

Overview of Honors Capstone Research

Mel Mark

Park University

Spring 2020

Table of Contents

I.	Introduction	3
II.	Gathering of Information and Literature	4
III.	Processing and Selecting Appropriate Methodologies	5-7
IV.	Metadata Extraction.....	8-9
V.	Conversion.....	9-10
VI.	Data Refinement.....	10-11
VII.	Training and Applying Models.....	11-15
VIII.	Looking Forward.....	15-16

INTRODUCTION

In the fall semester of 2019 I started the implementation phase of my honors research project at Park University with advisor Professor Joe Wang. As a computer science major who plays the bass in their free time, I was very interested in the field of music information retrieval. Music information retrieval simply involves extracting data from music and manipulating or analyzing it in some fashion. Specifically, I was interested in assigning human emotions to music from all over the world with the help of machine learning. This was not a small task, and required lots of research and collaboration with researchers all over the world, including Switzerland, California, Poland, Spain, and New Zealand (Image 1). This semester could be mainly divided into five sections respectively: gathering of information and literature, processing and selecting appropriate methodologies, metadata extraction, conversion, and analysis.

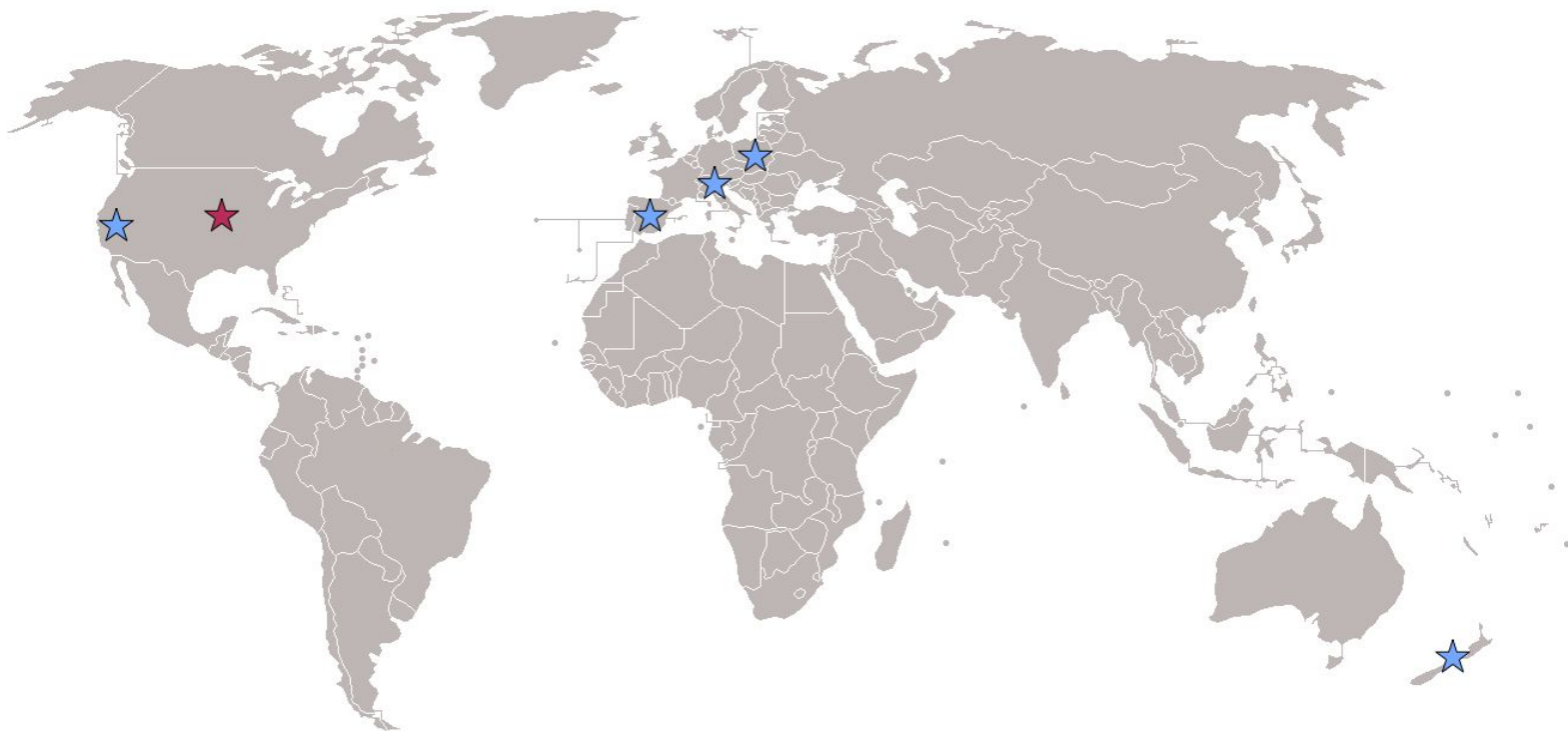


Image 1: Blue stars denote locations of some researchers I collaborated with, the maroon star is where Park University is located in Parkville, Missouri.

GATHERING OF INFORMATION AND LITERATURE

As any researcher knows, the first step to any problem is to gather relevant information and scholarly literature to aid in one's journey. This started with me browsing music information retrieval literature very broadly, looking carefully at datasets, intentions, and paths different researchers had taken. Then, as I began to narrow my search, I started to focus only on machine learning papers that specifically had to do with music and emotions. This proved to be difficult as it is a relatively new field that does not have a lot of substance in its libraries.

As I dove further and further into the literature, it became apparent that I was missing an understanding vital to this field: machine learning. As such, I began familiarizing myself with concepts and terms concerning machine learning by taking a course on Coursera and watching YouTube tutorials. This did help me understand some of the fundamental theories and practices behind machine learning, but it was