Classifiying Musical Valence using Song Lyrics and Spotify Audio Features

Kidd Jason Evangelista and Juan Miguel Bawagan III

***Abstract*— This study is aimed to discuss the existing problem of sentiment analysis from music, specifically from song lyrics, and to provide an alternative solution by classifying music through musical valence. The classification algorithms used in this study are Multinomial Naive Bayes and Support Vector Machine. Combined with the audio features retrieved from Spotify's API, lyrics were used to train models for classification. A web application was also made to supplement this research to showcase said models and to test the accuracy of predictions made versus real valence values.**

***Index Terms*— valence, musical valence, sentiment analysis, multinomial naive bayes, support vector machine, binary classification**

I. Introduction

A. Background of the Study

Machine learning is one of the many fields in Computer Science that is rapidly growing as an increasing number of researchers develop more efficient algorithms, particularly to help researches in its vast applications' development. Some of its interesting applications are Natural Language Processing (NLP) and Sentiment Analysis, wherein the latter detects and extracts subjective information (e.g. emotions, opinions) from text [1]. Sentiment Analysis is often used as means to classify the ideas of a certain population and to analyze trends currently existing. It is applied to a range of text-based models such as reviews, tweets, and social media posts which are good sources of emotionally-loaded opinions.

An interesting source of emotion-based text are present in music, specifically through songs. Recent studies have applied Sentiment Analysis in identifying and classifying emotions from lyrics. These studies share the similar struggle of accurately presenting the emotions conveyed in songs without considering the musical elements contained within them such as pitch, tempo, energy, danceability, etc. Text alone is not enough to accurately capture the emotions in a musical piece because music is highly subjective, thus it is challenging to quantify [2]. Fortunately, Spotify, a music streaming service, has an API that allows the retrieval of a track's audio features. Audio features include energy, danceability, loudness, tempo, and more. A certain feature that is of utmost importance to us is *valence*. It is defined as "a measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry)" [3]. Valence signifies a simplified method of classifying emotions and quantifying it means that we can directly use this to label the positivity or negativity of songs.

In this study, we will attempt to predict song valency through the use of classification techniques such as Multinomial Naive Bayes and Support Vector Machine. We shall measure and compare its accuracy from existing systems that predict song sentiments and determine whether it is suitable to use valency to classify songs or not.

B. Significance of the Study

Applications that deal with extracting emotions from texts present two problems. First, how do we know which emotions appropriately describe a specific text. And second, given a certain text passage and its corresponding emotion, how do we render its structure in order to know its truest meaning [4]. This is quite a challenge because lyrical text is distinct from ordinary text. Lyrical text possess the use of stylistic qualities such as rhyme, poetic form, and the use of figurative language like metaphor, irony, and sarcasm [5]. A conventional way of solving these problems is through the use of lexicons. We keep a dictionary of words that are created either manually or automatically, wherein the words are classified by the emotions they represent. The dictionary is then used to calculate the sentiment from the semantic orientation of words or phrases that occur in a text [6].

Identifying words and mapping them correspondingly to specific emotions are subjective. Hence, results will heavily depend on the means of creating the dictionary - whether it has captured the proper segregation of emotions from words or not. For example, Ahire identified three ways of building lexicons. First is the manual creation of lexicons and is the most subjective of the three. To achieve robustness in the results, multiple annotators can be asked to perform the task. Second is the automatic creation of lexicons which involves the automatic expansion of "seed words" that are incorporated with a known sentiment orientation. And third is through the use of SO-CAL or the Sentiment Orientation Calculator [7]. The variety of ways in creating lexicons contributes to the inconsistency of training a model for emotion classification.

Generally, the flow of existing sentiment analysis algorithms start by retrieving a set of data called the training dataset, where it can be labeled or unlabeled. The retrieved data is represented by a certain model (e.g. bag-of-words, lexicons, vectors) and are separated accordingly based on emotions (via labels). This model will be the basis for classification of test

Presented to the Faculty of the Institute of Computer Science, University of the Philippines Los Baños in partial fulfillment of the requirements for the Degree of Bachelor of Science in Computer Science

datasets and for extraction of sentiments from text through the use of Machine Learning (ML) algorithms such as the Naive Bayes, Maximum Entropy, and Support Vector Machine.

With the introduction of a measurable valence value through the Spotify API, we are able to present a justified and simplified classification label between positiveness and negativeness for our training data, while utilizing the track's musical attributes. Therefore, we eliminate subjectivity, and avoid the use of models that resort in manual segregation of emotion from words.

