Audio Sentiment Analysis

Alessandro Casella
Politecnico di Torino
Student ID: s306081
s306081@studenti.polito.it

Francesco Grandi
Politecnico di Torino
Student ID: s306272
s306272@studenti.polito.it

***Abstract*—In this report, we propose a possible approach for sentiment analysis on audio speech through different classification techniques. Both classic and more advanced machine learning models are compared and described in detail for this purpose. In particular, the preprocessing steps are carefully analyzed to enhance the most significant features.**

I. Problem overview

A. Aim of the competition

The objective consisted in classifying the audio files to emotions. Each audio file contains a short speech by a speaker.

B. Development dataset

It is composed of 9,597 records, each one described by two attributes:

- **emotion**: the sentiment felt by the speaker in the audio recording. This label can assume seven different possible values;
- **filename**: a string that represents the name of the audio file.

We will need to handle the development set to build a classification model to label the points in the evaluation set correctly.

C. Evaluation dataset

It contains 3,201 records and differs from the development dataset for the *emotion* column, which here represents our target to discover. The records are described only by the same *filename* attribute used in the development dataset.

The *filename* attribute of the datasets refers to the audio files in WAV format that are collected altogether in a separate folder.

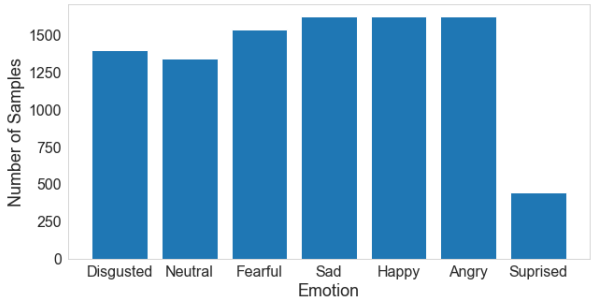

Fig. 1: Emotions distribution

Making an overview on given data, we noticed that neither dataset contained missing values. Secondly, analysing the target value, the class distribution of the training set was approximately balanced, except for one class, as shown in Figure 1: the *Surprised* samples were about four times fewer than the others. This fact might have an impact on our analysis. We can make some deeper considerations based on the nature of audio files, to better understand our development dataset's composition. Figure 2 shows how the audios are distributed in duration. The most common lengths are in the range of [2, 2.5] seconds, while there are few recordings above 5 seconds. It is also possible to note that the duration parameter follows a normal distribution. Since most classification models require a fixed number of features, we needed to draw up an approach to extract the same number of features from all the entries.

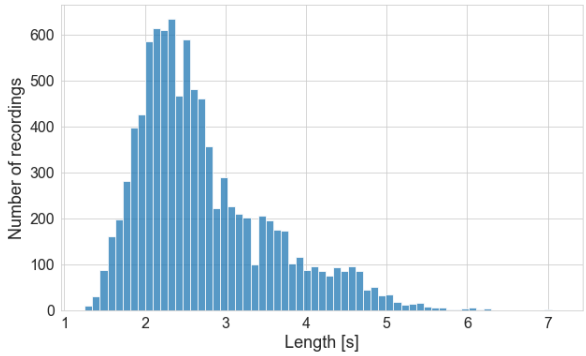

Fig. 2: Audios' length distribution

While recordings' duration differs, all signals have been sampled at the same rate, equal to 8 kHz. To better understand the data type we are dealing with, in Figure 3 and Figure 4 we visualize how the amplitude of a signal changes in the time domain. Since the amplitudes are bound in the range of values [-32,768, 32,767], we can derive that the sample width is 16 bits for all files [1]. In comparing two opposite emotions, there is a visible difference in amplitude. The waveform of the sound files is the main information with which to create features to train our model. However, the shape of a waveform does not yet carry enough discriminating information for our classification.

In our data exploration of the development set, we noticed that the *8005.wav* audio labelled as *Sad* does not contain any

