Data Mining II Project

Bernardo D'Agostino - Mohamed Arafaath - Vincenzo Rocchi

Academic Year 2022/2023

1 Module 1

In the initial module of the project, we employed both Decision Tree and K-Nearest Neighbors (K-NN) models, using KNeighborsClassifier and DecisionTreeClassifier from the sklearn library, for the binary classification task. We evaluated the performance of these models under different scenarios, including feature selection, outlier detection, and handling unbalanced learning tasks. The first dataset analyzed in this study is a reworking of the RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song) dataset, which contains audio-visual recordings of 24 actors saying short statements in English. Our dataset is already split into train and test sets, the train set contains a total of 1828, while the test set contains 624 instances. Each instance is associated with 434 features concerning synthesis measures and information of the audio files contained in the original dataset.

1.1 Classification and outlier detection

In this first part of the report, we aimed to build a K-nn and decision tree classifier on the emotional_intensity feature, to do so we tried different feature selection techniques and outlier detection methods to achieve the best performance. To perform feature selection we split our training data into training (0.75%) and validation (0.25%) sets.

1.1.1 Model training description

Our K-nn model was fine-tuned each time through a 5 folds cross-validated grid search on the train set which tried all possible numbers of neighbors in the range from 1 to the square root of the number of samples in the training set, comparing between Manhattan and Euclidean distance, and distance and uniform point weighting. The decision tree model was also fine-tuned through a 5 folds cross-validated grid search which tried the combinations of the parameters: Splitting criterion: [Gini, entropy] Max depth: [2, 5, 10, 15, 20, None], Min samples split: [2, 10, 30, 50], Min samples leaf: [1, 10, 30, 50].

1.1.2 Features description

The dataset includes 9 categorical variables, 'modality', 'vocal_channel', 'emotion', 'emotional_intensity', 'statement', 'repetition', 'actor', 'sex', and 'filename'. The 425 numerical variables in the dataset describe various statistics of the original audio signal and Five different transformations applied to it: Zero Crossing Rate (zc), Mel-Frequency Cepstral Coefficients (mfcc), Spectral Centroid (sc), STFT Chromagram (stft), Lag1. The extracted statistics computed on both the original data and its transformations are: sum, mean, standard deviation (std), minimum (min), maximum (max), kur (kurtosis), and skew (skewness). Furthermore, they have been calculated across the entire length of the waveform and across multiple quantiles.

1.1.3 Preprocesing

The first thing we did was apply min-max normalization to the continuous attributes. We fitted the scaler on the train data and then also applied it to the test data. We then dropped the filename, modality, and actor features since they are irrelevant. We then found and eliminated 51 features that only had 1 value. We then one-hot encoded the categorical variable emotions and binarized all the remaining categorical variables.

1.2 Feature selection

Since our dataset has a high number of features we tried different feature selection techniques, the aim of our feature selection was to find the best-performing subset of features for a K-nn classifier as the decision tree classifier has embedded feature selection. To have a benchmark for feature selection we first trained our K-nn model on the full dataset achieving a cross-validated accuracy of 0.731 on the train set and of 0.746 on the validation set.

1.2.1 Removing highly correlated variables

To avoid redundancy in the feature space we checked the correlations between the continuous features in the train set and found a very high number of highly correlated variables with 318 pairs of variables with an absolute correlation coefficient over 0.95. We selected five different thresholds for the correlation coefficient, namely 0.7, 0.8, 0.9, 0.95, and 0.99. For each threshold, we identified the highly correlated pairs of variables. Then, we iteratively eliminated the feature that had the highest number of remaining highly correlated variables. Then we trained our K-nn model on all subsets of features but as can be seen from Table 1 we achieved a significant drop in performance, so we discarded this feature selection technique.

Table 1: Threshold Comparison for Correlation

Threshold	C.V. Tra Acc.	Val Acc.	Elim. Att.
0.7	0.703	0.687	246
0.8	0.713	0.698	202
0.9	0.714	0.684	155
0.95	0.716	0.691	111
0.99	0.728	0.726	52
0	0.731	0.746	0

Table 2: Threshold Comparison for Low Variance

Threshold	C.V. Train Acc.	Val Acc.	Elim Att.
0.1	0.531	0.542	374
0.01	0.727	0.715	280
0.005	0.735	0.750	57
0.002	0.733	0.748	111
0.001	0.731	0.750	11
NO	0.731	0.746	0

1.2.2 Removing low variance features

We then performed feature selection based on the variance thresholds 0.1, 0.01, 0.005, 0.002, and 0.001. At each threshold, we trained the K-nn classifier using the subset of features that had a variance higher than the corresponding threshold. From Table 2 we can see that we achieved a slight improvement in the validation accuracy for the 0.001 threshold.

1.2.3 Recursive Feature Elimination

We then performed a Recursive Feature Elimination using the RFE library from sklearn, we first built a Random Forest classifier of 100 trees without any limit for depth or leaf and node samples and used the square root of our features to build each tree. We used this model to compute feature importance for the RFE model. At each iteration, we eliminated the 5 features with the lowest importance score computed as the sum of the impurity reduction for each feature across all decision tree nodes.

We then selected the 25, 30, 40, 50, 60, 75, 100, 125, and 150 most relevant features and trained the K-nn classifier using that subset. From Table 3 it's possible to see that the subset of 50 features significantly outperformed all previous feature selection techniques achieving a C.V. accuracy of 0.792 on the train set

and an accuracy of 0.781 on the validations set, so we choose to use that subset of features in the following sections.

Table 3: Feature Selection Performance

Selected Features	C.V. Train Acc.	Validation Acc.
25	0.754	0.722
30	0.754	0.730
40	0.771	0.741
50	**0.792**	**0.781**
60	0.786	0.776
75	0.786	0.754
100	0.784	0.763
125	0.775	0.774
150	0.770	0.759
All	0.731	0.746

1.3 Outlier Detection

We then applied different outlier detection techniques to the numerical features of the dataset for the Decision Tree classifier and to the subset of 50 selected features for the K-nn model. For this task, we used the combination of train and validation sets, optimized the hyper-parameters through cross-validation, and tested the final models on the test set with 1%, 5%, and 10% of outliers removed at each iteration.

1.3.1 Local Outlier Factor (LOF)

We calculated the Local Outlier Factor (LOF) for both datasets using the LocalOutlierFactor library from sklearn. To choose the correct number of neighbors we applied the strategy described in the original paper [1], we applied the algorithm using all numbers of neighbors in the range between 5 and 100, selecting for each observation its highest LOF score.

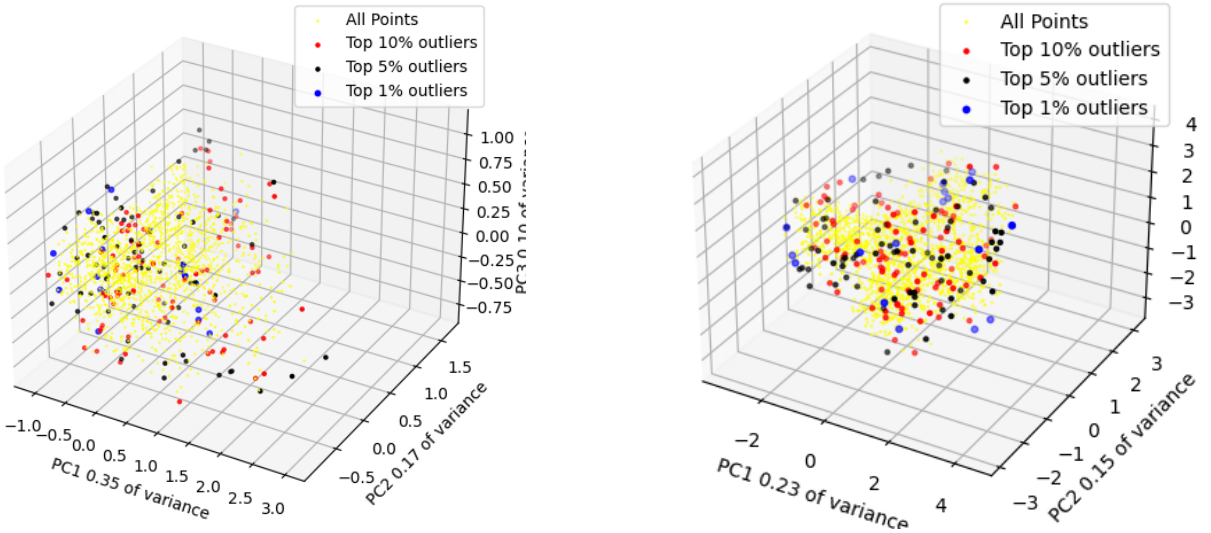
From Figure 1 it's visible that in both datasets the outlier points are located in the less dense regions of the PCA representation of the data. We observed that when increasing the number of neighbors from 5 to 100 in both datasets, only 5% of the points exhibited an absolute difference in the LOF (Local Outlier Factor) score greater than 10%. The highest increase in the LOF score was 0.3 points for both datasets, while the highest decrease was 0.3 for the K-nn dataset and 0.2 for the D.T. dataset, keeping in mind that the highest lof score we got was 1.2, demonstrating the robustness of this outlier detection techniques over the nn parameter.

We then selected the top 1%, 5%, and 10% of outliers, eliminated them, and trained both classifiers on the resulting datasets. Based on Table 4, the impact of outlier elimination on the performance of the K-nn model was marginal with a decrease in performance of 1% when removing 10% of outliers. We hypothesize that this is due to the majority of points belonging to a single high-density region depicted in Figure 1a. Consequently, most points do not have outliers (as identified by the LOF method) as neighbors. On the other hand, the D.T. model exhibited a noteworthy performance improvement of 3% more accuracy after removing the top 5% of outliers.

It can also be noted that the performance of the K-nn model decreased on the test set with an accuracy of 76% compared to the accuracy of 78% that we achieved on the validation set.

Table 4: Models Performance over LOF tresholds

Threshold	Elim. Obs.	Train K-nn	Test K-nn	Train D.T.	Test D.T.
1%	18	0.751	0.764	0.719	0.706
5%	91	0.755	0.761	0.719	**0.738**
10%	182	0.754	0.756	0.707	0.714
No	0	0.751	**0.766**	0.703	0.708



(a) PCA representation of outliers on K-NN. dataset

(b) PCA representation of outliers on D.T. dataset

Figure 1: PCA representation of outliers for LOF

1.3.2 Isolation Forest (IF)

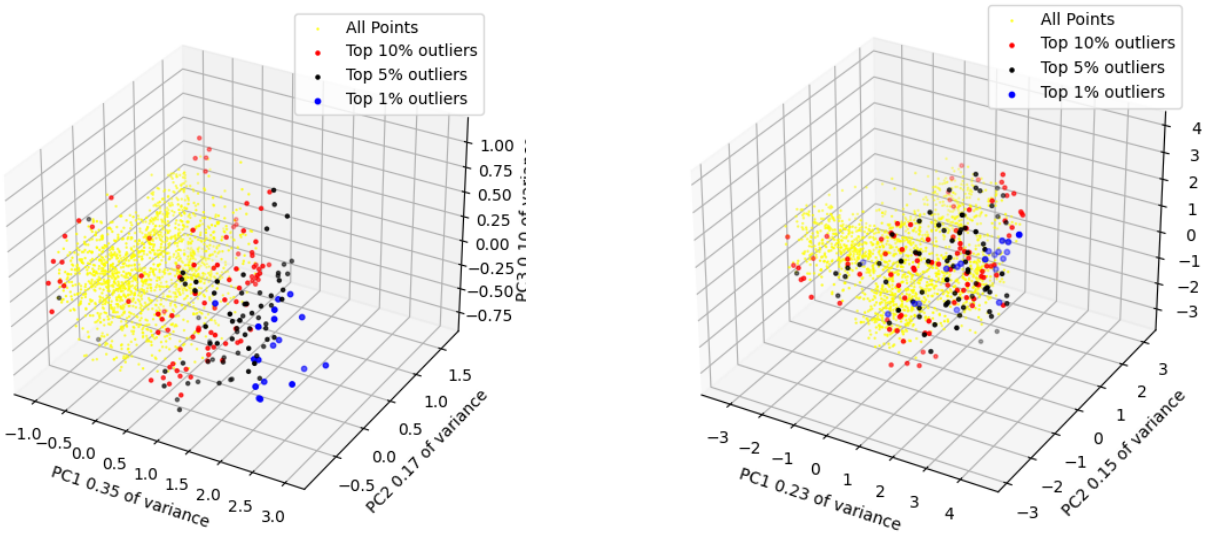
We then applied the Isolation Forest (IF) outlier detection method using the IsolationForest library from sklearn. We trained a forest of 100 trees identifying the top 1%, 5%, and 10% of outliers in both datasets and then trained and tested both models on the various subsets. From Figure 2, it is evident that the Isolation Forest algorithm identified outliers primarily at the edges of the data distribution, targeting data points with high positive values along the first principal component in both cases.