difficult for me to wrap my head around a lot of it without tinkering with it myself. As a result, I began sifting through all the different methods and approaches various researchers before me had taken, searching for the one that would best work for my problem and that would allow me to put my new knowledge to the test. Eventually, I settled on the New Zealand tool Weka. Weka reads ARFF files, a file type only used for Weka, and is able to preprocess data, classify data, cluster data, associate data, and visualize data. I specifically liked this tool because it had a GUI as opposed to just a command line tool. It is also widely used across the field of machine learning and has reputable and accredited documentation and research.

PROCESSING AND SELECTING APPROPRIATE METHODOLOGIES

While the literature regarding emotive machine learning with music information retrieval is not abundant, there is a great variety in the approaches that are reported. Because of this, it took a lot of reading and searching to find which methods supposedly had the lowest margin of error and best datasets. Between the different regression algorithms implemented to the myriad of differences among datasets, it was overwhelming to choose one way or another when there is more than one way to bake a cake. Two papers that heavily assisted me in picking methods are “Detecting Emotion in Music” by Li and Ogihara and “Music Emotion Recognition: A State of the Art Review” from Ithaca College and Drexel University. Both papers detailed methods researchers experimented with before and included the results and accuracy of each.

Another field I was underprepared in was psychology. I have never taken a class in psychology and was not aware of the various models used to plot mood. As such, I had to read up on the many methods of enumerating mood and emotions on both two dimensional and three dimensional graphs. Eventually, I settled on a model known as Thayer’s Model to two dimensionally plot emotions into four categories (Thayer, 2000). The two axes represent valence and arousal, two measurements that ultimately determine which quadrant and which emotion. I then further divided the four quadrants into nine subsections each, with three emotions representing three subsections in each quadrant. While there are 36 subsections in 4 quadrants, there are only 12 moods represented (pleased, happy, excited, annoyed, angry, nervous, sad, bored, sleepy, relaxed, peaceful, calm) because 36 divided by the 3 subsections occupied by each mood equals 12.

