Speech emotion recognition to develop a generalized emotion classifier for arbitrary audio files

Jere Lavikainen

Abstract

In this project, a combined dataset comprising data from six individual speech emotion recognition datasets is constructed. After pre-processing steps, 28 features are extracted per waveform. Supervised learning models are trained with five different classifier algorithms to predict the emotion label of audio waveforms in three classes: anger, happiness, and sadness. Neural network and support vector machine-based models performed the best among the tested algorithms with mean classification accuracies of 0.5837 and 0.5571 over six test sets, respectively. Although the models were capable of inter-language emotion recognition to some extent, accuracy in general was low and most likely attributed to suboptimal selection of audio features and classifier model hyperparameters. Several potential improvements to the workflow are presented but were not implemented in this project due to time constraints.

Work contribution

Jere Lavikainen was the sole contributor.

Introduction

Speech emotion recognition (SER) describes the task of grouping audio samples to different emotion classes. Usually, these classes are either discrete and intuitive (e.g., anger, disgust, fear, happiness, sadness) or dimensional (e.g., valence, activation, dominance). Acoustic features of speech are commonly used to classify the emotions [1, 2].

Supervised learning algorithms have been found useful in SER [1, 2]. They allow large numbers of data and input features to be included in the classifier model and depending on the algorithm, are capable of automatically emphasizing important features and excluding irrelevant ones (e.g., adjusting weights of neural networks during training) or are suitable to process image features (e.g., 2D convolutional neural networks to process spectrograms). Because my experience with audio processing stems only from this course, I figured that the flexibility of supervised learning algorithms would make up for my lack of expertise in input feature selection.

Although some prior models for SER exist [2], the goal of this project was to develop a new model to classify emotions into several classes. Furthermore, the model was designed to classify emotions regardless of the speaker's language, gender, or recording conditions. To this end, six existing datasets [] of five different languages were used. Additionally, to compare how different supervised learning algorithms perform in SER, classifier models were trained with five different algorithms. I hypothesized that at least some of them would result in a model that classifies emotions correctly in at least 50% of the cases.

While the results showed that neural network and support vector machine classifiers reached >50% classification accuracy with most test sets, the threshold was exceeded only barely. Furthermore, in some cases >50% prediction accuracy was not achieved for all emotion classes despite the general prediction accuracy being >50%.

Methods

The project comprised extracting class labels and audio files from six existing SER datasets, processing the audio files to ensure uniformity of sampling rate in the combined dataset, voice activity detection (VAD) to remove unvoiced parts of the waveforms, feature extraction to retrieve a set of 28 features for each audio file, the use of different classifier algorithms to train five models to recognize emotion, and evaluation of the models. Figure 1 shows the workflow of the project. All steps of the workflow were implemented as MATLAB scripts, attached with the report. MATLAB's built-in functionality and some of its toolboxes (specified below in "Used libraries") were utilized.