A notable distinction can be observed when comparing the IF algorithm to the LOF method: while the IF algorithm selected the top 1% of outliers within a clustered region of the space, LOF identified outliers primarily at the fringes of the overall distribution. This distinction holds for the other percentages of outliers and is likely attributed to the inherent dissimilarity between the two algorithms, as LOF focuses on detecting points in low-density regions of space, whereas IF targets points that are distant from the main distribution.

From Table 5, it is evident that contrary to the LOF method, both classifiers didn't show significant improvements after the removal of outliers.

Table 5: Models Performance over IF tresholds

Thresh.	N. Elim Obs.	Train K-nn	Test K-nn	Train DT	Test DT
1%	19	0.749	0.766	0.698	0.693
5%	91	0.740	0.766	0.693	0.708
10%	182	0.734	0.761	0.684	0.708
NO	0	0.751	**0.766**	0.703	**0.708**



(a) PCA representation of outliers on K-NN dataset

(b) PCA representation of outliers on D.T. dataset

Figure 2: PCA representation of outliers for Isolation Forest

1.3.3 Angle Based Outlier Detection (ABOD)

We utilized the ABOD model from the PYOD library, which estimates the outlying ness of data points based on their angles relative to other points in the dataset. Specifically, we employed the "fast" method, avoiding the normal version of Abod since it was too computationally expensive.

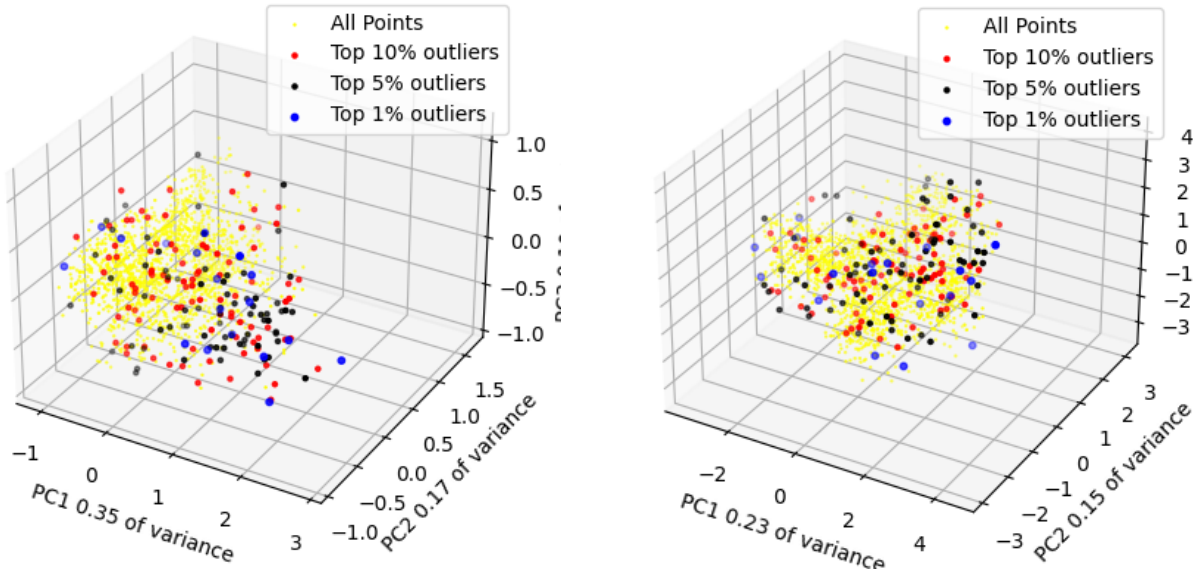
This method avoids calculating angles for all data points in the dataset and instead utilizes a specified number of neighbor points; we then proceeded to remove the outliers found by the algorithm, selecting and removing the top 1,5 and 10% and trained both classifiers on the dataset.

We then tried to find the best n_neighbors experimenting with the values 10, 20, 50, and 100 we found 100 exhibited slightly superior performance while not being too computationally expensive.

From Table 6, we can see that the outlier removal process resulted in a more or less equal performance for the k-nearest neighbors (KNN) model, while in the decision tree (DT) model improved the accuracy of the DT classifier by 4 p.p. when 10% of the outliers were removed.

Table 6: Models Performance over ABOD Thresholds

Threshold Elim.	Obs.	Train KNN	Test KNN	Train DT	Test DT
1.0	18	0.751	0.764	0.704	0.708
5.0	91	0.742	0.764	0.709	0.731
10.0	182	0.738	0.764	0.707	**0.745**
NO	0	0.751	**0.766**	0.703	0.708



(a) PCA representation of outliers on K-nn dataset

(b) PCA representation of outliers on D.T. dataset

Figure 3: PCA representation of outliers for ABOD

Upon analyzing Figure 3, it becomes evident that the ABOD model successfully identified the outliers, primarily locating them at the periphery of the data space, although not exclusively, like the IF model. Both models employ indeed a similar approach: Isolation Forest constructs isolation trees, while ABOD captures point angles and detect outliers based on these angles.

1.3.4 Lightweight on-line detector of anomalies (LODA)

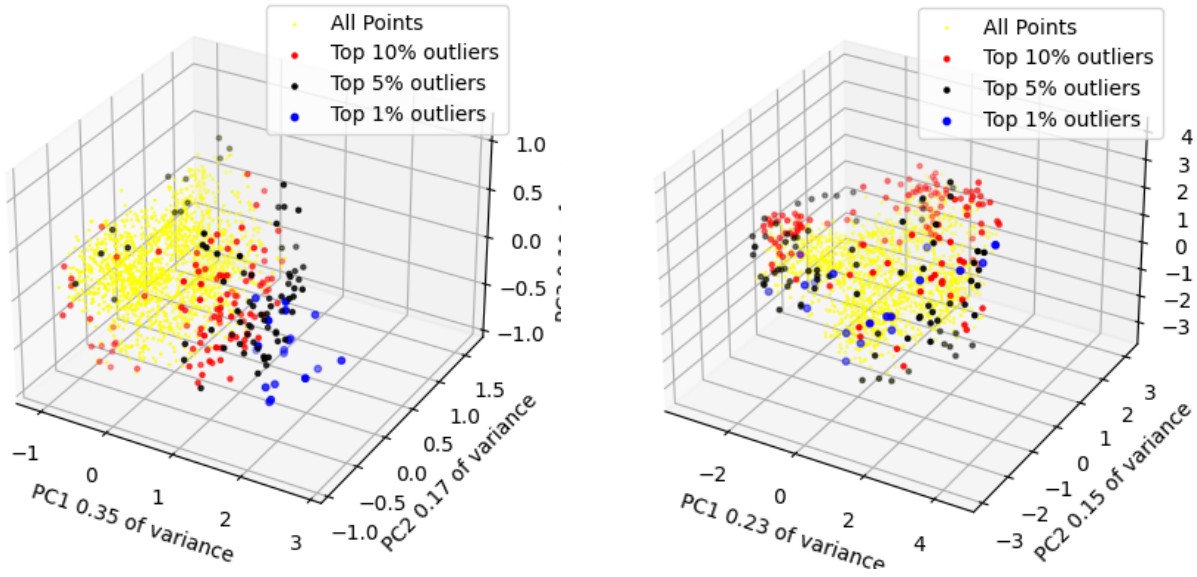
We used the LODA model, also from the PYOD library, to get an insight into a new method that uses an ensemble of very weak anomaly detectors, specifically one-dimensional histograms, which can result in a strong anomaly detector with performance equal to or better than state-of-the-art methods.

We compared two key parameters: the number of bins and the number of random cuts. We found that using 100 cuts and setting the number of bins to "auto" yielded the best results. We then analyzed the results of different outlier thresholds (1%, 5%, and 10%).

After training a classifier on the cleaned datasets obtained, we observed the results presented in Table 7. The analysis revealed that the DT model performed the best while removing the outliers, particularly when 10% of them were removed. This led to an increase in accuracy by 2 percentage points. Conversely, the KNN model demonstrated consistent performance across different outlier removal thresholds, exhibiting minimal variation in results.

Table 7: Models Performance over LODA Thresholds

Threshold Elim.	Obs.	Train KNN	Test KNN	Train DT	Test DT
1.0	18	0.7486	0.766	0.713	0.717
5.0	91	0.7443	0.764	0.718	0.709
10.0	182	0.7369	0.758	0.695	**0.721**
NO	0	0.751	**0.766**	0.703	0.708



(a) PCA representation of outliers on K-nn dataset

(b) PCA representation of outliers on D.T. dataset

Figure 4: PCA representation of outliers for LODA

When examining Figure 4, we can see that all the points, for both datasets, are located in the less dense regions of the PCA representation of the data, LODA identified in fact the outliers similarly to LOF.

1.4 Outlier detection Conclusions

A notable trend has emerged from our anomaly detection task, indicating that the KNN (K-Nearest Neighbors) model consistently achieved high accuracy without surpassing the baseline. Only a few models came close to matching the accuracy of the baseline KNN model, and this was mainly when removing just 1% of the outliers. This pattern strongly suggests that the KNN model is highly resilient to outliers. On the other hand, the decision tree model performed well overall, with accuracy scores consistently surpassing the baseline, except in the case of the IF (Isolation Forest) algorithm. This demonstrates that the decision tree model is more sensitive to outliers and can greatly benefit from a well-designed anomaly detection and removal task in our dataset. After conducting hyperparameter tuning and removing 10% of the outliers, the ABOD (Angle-Based Outlier Detection) algorithm performed the best in conjunction with the decision tree model. It achieved an accuracy of 0.745, which is 4% higher than the baseline accuracy of 0.708.

1.5 Unbalanced classification

We then proceeded with the unbalanced classification task, we used the same target variable "emotional_intensity" as in the previous section eliminating most of the observations in the positive class until they represented 4% of the distribution in the train set, for the features selected for both the datasets.

We evaluated the model's performance in the cross-validation phase using the average of the recall for each class which we'll call **balanced accuracy**.

We first tried classification without applying any re-balancing techniques with fine-tuning through cross-validation. From Table 9 and Table 8 we can notice how the Decision Tree model outperformed the K-nn model on all measures. And particularly in the precision over the underrepresented class.

Table 8: Performance of KNN Model on Unbalanced Dataset

KNN	Precision	Recall	F1 Score
Class 0	0.958	0.952	0.955
Class 1	0.059	0.067	0.062
Bal. Acc		0.509	
Accuracy		0.914	

Table 9: Performance of DT Model on Unbalanced Dataset

DT	Precision	Recall	F1 Score
Class 0	0.962	0.988	0.975
Class 1	0.333	0.133	0.190
Bal. Acc		0.560	
Accuracy		0.951	

1.5.1 Random balancing

The first algorithms we will present are the random ones, randomly undersampling the majority class and randomly oversampling the minority one using the RandomUnderSampler and RandomOverSampler from the imblearn library. We then found the best hyperparameters through cross-validation on the balanced data and tested them on the test set.

Table 10: Performance of KNN Model on Random Undersampled dataset

KNN	Precision	Recall	F1 Score
Class 0	0.65	0.95	0.77
Class 1	0.88	0.40	0.55
Bal. Acc.		**0.67**	

Table 11: Performance of DT Model on Random Undersampled dataset

DT	Precision	Recall	F1 Score
Class 0	0.67	0.62	0.64
Class 1	0.59	0.65	0.62
Bal. Acc.		**0.63**	

Table 12: Performance of KNN Model on Random Oversampled dataset

KNN	Precision	Recall	F1 Score
Class 0	0.58	0.96	0.72
Class 1	0.80	0.20	0.32
Bal. Acc.		**0.57**	

Table 13: Performance of DT Model on Random Oversampled dataset

DT	Precision	Recall	F1 Score
Class 0	0.57	0.92	0.71
Class 1	0.68	0.20	0.31
Bal. Acc.		**0.55**	

From Table 10 and Table 11 we can see that applying the undersampling on the class 0 we addressed the imbalance problem obtaining results that are more in line with the classification task performed prior to synthetically unbalancing the class 1 of the target variable. The values, both for KNN and DT, are improving for class 0 and dropping for class 1 indicating that we mitigated the bias towards class 1, we can also see that the K-nn model outperformed the D.T achieving a B.A. of 0.67.

As we can see from Table 12 and Table 13 that the oversampling technique performed worse than the undersampling technique in our case on all measures only achieving a B.A. of 0.57 on the K-nn model.

Undersampling can lead to a loss of information and may not fully represent the true distribution of the majority class, potentially affecting the model's ability to generalize to unseen data. While on the contrary, oversampling eliminates the potential loss of information but introduces duplicate instances. These duplicates can cause the model to give excessive weight to certain patterns. Random undersampling is generally more suitable when the dataset is already small, as it minimizes the potential loss of information. Conversely, if the class imbalance is significant, random oversampling may be a better choice to increase the representation of the minority class and better capture the data's distribution.

1.5.2 Condensed Nearest Neighbor (CNN)

Next, we employed the Condensed Nearest Neighbour method to explore a more intricate algorithm that takes into account the significance of the selected features for undersampling the class 0 instances. Unlike

the random techniques, this method does not assign equal weight to the classes.