Now that I had a model to plot emotions, I then had to find an appropriate dataset that had a good number of songs of the same length and annotated with valence and arousal values. This was vital to build my training set of data for the machine learning algorithms. This was a tall order to fill as this field is already so niche, there are not many datasets created, let alone public. After lots of digging, I eventually came across the DEAM: MediaEval Database for Emotional Analysis in Music (Soleymani, 2016). It was created by researchers in Switzerland and contained lots of information that was not relevant to my research. However, it did contain valence and arousal values for 1,745 songs and their 45 second audio files. It was not clear originally how they extracted the arousal and valence, so I attempted to contact them. The issue was that there was only one method of contact left on the paper, an email. I emailed the address listed but was sent back an error from the address's postmaster stating I was not whitelisted on their servers and could not send it emails. This was annoying because this was the only means of finding out this very important detail. I went to the Swiss university's website in which they conducted the research, and tried to find a technology office email so I could be either whitelisted or put in contact with this researcher. The entire website was in French, which I do not speak, so I had to translate it while navigating and eventually found an email to some generic office. I emailed them in both English and French but never got a response. In the meantime, I searched the Internet for the researcher's name and managed to find his personal website where he stated he was now a professor in California. I emailed him and he immediately got back to me stating that they had not extracted arousal and valence directly like I had thought, instead, they used turkers to manually extract the values. Of course, this is not ideal as it is subjective and not replicable. However, it was better than nothing.

The next step was plotting all the songs in the dataset on the two dimensional model by Thayer. To do this, I had to first find the origin of the graph, which was not (0, 0) as one might expect. Instead, I found the mean of each and made that the origin (4.9, 4.8). Then I took the minimum and maximum values of the arousal mean column and the valence mean column and subtracted it to find the differences. Once I had the differences, I divided them each by 6 (three subsections high and wide in each quadrant, times 2 equals 6) to find the increment. I then drew the boundaries for each subsection (Figure 1) and wrote a program in Python3.7 to automatically take in the valence and arousal mean columns and calculate and write each corresponding mood in a new CSV. Now that I knew they manually extracted the valence and arousal and I had annotated my dataset with moods according to Thayer's model, my next objective was to automatically extract the arousal and valence, which is much easier said than done.

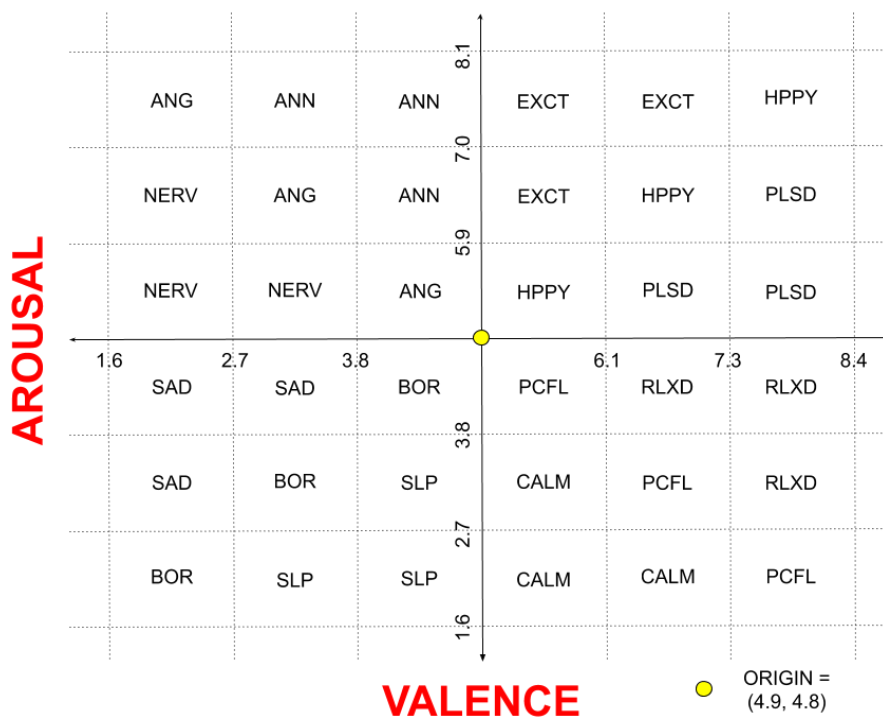
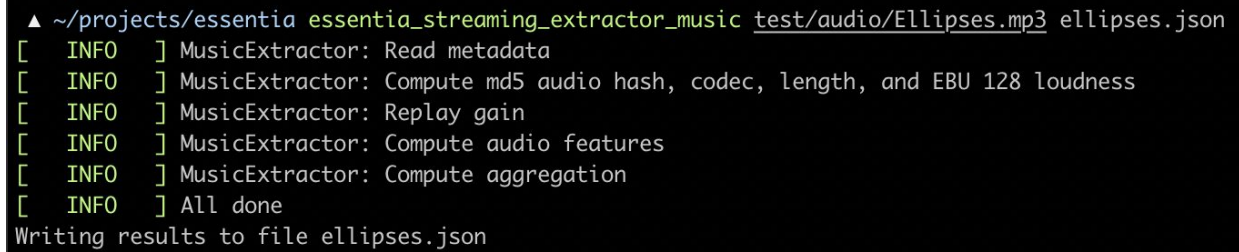


