CMLS Homework 1 - Group 9

10724182 10743181 10766268 10768481

`https://github.com/mttbernardini/cmls-hw1`

1 Introduction

The aim of this document is to describe the realization of a regressor able to predict the mood of a music piece. In particular, in accordance with the Music Emotion Recognition (MER) theory, we used a dimensional approach to face this problem. This approach considers emotions on a two dimensional plane: the *valence*, that describes the grade of pleasantness, and the *arousal*, which represents the level of intensity of the emotion.

Our work is based on a provided **dataset** containing songs, valence-arousal annotations and features extracted from the songs. Section 2 contains a description of the given dataset and of the operations we made on it: feature extraction, feature selection and annotation standardization. The next step was to choose the **regressors**, comparing different kinds of them and analyzing their parameters, in order to find the best model. This is analyzed in detail in section 3. To correctly **evaluate** and enhance our model, we split our dataset and took advantage of cross-validation tools, as described in section 4. The last step was to **test** the model against never seen data and draw our **conclusions** on the work, in section 5.

The following sections will analyze our workflow and the tests that led us to our solution of this problem. However the resolution approach was not linear, therefore the flow of this paper does not reflect our actions in a strictly chronological order. Further details can be found in the Jupyter Notebook containing the results of our experiment [3] and in the GitHub repository of this assignment [2].

2 Dataset description and operations

In our work we employed the DEAM Dataset (*MediaEval Database for Emotional Analysis in Music*) provided by the *Swiss Center for Affective Sciences of the University of Geneva, Switzerland*. The dataset consists of 2058 songs annotated with valence and arousal values, both continuously (per-second) and over the whole songs. Csv files containing features extracted from every song are also provided.

In particular, our work was focused, for the first 1744 songs, on the averaged static annotations of the dataset and on the features extracted from the music database. We used both the already provided features from the dataset and other features extracted ex novo.

2.1 Music database

The music database consists of royalty-free music from several sources: *freemusicarchive.org* (FMA), *jamendo.com*, and the *medleyDB dataset* [8]. There are 1,744 clips of 45 seconds from FMA and 58 full length songs, half of which come from medleyDB and another half from Jamendo [6]. The provided 45 seconds excerpts have all been re-encoded to have the same sampling frequency (i.e, 44.1 kHz) and have been extracted from a random (uniformly distributed) starting point for any given song. Both the 45 seconds clips and full songs are provided in MPEG layer 3 (MP3) format [10].

The music from the FMA is in rock, pop, soul, blues, electronic, classical, hip-hop, international, experimental, folk, jazz, country and pop genres. The music from the MedleyDB dataset, in addition, has music in world and rap genres, and the music from Jamendo also has reggae music. For 2014 and 2015 data set, each song has been manually checked and the files with bad recording quality or those containing speech or noise instead of music have been excluded. For each artist, no more than 5 songs have been selected to be included in the dataset. For medleyDB and Jamendo full-length songs, only songs which had emotional variation in them have been selected, using an existing dynamic MER algorithm for filtering them and doing a final manual selection [7].

2.2 Annotations

The dataset from 2013 and 2014 contains annotations on 45 seconds excerpts extracted from random points in songs. Each excerpt was annotated by a minimum of 10 workers and the static annotations were made on nine-point scale on valence and arousal for the whole 45 seconds excerpts [6]. Then, averaged annotations have been generated with 2 Hz sampling rate, providing both mean and standard deviation values [10].

We standardized annotations using the sklearn [9] `scale` function of the `preprocessing` module, which centers the values around zero to eliminate the effects of possible anomalies. Table 1 reports obtained ranges for each annotation after the standardization.

valence mean	valence std	arousal mean	arousal std
5.794	6.746	5.043	5.802

Table 1: Range (i.e. max-min) for each annotation after the standardization

2.3 Features

A set of features extracted by *openSMILE* [5] for 500 ms windows is provided with the dataset. We also extracted a set of additional features from the 45 seconds song excerpts using *Librosa* [4], computing for the latter 5 different statistical moments (mean, standard deviation, maximum, minimum and kurtosis).

We made several attempts in order to achieve best results in the cross-validation step (see section 4.3), so we first tried by separating tracks in two sets, a "high-annotation" set and a "low-annotation" set, for both valence and arousal mean/std values. Plotting the value distribution of a certain feature for some songs of each set, we could check if some differences were arising in the way those distributions gathered (e.g. see figure 1).