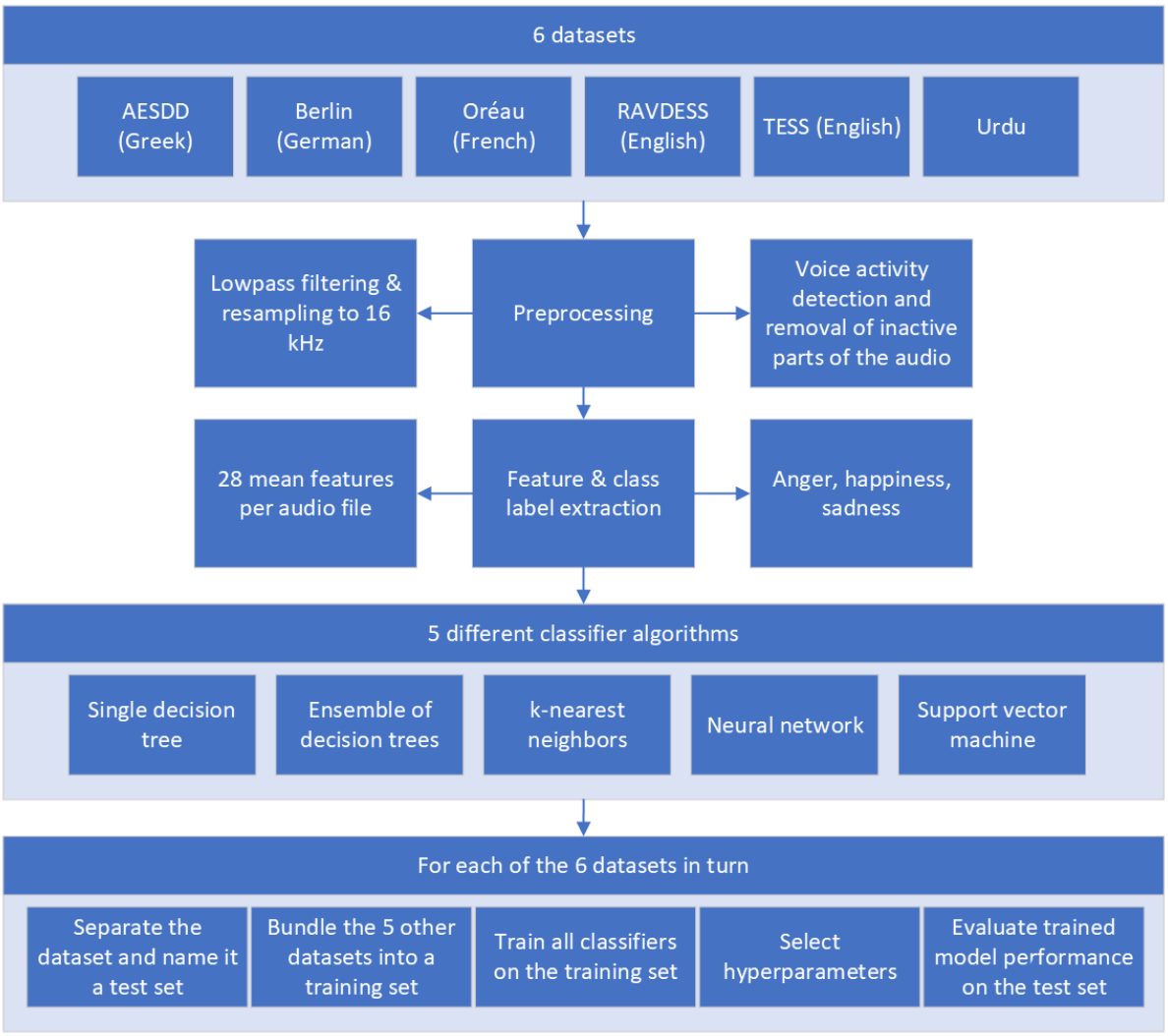

Figure 1. Workflow of the project.

Used libraries

MATLAB 2022a's Signal Processing Toolbox: for audio signal filtering and resampling

MATLAB R2022a's Audio Toolbox: for audio feature extraction

MATLAB R2022a's Statistics and Machine Learning Toolbox: for existing implementations of classification algorithms

MATLAB and its toolboxes are freely available with UEF's campus license for MATLAB.

Datasets

Data was extracted from six existing datasets: the Acted Emotional Speech Dynamic Database (AESDD) [1], the Berlin emotional speech database [2], the Oréau database [3], the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) [4], the Toronto Emotional Speech Set (TESS) [5], and an Urdu-language dataset of emotional speech [6]. Each dataset contained phrases spoken in different emotional states from several subjects. The languages present in the datasets were Greek, German, French, English, English, and Urdu for the datasets named previously, respectively.

