192083

Figure 4.6 Rules extraction using fp growth.

4.6.3. Optimization of Confidence and Support for Rules

An iterative technique has been used to maximize the confidence and support, and the apriori algorithm has been tested for various combinations of support and confidence. Figure 4.7 depicts a heatmap that shows how the number of extracted rules reduces as support increases. We extracted nearly the optimal number of rules at 20% support and 60% confidence.

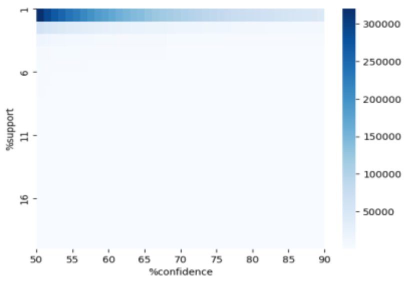
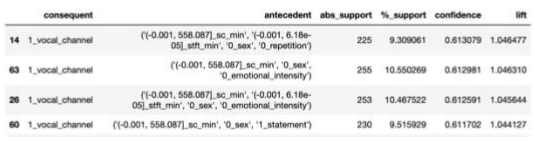


Figure 4.7 Optimization heatmap for rules extraction.

4.7. Target Variable Prediction

We have considered vocal_channel as our target variable to find consequences of speech i.e. 1_vocal_channel is equal to speech. The figure 4.8 shows that if sc_min is between -0.001 to 558.087, the stft_min is between -0.001 to 6.18e05, the gender is female, and the repetition is 1st the vocal channel would be “speech”.



	consequent	antecedent	abs_support	%_support	confidence	lift
14	1_vocal_channel	('(-0.001, 558.087]_sc_min', '(-0.001, 6.18e-05]_stft_min', '0_sex', '0_repetition')	225	9.309061	0.613079	1.046477
63	1_vocal_channel	('(-0.001, 558.087]_sc_min', '0_sex', '0_emotional_intensity')	255	10.550269	0.612981	1.046310
26	1_vocal_channel	('(-0.001, 558.087]_sc_min', '(-0.001, 6.18e-05]_stft_min', '0_sex', '0_emotional_intensity')	253	10.467522	0.612591	1.045644
60	1_vocal_channel	('(-0.001, 558.087]_sc_min', '0_sex', '1_statement')	230	9.515929	0.611702	1.044127

Figure 4.8 Vocal Channel as target variable for pattern mining.

4.8. Conclusion

An essential component of data mining is pattern mining. The patterns in the dataset were discovered using techniques like apriori and FP Growth. For the parameter settings of 20% support and 3 minimum items per set, a total of 32 frequent and closed itemsets as well as 13 maximal itemsets were discovered using the aprori and FP Growth algorithms. Additionally, association rules were extracted using both techniques, with a 60 percent confidence level. Both algorithms discovered 84 rules in total. The prediction of a few target variables, including "emotional intensity," "vocal channel," "statement," etc., was then accomplished effectively using the association rules.

Chapter 5

5. Regression

We have applied different types of regression algorithm on our dataset. For Univariate Regression, we have considered length_ms and intensity because we want to check the dependency of intensity on length_ms. And for Multivariate Regression, we have considered length_ms, intensity, and zero crossing sum to find out the dependency of zero crossing sum on both variables. We have divided the dataset in 75% training set and 25% test set. The following table 5.1 shows different matrixes such as Coefficients, Intercept, R2 score, Mean Square Error, and Mean Absolute Error for all possible regression algorithm for Univariate and Multivariate Regression.

Table 5.1 Regression Matrixes for different algorithms

	Coefficients	**Intercept**	**R2 score**	**MSE**	**MAE**
Univariate Regression					
Linear Regression	0.003	-49.385	0.003	59.408	6.210
Ridge	0.003	-49.385	0.003	59.408	6.210
Lasso	0.003	-49.385	0.003	59.408	6.210
Decision Tree Regressor			0.032	57.690	6.164
KNN Regressor			-0.070	63.757	6.428
Multivariate Regression					
Linear Regression			0.191	10783003.41	2521.56
Decision Tree Regressor			-0.584	21106927.18	3621.253
KNN Regressor			0.038	12813007.357	2799.27