Project Report TDDE19

Mattias Beming
Ted Gustafsson
Ludvig Joborn
Alfred Sporre
Caspian Süsskind

December 2021

1 Introduction

As the consumption and production of music moves toward digital forms, unprecedented opportunities for data collection and analysis have emerged. One interesting idea is to label music with the emotions that they bring to people. This idea, along with other music information retrieval, is explored in a paper by R.Castellon et al [1]. They manage to achieve promising results for several music information retrieval tasks, one of them being labeling the emotions of songs.

In order to retrieve and extrapolate information from music, there is typically a need for a lot of labeled data. The process of creating this data can be tedious and time consuming, and it is desirable to have to label as little data as possible. Furthermore, training models with a lot of data typically takes more resources either in terms of time, computing power or memory. One method that can be used to mitigate these problems is *active learning*. In active learning the goal is typically to choose what data to label so that as much as possible is gained in terms of model performance. The process is iterative, where an initial labeled dataset is the starting point. Then a model is trained using this data and a batch of unlabeled data is chosen for labeling based on some informativeness measure. The model is then retrained using the new larger labeled dataset and the performance is evaluated. This process is then repeated until some stopping condition is met.

The project described in this report is about combining active learning with the idea of labeling emotions in music. The aim is to create an application that can be used to efficiently choose songs for labeling so that a model can be trained to accurately predict the emotions of songs. To facilitate this, a dataset containing $1,000$ songs that are 45 second long with 90 emotion labels sampled at frequency 2. In order to get features that can be used to train machine learning models to predict emotions from these songs, feature extraction will be performed. To reduce the sheer amount of data used for training, feature selection will also be done using strategies such as principal component analysis (PCA) and variance threshold selection (VT). We also propose a way to capture the chronological correlation of emotions in music by enabling the data to be configured with a sliding window. To perform the active learning iterations several different regression models will be implemented as well several different approaches to choosing data points to label, so called *query strategies*. Finally, the different combinations of regression models, query strategies and other parameters will be evaluated and compared.

The hypothesis is that the results will vary quite a bit depending on which combinations of active learning query strategies are used along with which machine learning models and hyperparameters. It is also predicted that the active learning will help the model reach a good performance using only a subset of all the available data. Finally, it is predicted that the sliding window configuration will increase the performance of the models.

2 Methods

The methods and techniques that were used during the project are introduced in this section. First the data and some pre-processing steps are introduced followed by the architecture and the processes of the application. Finally some specifics about the methods and techniques that were used for active learning and machine learning are presented.

2.1 Dataset

The dataset used in the project is called *Emotion in Music Database* [5] or *Emomusic* for short. It consists of 1000 songs gathered from *Free Music Archive*[1], under CC license. However, duplicates were found in the data, and after their removal, only 744 songs remain. All the songs have a sampling frequency of 44100Hz and are 45 seconds in length, but due to instability of the annotations for the first 15 seconds, only the last 30 seconds are considered.

After successfully passing a qualification test, crowdworkers via Amazon Mechanical Turk[2] annotated the dataset. The labeling used are arousal and valence. Arousal measures how excited/annoying versus how calm/sleepy a song is, while valence measures how pleasing/relaxing versus how sad/nervous a song is, see Figure 1. The annotations were continuous throughout the song, and were

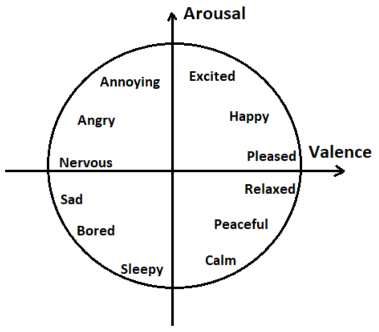


Figure 1: Emotional representation of valence and arousal.

done individually for arousal and valence. All annotations were provided on a scale from -1 to 1.

Along with the annotations, the standard deviation of the annotations were also reported. It was shown that the distribution of the annotations for each sample follows an approximate normal distribution. Thus, in an effort to remove outliers, only songs with all samples within the 99% confidence interval was used in this project.

Lastly, a dataframe with 6669 features [2] was provided along with the rest of the data.

[1]https://freemusicarchive.org/
[2]https://www.mturk.com/

2.2 Feature Extraction

In this project, we extracted features for all songs instead of using the provided set of features. The reasoning behind this was: 1) to enable feature extraction for songs not seen in the dataset; 2) the provided list is too large - ambiguity in what features to use.

For our own feature extraction we used the code provided from Free Music Archive [3], which needed a lot of modifications to fit this application. The library Librosa was used to extract interesting features [4].

Features were extracted in the following areas:

- Chroma
  Related to the 12 pitch classes.
- Tonnetz
  Computes the tonal centroid features.
- RMS - Root Mean Square
  Which is an indicator of the loudness in the music.
- MFCC - Mel-Frequency Cepstral Coefficients
  Which scales the signal to behave as in accordance with human hearing.
- Spectral
  Relates to analyses of different frequency spectrums.
- Zero-Crossing Rate
  Measures where signals cross between positive and negative.

And for each feature the following attributes were calculated:

- Mean
- Std
- Skew
- Kurtosis
- Median
- Min
- Max

This resulted 518 features instead of the 6669 from *Emo-music*, which significantly reduces the dimensions of the data.

[3] `https://github.com/mdeff/fma/tree/0ea2c9c83c84022fbf369e9dd258c7603baf33c4`
[4] `https://librosa.org/`

2.3 Feature Selection

This section aims to explain the chosen process of reducing the number of features, while still maintaining relevant information from the original features. Feature selection for this project consists of two different unsupervised feature selection techniques: *Principal component analysis* (PCA)[5] and *Variance threshold* (VT).[6]

2.3.1 PCA

PCA is a non-parametric technique for feature selection. As discussed by Schlens et al [4], this attribute is both a blessing and a curse: it is effective and easy to use but offers no way to use prior knowledge to control the principal components. The only choice that matters is the selection of the generated principal components.