During dataset extraction, audio files (waveforms) from each dataset were analyzed one by one. If the sampling rate of the waveform was not 16 kHz, the waveform was first lowpass filtered at 16 kHz using a 4^{th}-order Butterworth zero-phase filter and then resampled to 16 kHz. For each waveform, the information about its emotion label, subject identifier, and original dataset was also retrieved. This process is implemented in the main MATLAB file's sections 1.1 to 1.6.

After all original datasets were extracted, they were combined into a single set. Saved files containing waveforms and their information were retrieved for each individual dataset and concatenated into containers that combined them all, as described in the main MATLAB file's section 2.

The combined set then underwent voice activity detection (VAD) and feature extraction. All waveforms were iterated through. In the VAD phase, the indices corresponding to voiced segments of the waveform were determined. The waveform was squared and a 200-millisecond moving average smoothing window was applied over the squared waveform; indices corresponding to smoothed values that were greater than the median of the entire smoothed waveform were selected. Values corresponding to the selected indices in the unprocessed waveform were then concatenated together to form the trimmed waveform containing only voiced parts.

After VAD, the trimmed waveform was divided into 25 ms frames with 10 ms of hop between frame starting points. A Hamming window was applied to each frame and 28 features were extracted per frame: 13 Mel-frequency coefficients (MFCCs), 11 spectral parameters from the Mel-frequency spectrum (centroid, crest, decrease, entropy, flatness, flux, kurtosis, roll-off point, skewness, slope, and spread), fundamental frequency, harmonic ratio, zero-cross rate, and short-time energy. MATLAB's audioFeatureExtractor class was used to extract the features and its default settings were used aside from the settings mentioned here. The features for a single audio file were determined by taking the mean of the features from all frames of the audio file, resulting in 28 mean features per audio file. The VAD and feature extraction process is implemented in the main MATLAB file's section 3.

Each original dataset had its own set of emotions with some overlap between the datasets. Only three emotion classes were common to original datasets: anger, happiness, and sadness. It should be noted that they were not named thusly verbatim in all datasets: some datasets named their emotions in a non-English language, while in some the emotions were described by adjectives rather than nouns (e.g., “angry” rather than “anger”). In these cases, the emotion names were translated to English and changed to nouns. Anger, happiness, and sadness were then chosen as the classes to predict. Therefore, the combined dataset was trimmed to only include waveforms corresponding to those three emotions, as implemented in the main MATLAB file’s section 4.

The distribution of the data is shown in Figure 2. It should be noted that the TESS dataset comprised almost half of the audio files in the data, but only two of the 100 subjects. Its effects are evaluated in the Discussion section.

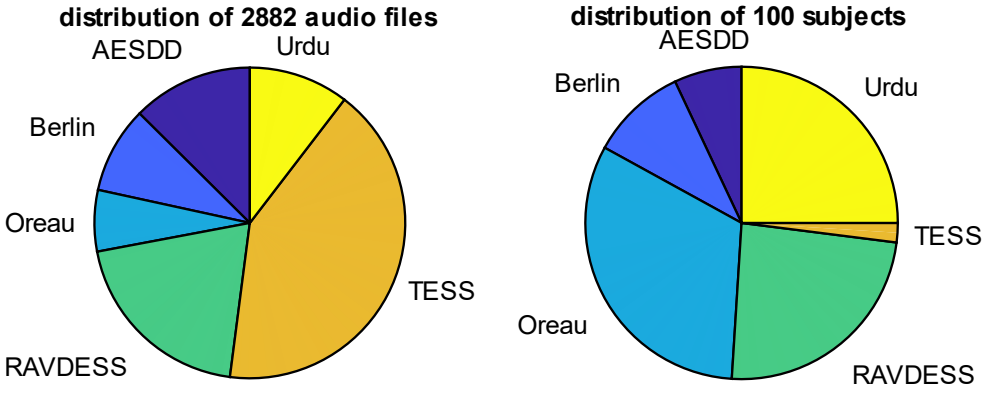

Figure 2. Distribution of audio files (left) and subjects (right) by dataset of origin.