Figure 1: *Hand-mapping each quadrant, section, and subsection for DEAM.*

METADATA EXTRACTION

At the beginning of trying to extract music data, I naively thought that arousal and valence could be directly extracted from songs. Instead, what I learned is that arousal and valence are psychological terms that mean nothing to music. In order to calculate arousal and valence, calculations would have to be made from other aspects of music data. I, of course, had no idea how to do this so I began reviewing scholarly literature once again. Eventually I came across a gem, a Polish researcher named Jacek Grekow published a paper detailing differences in music metadata extractors and the pros and cons of each (Grekow, 2018). The two extractors he focused on were Marsyas and Essentia. Marsyas is a Greek command line tool written in C++ and Essentia is a Spanish command line tool written also in C++. Both extracted metadata from music, but Grekow noted that Essentia is more detailed and provides more features, which would ultimately lead to greater accuracy in detection of emotions. Because of this, I chose to use Essentia over Marsyas (in addition to Marsyas not wanting to compile correctly on my computer).

Essentia has three high-level categories it puts its features in: low-level, rhythm, and tonal. It takes the audio file and quickly and efficiently outputs the data in a JSON or YAML file (Image 2). In addition, Essentia calculates thorough statistics for each feature extracted, including mean, geometric mean, power mean, median of an array, all its moments up to the fifth order, energy, and the root mean square (RMS). The only issue with Essentia is that it outputted the data in only JSON or YAML, with no option for ARFF files, which are the only kind Weka can read. As a result, I had to figure out how to convert the files, which would prove a challenge.



```
▲ ~/projects/essentia essentia_streaming_extractor_music test/audio/Ellipses.mp3 ellipses.json
[ INFO ] MusicExtractor: Read metadata
[ INFO ] MusicExtractor: Compute md5 audio hash, codec, length, and EBU 128 loudness
[ INFO ] MusicExtractor: Replay gain
[ INFO ] MusicExtractor: Compute audio features
[ INFO ] MusicExtractor: Compute aggregation
[ INFO ] All done
Writing results to file ellipses.json
```

Image 2: *Essentia reading and computing data within seconds.*

CONVERSION

Because ARFF files are only for Weka, there are not any converters publicly available. Weka does have a built in ‘converter’ tool but it only works with very specific file types with very specific data. After conversing with some researchers who developed Weka, it was clear that I would need to build my own tool. I had never built a conversion tool before, but luckily both the input and the output types were not too complicated. ARFF files basically have a set of labels or ‘attributes’ that is paired to a data value, depending on position within a comma separated list. I had a wide variety of languages to choose from to write this program but I ultimately chose Python3.7 for its simplicity and robust documentation. In addition, I had some experience with Python and was comfortable writing in it.