In this project PCA is used in two ways, as a feature selection tool, and as a visualization tool. The basic concept of reducing the amount of features the data have is applicable to both cases, what differs is the root cause. Furthermore, with the purpose of using PCA as feature selection, the dimension reduction should not be too aggressive, since reducing dimensions often goes hand in with losing information. With the assumption that information of features is strongly correlated with the variance of the features, then one could rephrase the goal of feature selection in terms of PCA; select principal components which maximises the variance of the data. PCA in the context of visualization works in a similar way, the number of selected principal components get severely restricted by the number of dimension an human can make sense of and the complexity to plot high dimension on the monitor to the computer. Thus, in this context, only three principal components are chosen.

Visualization was done mostly for the presentation and as a reality check to ensure our training and test data split are following the same distribution. In the feature selection context, the PCA yielded in a ∼99% reduction in amount of features while still maintaining 99% variance of the data, i.e., PCA reduced the amount of features to 7, down from 518.

2.3.2 VT

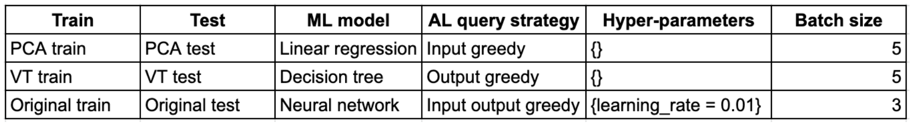
VT is often referred to as a baseline feature selection technique. Similarly as the PCA, it is based on the assumption that the variance of the data is correlated to the information contained in the data. VT calculates the variance for each feature and selects those features which have a variance over a specified threshold, this gives a similar control to the amount of features as with PCA. With a threshold of 100, the VT resulted in going from 518 features to 39.

[5] `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`

[6] `https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.VarianceThreshold.html`

2.4 Learning Profiles

The concept of a *learning profile* is something that is introduced in this project. A *learning profile* is an object that contains information about the following: what training and test datasets should be used for training and evaluation, what machine learning model should be trained and what active learning query strategy that should be used. A few examples of how a *learning profile* can look like is shown in Figure 2. Since there are several required fields in a *learning profile* there are many different combinations of *learning profiles* that can be created. In this section these different options will be further explained.



Train	Test	ML model	AL query strategy	Hyper-parameters	Batch size
PCA train	PCA test	Linear regression	Input greedy	{}	5
VT train	VT test	Decision tree	Output greedy	{}	5
Original train	Original test	Neural network	Input output greedy	{learning_rate = 0.01}	3

Figure 2: Examples of what learning profiles can look like.

2.4.1 Machine Learning Models

6 different multi-output regression models were implemented as part of the project. The implementations were made using the scikit-learn machine learning library [3]. The different models that were implemented are listed below.

- Linear Regression
- K-Nearest Neighbor (KNN)
- Neural Network
- Decision Trees
- Gradient Tree Boosting
- Ensemble Model

2.4.2 Active Learning

An important part of the project was pool-based active learning. Pool-based active learning is a form of active learning where the data to query on is chosen from a large pool of unlabeled data. All data in this unlabeled pool is considered for querying, and a batch of data points are selected according to some query strategy. An algorithm for pool-based active learning follows below.

1. Split the dataset D into an unlabeled pool U and a labeled seed S.
2. Train the chosen model using the seed S
3. Evaluate all data points in U and choose a batch of them according to a selected query strategy and a user defined batch size.

4. Query an oracle about the selected batch of data points
5. Move the selected batch of data points from U to S and add the oracle assigned labels to S as well.
6. Evaluate if more active learning iterations are required according to some pre-defined stopping condition. If so repeat the process from step 2, otherwise terminate the process.

In this project, 4 different query strategies were implemented with the purpose of comparing how well they work for the task at hand. Out of all the query strategies 3 of them are so called *greedy* strategies. The inspiration for the implementations of these came from a paper by D.Wu et al [6], but some modifications were made to fit the needs for this project. All 4 query strategies along with descriptions and details about their workings are found under the headings below.

Input Greedy Sampling
Input greedy sampling looks at the feature space when determining which points should be selected for querying. This has the advantage that the query strategy is not dependent on the model used in the active learning, and thus it has low computational cost. The process for selecting data points in the implementation used in this project is as follows.

1. Let U be the set of unlabeled data points and S be the set of labeled data points.
2. Calculate the distances $d_{ij} = ||u_i - s_j||$ for all data points $u_i \in U$ and all data points $s_j \in S$.
3. For each u_i find the minimum distance d_{ij} and assign this distance to u_i.
4. Sum up the assigned distances for every sample u_i belonging to a song and return the batch of songs that have the largest sums.

Output Greedy Sampling
Output greedy sampling looks at the labels and predictions when determining which points should be selected for querying. The aim is to increase the diversity in the output space. This strategy is model dependent and requires more computational power than input greedy sampling. The process for selecting data points in the implementation used in this project is as follows.

1. Let P be the set of predictions for the unlabeled data points and L be the set of labels for the labeled data points.
2. Calculate the distances $d_{ij} = ||p_i - l_j||$ for all predictions $p_i \in P$ and all labels $l_j \in L$.
3. For each p_i find the minimum distance d_{ij} and assign this distance to p_i.

4. Sum up the assigned distances for every prediction p_i belonging to a song and return the batch of songs that have the largest sums.

Input-output Greedy Sampling
Input-output greedy sampling looks at the labels and predictions as well as the features when determining which points should be selected for querying. The aim is to increase the diversity in the output space and input space. The process for selecting data points in the implementation used in this project is as follows.

1. Let U be the set of unlabeled data points and P be the set of predictions for the unlabeled data points, also let S be the set of labeled data points and L be the set of labels for the labeled data points.
2. Calculate the distances $d_{ij}^{u} = ||u_i - s_j||$ for all data points $u_i \in U$ and all data points $s_j \in S$.
3. Calculate the distances $d_{ij}^{p} = ||p_i - l_j||$ for all predictions $p_i \in P$ and all labels $l_j \in L$.
4. For each u_i find the minimum distance d_{ij}^{u}, also find the minimum distance d_{ij}^{p} and assign the minimum of the two distances to the sample.
5. Sum up the assigned distances for every sample belonging to a song and return the batch of songs that have the largest sums.