Model description

With the combined dataset in its final shape, the training of classifiers could begin (Section 6 of the main MATLAB file). For each original dataset in turn, the data from it was excluded and used as a test set. The data from the remaining five datasets was used to train a classifier model. The performance of the trained model was then evaluated by having it predict emotions based on input features from the test set and comparing how well those predictions matched actual emotions of the test set.

For classifier models, five algorithms were used: k-nearest neighbors (KNN), decision tree (DT), ensemble of decision trees, fully connected feedforward neural network (NN), and support vector machine (SVM). MATLAB’s existing implementations of these were used with default hyperparameters except for the settings in Table 1.

Table 1. Hyperparameter settings for different algorithms.

Algorithm	Nondefault hyperparameters	Optimized hyperparameters	
KNN	None	Distance type	11 different distance measures

		Number of neighbors	Log-scaled integers between 1 and 1441
DT	maximum number of splits = 100	Minimum leaf size	Log-scaled integers between 1 and 1441
Ensemble of DTs	Maximum number of splits = 100	Ensemble aggregation method	Bag, AdaBoostM2, and RUSBoost
		Number of learning cycles	Log-scaled integers between 10 and 500
		Learning rate for ensemble shrinkage	Log-scaled between 1e-3 and 1
		Minimum leaf size	Log-scaled integers between 1 and 1441
NN	3 hidden layers with 5 nodes each ReLU activation functions		
SVM	Gaussian kernel function	Box constraint	Log-scaled between 1e-3 and 1e3
		Kernel scale	Log-scaled between 1e-3 and 1e3

In the case of neural networks, 20% of the training set was randomly separated as a validation set and the training of the network was stopped when the validation error stopped decreasing for 6 consecutive epochs.

Therefore, we trained a total of 30 models (6 test sets, 5 algorithms). For each test set, the accuracy of a single model was calculated as the number of correctly predicted emotions divided by the number of emotions in the test set. Additionally, confusion matrices were calculated for each model.

Experiments and results

Results and Discussion

On classification accuracy and other results

Accuracy as defined in the methods was used as the main performance metric and is shown in Table 2. Best accuracy was reached with neural network and SVM classifiers, which over all datasets had mean ± standard deviation accuracies of 0.5837 ± 0.1135 and 0.5571 ± 0.1316, respectively. Therefore, roughly a little more than half of the predictions were correct with neural network and SVM models, and worse with other models.

Confusion matrices are presented for all models (Figure 3 to Figure 7). Because the accuracy remained low in most models, precision and recall are not explicitly presented; they can be visually interpreted from confusion matrices.

Figure 3 shows that in most cases, the KNN-based model ended up predicting only one emotion class (anger) irrespective of the input features. KNN is said to perform badly on high-dimensional data, so our input set of 28 features might've been too complex for it.

Although neural networks performed best on average, they predicted more than 50% of the test set correctly when inspecting emotion classes individually only on the AESDD and RAVDESS sets (Figure 6). Although the previously stated mean accuracy over all datasets may give the superficial impression of >50% accuracy, it does not apply to all emotions; even if one or two of the three emotion classes is predicted correctly more often than not, the third emotion class is likely to be predicted incorrectly. Thus, I would not use even the best models from this study in any proper application.

When inspecting the confusion matrices one dataset at a time, we can see that audio files in the Urdu dataset are often mislabelled as happiness when the true emotion is anger. The Urdu dataset consisted of audio samples from talk shows, while the other datasets were recorded by actors. It is therefore possible that people express anger and happiness more similarly in talk shows (exaggerated, loud voice) than they do in controlled, professional recording environments where anger and happiness are perhaps more easily conveyed by tone rather than volume of voice.