Then, we realized that a better way to visualize features might have been to plot feature-vs-annotation scatters (e.g. see figure 2). This way, we could notice several linear relationships between a feature and the respective valence and/or arousal values. We used these results to make a preliminary manual feature selection. In particular we selected different features depending on the annotation to be predicted and discarded unneeded features.

Finally, in addition to manual selection, we filter out unneeded or redundant features using automatic feature selection tools provided by sklearn, like “recursive feature elimination”, “select k-best” and “variance threshold”. In particular, the last two turned out to be the most effective.

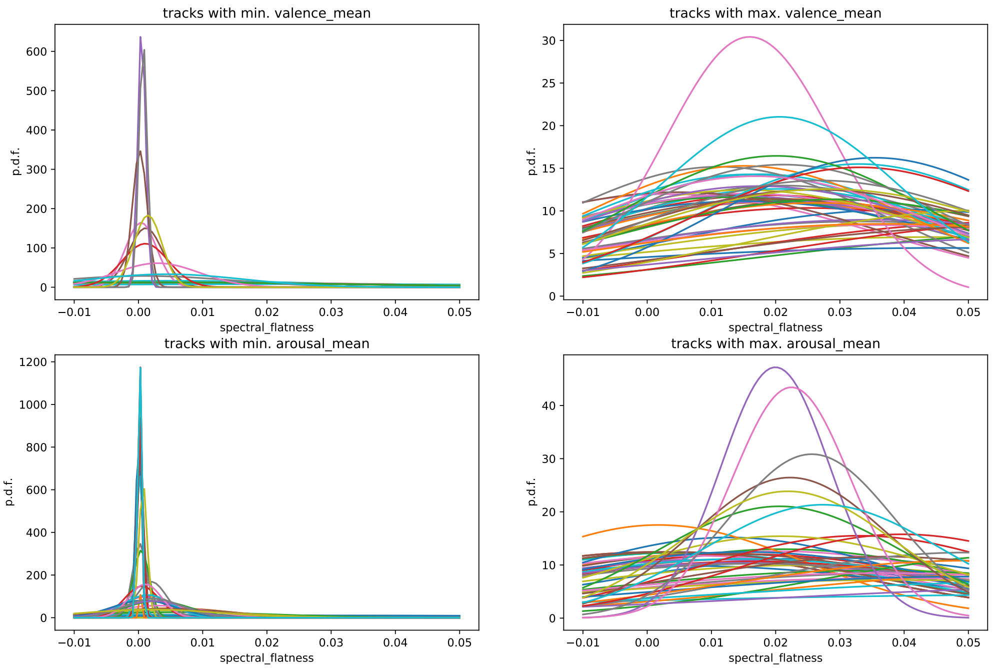

Figure 1: Spectral flatness distribution

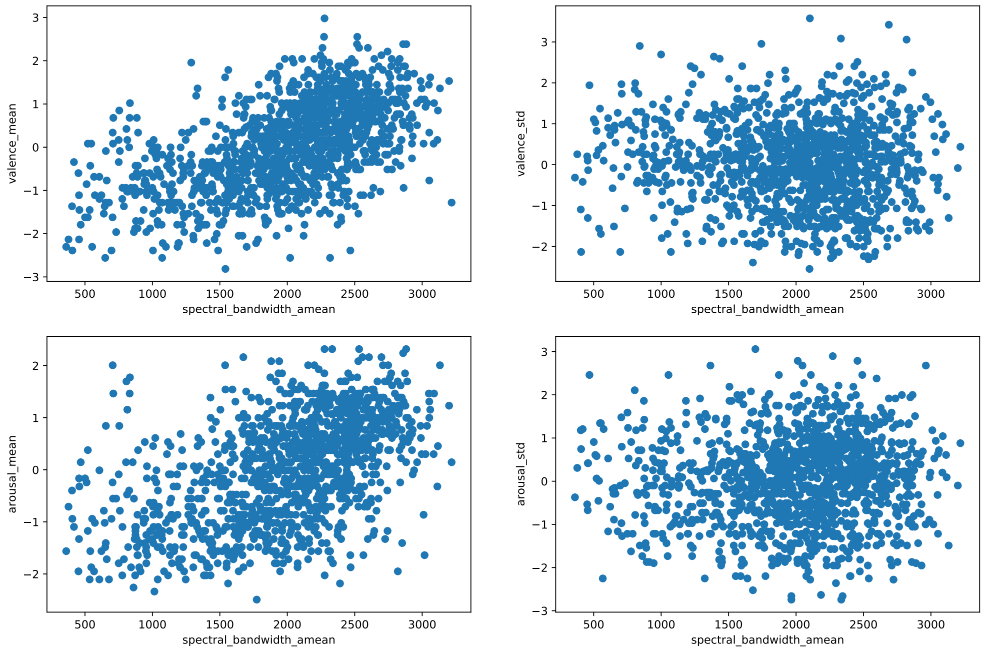

Figure 2: Spectral Bandwidth vs. Annotations scatter plot

For convenience, we arranged each feature processing step, including a standardization scaling (needed for some types of regressors, see section 3), in a sklearn `Pipeline` that is executed before each fitting stage and predicting stage of the regressors.

3 Regression models

Once features have been extracted, regression theory allows to predict a real value from a set of M given inputs, where M is the dimension of the features space. In this context, dimensional **Music Emotion Recognition** has been considered as a regression problem where distinct regressors are trained *independently* for valence and arousal. Although our focus was on Music Emotion Recognition for each song as a whole, the regression approach is also suitable for music emotion variation detection (**MEVD**), considering the frame-by-frame time evolution of features for each song.

In particular, four regressors

$$r\colon \mathbb{R}^M \to \mathbb{R}$$

will be trained to predict the four-element vector

$$[\text{valence mean, valence std, arousal mean, arousal std}]$$

Different regression models are used to find out which set of regressors best fit the data. In particular we focused on three families of regressors: *Linear Regressors*, *Support Vector Machines* and *K-Neighbors Regressors*. Performance evaluation will be achieved by means of the metrics represented by R^2 and MSE statistics (see section 4.1). For our experiments, we employed the Python library *Scikit-learn*, which integrates a wide range of state-of-the-art machine learning algorithms [9].