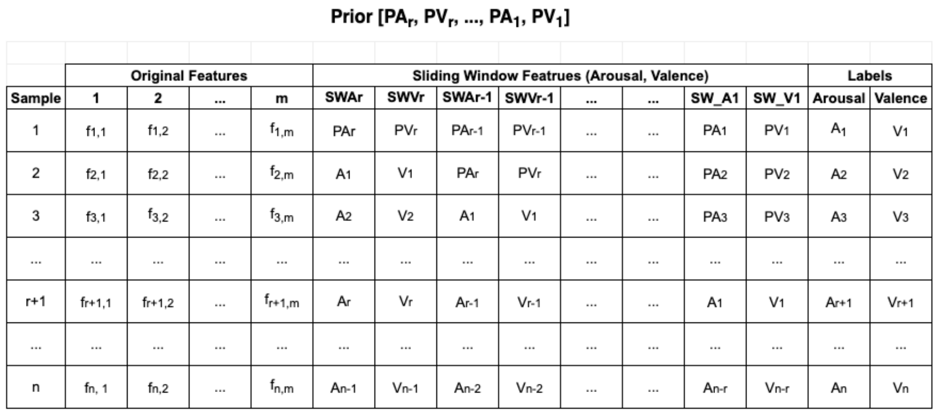
2.4.3 Datasets and Sliding Window

In the project three different kinds of datasets are used. One of these is a dataset that consists of extracted features from songs as well as labels for arousal and valence. The other two datasets consist of a subset of the features in the first dataset that are selected using PCA and variance threshold selection, as well as labels for arousal and valence.

In addition to the three kinds of datasets the ability to create more datasets exist by adding a sliding window to the different datasets. The motivation behind this is that there is a chronological correlation of emotion in music - if a song is sad, it is likely sad a few seconds later. In order to capture this correlation the data is augmented by adding indicators of the emotion in previous samples of a song to every sample. The idea is illustrated in Figure 3.

In the implementation a separation is made between how training and test data is configured with a sliding window. For training data the labels from the training data set are used as indicators of emotion for the previous samples, while for test data a basic linear regressor is used to predict the emotion of previous samples in order to get an indication of the emotion. The reason this separation is made is because, when using a model trained with a sliding window to predict on new unlabeled songs there is no way to configure the data in sliding window format using labels, since there are none. Thus, it made the most sense to use test data that is constructed without the use of labels.

In order to handle the fact that there is no natural good indicator of emotion for the first few samples in every song, an emotional prior was introduced. The prior was used to fill the slots for the sliding window features, as can be seen in Figure 3. The implementation allows for the prior to be passed in a few different formats, and the options vary depending on if the configuration is to be made for training or test data. The options for training data are, one prior for all songs and no prior at all. When no prior is passed, the labels from the first few samples are used and those samples are then not used during the training. The options for test data are either one prior for all songs or a different prior for each song. An overview of all the used datasets can be seen in Table 1.



Prior [PA_r, PV_r, ..., PA_1, PV_1]

	Original Features				Sliding Window Featrues (Arousal, Valence)								Labels	
Sample	**1**	**2**	**...**	**m**	**SWAr**	**SWVr**	**SWAr-1**	**SWVr-1**	**...**	**...**	**SW_A1**	**SW_V1**	**Arousal**	**Valence**
1	$f_{1,1}$	$f_{1,2}$	...	$f_{1,m}$	PA_r	PV_r	PA_{r-1}	PV_{r-1}	...	...	PA_1	PV_1	A_1	V_1
2	$f_{2,1}$	$f_{2,2}$	...	$f_{2,m}$	A_1	V_1	PA_r	PV_r	...	...	PA_2	PV_2	A_2	V_2
3	$f_{3,1}$	$f_{3,2}$	...	$f_{3,m}$	A_2	V_2	A_1	V_1	...	...	PA_3	PV_3	A_3	V_3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
r+1	$f_{r+1,1}$	$f_{r+1,2}$	...	$f_{r+1,m}$	A_r	V_r	A_{r-1}	V_{r-1}	...	...	A_1	V_1	A_{r+1}	V_{r+1}
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
n	$f_{n,1}$	$f_{n,2}$	...	$f_{n,m}$	A_{n-1}	V_{n-1}	A_{n-2}	V_{n-2}	...	...	A_{n-r}	V_{n-r}	A_n	V_n

Figure 3: Illustration of sliding window configuration for one song, with a prior.

Name	Feature selection	Sliding window	Train prior	Test prior.
ds1	PCA	No	-	-
ds2	VT	No	-	-
ds3	PCA	Yes	0	0
ds4	VT	Yes	0	0
ds5	PCA	Yes	T	0
ds6	VT	Yes	T	0
ds7	PCA	Yes	T	R
ds8	VT	Yes	T	R

Table 1: Description of the datasets, where 'T' = prior from target/labels of the data, and 'R' = prior predicted by linear regressor.

2.5 Model Selection Process

This process is used as a preliminary investigation into what learning profiles perform well on the *Emo-music* dataset. Our end goal is, however, to test the learning profiles against an annotating user, but doing this for every learning profile is unfeasible. (700+ songs with an optimistic annotation time per song on 1.5 minutes, we get over 16 hours of labeling. No thanks.) As such, we use this process to select a few promising learning profiles to use in model evaluation (see section 2.6).

2.5.1 Init Phase

The main purpose of the init phase is to initialize the *learning profiles* (see 2.4) that will be used in either viability and user phase. Here, an active learning method is bundled with a machine learning method, a training dataset, and a validation dataset (with or without sliding window applied, see 2.4.3). When a machine learning method has hyperparameters to vary, we create a learning profile for each set of hyperparameters we test for. All learning profiles have a batch size as well, but we use the same batch size for all learning profiles to simplify comparison between them. The created learning profiles is then used in the viability phase.