Another dataset that stands out for low accuracy while testing on it is the French Oréau dataset (Table 2). When it is used as a test set, the training data includes no French samples; if the feature set includes language-specific information, the low accuracy makes sense. However, the accuracies are not particularly low when using the AESDD or the Berlin set as the test set, even though in those cases the training set also doesn't include the language that is present in the test set. It is therefore possible that in the feature set used in this study, there is more similarity of information between English, German, and Greek than there is between French and the other languages.

Table 2: Accuracies for different algorithms when each of the six original datasets in turn was used as the test set.

	AESDD	Berlin	Oreau	RAVDESS	TESS	Urdu
kNN	0.3343	0.4885	0.3641	0.5295	0.3333	0.3333
DT	0.4917	0.6615	0.4348	0.4080	0.6242	0.2633
ensemble	0.4503	0.6615	0.4728	0.5052	0.5500	0.4900
NN	0.6354	0.7462	0.4511	0.5538	0.4883	0.6167
SVM	0.5608	0.7423	0.4402	0.5556	0.5250	0.4167

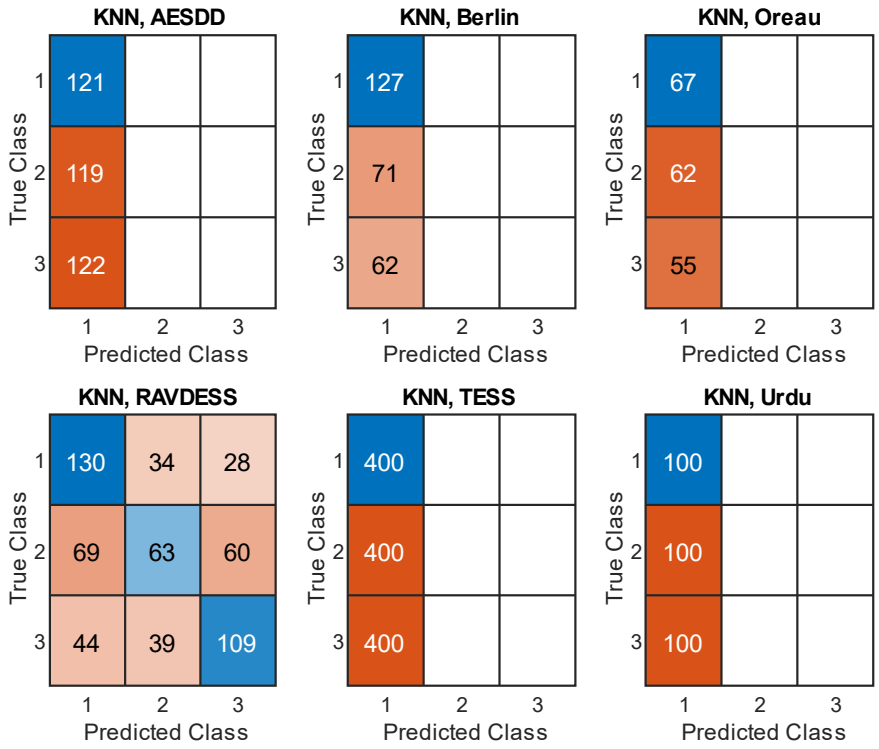

Figure 3. Confusion matrices for different test sets with k-nearest neighbors classifiers. Classes 1, 2, and 3 are anger, happiness, and sadness, respectively.