The first challenge was extracting just the numbers of the JSON file Essentia outputted. I was not interested in the labels, just the numerical values and two non-numerical values (key signature and major/minor key). This was made even more difficult by the fact JSON files from Essentia contain multiple levels, and it is not possible to grab just the values for this to work. There was another challenge within this: some labels contained numbers, such as ‘dmean2’ or ‘melbands128’, so I had to somehow figure out how to differentiate numbers in labels and actual

numerical values. Also, a lot of numbers contained e+ or e- to denote scientific notation, so in addition to some labels containing numbers, some numbers contained letters. This was circumvented by a few conditional statements checking for those exceptions listed above. I also used some temporary lists and strings as placeholders to manipulate data before assigning the finalized data to the output variables.

Now that I was able to just extract the numbers, I had to write the actual converter that took the JSON data and output it to an ARFF file. In Essentia, all labels are the same for each song except one. There is a label called 'beats_position' that can vary with how many times it appears. Sometimes there are over 150 beats_position numbers, but sometimes there are under 75. I was able to write a loop that detected how many beats_position labels to generate in the ARFF files that solved that problem. Finally, I was able to open up successfully converted JSON → ARFF files in Weka and analyze them. Now I just needed to clean up the dataset acquired previously.

DATA REFINEMENT

Before I began analyzing and classifying data, I had to first ensure that my dataset was of the highest quality, so that my predictions would be as accurate as possible. Some of the moods the program assigned to songs were a little questionable. To avoid outliers or inaccurate data, I began the process of going through and listening to each 1,745 45 second audio clip. I marked each one I found accurate and took those songs and created an improved dataset (Image 3). Once this was done, I was able to more accurately predict emotions in songs in Weka.

song_id	valence_mean	arousal_mean	mood	accurate? y/n				
2	3.1	3	bored	y				
3	3.5	3.3	bored	y			number	percentage
4	5.7	5.5	happy	y		favor	15	35.71%
5	4.4	5.3	angry	n		disagreement	27	64.29%
7	5.8	6.4	excited	y		total	42	100.00%
8	3.2	4.8	nervous	n				
10	4	4.7	bored	y				
12	5.5	5.8	happy	n				
13	3.2	4	sad	n				
17	4.4	6	annoyed	n			ranked:	
18	4.8	3.9	bored	y		mood	# of instances	contribution
19	5.9	4.3	peaceful	n		happy	404	23.15%
20	5.4	6.5	excited	n		bored	243	13.93%
21	6.6	6.3	happy	n		sleepy	222	12.72%
22	4	4.8	angry	n		excited	189	10.83%
24	5.3	3.9	peaceful	n		angry	152	8.71%
25	7.9	4.7	relaxed	n		peaceful	125	7.16%
31	4.7	3.8	bored	n		sad	121	6.93%
32	4.2	3.3	sleepy	y		pleased	94	5.39%
35	4.8	6.5	annoyed	y		calm	57	3.27%
37	6.6	5.3	pleased	n		annoyed	57	3.27%
39	7.3	5.4	pleased	n		nervous	44	2.52%
40	4.6	4.8	angry	n		relaxed	25	1.43%

Image 3: *Going through each song individually and analyzing accuracy; this will take some time.*

TRAINING AND APPLYING MODELS

Finally, after fine-tuning my dataset, I had to decide which features to include in training my models. What I learned was that more input features do not necessarily equal more accurate predictions. Not all features are relevant to the predictions, and therefore can obfuscate the actual results. I referred to Jacek Grekow's paper, *Audio features dedicated to the detection and tracking of arousal and valence in musical compositions*, where he defines a list of features that are important for predicting valence and arousal, which is what I wanted based on Thayer's model. The 16 critical features he defines are segmented into three categories based on

Essentia's output: lowlevel, rhythm, and tonal. The 10 lowlevel critical features are melbands crest and kurtosis, spectral energy, entropy, flux, rolloff, and skewness, zero crossing rate, gfcc, and mfcc. The 3 rhythm critical features are danceability, onset rate, and beats loudness band ratio. The remaining 3 tonal critical features are chords strength, hpcp entropy, and key strength.