2.5.2 Viability Phase

In the viability phase, our learning profiles are tested against the labels in *Emo-music*. Using the methods and datasets bundled in a learning profile, we do the following:

1. Initialize a seed dataset with one random song.
2. Perform machine learning to get predictions on unlabeled data points. (Used in some active learning methods.)
3. Perform active learning to select a set of songs to query for labels from the *Emo-music* dataset. The number of songs selected is equal to the batch-size.
4. Add the selected songs to the seed dataset.
5. Perform machine learning to get new predictions on unlabeled data points and stores the MSE value that is measured on the validation data.
6. Repeat step 3 to 5 until there are no more unlabeled songs in the *emo-music* dataset.

The process above is performed with every learning profile.

2.5.3 Evaluation Phase

The evaluation phase is responsible for summing up the MSE values for each learning profile acquired in the validation phase. This corresponds to calculating the area-under-curve of the graphs discussed earlier. The idea is that the lower the area-under-curve, the better the learning profile performs on our task. We can use this to compare our dataset modifications, machine learning methods, and active learning methods. This can be presented visually to the user, which will be explained further in the *2.7 Presentation Process*.

2.6 Model Evaluation Process

This process involves users annotating songs, which is very time consuming. As such, we would like to evaluate only a selection of learning profiles in this process. In fact, we select a subset of the learning profiles that performed well in the model selection process.

To save time with the annotating part, a dictionary of all completed annotations are stored to disk. This is done to ensure that the user does not have to annotate the same song multiple times.

2.6.1 Init Phase

This phase is identical to the init phase described in section 2.5.1, except that we bundle each learning profile with a training+validation dataset and a testing dataset.

2.6.2 User Phase

This phase is very similar to the viability phase described in section 2.5.2, with a key difference. The difference is that in step 3, we do not query the labels from the *Emo-music* dataset. We instead query a user for labels using a GUI we developed for data annotation. These labels are then used to train the machine learning models. The GUI starts in one of two modes: arousal or valence. An image of the GUI can be seen in Figure 4.

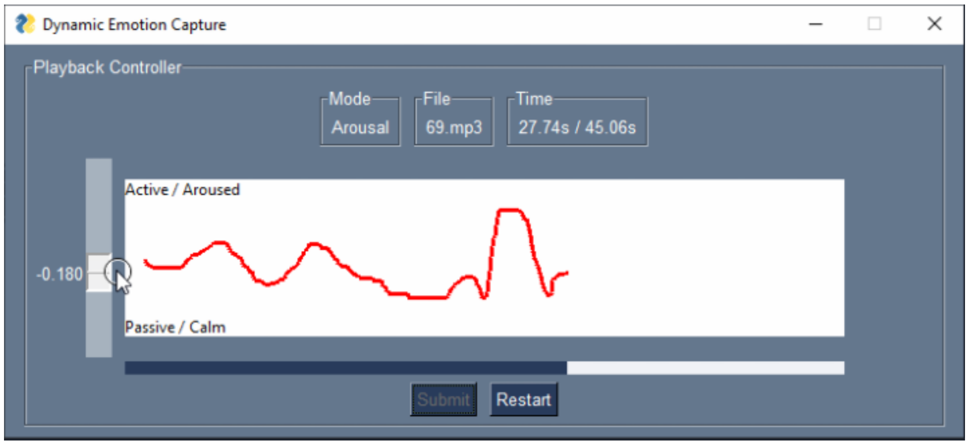


Figure 4: Annotation of arousal using the GUI for song 69.*mp*3.

2.6.3 Evaluation Phase

This phase is fully identical to the evaluation phase described in section 2.5.3.

2.7 Presentation Process

The presentation process is responsible for presenting the results to the user. The profiles that are presented here can either come from the output of the *2.5 Model Selection Process* or the *2.6 Model Evaluation Process*.

In this process, plots are shown to the user. This can be done based on three different evaluation modes: arousal, valence or mean. Furthermore, the possibility to choose from three different presentation modes exist. These modes will let the user filter learning profiles in order to compare either different datasets, machine learning models, or active learning techniques more clearly.

2.7.1 Presentation Phase

This phase allows the creation of graphs that plot the MSE scores of each model. One can choose between plotting all profiles in one graph, or plot based on a choice of a specific machine learning method; active learning method or specific dataset. The user is given the option to specify the number of profiles to be shown, which displays the best ones according to their area-under-curve score (calculated by summing the MSE score for each profile).

3 Results

In this section we present the results from Model Selection and Model Evaluation.

3.1 Model Selection Process

All models described below were tested with a batch size of 100 songs and a sliding-window size of 5.

A total of 384 learning profiles were tested with the *Emo-music* dataset. Among these, the best performing learning profiles used VT without sliding window and gradient tree boosting. The best learning profile achieved a RMSE of 0.1935 (MSE of 0.03744) for arousal and 0.2187 (MSE of 0.04782) for valence after full training.

Due to the sheer number of learning profiles, we will only present a subset of our results. For a full presentation of the performance of all learning profiles we refer the keen reader to run the presentation process and explore for oneself.

Figure 5 and 6 show the best learning profile for each machine learning method for arousal and valence respectively. Figure 7 and 8 show the best learning profile for each active learning method for arousal and valence respectively. Figure 9 and 10 show the best profile for each dataset.