In our experiment, we tested various values for the number of neighbors (nn) from 5 to 100. For the DT dataset, the best nn was 8 with 62 selected records for class 1. For the KNN dataset, the optimal nn was 20 with 60 selected records in class 1. We then fine-tuned the classification models using these datasets.

Table 14: Performance of KNN Model on CNN balanced dataset

KNN	Precision	Recall	F1 Score
Class 0	0.65	0.83	0.48
Class 1	0.71	0.48	0.57
Bal. Acc.		**0.655**	

Table 15: Performance of DT Model on CNN balanced dataset

DT	Precision	Recall	F1 Score
Class 0	0.72	0.61	0.66
Class 1	0.61	0.72	0.66
Bal. Acc		**0.666**	

Based on the findings presented in Table 14 and Table 15 , we can see that the CNN model demonstrated a clear improvement over the baseline performance using unbalanced data. For the D.T. classifier, it surpassed the results obtained through random oversampling while not getting better results than undersampling in terms of B.A. but achieving a more balanced recall for the two classes, even having a higher recall for the minority class. While it didn't achive a good performance for the K-nn model getting worst B.A. results than random undersampling.

1.5.3 Synthetic Minority Over-sampling Technique (SMOTE)

We used the Synthetic Minority Over-sampling Technique (SMOTE) to oversample the training set. We tested different values for the number of neighbors (5, 10, 20) and the percentage of observations to oversample (0.5, 0.75, 1). For each combination of parameters of the SMOTE algorithm performed hyperparameter tuning for both classifiers using a 5-fold cross-validation grid search approach.

Table 16 and Table 17 show the performance comparison of different rebalancing techniques. The results indicate that the SMOTE technique had the worst performance on the D.T. classifier. It had a significantly higher recall for the majority class but a lower Balanced Accuracy compared to the CNN rebalancing technique. However, the SMOTE technique outperformed all other rebalancing techniques when applied to the K-nn classifier. It achieved a Balanced Accuracy of 0.68, with only a small difference of 0.03 between the recalls. This outcome can be attributed to the similarity between the K-nn classification algorithm and the oversampling method of SMOTE. Both approaches consider the nearest neighbors, leading to the creation of dense regions of minority observations in the feature space near other minority observations.

Table 16: Performance of KNN, NN 20, Samp 100%, on SMOTE balanced dataset

KNN	Precision	Recall	F1 Score
Class 0	0.979	0.699	0.816
Class 1	0.090	0.666	0.158
Bal. Acc		0.68	

Table 17: Performance of DT NN 10, samp. 75% on SMOTE balanced dataset

DT	Precision	Recall	F1 Score
Class 0	0.592	0.967	0.734
Class 1	0.853	0.222	0.353
Bal. Acc.		0.594	

1.5.4 Outlier Detection Techniques

We then proceeded to do unbalanced classification with the isolation forest outlier detection technique. We trained the outlier detection technique on the training data setting a contamination value equal to the percentage of observations in the minority class. We then used the model to predict the observations in the test data considering the observation predicted as outliers as belonging to the minority class.

From Table 18 it's possible to see that the outlier detection technique performed worst than the other rebalancing technique with an extreme difference in the recalls of the two classes.

Table 18: Performance of I.F. outlier detection techniques on unbalanced dataset

Isolation Forest	Precision	Recall	F1 Score
Class 0	0.96	0.97	0.96
Class 1	0.15	0.13	0.14
Balanced Accuracy Test set		0.550	

2 Module 2: Advanced Classification

We then applied multiple classification models to the emotions classification task, we used the dataset as preprocessed for the Decision Tree classifier. The 8 classes of emotions are not distributed equally as disgust, surprise, and neutral have half the observations as the other emotions, and they are correlated to other measures as there are no neutral observations with strong intensity and no surprised and disgust observations that are sung.

2.1 Logistic Regression

We used scikit-learn's Logistic Regression classifier to tackle the task. Our approach involved training a multinomial logistic regression model with cross-entropy loss. This model predicts class probabilities and assigns observations to the class with the highest probability. With 379 features and 8 classes, the model estimated a total of 3040 parameters. To mitigate overfitting, we explored L1 and L2 regularization as well as different feature selection methods.

To optimize performance, we performed 5-fold cross-validation and tested 100 different values of the C coefficient (inverse of regularization strength) ranging from 0.01 to 10, evenly spaced on a logarithmic scale. We considered L1, L2, and no regularization.

Initially, we trained the Logistic Classifier using all 379 features. However, due to multicollinearity, we then trained the model using subsets of uncorrelated features based on various correlation thresholds (0.7, 0.8, 0.9, and 0.99). We also tested the model using feature subsets selected by recursive feature elimination with feature importance from a random forest model, aligning with the K-nn model's feature selection process.

To address overfitting, we performed feature selection by selecting features with non-zero coefficients after L1 regularization across all classes. We tried different C coefficient values (1, 2, 5, 7, and 10) and evaluated model performance using the selected features for each regularization parameter value.

Among the models compared, the Logistic model trained on the complete dataset with L1 regularization and C = 4.64 performed the best on the test set. This model had a total of 2517 non-zero parameters.

From Table 19 we can see that the classifier encountered difficulties in accurately classifying sad emotions. However, From Figure 6a we can see that the worst ROC curves where for the happy, fearful and sad emotions.

2.2 Neural Network

We attempted to solve the classification task using a feed-forward multi-layer NN built using the Keras API in TensorFlow.

We chose to use the Adam optimizer and all 379 features as input for the neural network. After some preliminary testing, we decided to use a tanh activation function and Glorot Normal initialization. For the last hidden layer, we utilized a softmax activation function and categorical cross-entropy loss function. Additionally, we incorporated L1 regularization and dropout in each layer. During the experimentation phase, we explored various network architectures and different values for the learning rate, dropout rate, L1 regularization, batch size, beta_1, and beta_2 parameters.

For the training of each model, we set the maximum number of epochs to 600. We implemented Early Stopping based on the validation accuracy, utilizing 10% of the training observations to mitigate over-fitting and to restore the weights corresponding to the best validation accuracy achieved by the model. To determine

Table 19: Performance of the best Logistic model on the test set

Class	Precision	Recall	F1-score
angry	0.61	0.78	0.69
calm	0.55	0.68	0.60
disgust	0.46	0.73	0.56
fearful	0.60	0.35	0.44
happy	0.41	0.52	0.46
neutral	0.55	0.44	0.49
sad	0.47	0.23	0.31
surprised	0.52	0.48	0.50
Accuracy			0.52
Balanced Accuracy			0.53
Mean AUC			0.88

Table 20: Performance of the best NN model on the test set

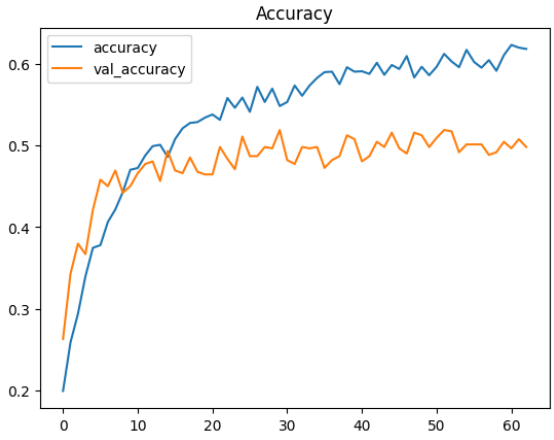
Class	Precision	Recall	F1-Score
angry	0.61	0.74	0.67
calm	0.53	0.67	0.59
disgust	0.57	0.56	0.57
fearful	0.69	0.36	0.48
happy	0.45	0.43	0.44
neutral	0.46	0.56	0.50
sad	0.42	0.33	0.37
surprised	0.50	0.62	0.56
Accuracy			0.52
Balanced Accuracy			0.53
Mean AUC			0.88

the optimal combination of the remaining hyperparameters, we conducted a 5 folds C.V. Grid search with the following values, the ones in bold are the optimal values found.

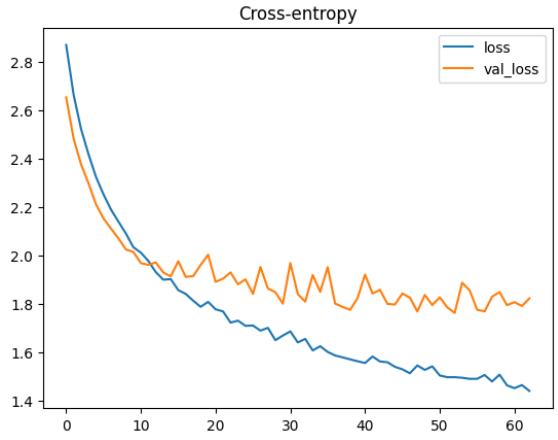
- Network Topology: **(60,60)**, (60, 10), (10, 10)
- Learning rate: **0.001**, 0.005, 0.01
- Dropout rate: **0.2**, 0.5, 0.7
- Lambda parameter of L1 regularization: 0, 0.001, 0.005, **0.0005**
- Beta_1: **0.9**, 0.99, 0.8
- Beta_2: 0.999, **0.995**, 0.990
- Batch Size: **128**, 256

Based on Table 20, it is evident that the NN model exhibited similar performance to the logistic model across all evaluation metrics. However, like the logistic classifier, the NN model encountered challenges in accurately classifying sad emotion, while having a better performance on it, and had the worst ROC curves where the same, for fearful, happy and the sad emotions .

Looking at Figure 5 we can see that the model early stopped after 60 epochs and that the validation loss and accuracy diverged from the training accuracy after only 10 epochs, but the validation accuracy and loss still kept getting better even if at a lower rate than the training one indicating that we haven't over-fitted the model which we believe is caused by the early stopping introduced in the model. We can also see that the loss curve is smooth indicating that the learning rate step is not so large that it causes abrupt changes in the loss function.

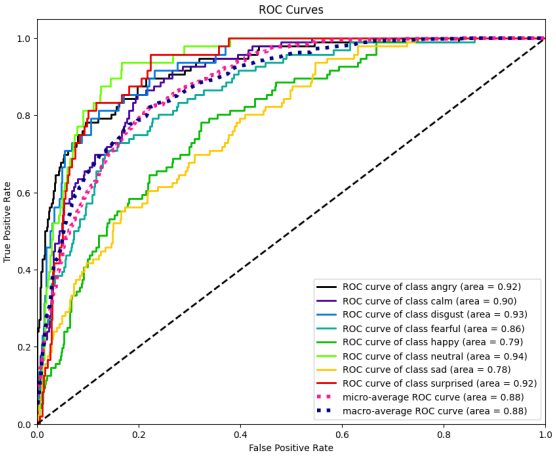


(a) Accuracy of the NN classifier during training

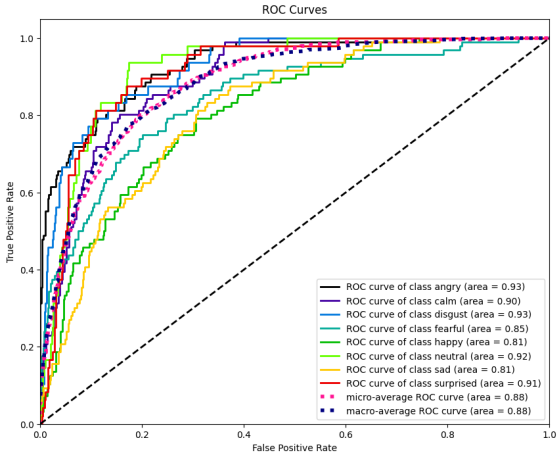


(b) Loss of the NN during training

Figure 5: Loss and Accuracy of the NN on train and validation set during training



(a) ROC plot for the Logistic Classifier



(b) ROC plot for NN classifier

Figure 6: ROC plots for Logistic and NN classifiers

2.3 Support Vector Machines

We applied the Support Vector Machines (SVM) algorithm to the classification task. We experimented with both linear and non-linear kernels to determine their effectiveness in separating different emotion classes. We started with the dataset containing 379 features and first applied an SVM with a linear kernel on the dataset.

We conducted a 5-fold CV Grid search with the following values, the ones in bold are the optimal values found, achieving an accuracy of **0.48** on the test set.

- C: 0.01, 0.1, 1, 10, **100**
- max_iter: **1000**, 5000, 10000

We then tried the SMV with a non-linear SVM Kernel. We conducted a 5-fold CV Grid search with the following values, the ones in bold are the optimal values found, achieving an accuracy of 0.51 on the Test

set.