Linear models In the simplest linear model, coefficients are computed to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation. Other linear models allow to tune different open variables in order to better fit to the data, in particular we tried the *Ridge* regressor with built-in cross-validation and the *Stochastic Gradient Descent* regressor. The latter is set to be ϵ-insensitive, which is the same loss function used in SVR. The open variables of SGD are the threshold ϵ and the α constant that multiplies the regularization term, while Ridge has only α as open variable. The higher the value of α, the stronger the regularization. In particular, `RidgeCV` performs a cross-validation using, by default, values for $\alpha = 0.1, 1, 10$ (see section 4.2 for a complete description on cross-validation). It represented our first attempt because of the efficiency in computational terms and consequently had a role in facilitating the feature selection.

Support Vector Machines The idea behind SVM is to map the feature space to a higher dimension space, in order to be able to learn a nonlinear function by a linear learning machine in the kernel-induced feature space, where data are more separable [11]. Sklearn provides two types of support vector regressors, i.e. `NuSVR` and `SVR`. The former is an implementation of a ν-based SVM, while the latter implements an ϵ-based SVM. Both algorithms also have a penalty parameter C, which is inversely proportional to α of the previously mentioned linear models. Another free parameter in the SVM model is the kernel function. In our experiments we tried radial basis function (RBF) kernels and sigmoid kernels, as they were the ones requiring less computational power.

K-neighboors For variety, we also attempted using neighbors-based learning methods. The idea behind *K-Neighbors* models is to detect the closest sample points in distance to the considered sample point, and predict from them the target value. The main open parameter is thus k, which is the number of neighboors points to consider. Each sample's contribution can also be weighted. In particular, the sklearn implementation provides a “uniform” or a “distance” weighting. In the first mode each point has the same weight, whereas in the second one each point is weighted in an inversely proportional manner in respect to its distance to the considered sample point.

Cross-validation is used to find the best parameters for each regressor and the procedure will be described in detail in the next section.

4 Model training and evaluation

In order to avoid over-fitting and be able to evaluate our model on a real-world scenario, the dataset has to be initially shuffled (because the original dataset is ordered by genre) and split into a *training set* and a *testing set*. Consequently, every kind of consideration, analysis and training was made by using the training set only, to avoid information leaking from the testing set to the training set. A final testing can be done at the very last stage, once we are satisfied with the trained model and we are willing to test it against never seen data (see next section).