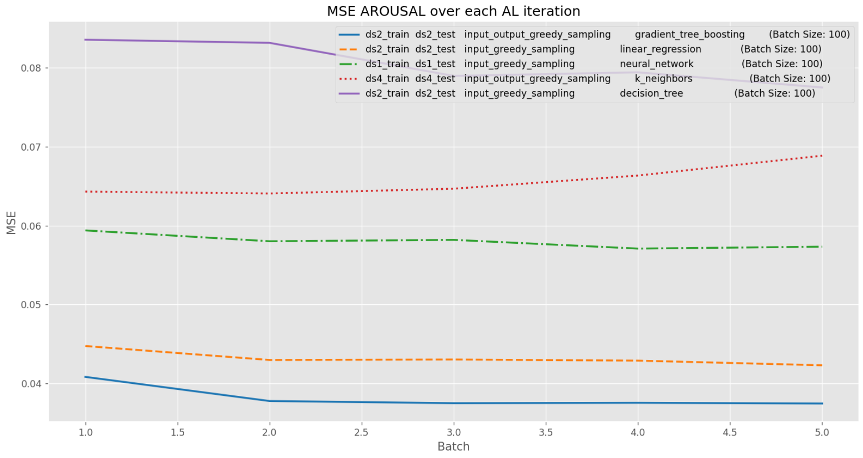


Figure 5: MSE values for arousal over each machine learning method's best learning profile (model selection).

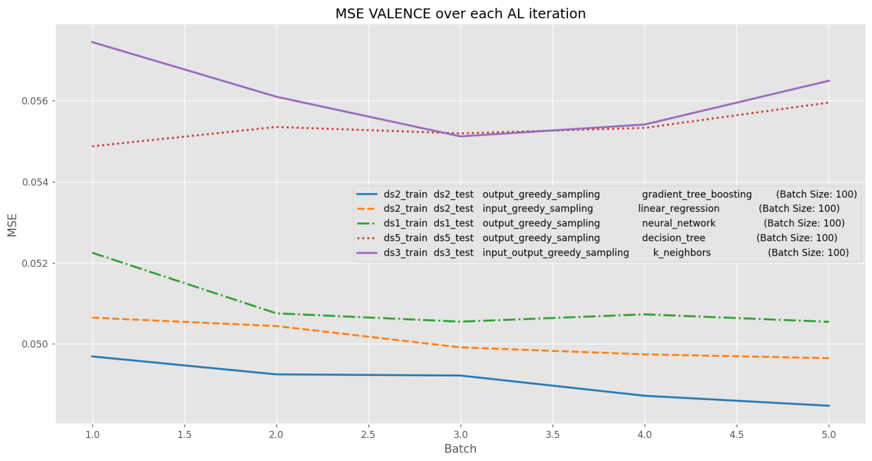


Figure 6: MSE values for valence over each machine learning method's best learning profile (model selection).

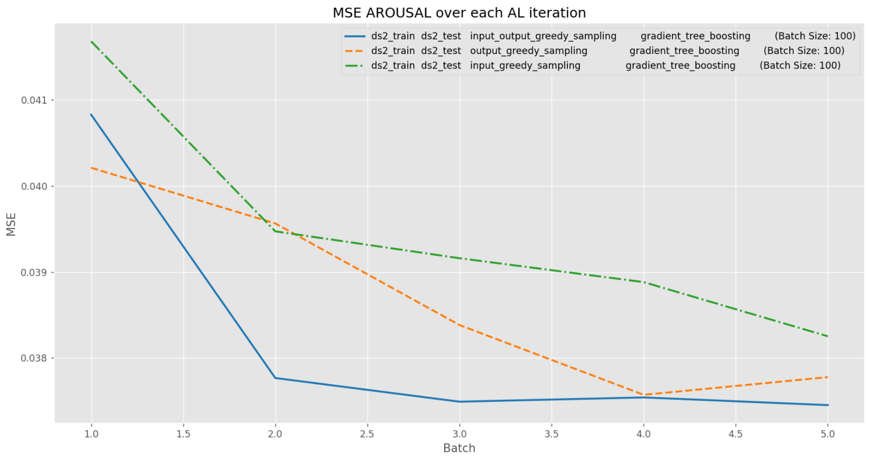


Figure 7: MSE values for arousal over each active learning method's best learning profile (model selection).

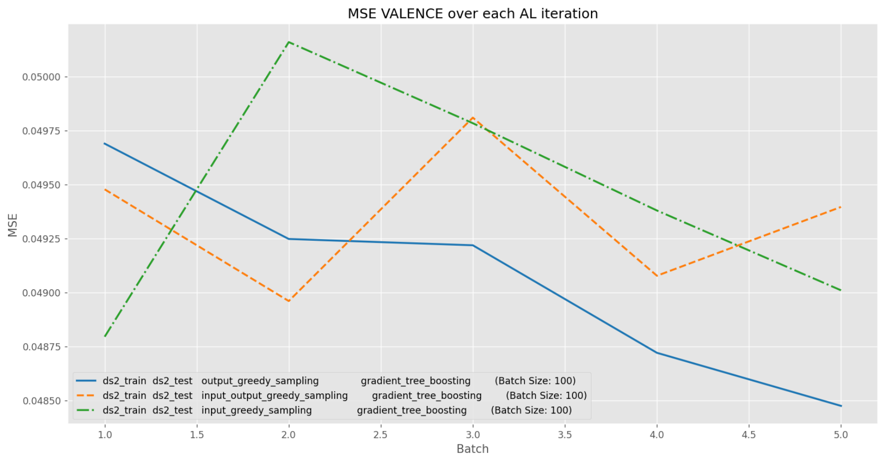


Figure 8: MSE values for valence over each active learning method's best learning profile (model selection).

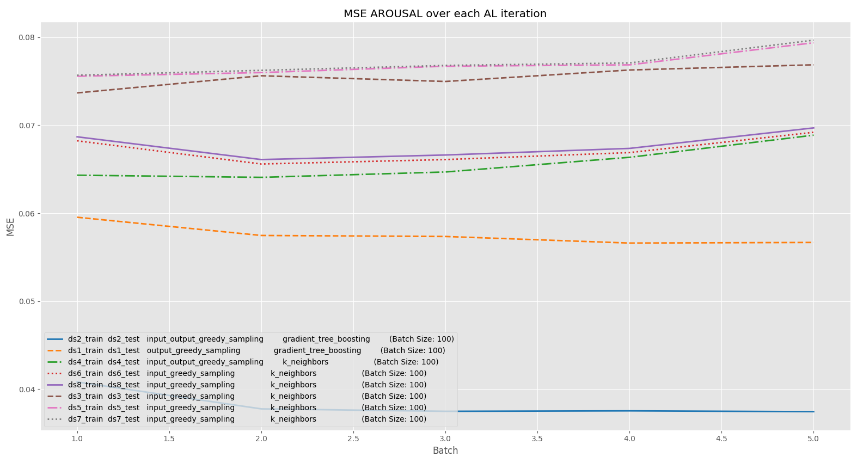


Figure 9: MSE values for arousal over each dataset's best learning profile (model selection).