C. Objectives of the Study

The primary objectives of this study are as follows: a) to predict positive or negative valence of a song given its lyrics, and b) to determine if valence is a suitable classification label for song sentiment analysis, and c) to criticize Spotify audio features as classification features and discuss possible future works.

D. Scope and Limitations

While this study presents an alternate solution to the problems present in sentiment extraction, it suffers from a number of deficiencies, primarily because of its simplistic nature.

For instance, valence does not capture music's figurative nature. A happy-sounding song will have a positive valence, and conversely, a sad-sounding song will have a negative equivalent. By using, valence as a measure of positiveness and negativeness, we disregard the figurative elements of music, most especially in terms of sarcasm and irony in its lyrics. A sad-sounding song may appear to have lyrics that uplifts one's spirit, and a happy-sounding song may appear to have the most depressing lyrics. We eliminate these possibilities as we only examine the trend in lyrics that is present within the songs, regardless of their true emotional content.

Another problem that arises is the simplicity of valence to "classify" emotions. While we can hypothetically classify and label songs accurately, we only classify as either positive or negative. It does not determine which specific emotion is being portrayed in a given lyrical text.

However, the most obvious drawback is that we can not classify music that does not contain lyrics and those music that contain gibberish words as lyrics. Analyzing the musical features of a song itself and actively predicting a value for valence is not included in this study. For simplistic purposes, we shall only consider songs in the English language only. Though, it is also possible to use the same methodology when classifying songs that are in other languages.

II. Review of Related Literature

Digital music libraries, undeniably plays an important role music feature extraction research. With the rapid development of vast and easily-accessible digital music libraries over the past decade, there has also been a rapid expansion in music information retrieval systems [2]. Existing APIs provide researchers musical information such as lyrics, genre, song reviews, etc. These researches heavily depend on these information retrieval systems. In the vastness of this field, however, emotion classification from text prove to be more difficult compared to systems involving genre classification. Musically, lyrics are created with such freedom that it can either be created with such structure as the likes of writing poetry, or it can created without the restraints of having a form. Thus, emotion extraction from lyrics tend to be problematic. Some studies have acknowledged this, and as a result have shied away from solving the problem through sentiment analysis alone. Others, however, continued to present new approaches in this field.

Oudenne and Chasins realized that lyrics are difficult to analyze for sentiment and, as a result, systems that rely on lyrical analysis alone for classification will yield poor results [8]. In their study, training data were classified by positive and negative emotions using Jan Wiebe's subjectivity lexicon and noted that lexicons used [in any system] should be corpus-specific because general lexicons that are applied in specific documents tend to wield less accurate results.They then tested their data with different algorithms including Present In One, Prevalent In One, Naive Bayes, TF-IDF Cosine Similarity, and WordNet Cosine Similarity algorithms, achieving 52%, 67%, 66%, 57%, and 52% maximum accuracy respectively.

Instead of using a model like lexicons, some researchers use social tags instead. Tags such as "happy" and "sad" are clearly useful for emotion recognition. Research involving the analysis of social tags has focused on grouping them into distinct emotions since their are numerous tags containing similar terms such as "merry", "joyful", and "cheerful" [2]. It is also problematic because social tags contain numerous irrelevant information within a song because originally, tags are designed to organize personal content. Twitter hash tags, for example, are used to organize data, and to group tweets that are related by a central topic.

While there are related studies that focus only in analysis of lyric-based text, some studies utilize the audio analysis of songs. In 2003, features like tempo, timbre, harmony, and loudness were used to train Support Vector Machines using a hand-labeled library of 499 music clips to classify music into one of 13 mood categories [9] by Li and Ogihara and have achieved an accuracy of 45%. A similar study conducted by Lu et al. made use of acoustic features including intensity, timbre, and rhythm with the introduction of the Arousal-Valence model (A-V model). The A-V model is a mapping of continuous emotions over two dimensions - Arousal and Valence, contrasting and shifting between positivity and negativity (Valence), and the low and highs of energy (Arousal) [10]. Their classifier used Gaussian Mixture Models (GMMs) for the four principal mood quadrants of the A-V model and achieved an accuracy of 85% which is surprisingly better than Li and Ogahira's work [2]. Note that, valence in this context is used in a textual manner. It still measures positivity and negativity of a word, but is drawn from the ANEW dictionary, representing an average person's perceptions of the positivity or negativity of a word. The A-V model may have performed way more accurately than any other models, but it still does not escape subjectivity since it relies on a dictionary. We need a model that eliminates the use of subjective classification of words if we want to determine which words are incorporated

with positive and negative songs.