Based on these critical features, I was now ready to preprocess my data. Originally there were 4,377 features outputted by Essentia for each of the 120 MP3s, so I needed a way to quickly select thousands of features and remove them to slim down the data. Weka integrates PERL RegEx (Regular Expression) as a way to pattern-match features and manipulate them. I had never used PERL RegEx but I quickly learned and was able to trim it down quite sizably (Image 4).

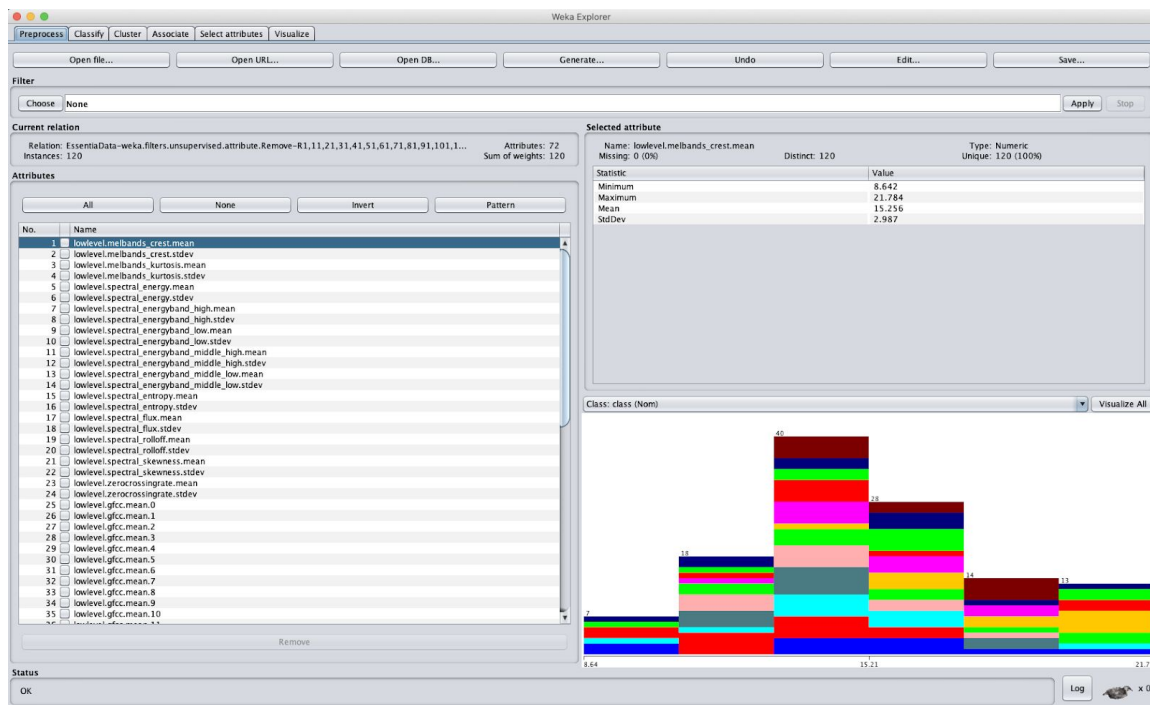


Image 4: Cherry picked features preprocessed with PERL RegEx in Weka.