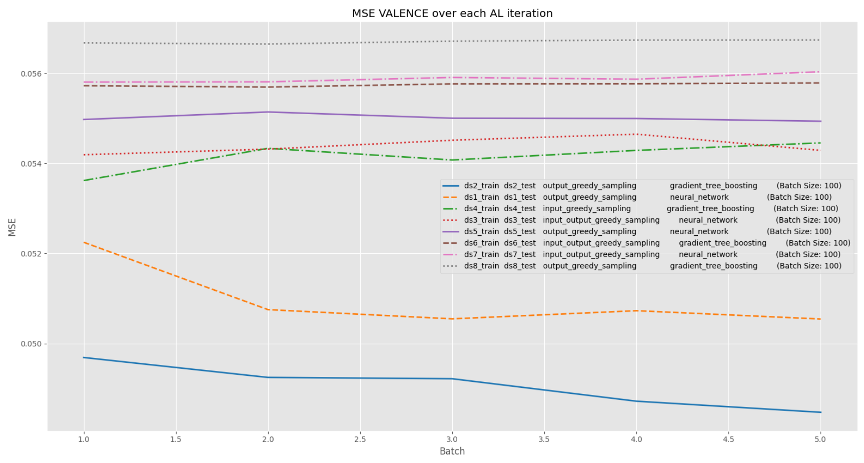


Figure 10: MSE values for valence over each dataset's best learning profile (model selection).

3.2 Model Evaluation Process

All models described below were tested with a batch size of 2 songs and a sliding-window size of 5. As described in 3.1, the interested reader should try running the presentation process oneself for a more complete presentation of the results.

Figure 11 and 12 show the best learning profiles for each machine learning method, Figure 13 and 14 show the best profiles for each active learning method, and Figure 15 and 16 show the best profiles for each dataset.

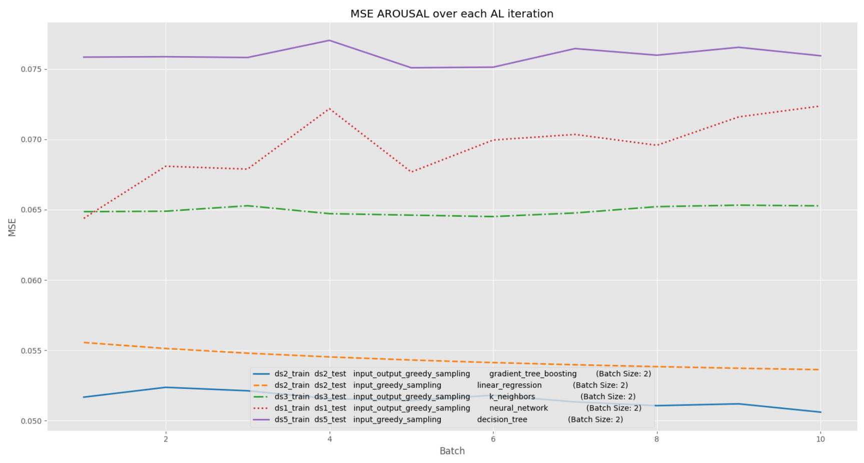


Figure 11: MSE values for arousal over each machine learning method's best learning profile (model evaluation).

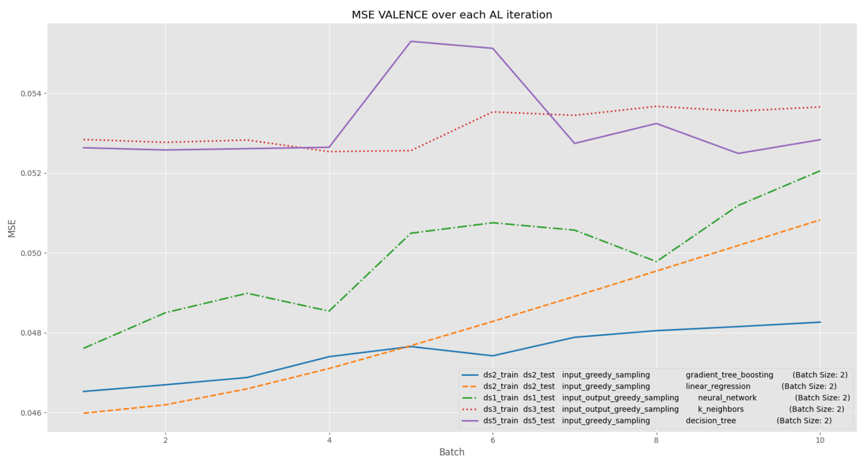


Figure 12: MSE values for valence over each machine learning method's best learning profile (model evaluation).

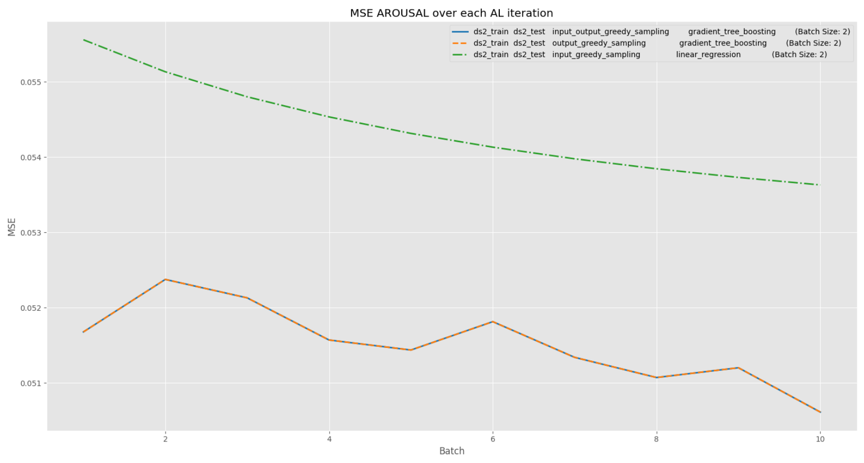


Figure 13: MSE values for arousal over each active learning method's best learning profile (model evaluation).