On a more recent and relevant work, a data scientist and an R-blogger, under the name of "RCharlie", used the Spotify and Genius Lyrics APIs to quantify the sadness in some of his favorite Radiohead songs. He specifically used Spotify's valence measurement and combined it with the NRC lexicon and tidytext. He also factored out lyrical density - a measurement of how "important" lyrics are to a given song. Lyrical density is computed by the number of lyrics per song over the track length. Considering all of these together wielded an astonishing result as he was able to visualize which songs were the saddest and the gloomiest [11]. Here, we can see that musical valence has had a major impact in his success of his work. When combined with sentiment analysis, musical valence could potentially answer some of the problems present in song emotion extraction and in music information retrieval systems in general. In the next chapter, we shall discuss the methods and procedures to be applied in this study.

III. Materials and Methods

A. Gathering the Data

We used a dataset of 1090 songs, retrieved from The Billboard Hot 100 from 2006-2016. 10 of these songs are unavailable through Spotify. Hence, only 1090 of them were considered. With these data, we searched for the lyrics of each song using the Genius API, an interface for the retrieval of user-annotated lyrics, poems, and other text. We then checked for each lyric's spelling and removed extra characters and stored it in a database along with the track's details such as artist, title, and year released.

B. Classifying the Data

We also searched for the track through the Spotify API and tap into its ability to retrieve audio features. The Audio Features object contains the property "valence" which we will use to classify our data. A value greater than 0.5 indicates a positive song, while a value less than 0.5 indicates a negative song.

C. Feature Selection

After classification, we collected the lyrics and transformed it into a vector containing the frequencies of words. The term frequency-inverse document frequency (TF-IDF) model is used to transform the lyrics into a vector instead of the bag-of-words model. TF-IDF is intended to give weight to certain words in a document based on the number of times those words appeared on a document. In Scikit-learn, the TfidfVectorizer is readily available to transform a set of corpora into a vector.

Along with the lyrics of the songs gathered in this study, their respective audio features are also used as features. The Spotify API offers the retrieval of an Audio Features object in its API. Features that were considered in this study were as follows: *danceability*, *energy*, *loudness*, *acousticness*, *instrumentalness*, and *tempo*. Spotify offers other features, but these features were selected because they are intuitively important.

Acousticness is defined as a confidence measure if a track is acoustic. A score of 1.0 for acousticness denotes high confidence that the track is acoustic. Danceability, on the other hand, denotes if a music is danceable. It is based on certain elements of music like tempo, rhythm stability, beat strength and overall regularity. A score of 1.0 for danceability is most danceable and 0.0 is least danceable. Energy, meanwhile, represents a perceptual measure of intensity and activity. A score of 1.0 for energy means high energy and 0.0 means low energy. Loudness is the overall loudness of a song in decibels (dB). Loudness are averaged throughout the track and is useful in comparing loudness from other tracks. The range for loudness goes from -60 db to 0 db. Tempo is the overall estimated tempo of a track in beats per minute, It is the speed or pace of a given piece. Last is the instrumentalness, which is the measure for vocal content, The values range from 0.0 to 1.0. The closer it is to 1.0, the higher the probability that the piece does not yield vocal content [3].

D. Training the Model

In training, we used two classification models, Multinomial Naive Bayes and the Support Vector Machine. We selected these two because they are the simplest yet effective methods in text classification. First, we trained a model using the Multinomial Bayes and SVM without Spotify's audio features. Next, we trained an SVM model and included the audio features as its features. This is done to have a comparison in results between using and not using the audio features.

1) Multinomial Naive Bayes: The Naive Bayes classifier is one of the fastest and easiest ways of implementing data classification. The Naive Bayes is based on Bayes' theorem that describes the probability of an event based on prior knowledge of conditions that might be related to the event. The Bayes' Theorem is as follows:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1}$$

A and B are events independent of each other and $P(A)$ is not equal to 0. $P(A|B)$ is the likelihood of A given that B is true. This shows conditional probability. We can then replace the variables in this equation to fit our data.

$$P(C|lyr) = \frac{P(lyr|C)P(C)}{P(lyr)} \tag{2}$$

In Equation 2, C represents the class (either positive or negative) and lyr represents the lyrics. This equation describes the probability of a song belonging to a class given its lyrics. In Equation 1, we see that it only describes the probability of a singular events, regardless if the probability is conditional or not. In our study's case, we consider the probability of each word that appears in the lyrics as a collective. For this reason, we adopt the Multinomial Naive Bayes - a variant of the Naive Bayes, that will satisfy .