A powerful tool that can help us understand how to properly choose the features, the regressor and its parameters, and effectively evaluate the model before the final testing, is the *cross-validation*, explained in the following.

4.1 Metrics

In regression theory, there are two possible metrics that can be used to assess the accuracy of a regressor, which are different from the metrics used in classification theory, since the ground truth domain is continuous.

Mean Squared Error The MSE is a measure of the expected value of the quadratic error between each predicted value y_i and its true value $\hat{y}_i$, therefore a lower value indicates a better accuracy. Given N as the number of samples, it is mathematically computed as follows:

$$\text{MSE}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \qquad [0, \infty)$$

Since it is a scale-dependent metric, it is important to contextualize it into the scale of reference (i.e. a MSE of 0.2 is worse in a domain ranging from 1 to 10 than in one ranging from 1 to 100).

R^2-score The coefficient of determination is a measure of how well unseen samples are likely to be predicted by the model, and it represents the ratio between the variance of the predicted values and the variance that would be predicted by the model. Given N as the number of samples, and $\bar{y}$ as the mean of the predicted values, it is mathematically computed as follows:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{N} (y_i - \bar{y})^2} \qquad (-\infty, 1]$$

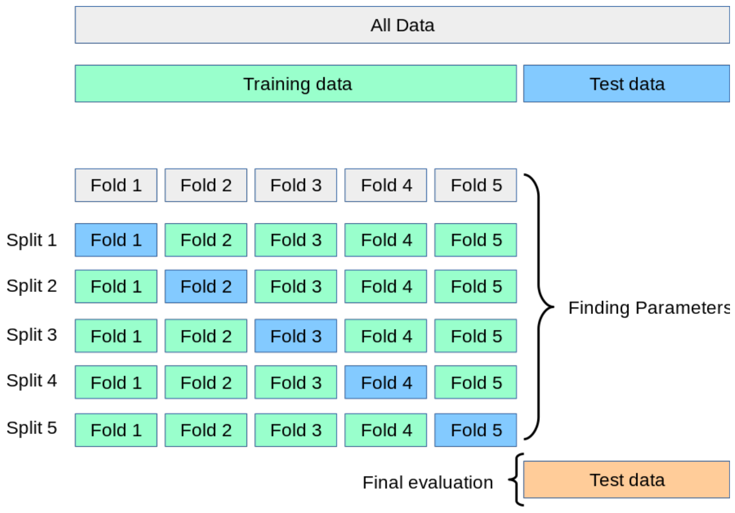

Figure 3: k-fold cross-validation example with $k = 5$ [1]

It is a scale-invariant metric, therefore it is possible to compare R^2-scores across different domain ranges, however it is variance-dependent, so it might not be meaningful to compare it across different datasets. A value of 1 indicates that the regressor perfectly predicts the ground truth. A value of 0 indicates that the regressor always predicts the mean value $\bar{y}$, disregarding the input features. A negative value indicates that the regressor behaves arbitrarily worse than predicting a constant.

4.2 Cross-Validation

It is a methodological error to evaluate metrics against the testing set and then use them to enhance the model, since this leaks information about the testing set which is supposed to be left unknown. This phenomenon is known as *over-fitting* and should be avoided.

One possible option to evaluate the model without using the testing set is to cut a separate sub-set from the training set, called *validation set*, and compute the metrics against this one instead. The drawback, however, is that this option further reduces the training set usable size.

A better approach consists in iteratively cutting a different small sub-set for validation from the training set, collect the metrics about each possible cut, and aggregate them in terms of mean and standard deviation (see figure 3). This way, the whole training set can be used. This procedure is known as *k-fold cross-validation*.

The gathered metrics can effectively be used to evaluate the performance of the chosen regressors over the dataset. At this point, it is possible to tune the regression model (i.e. by changing the free parameters of the regressors) without the risk of over-fitting (see figure 4).

4.3 Enhancing the model

The sklearn library provides a function `GridSearchCV()` for automatically selecting the best parameters for a given estimator, based on a grid of possible combinations.

At first, we computed the scores over a simple `LinearRegression` to evaluate the feature selection stage (see section 2.3). Then we took advantage of the grid search to first optimize the linear model by using a Stochastic Gradient Descent regressor and a Ridge regressor with integrated cross-validation optimization. We finally used again the grid search to find the suitable parameters (see section 3) for ν-SVM, ϵ-SVM and KN-regressor. For example, the best parameters found for the ν-SVM regressor are reported in tables 2.