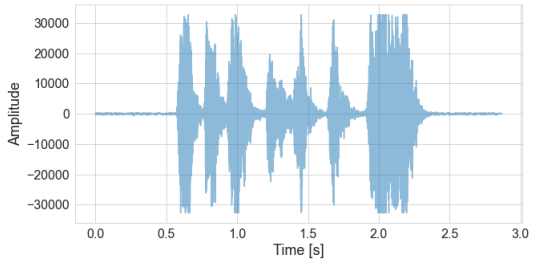

Fig. 3: Amplitude of an *Angry* recording

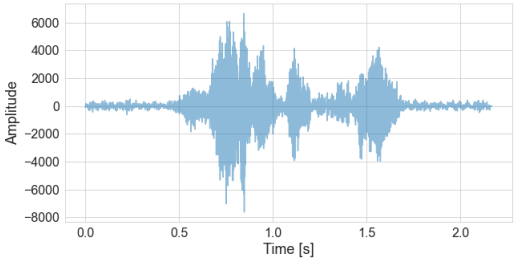

Fig. 4: Amplitude of a *Happy* recording

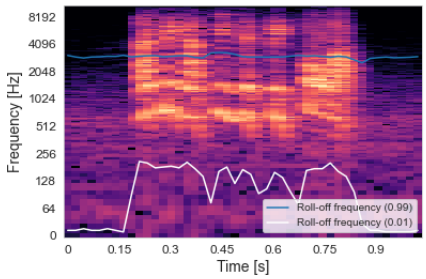

Fig. 5: Spectogram with rolloff of an *Angry* recording

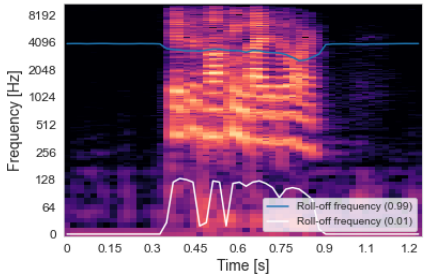

Fig. 6: Spectogram with rolloff of a *Happy* recording

kind of information: its NumPy array representation consists in all zeros. Since we believe it as a misclassification error, we decided to exclude it from our analysis. In conclusion, after analyzing the amplitude and the length distribution, we assumed no outliers were present. Indeed, we didn't find any audio with a too-far-from-the-average amplitude or length.

II. Proposed approach

A. Data preprocessing

We have discovered that the time domain contains valuable information regarding the recordings. However, we are also aware that we could use other features to distinguish the emotions in a better way. For this purpose, we evaluate the link between time and frequency through *Short-Time Fourier Transform - STFT*. This method cuts our audio into short, overlapping and equal-length segments. The Fourier transform elaborates each of these segments and produces spectrograms: by their union, the multiple power spectrogram identifies the frequency evolution of the audio along with the time domain (Figure 5 and Figure 6).

To extract relevant information from data, we relied on *librosa* library, which provides functions to extract audio features. We tried to get an idea of some of them by first considering past works about audio classification, then visualizing the differences among the classes as we did for spectrogram and finally testing their relevance to our model performance. We show all the ones we used below:

- ***Mel-Frequency Cepstral Coefficients - MFCC***: this feature exploits the human auditory frequency response with the help of Mel-Scale frequency response. Mathematically it transforms the power spectrum to a small number of coefficients, that represent the power of the audio signal in a frequency region, taken w.r.t. time. MFCC uses STFT to produce audio power spectrums, to which overlapping window functions are applied. The sums of energy at each window represent the mel filterbanks, mapped to the mel scale [1]. The discrete cosine transform applied to the log of power in each filterbank.
- ***Mel Spectrogram***: when we mapped the frequencies to the mel scale, we even produced a new spectrogram, with the frequency scale in mels [2]. We took it into account because we saw that Angry voices presented clearer transitions between frequency peaks compared to Happy voices: this also makes intuitive sense.
- ***Chromagram***: on each power spectrogram computed by STFT, builds a representation of an audio signal w.r.t. time, mapping it to 12 pitch classes, i.e. the musical scale CDEFGAB + 5 semitones. The pitch distribution of the Angry voice has a much lower dispersion than the Happy one, whose pitch has a higher dispersion at any point in time.
- ***Rolloff***: the frequency under which a specified percentage of the total energy of the spectrum is contained (Figure 5 and Figure 6). This measure is worthwhile in distinguishing voiced speech from unvoiced: the former has a high proportion of energy contained in the high-frequency range of the spectrum, while lower bands include the latter.

Because the chromagram, mel spectrogram and MFCC are calculated on audio frames produced by STFT, we got a matrix back from each function. Since the recordings differ in length,

we took the mean and the standard deviation from the matrix. In this way, for each feature and for each audio sample, we managed a single array equal in length to all the other ones. Defining our features as explained takes the benefit of having a uniform number of features for all signals.