Based on my initial literature gathering, I knew which machine learning algorithms I wanted to choose that would best match my specific problem. Of the dozens of algorithms, I settled on 4: sequential minimal optimization (SMO), neural network (what Weka calls multilayer perception), simple logistic, and J48 tree. The SMO algorithm is an iterative algorithm designed to solve the optimization problem regarding binary classification. It does this by breaking the problem into the smallest possible sub-problems before being analytically solved. Basically, it has to find a minimum of a one-dimensional quadratic function using Lagrange multipliers that meet the Karush–Kuhn–Tucker (KKT) conditions. Neural networks are loosely modeled off biological brains in its abilities to pick up new patterns without being explicitly taught. The simple logistic algorithm presumes that the connection between the measurement variable and the natural log (\ln) of the odds ratio is linear. Lastly, the J48 tree has the structure of a flowchart with nodes and internodes. Decision trees, like J48 trees, were more popular about ten years ago but still used today. Some of these algorithms are far more complicated than others, and all operate very differently, but they all predict fairly similar results most of the time. I applied each of the 120 songs' data to each algorithm and generated 4 models (Image 5). Finally, I was then able to apply those models to new songs and output predictions.

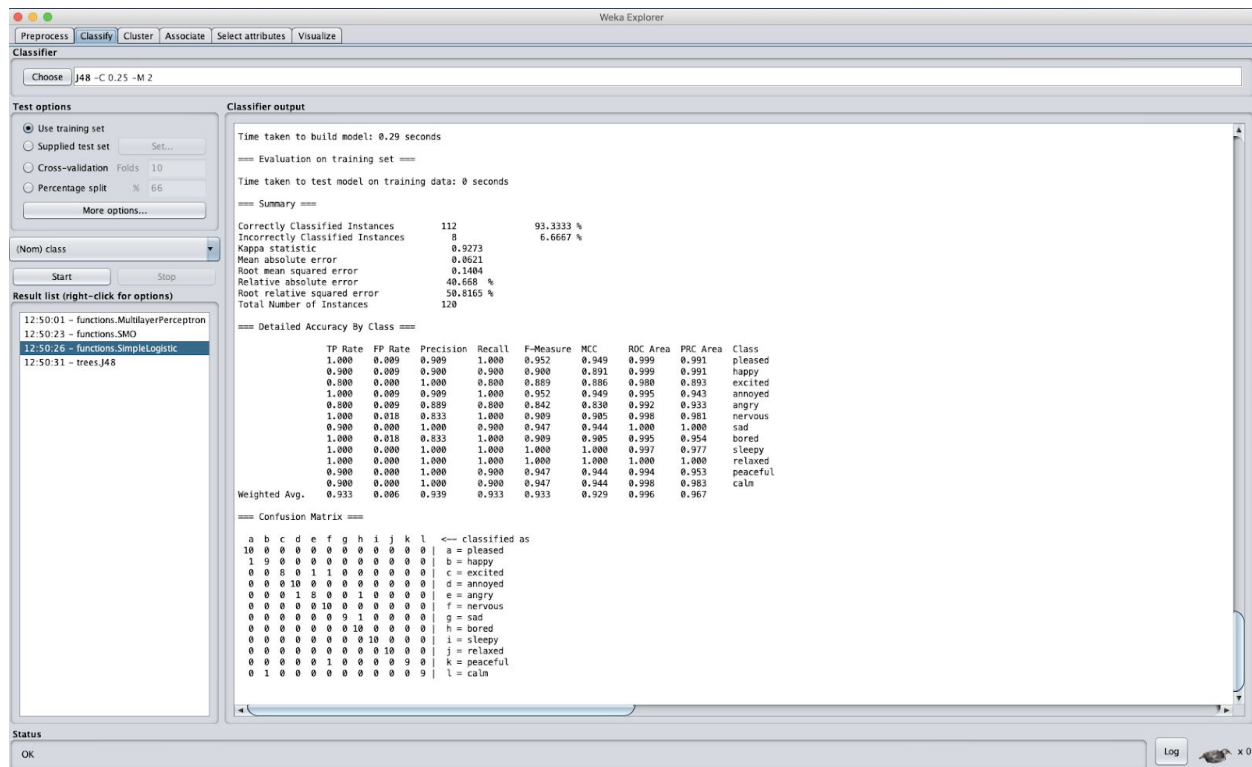


Image 5: Classifications and confusion matrices with SMO, J48 tree, simple logistic, and neural network algorithms.