$$P(lyr|C) = P(w_1, w_2, ..., w_k|C) = \prod_{i=1}^{k} P(w_i|C) \tag{3}$$

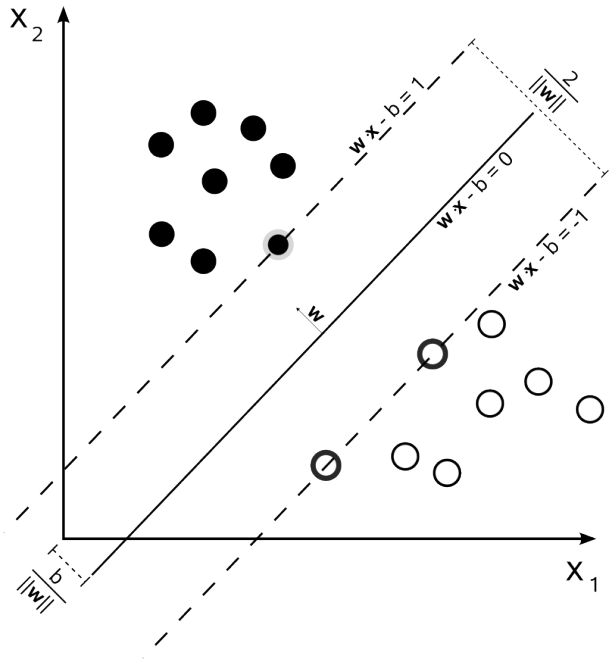


Fig. 1: Support Vector Machine with Optimal Hyperplane [11]

$$P(w_k|C) = \frac{n_k + 1}{n + |V|} \tag{4}$$

In Equation 3, we equate $P(lyr|C)$ as a product of the probabilities of the words that appear in the lyrics given the class. Each individual probability of the words given a class is one plus its frequency in the dictionary (bag-of-words) over the total number of words present in the dictionary plus the dictionary size (see Equation 4). We add one to the numerator to avoid having a 0 probability and affect the whole probability of the lyrics given a class. This technique is called Laplace Smoothing.

$$P(C|lyr) \propto P(C) \prod_{i=1}^{k} P(w_i|C) \tag{5}$$

The final formula for finding the class is given in Equation 5. We can disregard the probability of the lyrics since $P(C|lyr)$ is directly proportional to the numerator on the right side of the equation (see Equation 2). We apply Equation 5 to both the positive and negative dictionaries and the probability with the highest value will be the class of the song.

2) Support Vector Machine: A Support Vector Machine is a supervised machine learning technique used in classification and regression that involves finding for an optimal hyperplane that divides the a set of data points into two classes through the use of support vectors. The set of data that we retrieved was converted into numerical form to classify using this method. We shall use python's library, scikit-learn, for preprocessing of data and the application of Support Vector Machines.

Given training data $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ where $x_i = (x_1, x_2, ..., x_n)$ is a vector of features in a song (e.g. lyrics, valence) and y_i is one of $\{-1, +1\}$ that denotes positivity or negativity of a song. SVM finds a linear function w (weight vector) such that $f(x_i) =< w.x_i > +b$ where b is bias. If $f(x_i) \geq 0$ then $y_i = +1$; else $y_i = -1$ [12]. We used Spotify's measure of audio features and they were included in the feature vector that will represent all of the points in the data.

In Figure 1, we see that a set of points are divided by an optimal hyperplane. The two sides that are divided by the line symbolize the decision boundaries for each of the classes (hollow circle and solid circle). Though it might seem that SVM is only effective in binary classification, it is also used in multiple classification.

E. Data Splitting and Testing

The initial dataset retrieved was split into two, one for training and one for testing. The training and testing of data are cross validated using the K-fold Cross Validation where K=10. This is to ensure that every data is trained and tested so that we can obtain the maximum accuracy for our classifiers. Every iteration of the K-fold Cross Validation with K=10 means that data will be split 90% for training and 10% for testing. We then applied four performance measures: *accuracy*, *precision*, *recall*, and *f1 score* or *f-score*. Accuracy is the most intuitive performance measure that measures the ratio between correctly predicted observations and all observations made. Next, the precision is the ratio between the true positive observations and the all observations marked as positive. Recall, meanwhile, is the ratio between the true positive observations and all objects classified as positive. Finally, the F1 score is the weighted average between recall and precision.