- C: 0.01, 0.1, 1, 10, **100**
- kernel: **rbf**, poly, sigmoid
- gamma: scale, **auto**
- max_iter: **1000**, 5000, 10000

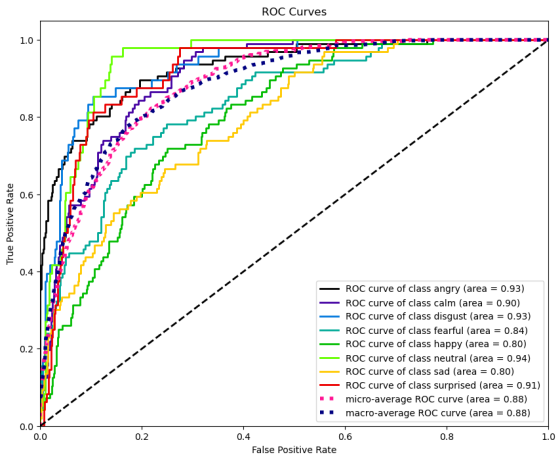
For the non linear SVM we can see from both Table 22 and Figure 7a, which contains the ROC curve, that the SVM model struggled to classify the happy and sad emotions. While we can see from Table 21 and Figure 7b that the linear SVM struggled in classifying sad, same as the nonlinear kernel, along with neutral, while the ROC plot the classes with the lowest curves are the same as in the nonlinear SVM, while achieving the same mean AUC score. We can also see that using a nonlinear kernel improved the accuracy of 3.

Table 21: Performance of SVM with Non-Linear kernel on Test data

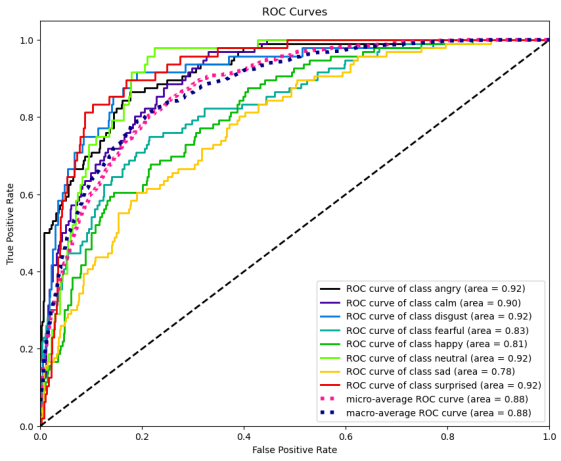
Class	Precision	Recall	F1-score
angry	0.56	0.80	0.66
calm	0.54	0.69	0.61
disgust	0.52	0.71	0.60
fearful	0.52	0.35	0.42
happy	0.39	0.38	0.38
neutral	0.44	0.42	0.43
sad	0.56	0.31	0.40
surprised	0.46	0.40	0.43
Accuracy			**0.51**
Balanced Accuracy			0.51
Mean AUC			0.882

Table 22: Performance of SVM model with Linear kernel on test data

Class	Precision	Recall	F1-score
angry	0.51	0.77	0.62
calm	0.52	0.66	0.58
disgust	0.44	0.58	0.50
fearful	0.55	0.35	0.43
happy	0.45	0.47	0.46
neutral	0.36	0.33	0.34
sad	0.44	0.24	0.31
surprised	0.50	0.38	0.43
Accuracy			0.48
Balanced Accuracy			0.47
Mean AUC			0.87



(a) ROC plot for SVM with non-linear kernel



(b) ROC plot for SVM with linear kernel

Figure 7: ROC plots for SVM classifiers

2.4 Ensemble Methods

2.4.1 Bagging

We then applied a Bagging model to our task. We conducted a 5-fold CV Grid search with the following values, the ones in bold are the optimal values found, achieving an accuracy of 0.5 on the test set.

- base_estimator__max_depth: 2, 4, 6, 8, **10**, 15
- base_estimator__min_samples_split: **2**, 5, 10, 50
- n_estimators: 50, 100, **150**, 200

From Table 23 and Figure 9a we can see that it too struggled in classifying the sad and fearful emotions. Having a similar ROC pattern as the previous classifiers.

From Figure 8, we can see the learning curve of the classifier trained on a small subset of the original training set to reduce computation time. Initially, the Bagging model shows strong overfitting tendencies, as indicated by a training accuracy of 1. As the number of training instances increases, we observe a reduction in training accuracy, indicating improved generalization capability. The cross-validation score, on the other hand, gradually increases as more training instances are used. However, it reaches a plateau at approximately 600 training instances, achieving an accuracy of 0.38. This suggests that there is limited improvement in generalization beyond this point.

Table 23: Performance of Bagging on Test set

Class	Precision	Recall	F1-score
angry	0.58	0.75	0.65
calm	0.55	0.73	0.63
disgust	0.50	0.46	0.48
fearful	0.58	0.27	0.37
happy	0.39	0.45	0.42
neutral	0.60	0.50	0.55
sad	0.34	0.29	0.31
surprised	0.48	0.52	0.50
Accuracy			**0.50**
Balanced Accuracy			0.49
Mean AUC			0.8775

Table 24: Performance of AdaBoost on Test set

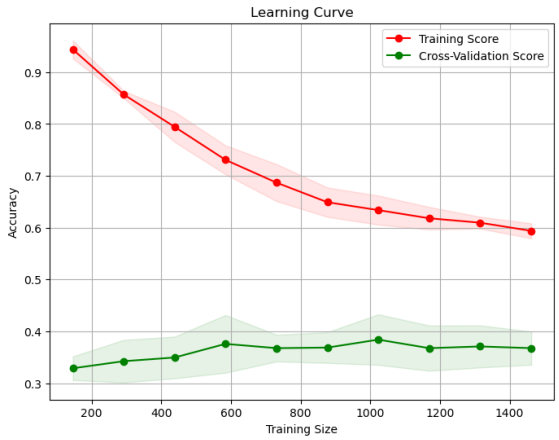
Class	Precision	Recall	F1-score
angry	0.58	0.70	0.63
calm	0.49	0.48	0.48
disgust	0.38	0.25	0.30
fearful	0.51	0.36	0.42
happy	0.27	0.40	0.32
neutral	0.43	0.12	0.19
sad	0.34	0.42	0.38
surprised	0.43	0.40	0.41
Accuracy			0.42
Balanced Accuracy			0.39
Mean AUC			0.835

2.4.2 AdaBoost

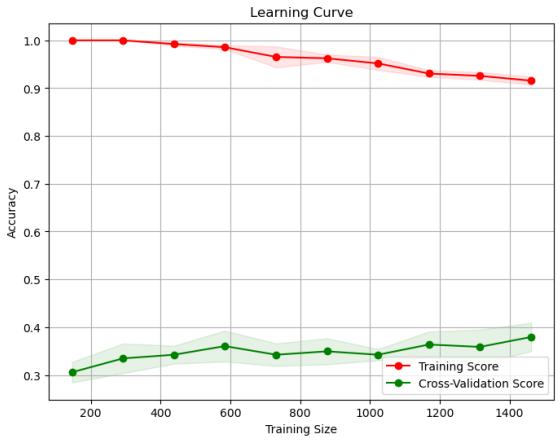
We then tackled the classification problem using the Adaboost classification model. We conducted a 5-fold CV Grid search with the following values the ones in bold are the optimal values found which achieved accuracy on the test set of 0.42.

- base_estimator__max_depth: 2, 4, 6, 8, **10**
- base_estimator__min_samples_split: 2, 5, **10**
- n_estimators: 50, **100**, 150, 200

From Table 24 and Figure 9b that the AdaBoost model struggled to classify the sad, like most classifiers, and neutral disgust and happy, performing worst than any other model so far with an accuracy of 0.42. From the ROC plot, we can see that we achieved the worst MAUC so far 0.835 with the worst emotions being happy sad, and fearful as in the other classifier.



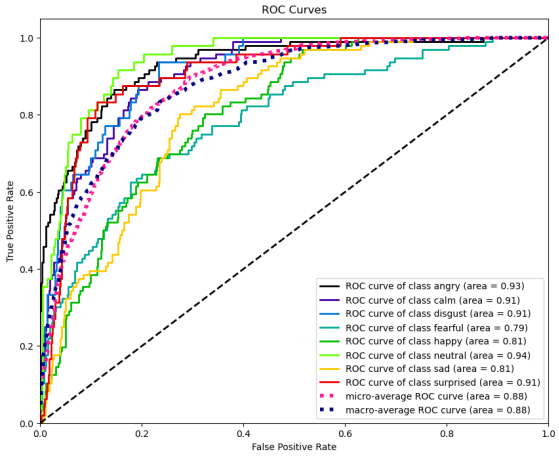
(a) Learning Curve of Bagging Method



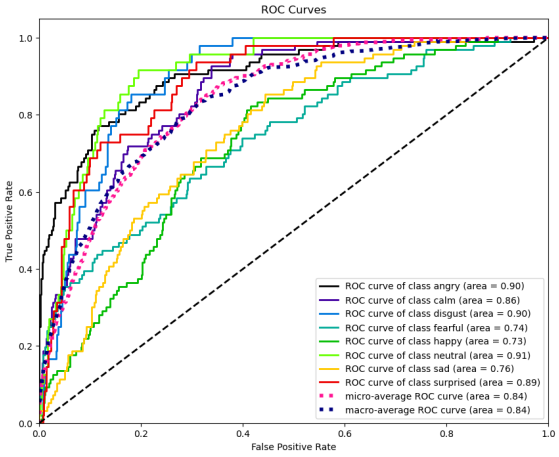
(b) Learning Curve of Boosting Method

Figure 8: Learning Curves of Bagging and Boosting Methods

From Figure 8, we can see the learning curve of the classifier trained on a small subset of the original training set to reduce computation time. Initially, the AdaBoost model shows strong overfitting tendencies, as indicated by a training accuracy of 1. As the number of training instances increases, we don't observe a gradual reduction in accuracy as in Bagging, indicating that the model is overfitting the data. The cross-validation score, on the other hand, gradually increases as more training instances are used same as Bagging.



(a) ROC plot for Bagging classifier



(b) ROC plot for AdaBoost

Figure 9: ROC plots for Bagging and Boosting classifiers

2.4.3 Gradient Boosting Machine

As our last ensemble method we employed the GBM classifier. Since it's an ensemble method of the decision tree classifier we performed a basic DT classification on the data for different parameters of the tree depth to set a baseline. We then proceeded to set up a grid search to optimise the GBM model with the following values, the optimal ones are in bald achieving an accuracy of 0.52 on the test set.

- `n_estimators`: 10, 20, 50, **100**
- `subsample`: 0.5, **0.75**, 1.0
- `learning_rate`: **0.1**, 0.01, 0001, 00001
- `validation_fraction`: 0.05, 0.10, **0.15**
- `max_depth`: 3, 4, 5, 6, **7**

From Table 25 and Figure 14 we can see that it too struggled to classify the sad, fearful, and happy emotions while having the same ROC pattern has the other classifiers.

Table 25: Performance of GBM on Test set

Class	Precision	Recall	F1-score
angry	0.62	0.83	0.71
calm	0.60	0.60	0.60
disgust	0.49	0.60	0.54
fearful	0.51	0.27	0.35
happy	0.34	0.44	0.39
neutral	0.67	0.42	0.51
sad	0.35	0.33	0.34
surprised	0.47	0.42	0.44
Accuracy			0.516
Balanced Accuracy			0.428
Mean AUC			0.86

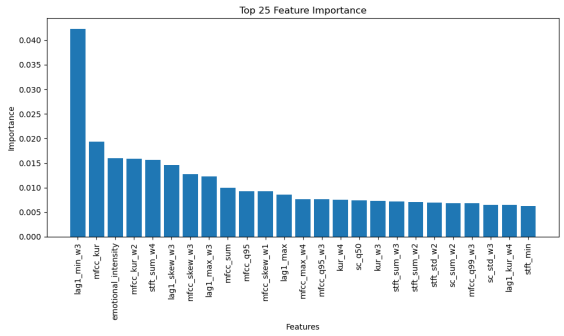
Table 26: Performance of Random Forest on Test set

Class	Precision	Recall	F1-score
angry	0.57	0.71	0.63
calm	0.47	0.66	0.55
disgust	0.49	0.54	0.51
fearful	0.54	0.26	0.35
happy	0.41	0.54	0.46
neutral	0.57	0.27	0.37
sad	0.37	0.25	0.30
surprised	0.46	0.52	0.49
Accuracy			**0.47**
Balanced Accuracy			0.47
Mean AUC			0.835

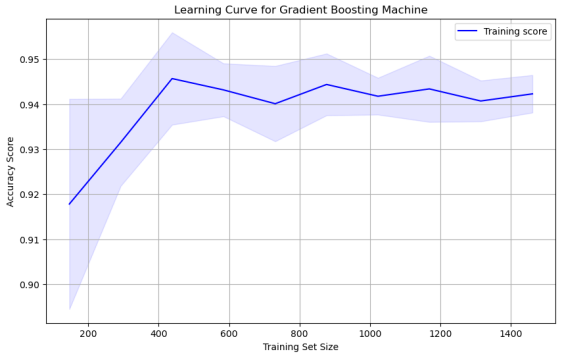
We chose a relatively low number of estimators (trees) due to the larger learning rate, as these parameters are interdependent. The choice of subsample parameter helped balance computational time and performance, while also adjusting the number of trees and learning rate.

For maximum depth values, we experimented with 1 to 10 and found that 7 produced better results. We mitigated overfitting risks by randomly selecting subsets of the dataset for each tree, despite using a high learning rate and large max depth.

To prevent overfitting, we used a 0.15 validation fraction during training. Previous tests without a validation fraction resulted in severe overfitting, but our final model, shown in Figure 10b, exhibits a lesser degree of overfitting with potential for further improvement.