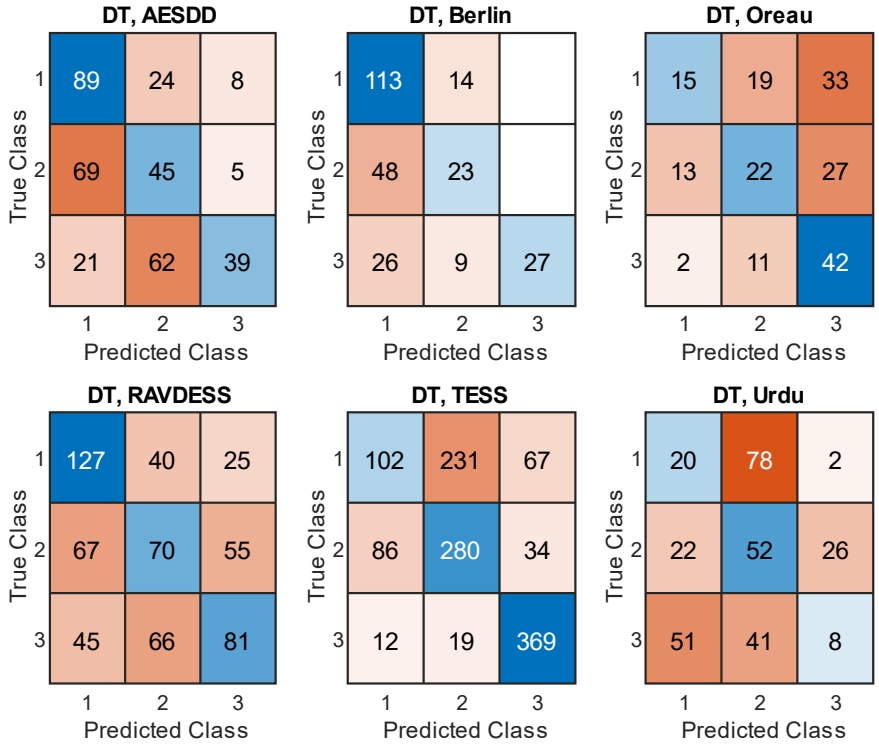

Figure 4. Confusion matrices for different test sets with decision tree classifiers. Classes 1, 2, and 3 are anger, happiness, and sadness, respectively.

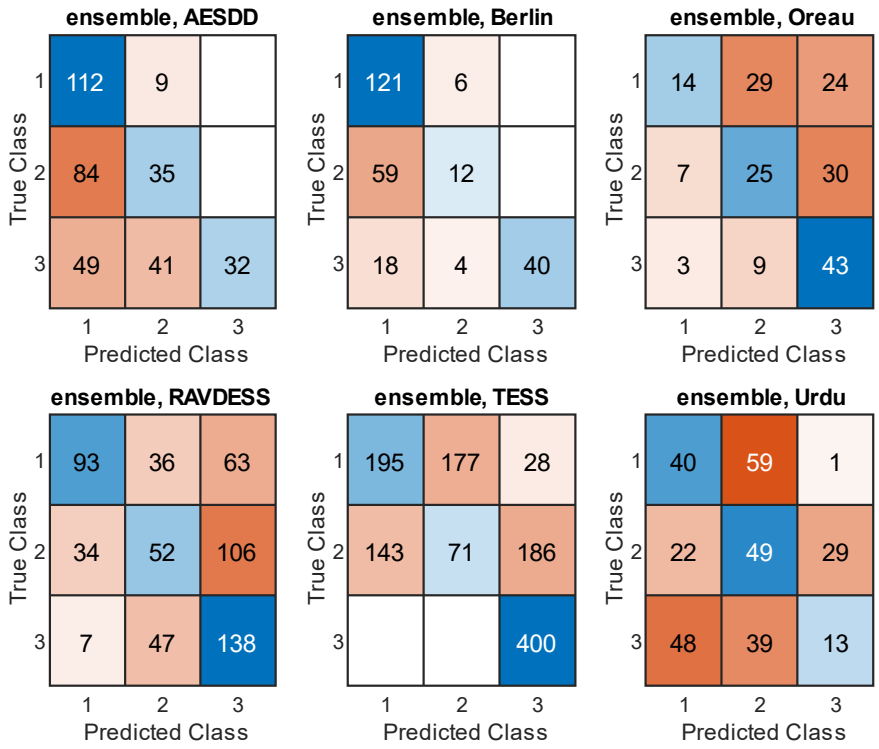

Figure 5. Confusion matrices for different test sets with ensembled decision tree classifiers. Classes 1, 2, and 3 are anger, happiness, and sadness, respectively.