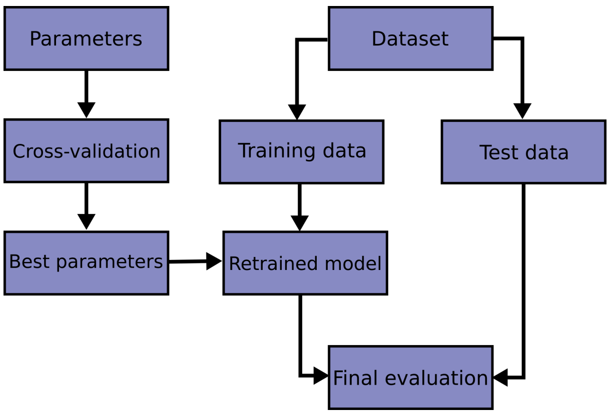

Figure 4: Grid Search Workflow [1]

	kernel	C	ν
valence mean	rbf	10	0.75
valence std	rbf	0.1	0.25
arousal mean	rbf	1	0.5
arousal std	rbf	0.1	0.75

Table 2: Best parameters for ν-SVM regressor

After refining the feature selection, we extracted the metrics of the best found estimators (see table 3). It can be observed that, even after several attempts of feature selection (both manual and automatic), and optimization through cross-validation, no regressor could obtain a significant positive R^2-score for the values of "valence std" and "arousal std", and the obtained R^2-scores for "valence mean" and "arousal mean" were still quite low. This might be a consequence of the poor quality of the provided dataset, which required some preprocessing step on it that we didn't manage to accomplish.

However, a meaningful evaluation of the trained model can be still obtained by comparing the scores of the cross-validation against the scores obtained by the final testing stage. This way, we can assess how worse the model will behave when we input never seen data (see section 5). In particular, given the results in table 3, we decided to use Ridge, ν-SVM and KN-regression as candidates for the final testing stage, as they seemed to be the best scoring regressors for each family of regression models considered.

	valence mean	valence std	arousal mean	arousal std
Linear	0.44 ± 0.19	-0.10 ± 0.12	0.39 ± 0.13	-0.13 ± 0.18
SGD	0.41 ± 0.25	-0.11 ± 0.14	0.36 ± 0.19	-0.16 ± 0.20
Ridge	0.46 ± 0.16	-0.04 ± 0.09	0.41 ± 0.14	-0.10 ± 0.16
ν-SVM	0.40 ± 0.18	0.01 ± 0.06	0.48 ± 0.16	0.01 ± 0.07
ϵ-SVM	0.42 ± 0.18	0.02 ± 0.05	0.41 ± 0.13	0.01 ± 0.07
K-neighbors	0.40 ± 0.08	-0.06 ± 0.09	0.40 ± 0.16	-0.07 ± 0.10