(a) GBM top 25 feature Importance



(b) GBM learning curve

Figure 10: GBM learning curve and top features

Figure 10a reveals that among the top 25 features for the GBM classifier, there are both categorical and numerical features. This distinguishes it from the behavior observed in the importance of features (IF), where only numerical features are considered.

2.4.4 Random Forest

During the Random Forest ensemble method to improve the performance, we conducted a grid search testing the following hyperparameters, the resulting best ones are in bold, achieving accuracy on the test set of 0.47.

- n_estimators: 50, **100**, 200
- max_depth: **None**, 5, 10
- min_samples_split: **2**, 5, 10
- min_samples_leaf: 1, **5**, 10
- max_features: **sqrt**, log2

From table 26 and Figure 15 we can see that with the best parameters Random Forest achieved accuracy on the test set of **0.471** lower than most of the previously seen models, struggling to classify the sad, fearful and neutral emotions, having once again a similar pattern on the ROC curve.

We plotted the feature importance for the top 25 important features using the Mean Decrease Impurity measure and the Permutation importance. We observed a bias towards high cardinality features, with all the top 25 features being numerical in both measures indicating that the categorical variables have less predicting power. The highest feature importance on the MDI index was **0.011** for **lag1_q01_w3** and **0.048** for **mfcc_std_w3**, while the most important features for the permutation index were **q01_w3** and **mfcc_mean_w3**. It's interesting to notice that the permutation and MDI index selected a very different subset of features.

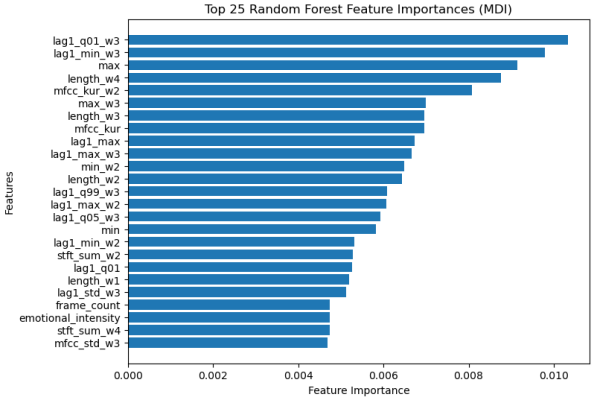


Figure 11: RF Feature Importance

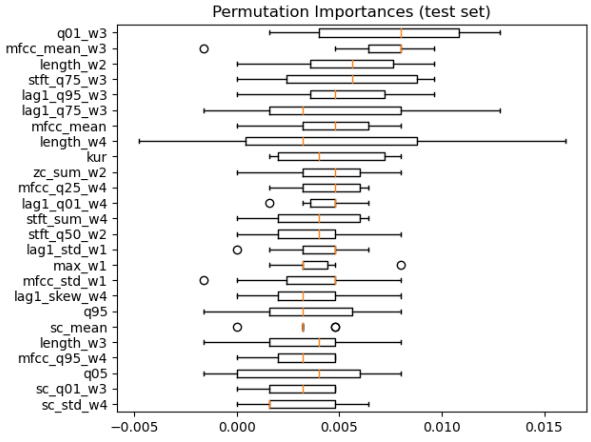


Figure 12: RF Permutation Importance

Figure 13: Random Forest Feature importance

2.5 Model Comparision with ROC

From the previous analysis, we can see that the Logistic, NN, and GBM achieved the best accuracy of 0.52 closely followed by nonlinear SVM and bagging with 0.51 and 0.50. And we can say that the Adaboost performed the worst with an accuracy of 0.42.

From Figures 6, 7, 9, 14 and 15 we can see that all the models followed the same pattern in ROC curves with the sad, fearful and happy emotions having a curve lower than the average, and we can also note that the nN classifier was the one with the more balanced curves closer to the mean.

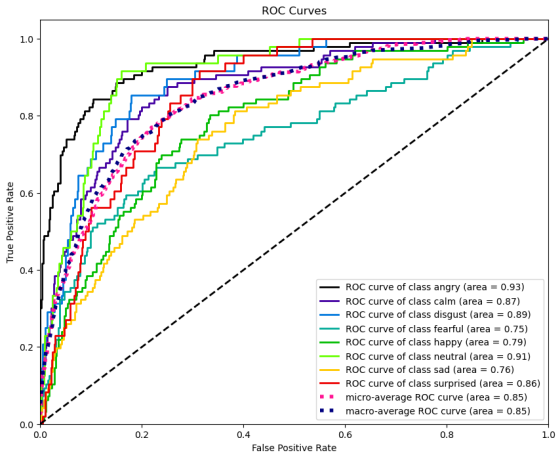


Figure 14: ROC plot for Gradient Boosting Machine

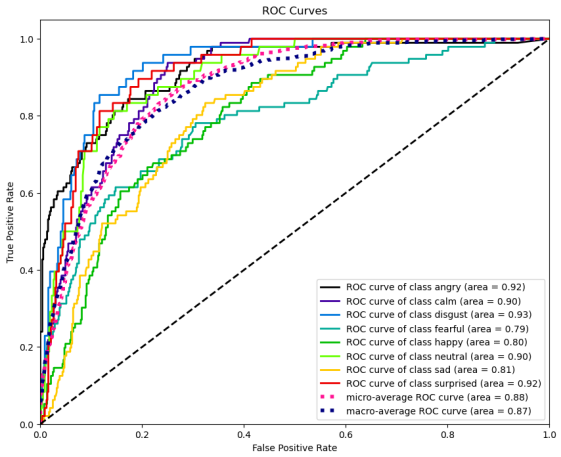


Figure 15: ROC plot for Random Forest Classifier

Figure 16: ROC plots for GBM and RF models

3 Module 2 Advanced Regression

The main difficulty we found in choosing our target variable for the advanced regression task was that as we previously said most numerical features are highly correlated amongst each other since they are features extracted from the same audio files.

We selected "std" as our target variable, which serves as a proxy for intensity. Using a simple linear model, we obtained an R-squared value of 0.9997, a Mean Absolute Error (MAE) of 0.00116, and a Root Mean Square Error (RMSE) of 0.00245.

To have a more challenging regression task, we decided to eliminate all variables that had an absolute correlation higher than 0.5 with our target variable. This resulted in a dataset of 316 features. After this feature selection step, our model achieved an R-squared value of 0.8477, an RMSE of 0.0577, and an MAE of 0.03738. We considered this model as our benchmark.

Subsequently, we trained advanced classifiers using this dataset.

3.1 Random Forest Regression

We tried to use the Random Forest regression algorithm on the task. From preliminary testing we immediately saw that it consistently reached a worst performance than the linear regressor, but we choose to use the following parameters on a 5 folds grid search to try and achieve the best performance possible.

- 'n_estimators': 100, **200**, 400
- 'max_depth': 5, **10**, 20,
- 'min_samples_split': 2, **20**, 50,
- 'min_samples_leaf': 1, **20**, 50.

We choose to try a minimum depth of 5 since we have a fairly large dataset of 1800 observations and for this reason we choose the minimum number of samples per leaf and split as 50. We choose to use the squared error as criterion since it was our target parameter to compare this model to the other.

The best model had 200 estimators with 20 min samples for a leaf and a split and a max depth of 10. Achiveing a $R\hat{2}$ of 0.7572, a MAE of 0.0418 and a RMSE of 0.0729 with a significantly worst performnce than the simple linear model.

The most important features for the R.F. model on both permutation and impurity importance where the angry emotion, the emotional intensity, which are self explanatory has the observations belonging to those classes have the highest intensity, mfcc_q99, and mfcc_mean_w2.

3.2 Neural Network

We then Choose to use a NN on the task, we choose to use the Adam optimiser. After some experimentation we found the working range of parameters and fount the relu activation function to be far more performing than any other activation function. We also choose to give each model 900 epochs max and to early stop them with a patience of 30, higher than the previouse model. We then used a batch size of 500, and gloroth normal initialization.

We choose to run a 5 folds grids search routine on the following parameters

- Learning rate: **0.001**, 0.0025, 0.0001
- Beta 1: **0.99**, 0.999
- Beta 2: **0.999**, 0.995
- L1 reg: 0, 0.001, **0.0001**
- Hidden Layer Size: **(60, 60)**, (10, 10)
- Dropout Rate: 0.5, 0.2, **0**

The best NN model had a learning rate of 0.001, a Beta 1 of 0.99 a Beta 2 of 0.999 a L1 regularization strength of 0.0001 a 0 dropout rate a a two 60 nodes layers topology.

When trainig the network on the whole trainig set we saw that it hadn't early stopped and so we decided to augment the maximum number of epochs to 2000, the model early stopped on epoch 1751 achieving a $R\hat{2}$ of 0.9672, a RMSE of 0.02679, and a MAE of 0.0167. This model had a much better performance than both the R.F. model and the benchmark arriving close to the performace of the linear model trained on all features.

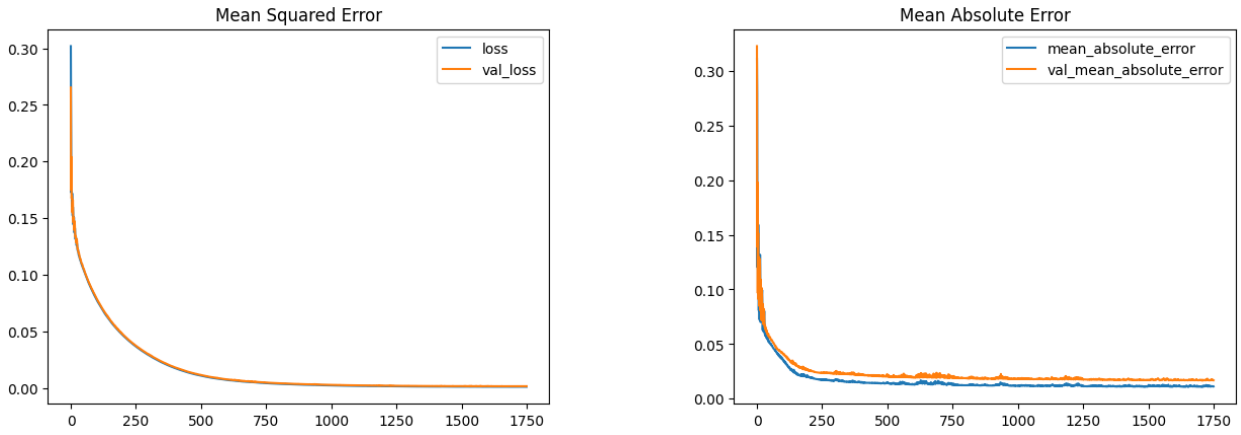
From Figure 17 we can see that we have a smooth learning curve indicating a good learning rate and momentum for the model, and we can also see that there is a difference between the MAE over the train and validation set, but we can also see that they are both decreasing with the epochs so we are not incurring in overfilling for this model.

3.3 Model comparison

It's clear than NN and Linear model strongly outperformed the RF model, and we believe that this could be caused by the fact that the features are all statistics extracted from the same audio file and so a linear (or non linear in the NN case) combination of them is much more effective in predicting our target variable than a segmentation of the feature space as in the case of the R.F. model.

4 Module 3 Time series analyis

In this section of our work, we changed the database and used the original audio files of the RAVDESS dataset.



(a) Train and validation MSE over training

(b) Train and validation MAE over training

Figure 17: Learning curve of Regression NN

4.1 Preprocesing

We first converted the audio files as NumPy arrays and split them into test and train sets. We then extracted the categorical features of the audio files and split them again in train and test set.

We then found the longest time series in the training dataset and padded each of the shorter observations with zeros and truncated the longer observations in the test set. Ending with a dataset of time series of length 304304.

4.2 Motif and anomaly discovery

First, we began by selecting the initial observation from the training set. To remove the sections of the sequence that mostly contained values of 0, we looked for a window of 100 samples where the mean value exceeded the mean of the entire time series on both sides. We truncated the sequence at the indices of these two windows. As a result, we obtained a sequence consisting of 6000 samples c.a.

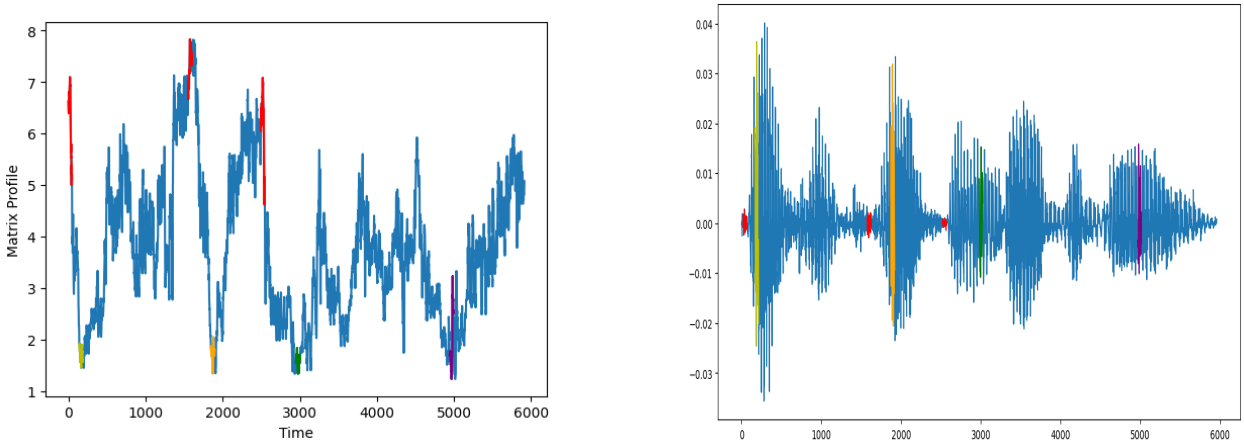
Next, we used the STOMP algorithm from the Stumpy library to calculate the matrix profile of the time series. We experimented with different window sizes and found that a window of 50 samples yielded the most intriguing results.