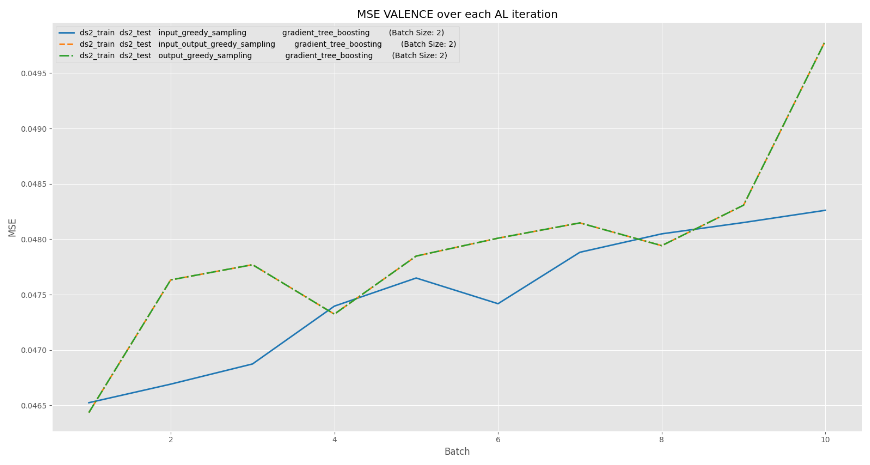


Figure 14: MSE values for valence over each active learning method's best learning profile (model evaluation).

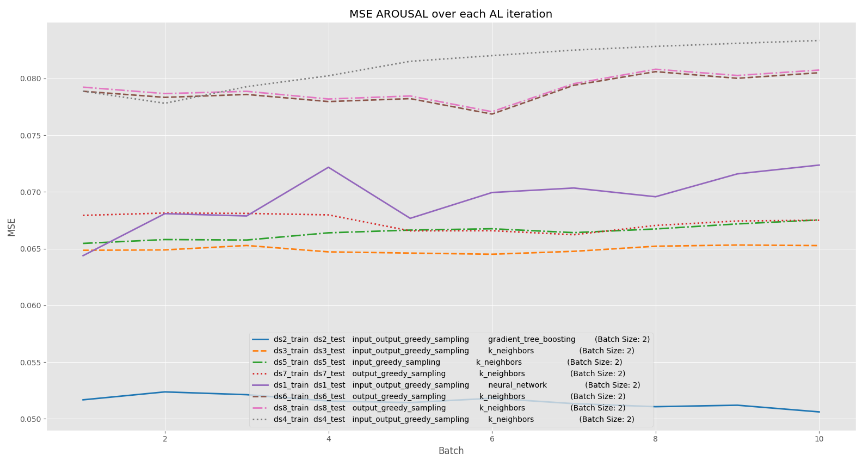


Figure 15: MSE values for arousal over each dataset's best learning profile (model evaluation).

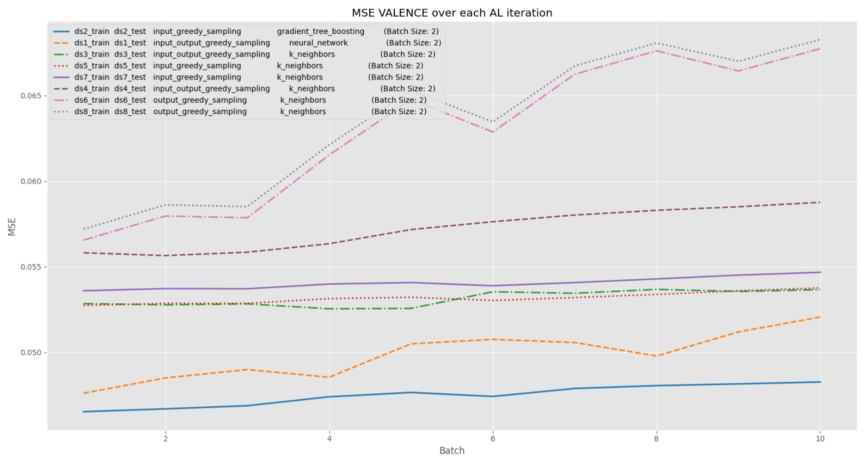


Figure 16: MSE values for valence over each dataset's best learning profile (model evaluation).

4 Discussion

In this sections the results are discussed and some conclusions about the project as well future work are presented.

4.1 Results

This section presents our discussion of the results.

4.1.1 Model Selection

It is of great interest that the best performing datasets did not use a sliding window. It seems then that most machine learning models fail to make good use of the sliding window values, which is not what was expected. This may in part be because of how test datasets with a sliding window are generated. The sliding window features are not actual target labels, they are predictions generated using a linear regression model. Thus, if the linear regressor provides bad predictions, the test data used for evaluation will not be similar to the data used when training the model. This difference in train and test data could explain why the datasets with a sliding window did not perform better. It is worth mentioning that it is not ideal to use a linear regressor when generating the test data, it would be better if the trained model that is to be evaluated was used instead. However, for implementation reasons this could not be realized in this project, but it is definitely something that should be considered in any future work.

Changing between the greedy active learning methods seemed to make very little difference in performance. At times one method selects better songs early, while others may do better later (as illustrated in Figure 17). None of these consistently outperformed the others. Unfortunately we did not have time to get ensemble models working, and could therefore not test maximum standard deviation sampling.