I chose a few songs to initially analyze against the models, and I was satisfied with the models' predictions because I generally agreed. A song that I personally really like but is also popular is *Juice* by Lizzo, also known as Melissa Jefferson. I divided the song into 9 sections: intro, first verse, first pre-chorus, first chorus, second verse, second pre-chorus, second chorus, bridge, and last chorus respectively. After segmentation I analyzed each one against the models in addition to the entire 3 minute song as a whole (Figure 2). What was interesting was that the sections that repeated, like pre-choruses and choruses, had pretty much the same results, with the same levels of confidence. The song ranked 'pleased' across the board with the exception of the

second verse, which was predicted as 64% ‘angry’ and 36% ‘happy’. The second verse of the song is when the mood shifts a bit as Lizzo gets more sassy, and it is noticeable when listened to. Therefore, I was very satisfied with the models’ results with this song, as it was able to, in my opinion, accurately map the mood of the song over time.

type of ML	juice_o_g	juice_1_intro	juice_2_verse1	juice_3_prechorus	juice_4_chorus	juice_5_verse2	juice_6_prechorus	juice_7_chorus	juice_8_bridge	juice_9_chorus
SMO	pleased, .167	pleased, .167	pleased, .167	pleased, .167	pleased, .167	happy, .167	pleased, .167	pleased, .167	pleased, .167	pleased, .152
simple logistic	pleased, .821	pleased, .972	pleased, .969	pleased, .664	pleased, .737	happy, .426	pleased, .341	pleased, .678	pleased, .976	pleased, .311
neural network	pleased, .962	pleased, .541	pleased, .845	pleased, .739	pleased, .777	angry, .406	pleased, .718	pleased, .81	pleased, .978	pleased, .64
J48	angry, .667	happy, 1	excited, .667	pleased, 1	angry, .667	angry, .667	pleased, 1	angry, .667	angry, .667	angry, .667
result with weights	75% pleased	63% pleased	75% pleased	100% pleased	72% pleased	36% happy	100% pleased	71% pleased	76% pleased	62% pleased
	25% angry	37% happy	25% excited		28% angry	64% angry		29% angry	24% angry	38% angry

Figure 2: *Juice by Lizzo segmented based on song structure and analyzed by each model.*

LOOKING FORWARD

Ideally, I would like to further segment more songs of a greater diversity to make more predictions over time. I would also like to explore more ways to tweak my models to make them more accurate. Another thing I would like to do is automate the process. It currently takes me hours to take the MP3, trim it, run it through Essentia to generate a JSON, put the JSON file through my Python converter to generate an ARFF, then preprocess the ARFF in Weka using

Perl RegEx, apply it to each model, and record the results. I have the experience in Python automation and I know I could do it, but it is simply a matter of time and resources.

There are a wide variety of applications for music, and therefore lots of applications for this research. It would satisfy me to apply this research to something that would benefit the community, however that may be.

References

- Grekow, J. (2018). Audio features dedicated to the detection and tracking of arousal and valence in musical compositions. *Journal of Information and Telecommunication*, 2(3), 322-333.
- Jefferson, M. (2019). Juice [Recorded by Lizzo]. On Cuz I Love You [MP3]. New York City, NY: Atlantic Records. (2018 December)
- Kim, Y. E., Schmidt, E. M., Migneco, R., Morton, B. G., Richardson, P., Scott, J., ... & Turnbull, D. (2010, August). Music emotion recognition: A state of the art review. In *Proc. ISMIR* (Vol. 86, pp. 937-952).
- Li, T., & Ogihara, M. (2003). Detecting emotion in music.
- Soleymani, M., Aljanaki, A., & Yang, Y. H. (2016). DEAM: MediaEval Database for Emotional Analysis in Music.
- Thayer, J. F., & Lane, R. D. (2000). A model of neurovisceral integration in emotion regulation and dysregulation. *Journal of affective disorders*, 61(3), 201-216.