From Figure 18a we can see that there are 4 regions of the audio file where we have local minima in the matrix profile and we choose to pick one motif from each of them. From Figure 19 we can see the motifs in a close-up. And we can notice that while the motifs are not similar to each other they represent local recurrent patterns in the time series. We can also see that there are 3 regions in the matrix profile that are higher than the others and we choose them as anomalies. From figure 20 we can see that these three anomalies are random patterns of noise and that two of them are in sections of low intensity of the audio.

From the plot of the matrix profile, we can see that there are 3 regions where we have particularly high local maxima, these are represented in red in Figure 18a. From their close-up representation, it's possible to see that they are patterns of random oscillations in the data and as we can see from Figure 18b they are present in regions of low intensity or the audio file, which represent pauses between the words spoken in the sentence.

While in contrast, the motifs are in regions of high intensity of the audio, which are also more regular. In particular, all 4 motifs are present in a section of the audio where the actor is saying a vowel.

We then choose to compare the motifs among all audio files, to do so we Min-Max standardised all observations in the dataset, and we extracted the audio parts ignoring the regions where the time series was composed of only 0. We then calculated the matrix profile for each observation with a window of 50 and



(a) Matrix profile

(b) Motifs and anomalies on original audio

Figure 18: Motifs and anomalies on Matrix Profile and original data

selected one motif at the minimum of the matrix profile from each time series.

We then performed SAX using the Symbolic_Aggregateb_Approximation algorithm from pyts, extracting 10 segments from each observation with an alphabet size of 5.

We then selected the 4 patterns with the highest support:

- eeddccbbaa: Support 39
- aabbccddee: Support 37
- ecabddbace: Support 24
- adecbbedca: Support 19

Figure 21 shows the plot of 10 motifs from each of the most common SAX sequences. We can see that the first two represent a constant descending and ascending patterns, while the last two represent an oscillating pattern of two and 3 waves.

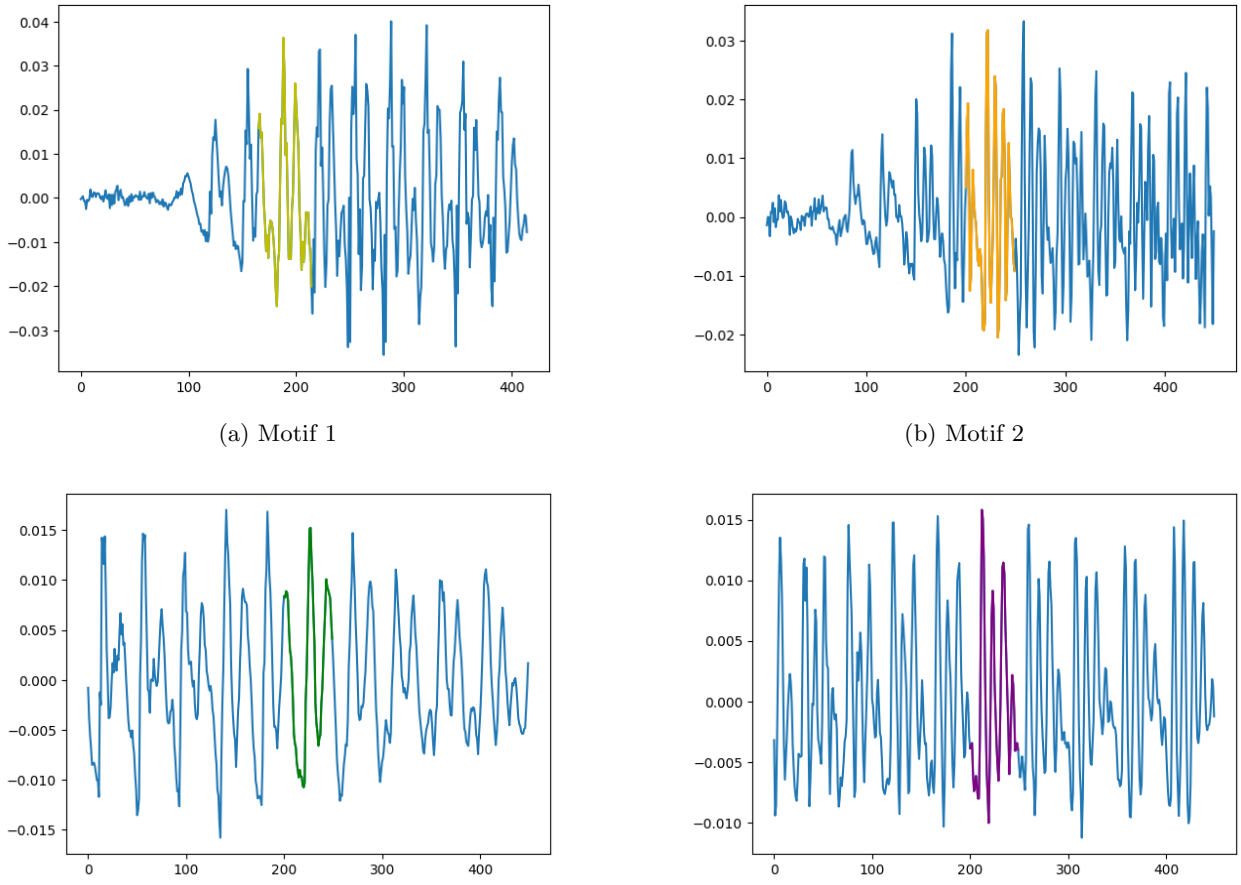
It's also interesting to note that the first two motifs are predominantly found in speech, 66%, 80%, and strong 80%, 66% observations and have both fearful as the most common emotion 50%. While the last two are predominantly found in song observations 66% and 78%. All four motifs are predominantly from male observations, with more than 70% in each.

We can infer that audio files with strong speech have recurrent sudden changes in intensity, while audio files that are sung tend to have more oscillatory patterns. This makes sense as in speech recordings there are pauses between words after which there are sudden spikes of intensity while sung observations have more constant intensity.

4.3 Clustering

4.3.1 Feature-based Clustering

During the preprocessing step, we scale the time series using the TimeSeriesScalerMinMax algorithm from tslearn. We extracted from the audio time series the features: Mel-Frequency Cepstral Coefficients (MFCC), chroma, spectral contrast, tonnetz, zero-crossing rate, and pitch. We then calculated the mean, median, and standard deviation of these features. We then performed k-means clustering with the Euclidean distance as a metric, utilizing the KMeans algorithm from sklearn. To determine the number of clusters, we used the elbow method on within-cluster sum of squares and the silhouette score.



(a) Motif 1

(b) Motif 2

(c) Motif 3

(d) Motif 4

Figure 19: Close up on 4 motifs

From Figure 22 we can see that the silhouette score is maximal at k=2 with a score of 0.53. However, the Elbow method suggests 3 as the elbow point. The Elbow method considers the optimal within-cluster sum of squares (WCSS) and a lower distortion score, indicating that the data points are closer to their cluster centroids. So we choose to try 3 clusters.

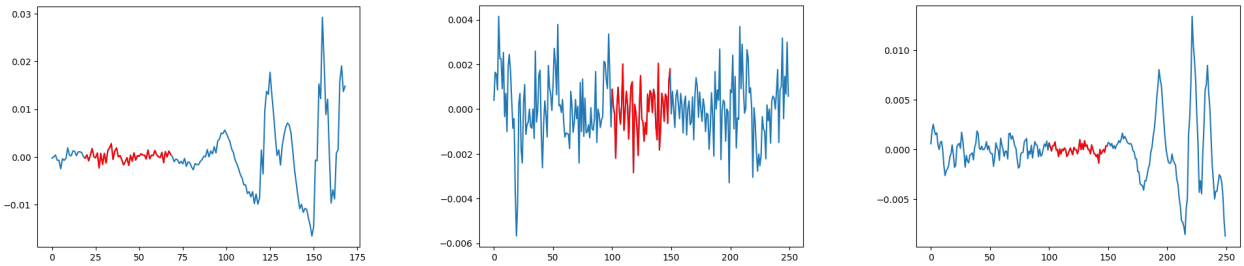
From Tables 27 and 28 we can see that we have two larger clusters of more than 500 observations and one smaller one.

The first cluster, of 636 observations, contains almost twice the number of male observations with respect to females and has more song and normal observations while having the same number of kids and dogs statements.

The second cluster, of 769 observations, is the biggest one containing most of the angry, happy and sad emotions. It's composed by 75% of female observations, with an almost equal number for the other categorical attributes.

The third cluster, of 423 observations, is the smallest one containing almost no angry, disgusted or surprised observations. It is composed of 80% males and has twice the number of songs and normal observations.

We can say that the driving categorical features behind the clustering were sex, with most of the female observations contained in the second cluster and males in the first and third, and angry with 60% of observations being in cluster 2 and happy with 50% of observations being in cluster 2. While the other emotions are more equally distributed over the clusters. The Statement variable in particular was almost equally distributed in the three clusters.



(a) Anomaly 1 (b) Anomaly 2 (c) Anomaly 3

Figure 20: Close up on 3 anomalies

Cluster	Angry	Calm	Disgust	Fearful	Happy	Neutral	Sad	Surprised
Feature-Based Clustering								
Cluster 1 (636)	96	98	54	111	85	56	82	54
Cluster 2 (769)	164	88	57	114	136	37	115	58
Cluster 3 (423)	20	94	33	55	59	47	83	32
Distance-Based Clustering								
Cluster 1 (380)	123	8	6	82	106	4	29	22
Cluster 2 (97)	58	28	0	28	9	0	1	1
Cluster 3 (1351)	99	272	138	170	165	136	250	121

Table 27: Distribution of emotions over the clusters

We then plotted the data in 2 dimensions using PCA and TSNE dimension reduction techniques. From figure 23 we can see that the PCA plot reveals three densely filled and well-separated over the PC1 component. The t-sne plot also shows a clear separation between the clusters. We believe that this clear separation is due to the nature of the features which are extracted from the same audio file and are very correlated to each other as we said before, and so the First principal component explains 90% of the variability and so it demarks the separation between the two clusters, and this clear separation can be seen for the same reason in the t-sne plot.

4.3.2 Distance-based Clustering

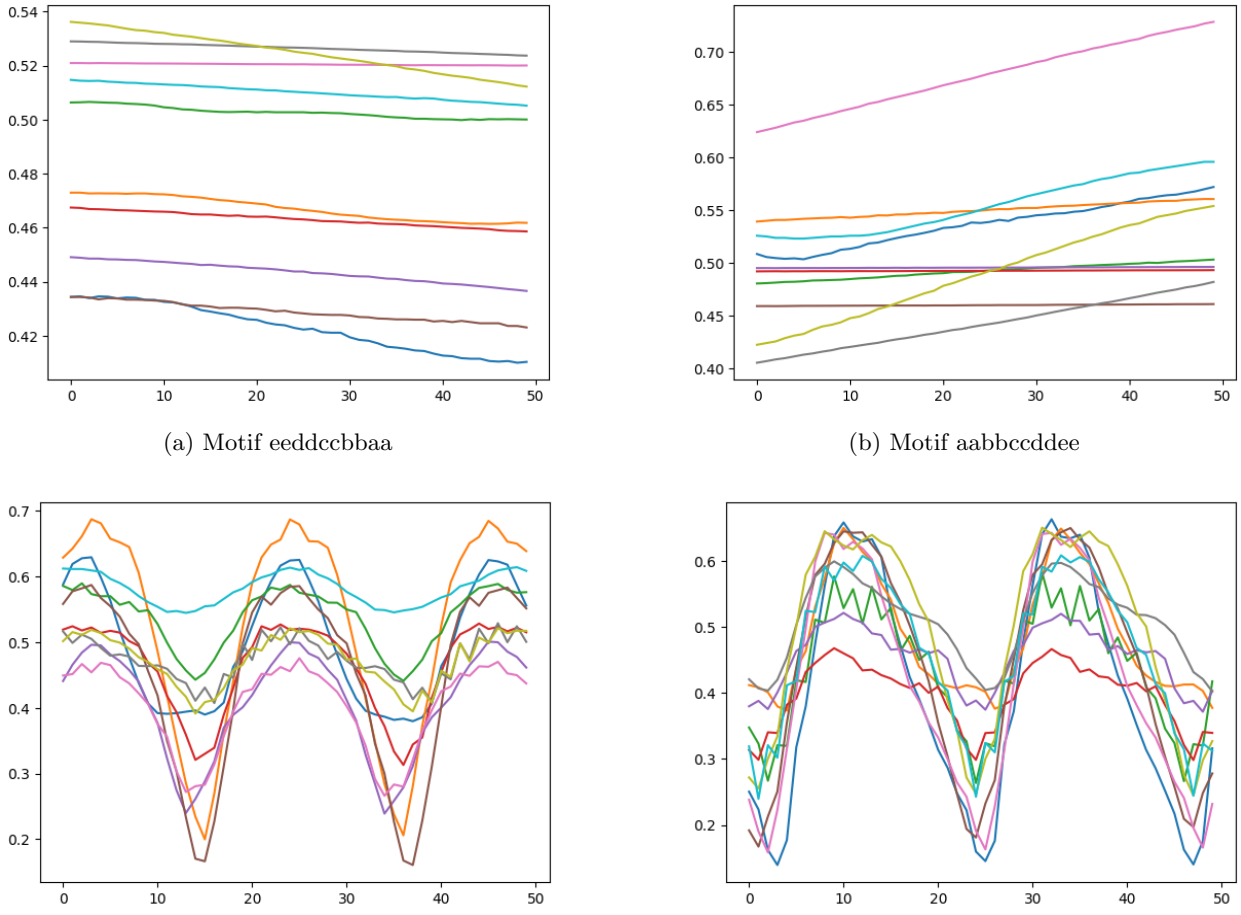
Secondly, we perform clustering with k-means on the raw time series using dynamic time warping (DTW) as a metric. To do this we choose to reduce the dimension of the time series dataset, we cut each time series eliminating the zero-inflated regions as previously said, and trimmed each of them to make them as long as the shortest one resulting in a dataset of time series of length 4605.