Table 3: Cross-validation R^2-scores with 95 % confidence intervals

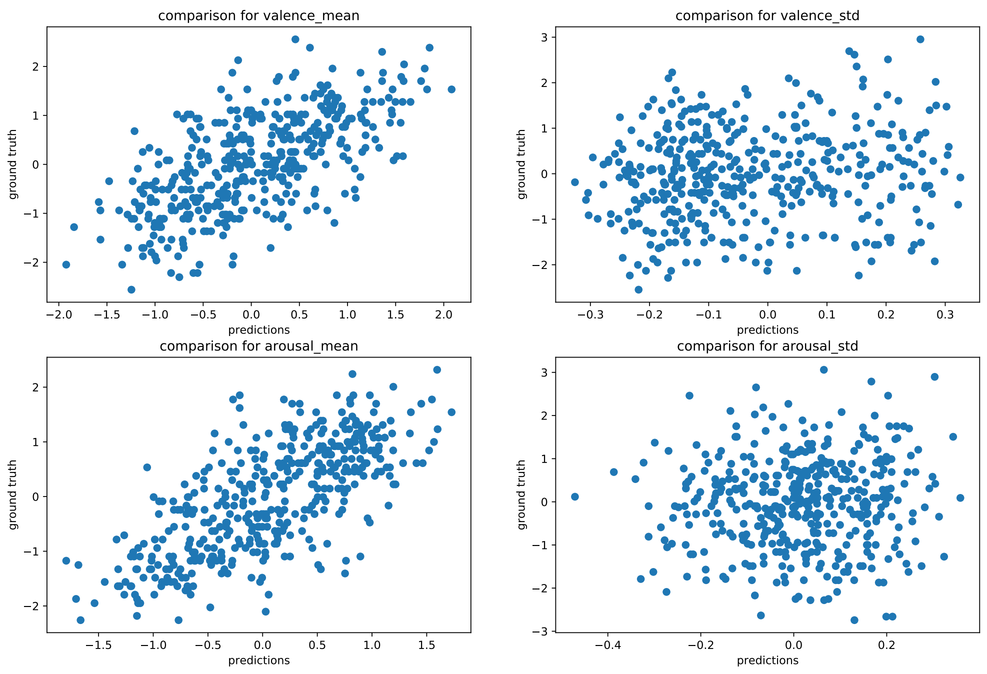

Figure 5: predictions vs. ground-truth scatter for SVM regression

5 Final testing and conclusions

Once we refined our model at the best could, we evaluated it against never seen data, by comparing the values predicted by the model, when the testing set is provided as input, against the true values of the annotations in the testing set. As already mentioned in section 4.2, once the final testing was done, we avoided going back to the model and make further tweaks, as this would have led to over-fitting.

For the final test we used the metrics of MSE and R^2-score (see section 4.1) and obtained the values reported in table 4, providing the differences in respect to table 3. It can be noticed that such differences fall between the confidence intervals provided by the cross-validation, therefore we can conclude that we still obtained a robust, albeit inaccurate, model, providing similar results in new unknown scenarios.

We also extracted scatter plots comparing the predictions of each regressor against the true values of the testing set. In figure 5 is depicted the case of ν-SVM, which turned out to be the best model fitting the data. In the best case scenario, these scatters should draw a 45° line. In our case, the scattering of “valence std” and “arousal std” clearly outline the bad R^2-scores obtained for these annotations.

In our assignment, it was requested to use the static annotations and elaborate a regression model over static features. A possible way to develop and improve the accuracy of the obtained models could be to adopt a dynamic approach, in which values related to temporal windows are considered instead, and it is left as future work.

	valence mean	valence std	arousal mean	arousal std
Ridge	0.53	1.09	0.54	1.18
ν-SVM	0.58	1.00	0.49	1.09
K-neighbors	0.62	1.00	0.61	1.08

(a) MSEs

	valence mean	valence std	arousal mean	arousal std
Ridge	0.45 (-0.01)	-0.08 (-0.04)	0.45 (+0.04)	-0.09 (-0.01)
ν-SVM	0.40 (+0.00)	0.01 (+0.00)	0.50 (+0.02)	0.00 (-0.01)
K-neighbors	0.35 (-0.05)	0.01 (+0.07)	0.39 (-0.01)	0.00 (+0.07)

(b) R^2-scores

Table 4: Final evaluation metrics

References

[1] Cross-validation: evaluating estimator performance. `https://scikit-learn.org/stable/modules/cross_validation.html`.

[2] Github repository of the assignment. `https://github.com/mttbernardini/cmls-hw1`.

[3] Jupyter notebook of the assignment. `https://mttbernardini.github.io/cmls-hw1`.

[4] Librosa: audio feature extraction. `https://zenodo.org/record/3606573`.

[5] Opensmile: audio feature extraction. `http://opensmile.sourceforge.net`.

[6] Anna Aljanaki, Yi-Hsuan Yang, and Mohammad Soleymani. Developing a benchmark for emotional analysis of music. *PloS one*, 12(3), 2017.

[7] Aljanaki Anna, Yang Yi-Hsuan, and Mohammad Soleymani. Emotion in music task at mediaeval 2015. In *Working Notes Proceedings of the MediaEval 2015 Workshop*, 2015.

[8] Rachel M Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. Medleydb: A multitrack dataset for annotation-intensive mir research. In *ISMIR*, volume 14, pages 155–160, 2014.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[10] M Soleymani, A Aljanaki, and YH Yang. Deam: Mediaeval database for emotional analysis in music, 2016.

[11] Yi-Hsuan Yang, Yu-Ching Lin, Ya-Fan Su, and Homer H Chen. A regression approach to music emotion recognition. *IEEE Transactions on audio, speech, and language processing*, 16(2):448–457, 2008.