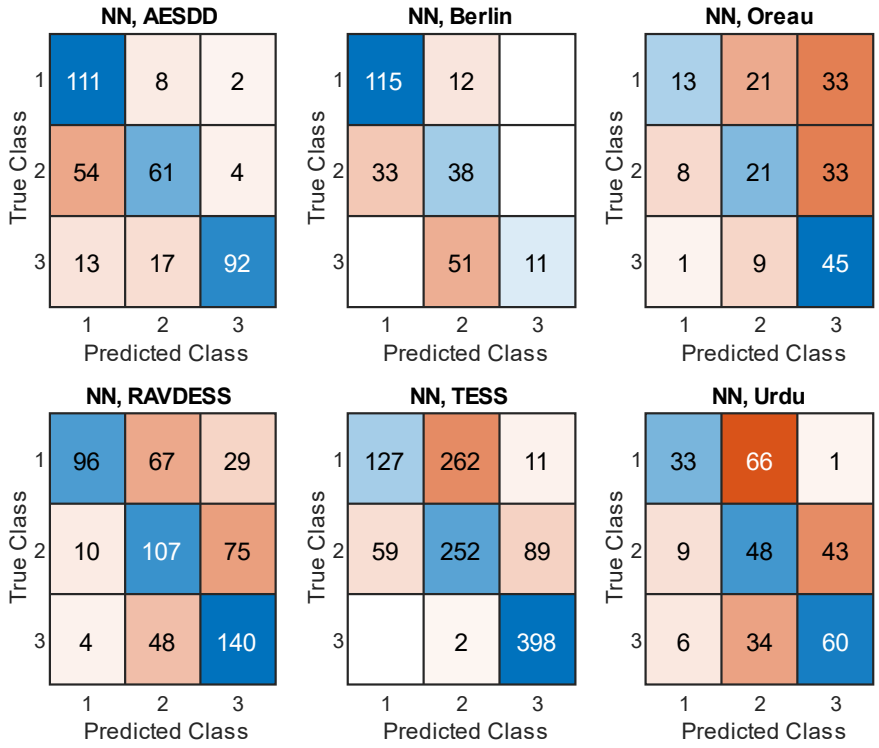

Figure 6. Confusion matrices for different test sets with neural network classifiers. Classes 1, 2, and 3 are anger, happiness, and sadness, respectively.