We then tried to use the PAA algorithm to reduce our dataset size for computational purposes but we found a problem, since audio files are oscillations around the value 0 the mean of a segment of audio will always be very close to 0, and so blindly applying PAA to our dataset would result in noise. So we choose to transform the values of each time series into absolute values before performing PAA, this way resulting in an approximation of sensible values. We choose to compress each time series into 100 segments from 4605.

We then employed the *TimeSeriesKMeans* algorithm from tslearn using DTW as a distance measure. Similar to the previous approach, we compute both the silhouette and the within-cluster sum of squares (WCSS) score. From Figure 24 we can see the presence of an elbow at k = 3 and 4 but we choose to analyze 3 clusters as the silhouette score for 3 clusters was much higher.

From Tables 27 and 28 we can see that we have one large cluster containing 74% of all observations with a second large cluster with 21% and a remaining small cluster with just 5% of the observations.

The first cluster is the medium one, it contains 44% of all angry observations and 37% of all happy



(a) Motif eeddccbbaa

(b) Motif aabbccddee

(c) Motif ecabddbace

(d) Motif adecbbedca

Figure 21: Most common SAX representations of motifs

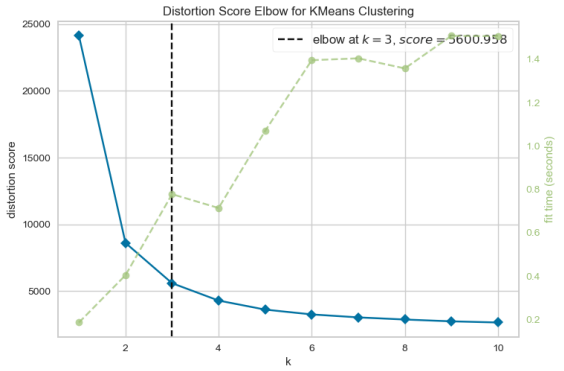
observations, and a consistent amount of fearful while having almost no observations for all other emotions, 77% of its observations are normal, and it has the majority of female and song observations while being balanced on the other categorical features.

The second cluster is the smallest, it is composed of more than 50% angry observations, and 28 calm and 28 fearful observations. It has a majority of Female, Speech and Normal observations.

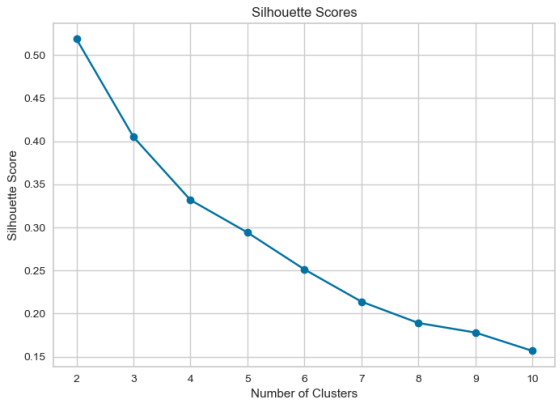
The third cluster is the biggest by far, it contains almost all calm, sad, disgust, and neutral observations in the dataset, having a majority of speech and strong observations.

As before we can see that the statement class didn't have any influence on the clustering. In this case the driving factor for the clustering where emotions, with the third cluster containing almost all elements of 4 emotion classes, and the other clusters being strongly unbalanced toward the others. Also, emotion intensity was of particular interest in separating the first and third clusters.

From Figure 25b we can see both the cluster centroids and the mean time series for each emotion class. We can see that the centroids for clusters one and two have a much higher intensity with peaks over time, this is caused by the high presence, for clusters one and two of angry and fearful emotions, and of happy emotions for only cluster one, which as can be seen from Figure 25b are the loudest emotions with peaks at the beginning and at the end. While cluster 3 has a lot of observations from all emotions, in particular calm, disgust, and sad, which have a low mean intensity, for this reason, it has a lower mean intensity comparable to the mean intensity of the whole dataset.



(a) Elbow Plot (WCSS Plot)



(b) Silhouette Plot

Figure 22: Feature Based Clustering Elbow and Silhouette Plots

Cluster	F	M	Song	Speech	Normal	Strong	Kids	Dogs
Feature-Based Clustering								
Cluster 1 (636)	232	404	387	249	355	281	318	318
Cluster 2 (769)	579	190	395	374	352	417	424	354
Cluster 3 (423)	81	342	298	125	277	146	172	251
Distance-Based Clustering								
Cluster 1 (380)	213	167	213	167	293	87	195	185
Cluster 2 (97)	61	36	23	74	94	3	45	52
Cluster 3 (1351)	618	733	512	839	457	894	674	677

Table 28: Distribution of categorical variables over the clusters

4.4 Classification

We then tackled the emotion classification task using the original audio data.

4.4.1 K-nn

We first applied a simple K-nn model on the decimated padded dataset applying a 5 folds CV on the train set to find the best parameters in the same fashion as previously described in section 1.1.1.

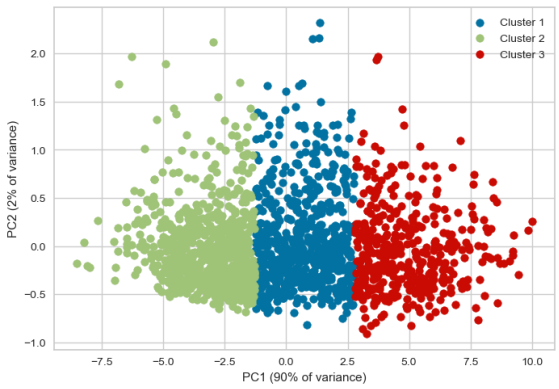
The best model had 38 NN, Manhattan distance and distance weighted voting and achieved a C.V. accuracy of 17%.

The model achieved a terrible performance predicting only 2 classes out of 8 and having an accuracy slightly higher than random guessing, only due to the fact that disgust, surprise, and neutral emotions are much less common as in fact its balanced accuracy is 0.12 equal to random guessing.

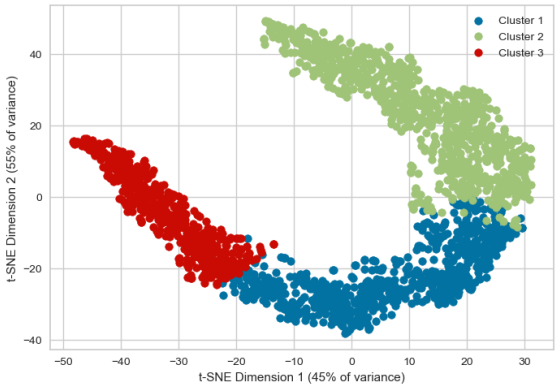
We hypothesized that the model's poor performance might be attributed to the time series being unsynchronized, with some starting earlier than others. To address this issue, we removed the sections of the time series that contained mostly zeros for each observation, as explained in the motifs section. However, despite this effort, we did not observe any improvement in the model's performance.

So we choose to use Dynamic Time Warping distance dtw from tslearn.metrics on the K-nn model, we immediately saw that we couldn't apply it on the dataset since the computation was too expensive, so we tried different approaches to reduce the size of the time series.

We first took the original audio files and eliminated the zero inflated regions and sampled one observation every 50, leaving us with a dataset of time series with an average length of 1892.

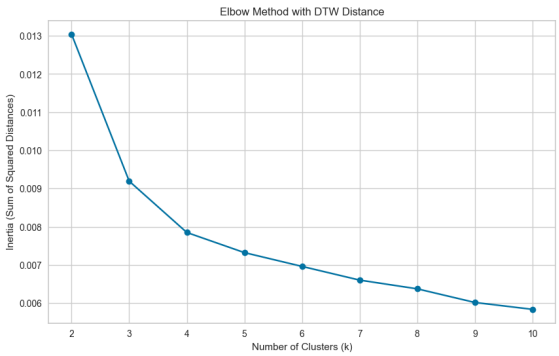


(a) PCA plot

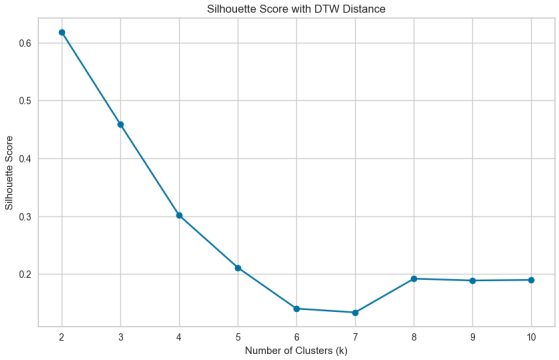


(b) TSNE Plot

Figure 23: Feature Based Clustering Dimensionality Reduction Plots

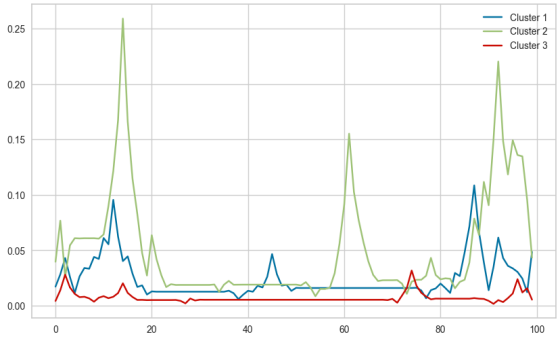


(a) Elbow Plot(WCSS Plot)

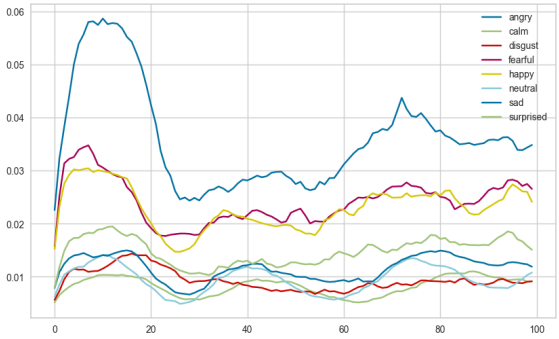


(b) Silhouette Plot

Figure 24: Distance Based Clustering Elbow and Silhouette Plots



(a) Plot of centroids of each cluster



(b) Plot of the mean of emotions

Figure 25: Plot of mean time series of emotions and of cluster centroids

We also took the padded dataset and reduced the dimensionality of the dataset by sampling one observation every 300 ending with a dataset of time series of length 1015.

We then wrote our own K-nn classifier class to compare time series of different lengths and parallelise the computation of the nearest neighbours and then splitted the training set into train 80% and validation

20% and found the best number of neighbours between 1 and 200 on the validation set and then tested the model on the Test set.

From Tables 30, 31 we can see that the DTW K-nn classifier achieved the same accuracy on the test set on both datasets, on the first one the best nn was 35 while on the second one was 88, both of them had the best performance on calm emotions and the worst on disgust, but the model trained on the padded data had a much more unbalanced performance over the classes, achieving a lower balanced accuracy and having an recall lower than 20% on 4 classes while the other model only had it for the disgust emotion.