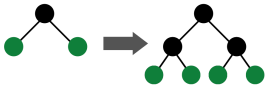

Fig. 7: level-wise tree growth

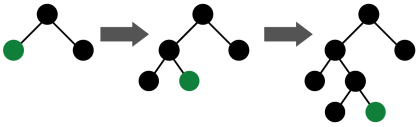

Fig. 8: leaf-wise tree

B. Model selection

We approach the problem by testing different types of algorithms. The following part highlights them in order of increasing effectiveness.

- ***Gaussian Naïve Bayes***: implements the naïve Bayes assumption by considering conditional independence between every pair of features.
- ***Linear Support Vector Machine - SVM***: finds a linear hyperplane separating points belonging to different labels. This is achieved by maximizing the distance from the nearest points to the hyperplane.
- ***K Nearest Neighbors - KNN***: this algorithm classifies elements by computing the distance from the nearest neighbours. The choice of the assigned class is made by majority voting.
- ***Random Forest - RF***: employes a set of decision trees trained on random subsets of the dataset. By majority voting is then chosen the best decision tree to deploy. We selected random forests as it's proven to perform well on audio signal classification problems [1].
- ***Multilayer Perceptron Classifier - MLP***: is a feedforward artificial neural network. An MLP is characterized by several layers of input nodes connected as a directed graph between the input and output layers. MLP uses backpropagation for training the network.
- ***LightGBM - LGBM***: is a gradient boosting algorithm as such an ensemble model of decision trees, which are trained in sequence [2]. In each iteration LightGBM learns the decision trees by fitting the negative gradients (also known as residual errors). The main difference between other gradient boosting algorithm is that the decision trees in LightGBM are grown leaf-wise. In fact, instead of checking all of the preceding leaves for each new leaf as in level-wise tree growth (Figure 7), it is enough to refer to the previous node (Figure 8). Moreover all the attributes are sorted and grouped as bins. This implementation is called histogram implementation. These kinds of algorithm have several advantages, such as better accuracy, faster training speed, and are capable of large-scale handling data [3].

Model	Parameters	Values	Best F1 score
GaussianNB	*default*	default	0.258
SVM	*C* *gamma* *kernel* *random_state*	10 auto rbf 42	0.601
KNN	*n_neighbors* *weights* *algorithm* *leaf_size* *n_jobs*	5 distance brute 30 4	0.603
RF	*criterion* *max_features* *min_samples_split* *max_depth* *random_state* *n_jobs*	entropy, gini auto, sqrt,log2 2, 5 100, 200 42 -1	0.657
MLP	*hidden_layer_sizes* *activation* *solver* *alpha* *epsilon* *learning_rate*	(8,), (180,), (300,),(100,50,), (10,10,10) tanh, relu, logistic sdg, adam 0.0001, 0.001, 0.01 1e-08, 0.1 adaptive, constant	0.676

TABLE I: Hyperparameters considered

boosting_type	gbdt	gbdt	gbdt	gbdt	gbdt
objective	multi class	multi class	multi class	multi class	multi class
num_class	7	7	7	7	7
metric	multi _logloss	multi _logloss	multi _logloss	multi _logloss	multi _logloss
learning_rate	0.16	0.16	0.1	0.12	0.025
max_bin	600	600	600	600	800
max_depth	30	30	30	30	37
num_iterations	700	700	1500	1200	100
F1-Macro score	0.69	0.708	0.709	0.71	0.712

TABLE II: LightGBM, Hyperparameters considered

Since SVM and MLP typically benefit from a normalization step, we applied a *standard scaler* to transform the features in these states. The last three models gave us better results than the previous ones; for this reason, from this moment on, we analyze these classifiers. However, as shown in Table I, just a simple or a default setting has been performed in the algorithms that showed less promising results to have a general comparison among all the classifiers.

The best-working configuration of hyperparameters has been identified through a grid search, as explained in the following section.

C. Hyperparameters tuning

Two hyperparameters' sets are tuned, each corresponding to the preprocessing and final prediction phase.