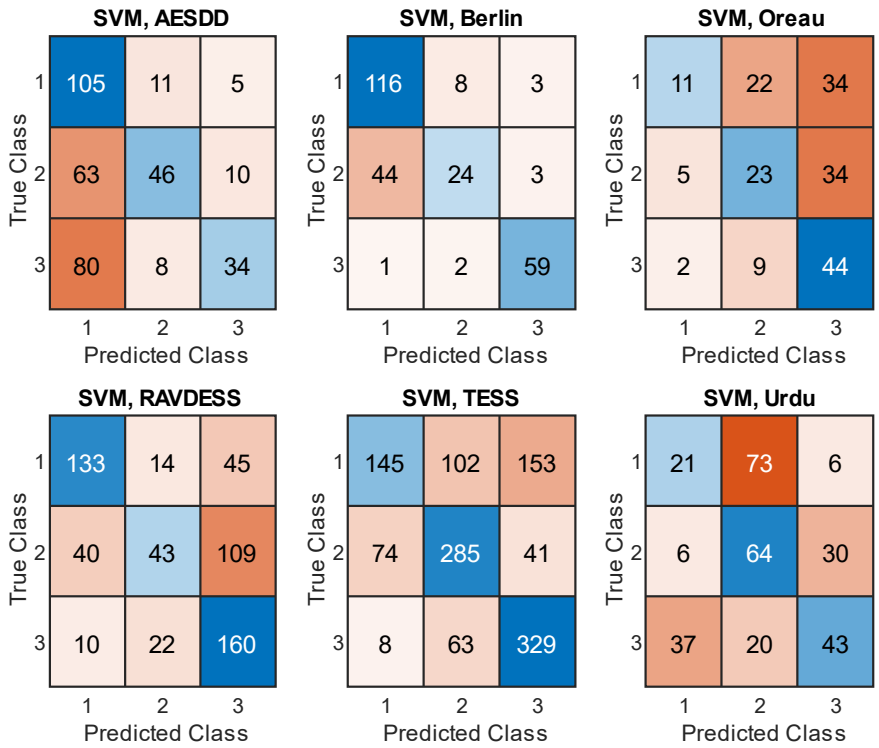

Figure 7. Confusion matrices for different test sets with support vector machine classifiers. Classes 1, 2, and 3 are anger, happiness, and sadness, respectively.

Weighted training of samples

The TESS dataset comprises only 2% of the subjects, but almost half of all waveforms (samples). As a result, during classifier training, it is likely that the subject-specific bias of the TESS subjects is heavily emphasized. Additionally, both TESS subjects are women, so the training process will emphasize feminine attributes in terms of features (e.g., higher pitch). To fix this imbalance of input data distribution, we should use custom weights. Whenever the classifier implementation (e.g., neural network) allows weighting of samples during training, it should be used. The weights should be designed so that unique subjects have approximately equal weights. For example, because the TESS subjects comprise 2% of all subjects, the samples from the TESS dataset should be weighted so that the sum of their weights equals 2% of the sum of the weights of all utilized data (from all datasets).

Similarly, we should aim to weight different languages equally. Therefore, the language weight mask should be designed so that the sums of the weights of samples of individual languages are equal. To obtain a compromise between weighting for equal subjects and equal languages, the subject weight mask can be multiplied elementwise with the language weight mask.

Hyperparameter selection

Currently, the validation sets for hyperparameter optimization are separated randomly from the training set. As a result, both the training set and the validation set may contain samples from the same dataset or even same subject. This can lead to distorted performance metrics during hyperparameter optimization and cause suboptimal hyperparameters to be selected.

A solution would be to implement the validation set separation manually so that samples from a single subject are present only in the training set or the validation set, but never in both. Even better

would be to enforce a similar requirement dataset- or language-wise. Because the training set always contains 5 datasets, one of the datasets could be separated as the validation set, which would ensure no samples from the same subject or dataset exist in both the post-separation training set and the validation set.

Finally, adding more hyperparameters to optimize would probably result in more accurate models but they would take more time to train.

Different classifiers and features

During the early stages of the project, superficial correlation analysis was done by calculating Pearson correlation coefficients between various features and the three emotion classes that were converted into numeric values, but no clear correlation was observed. Thus, the features were chosen on a hunch from features studied during the course (e.g., MFCCs and fundamental frequency) and features that seemed like they could be descriptive (e.g., spectral parameters of the Mel-scale spectrum). It is quite likely that more accurate models could be trained if features are selected with more care, for example by trying out combinations of different feature sets. In this project, however, we settled for the initial feature set of 28 mean features due to time constraints. Similarly, the results could probably be honed if more algorithms were included instead of just the currently used five.

Conclusion

In summary, there are many possible improvements to be done to the workflow and the steps to implement them are clear in some cases. The reason that these steps were not taken in the methods of this report is mostly that they are time-consuming to implement. With other courses and work during this spring, I had to prioritize how much time I can put into this project. As a result, I had to make conscious decisions to occasionally use suboptimal but quick methods to finish the project in time. With more time, I would’ve searched for more datasets of emotional speech, tested more audio feature combinations and implemented some kind of correlation analysis for them, used more classifiers and hand-picked their hyperparameters (or implemented a fine grid search to automatically optimize them), and evaluated the results with different frame sizes. Regrettably, running the codes in the current workflow already took more than 30 minutes, so the aforementioned actions were not an option.