Table 29: Performance of the best DTW K-NN models

Table 30: Cutted Data

Emotion	Precision	Recall	F1-score
angry	0.64	0.41	0.50
calm	0.42	0.73	0.53
disgust	0.42	0.16	0.23
fearful	0.39	0.32	0.35
happy	0.35	0.28	0.31
neutral	0.28	0.45	0.34
sad	0.27	0.24	0.26
surprised	0.35	0.42	0.38
Accuracy			0.38
Balanced Accuracy			0.37

Table 31: Padded Data

Emotion	Precision	Recall	F1-score
angry	0.74	0.42	0.53
calm	0.38	0.97	0.54
disgust	0.33	0.08	0.13
fearful	0.38	0.16	0.23
happy	0.41	0.34	0.37
neutral	0.23	0.18	0.20
sad	0.22	0.19	0.20
surprised	0.37	0.52	0.43
Accuracy			0.38
Balanced Accuracy			0.35

4.4.2 Mini Rocket

Next, we addressed the classification problem by employing the Mini ROCKET (Random Convolutional Kernel Transform) feature extraction method. To handle the high dimensionality of each padded time series, we implemented a decimation technique, sampling one observation every 10 data points. This resulted in a dataset with time series of length 30,431. It's worth noting that this is the largest dataset size used in our project, highlighting the impressive speed of the feature extraction process using the ROCKET algorithm.

We then tried 100, 1000, 5000, 10000, 20000 as parameters for the number of kernels in the Mini-Rocket algorithm.

For each sample of extracted features we then trained with 5 folds CV a Logistic Regression Model with l2 regularization trying 100 different regularization parameters evenly spaced in a logarithmic scale from 0.001 to 100.

The best model had a regularization parameter of 40.3 and used 10.000 kernels in the Mini-Rocket Feature extraction achieving a C.V. accuracy of 0.661. We then tested this model on the test set, transformed with the Mini-Rocket model fitted on the train set.

From Table 32 we can see that this feature extraction method had a significant decrease in performance when applied to the test set, but still significantly outperformed each previously seen model on the emotion recognition task in terms of accuracy, while not in terms of mean AUC.

4.4.3 Shapelet

We then approached the classification task with a shapelets-based classifier. We chose to take the original time series data, to cut the zero-inflated regions, we then found the shortest observation and trimmed all time series to that length, and then decimated them until we had a dataset of time series of length 921. This was done to make the computation on the shapelets feasible.

We choose to use as a shapelet extraction method the LearningShapelets model from tslearn, this model finds the optimal shapelets through gradient descent, using a categorical loss computed through a logistic model trained on the shapelets encoding of the data.

Table 32: Performance Ridge clas. on the Rocket test set

Emotion	Precision	Recall	F1-score
angry	0.59	0.78	0.67
calm	0.63	0.62	0.62
disgust	0.44	0.64	0.52
fearful	0.62	0.33	0.43
happy	0.55	0.49	0.52
neutral	0.83	0.61	0.71
sad	0.52	0.57	0.54
surprised	0.46	0.54	0.50
Accuracy			0.57
Balanced Accuracy			0.57
Mean AUC			0.869

Table 33: Performance of RF on the shapelets test set

Emotion	Precision	Recall	F1-score
angry	0.50	0.68	0.58
calm	0.50	0.52	0.51
disgust	0.23	0.12	0.16
fearful	0.35	0.31	0.33
happy	0.33	0.44	0.38
neutral	0.52	0.31	0.38
sad	0.34	0.37	0.35
surprised	0.21	0.12	0.16
Accuracy			0.40
Balanced Accuracy			0.35
Mean AUC			0.79

We did an initial experimentation phase where we decided to utilize the Adam optimizer for our gradient descent problem using 200 epochs. Additionally, we set a minimum shapelet length of 2% of the time series. And decided to train our model to learn 6 shapelets for each size, the parameter Number of Shapletes sizes (NSS) determines the number of shapelet sizes (as 2% times 1, 2, 3 ... until NSS) that the model will create.

We first split our data into train 80% and validation 20% sets and tested through grid search a combination of the parameters choosing the one that achieved the best accuracy for the logistic model in the LearningShapelets algorithm on the validation set. The parameters we tried are:

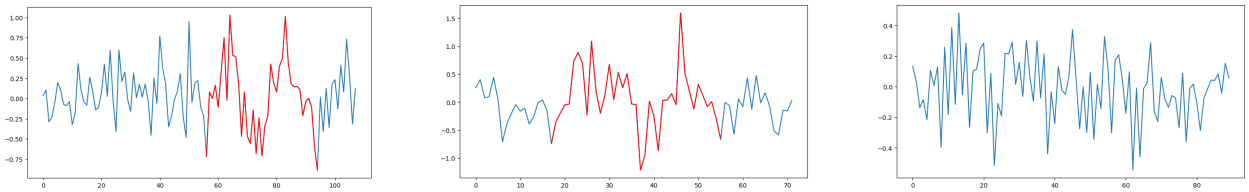
- Learning rate of Adam Optimiser: 0.02, **0.005** ,0.001
- L2 regularization of Optimiser: **0**, 0.01, 0.001
- Number of Shapletes sizes (NSS): 5, **10**
- Min-Max Standardisation: True, **False**

The best model we found was not min max scaled, had no regularization, estimated 60 shapelets, and had a learning rate of 0.02. This model achieved an accuracy of 0.31 on the validation set and 0.34 on the test set.

We then extracted the shapelets from the model and transformed the train and test data with the distances from each shapelet resulting in a dataset of 60 features.

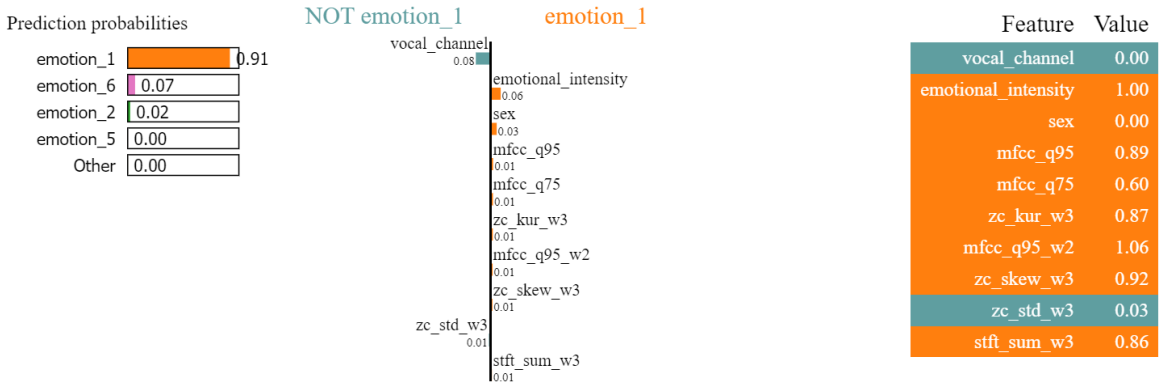
We then trained a Random Forrest classifier with gird search and 5 folds CV, from Table 33 we can see that this classifier achieved a better accuracy on the test set than the logistic model trained by the LearningShapelets.

We then calculated the feature importance indexes mentioned in the previous section for the random forest classifier finding the most important shapelets to be 34, 20, 26. From their plot in Figure 26.



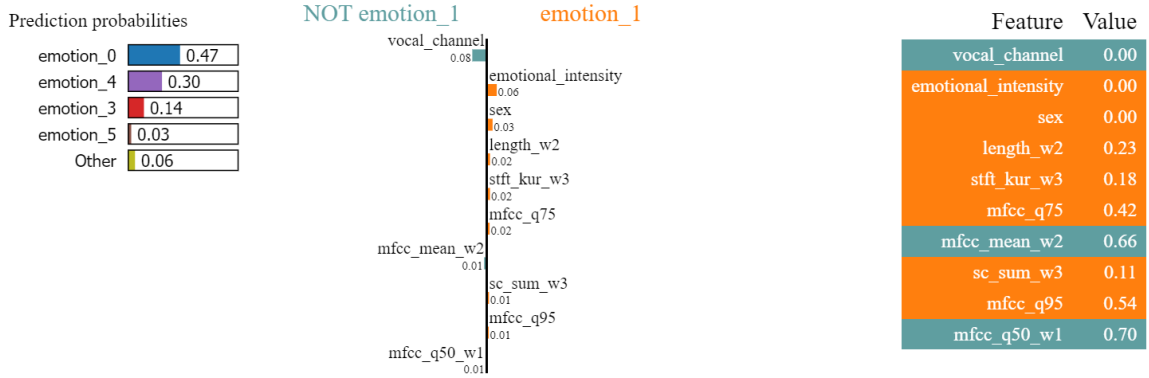
(a) Shapelet 34 (b) Shapelet 20 (c) Shapelet 26

Figure 26: 3 most important shapelets identified by Impurity based and Permutation Feature importance



(a) LIME XAI analysis 1

Figure 27: LIME XAI analysis (1st image)



(a) LIME XAI analysis 2

Figure 28: LIME XAI analysis (2nd image)

When comparing these shapelets to the most common motifs identified in the previous section in Figure 21 we can see that in the shapelets 34 and 20 we have a two waves patterns, marked in red, similar to the motif c and d. But we don't find increasing or decreasing patterns like in motifs a and b.

4.4.4 Comparison Between models

From the previous analysis we can infer that the Mini Rocket Feature extraction method followed by a ridge regression was the best preforming model we found in this whole work achieving the highest accuracy and balanced accuracy of all models and predicting all classes with a support of at least 0.3. While both DTW K-nn and Random Forest trained on shapelet data performed worst than the best models trained on extracted features from the time series having an accuracy lower than 0.5.

5 Module 4 - Explainable AI

As for the last module we employed two techniques to better explain and make more readable the process employed by the Deep NN classifier we already implemented in Paragraph 3.2.

5.1 BLACK BOX visualization via Decision Tree

We initially trained a Decision Tree using the predictions obtained from the train set by our Neural Network This allowed us to create a visual representation of the tree, showing the parameters involved in each split and how they influenced the decisions made by our black box model. The Decision Tree was constructed with default settings, except for the depth of the tree, which we set to 4 to make the visualization more readable. You can see the plot in Figure 29. In a real-world scenario, we can provide an example of an analysis that could be conducted. Upon examination, we observe that the features 'lag1_min_w3' plays a crucial role as the initial splitting criteria for distinguishing emotions. These features enable us to group the four emotions into two categories: sad-calm and angry-fearful. By examining the specific values associated with these features (whether they are greater or lower than 0.954), we can gain insights into the decision-making process of our Neural Network. This ability to easily interpret the values allows us to detect any potential biases present in the model, which could be detrimental if the model were involved in making critical decisions for a company. The same can also be ampliated with more leaf to get a deeper dive into the criteria chosen but with the flaw of being less readable.

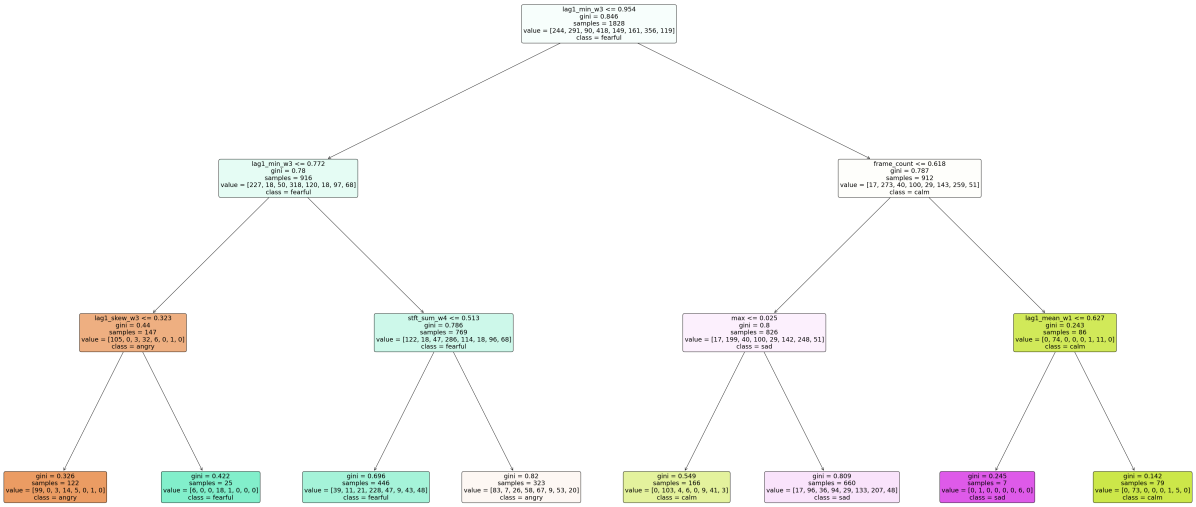


Figure 29: Decision Tree on predicted classes

5.2 Lime

We then implemented the Lime tabular explainer from the lime library. This method is a local one and so we applied it to two correctly classified observations of the test set using the predicted probabilities of our NN. The results of the analysis can be seen in Table 27a, in which we can see that for both observations the vocal channel feature was arguing for the incorrect class while emotion intensity, sex, and most other features were arguing for the correct labeling and that the most important features where categorical.

References

[1] Markus M Breunig et al. "LOF: identifying density-based local outliers". In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 2000, pp. 93–104.