- Preprocessing:

  - *n_mfcc*
  - *n_mels*
  - *roll_percent*

  The best performance was obtained using 40 mel filterbanks (*n_mfcc* = 40), 128 mel frequency bands (*n_mels* = 128) and two different percentages equal to 5% and 95% of the total energy of the spectrum to respectively approximate maximum and minimum frequencies (*roll_percent* = 0.05, *roll_percent* = 0.95). For each audio, we extracted 364 features in total, resulting from summing up 40+40 MFCCs, 128+128 mels spectrogram, 12+12 chromagrams and 2+2 rolloffs. Unlike LBGM and MLP, we made a feature selection in the random forest classifier, since we noticed the performance improvement taking only the most 100 important features. This selection considers the fitted attribute *feature_importances_*, which highlights how relevant each feature is. The rolloff features were the main impact on the classification, followed by MFCC both in mean and standard deviation; the less significant were chromagram's and mel coefficient's standard deviations. We tried to apply the same scrap to LGBM, but this caused a worse performance.
- Hyperparameters' predictions in:
  - *Random Forest*
  - *MLP*
  - *LightGBM*

  To find the best hyperparameters we applied cross-validation with GridSearchCV using 4 folds, which was enough to avoid overfitting on the training set. In doing so, we considered the combinations of parameters summarized in Table I and Table II. We split the development dataset in the following proportions:
  - **80% Training Set**: this data was used to train the model with various hyperparameters
  - **20% Test Set**: this data was used to test model and select the hyperparameters

III. Results

The chosen hyperparameters were tested in the test set described above. Unexpectedly, the least frequent class (*Surprised*) was the one predicted with more accuracy; consequently, we didn't take any step towards balancing the class distribution in the development dataset. The best configuration for random forest was found for {*criterion: entropy, max_features: 'sqrt', n_estimators: 2000, max_depth: 200, min_samples_split: 2, n_jobs: -1, random_state: 42* } with a public f1 macro score of ≈ 0.684, whereas the best configuration for the MLP was found for { *activation: logistic, alpha: 0.01, epsilon: 1e-08, hidden_layer_sizes: (300,), learning_rate: 'adaptive', solver: 'adam'* } with a public f1 macro score of ≈ 0.688. The two classifiers achieve well above the baseline of 0.544. Nonetheless, the best performing algorithm was found to be LightGBM. The best combination of hyperparmeters of this model is {*learning_rate: 0.025, boosting_type: gbdt, objective: multiclass, metric=multi_logloss, vcmax_depth=37, max_bin = 800, n_estimators = 2000, num_leaves = 22*} resulting in a public f1 macro score of ≈ 0.726.

IV. Discussion

During the implementation of the algorithms, it was easy to see that raw data was not enough to achieve good performance for any model. Therefore, extract new features by analyzing the data in other dimensions and scaling them through mathematical tools was very helpful. We implemented different algorithms to take full advantage of the extracted features. Among these, the best performing models were Multilayer Perceptron Classifier and Random Forest. However, we obtained the best scores only with an advanced implementation of the latter through LightGBM. Later on, some aspects might be worth taking into consideration to improve the results:

- **Extract more features from the raw data.** We tried a fair number of them and selected the most significant ones. Further improvement could be achieved using spectral bandwidth and spectral contrast.
  In addition, using other statistical tools such as Root Mean Square can help to characterize better the extracted features.
- **Implementing a Convolutional Neural Network (CNN)** with various layers[2]. It would allow studying features extracted in the form of images more in depth.
- **Try more grids-search.** Particularly promising is the LightGBM model, which improved every time the learning_rate parameter decreased. In addition, according to literature related to the model [4], setting the *boosting_type* parameter to *dart* could improve the overall score further.

In conclusion, we reached higher-than-baseline private scores for every model (except GaussianNB), better than the average public score, thanks to the use of LightBGM. The fact that our private score is always less than the public one hints at avoiding overfitting.

References

[1] Lalitha, S., Geyasruti, D., Narayanan, R., Shravani, M. (2015). Emotion detection using MFCC and cepstrum features. Procedia Computer Science, 70, 29-35.

[2] García-Ordás, M. T., Alaiz-Moretón, H., Benítez-Andrades, J. A., García-Rodríguez, I., García-Olalla, O., Benavides, C. (2021). Sentiment analysis in non-fixed length audios using a Fully Convolutional Neural Network. Biomedical Signal Processing and Control, 69, 102946.

[3] F. Saki, A. Sehgal, I. Panahi, and N. Kehtarnavaz, "Smartphone-based real-time classification of noise signals using subband features and random forest classifier," in 2016 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, 2016, pp. 2204–2208.

[4] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems, 30.

[5] Essam Al Daoud, "Comparison between XGBoost, LightGBM and CatBoost Using a Home Credit Dataset" World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering Vol:13, No:1, 2019

[6] Vinayak, R. K., Gilad-Bachrach, R. (2015, February). Dart: Dropouts meet multiple additive regression trees. In Artificial Intelligence and Statistics (pp. 489-497). PMLR.