In general, predictions for arousal were better than those for valence. For example, the best model for arousal had a summed MSE of 0.191, while the best model for valence had a summed MSE of 0.245. These results match those derived by Soleymani et al [5]. This indicates that valence is to a larger extent prone to subjectivity, compared to arousal.

For most combinations of datasets and active learning methods, gradient tree boosting performed the best. The second best model was linear regression, outperforming neural nets and k-neighbors. Most machine learning seemed to already have good default values for their hyperparameter. k-neighbors performed the worst, and in fact gave increasingly worse prediction as the songs in the seed dataset continued to increase above 200.

One thing of great interest is that different machine learning models performed best on different variations of our datasets. For example, gradient tree boosting preferred the dataset using VT without sliding window, while k-neighbors preferred the dataset with PCA and sliding-window with no prior.

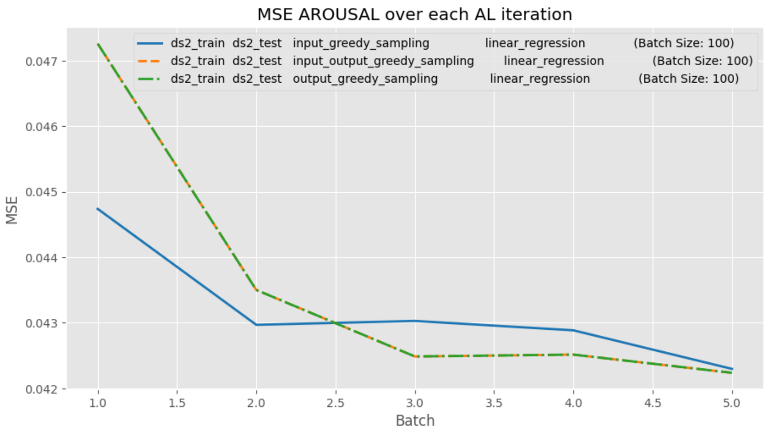


Figure 17: Comparison of active learning methods using linear regression (model selection).

But the best performing machine learning models (gradient tree boosting and linear regression) prefer datasets without sliding window. Additionally, datasets with VT consistently outperform those with PCA, indicating that PCA overly reduces the dimensionality of the data.

The most decisive factors in determining performance was which machine learning model and which dataset variation were used. The least important factors for performance were hyperparameters and active learning method.

4.1.2 Model Evaluation

It is of great interest that arousal MSE values stays even or decrease as we add songs to the dataset, while valence MSE increase as we add songs. This indicates that it is easier to have consensus on the arousal values than on the valence values.

It is unfortunately rather difficult to draw solid conclusions from the results of this process due to the limited number of annotated songs. Perhaps the initial seed dataset is too large in relation to the annotated songs. Perhaps it was a poor idea to split the annotating between multiple people, introducing additional inconsistency in the annotations.

4.2 Validity of Dataset

The dataset used in the project is somewhat diverse in what songs are included. Some songs are poor in recording quality, while others are not really songs, or parts of songs that does not contain clear characteristics of music. The genres of songs included in the dataset are ranging from classical- to rock-music, with everything in-between. Due to some quality inconsistencies, the overall results of the project might not be the most accurate. To conclude which learning profiles that performs best, different datasets should be tested to conclude a more balanced result.

4.3 Conclusions and Future Work

The implemented sliding window configurations did not increase performance of the models, the baseline datasets outperforms the sliding window variants in all cases. The lack of a baseline for active learning makes it harder to evaluate and compare different techniques. Thus, a baseline that samples random songs to be annotated instead of using active learning query strategies would be a great addition. This enables a fair comparison between the implemented query strategies and the baseline, and thus shows if our implementation was of benefit to the project.

As a consequences of the project's limitations and delimitations discussed in section 4.1.2, we are unable to use our results to answer whether active learning models can help find subsets of data such that test error converges. This is rather unfortunate, but due to time limitations we were unable to address this.

We would have liked to explore more active learning models, which would be a great way to extend to the project. We also had quite the explosion of number of learning profiles, which limited our ability to try various hyperparameters for our models. As such, they may not be fully optimized, which somewhat impairs our ability to draw general conclusions.

We would also like to test supervised feature extraction, as this would help us find key features while still reducing the dimensionality of our data. Also, another extension would be to extract more diverse features, such as timbral and rhythmic features. We could even include features such as genre or subgenre.

We would also like to try to use our processes to create personalized models. This would mean that we assign a model to a user, where the model queries the user and becomes better and better at predicting the emotions the user will feel on any given piece of music.

References

[1] Rodrigo Castellon, Chris Donahue, and Percy Liang. "Codified audio language modeling learns useful representations for music information retrieval". In: *arXiv preprint arXiv:2107.05677* (2021).

[2] Florian Eyben et al. "Recent Developments in openSMILE, the Munich Open-source Multimedia Feature Extractor". In: *Proceedings of the 21st ACM International Conference on Multimedia.* MM '13. Barcelona, Spain: ACM, 2013, pp. 835–838. ISBN: 978-1-4503-2404-5. DOI: `10.1145/2502081.2502224`. URL: `http://doi.acm.org/10.1145/2502081.2502224`.

[3] scikit-learn. *Machine Learning in Python.* URL: `https://scikit-learn.org/stable/`. (accessed: 2021-12-15).

[4] Jonathon Shlens. "A Tutorial on Principal Component Analysis". In: *Educational* 51 (Apr. 2014).

[5] Mohammad Soleymani et al. "1000 Songs for Emotional Analysis of Music". In: *Proceedings of the 2Nd ACM International Workshop on Crowdsourcing for Multimedia.* CrowdMM '13. Barcelona, Spain: ACM, 2013, pp. 1–6. ISBN: 978-1-4503-2396-3. DOI: `10.1145/2506364.2506365`. URL: `http://doi.acm.org/10.1145/2506364.2506365`.

[6] Dongrui Wu, Chin-Teng Lin, and Jian Huang. "Active learning for regression using greedy sampling". In: *Information Sciences* 474 (2019), pp. 90–105.