F. General Workflow

To train a classifier for our data, we followed three steps: Initialization, Learning, and Evaluation. In Figure 2, we see the different processes involved in these steps. In Initialization, we collected our data and pre-processed them in preparation for feature extraction which will be used in our classification algorithms. In preprocessing, we "clean" the raw data to transform it into an understandable format, removing unnecessary details. Feature extraction, on the other hand is the further breaking down of our preprocessed data and identifying what attributes of this data is deemed important. Features in machine learning are the measurable properties of a phenomenon.

Next, we proceeded to the Learning step where we shaped our model using machine learning algorithms (Naive Bayes and SVM) and the preprocessed data (or training data) and its features. This step is also called Training step because this process involves training an ML model given and ML algorithm and a set of training data. After successfully training our model, we obtained our classifiers. Finally, in Evaluation step we tested the classifier with the test data and checked its accuracy, recall, precision, and f1-score. K-fold Cross Validation is done where K=10 so that every data is used for training and testing. The performance measures of all the classifiers are then compared to one another.

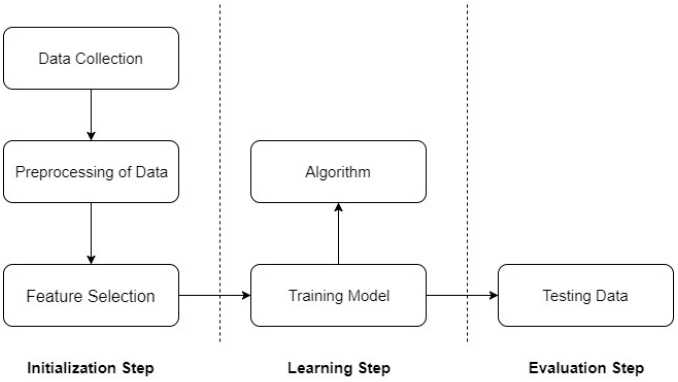


Fig. 2: General Workflow in Training Classifiers

Classifier Performance				
Classifier	Accuracy	Precision	Recall	F1 score
MNB	0.6248	0.6755	0.6255	0.6495
SVM w/o AF	0.6615	0.6621	0.6610	0.6615
SVM w/ AF	0.7220	0.7245	0.7182	0.7213

Fig. 3: Classifier Performance Evaluation

G. Technologies To Be Used

To help us in our study, we created a web application that will classify song valence. For this study, we used Django, a high-level Python Web framework, and PostegreSQL as the DBMS. We chose Python as the base programming language for its rich machine learning libraries that ultimately helped us in the Initialization, Learning, and Evaluation steps. Specifically, we used the scikit-learn library for its various classification, regression, and clustering algorithms. We also used scikit's K-Fold Cross Validation model in evaluating our classification models.

IV. Results and Discussion

A. Classifier Performance Evaluation

In Figure 3 we see that the Multinomial Naive Bayes Classifier has gained an accuracy of up to 62% while the SVM w/o Audio Features (AF) produced a better score of 66%. The SVM w/o AF has done a relatively better job compared to the MNB in terms of Accuracy, Recall, and the F1 score, only losing to MNB by precision for only a small amount of points. This can be due to the fact that SVM can work well with high-dimensionality classification problems. We can interpret from our results that our feature space (of 13,697 words) worked well for our classifier. Compared to the work of Li and Ogihara, which yielded a disappointing 45%, the SVM w/o AF has done a much better job at classifying musical valence but only in the sense of positivity and negativity. While Li and Ogihara has had a low accuracy score, they attempted to classify into one of 13 groups. Had they reduced the number of classifications, they would have gotten a much better performance score.

The SVM w/ AF, however, has had the best performance among the three classifiers. It outperformed the two classifiers, reaching up to 72% accuracy, precision, recall, and F1 score. Initially, the classifier did not perform well relatively to the two. It only yielded about 60-63% accuracy and is mainly because the audio features combined with the vectorized lyrics contain different values that are not normalized. For example, the audio feature 'loudness' can have values ranging from -60 to 0 depending on the actual loudness of the music, whilst vectorized lyrics from scikit's TdidfVectorizer can have low values like 0.09 or 0.1 for words that appear seldomly. The workaround for this problem is normalization and dimensionality reduction so that we can be able to describe data in a uniform and effective manner. Dimensionality reduction is done to further breakdown 13,697 words + 6 audio features in our feature space.

Normalization is required for dimensionality reduction, specifically in Principal Component Analysis (PCA). PCA is used to a set of observations of possibly correlated variables to transform them to a set of linearly uncorrelated values of variables called the *principal components*. PCA selects the features in such a way that the variance in the new low-dimension representation is high.

This is beneficial to us especially when we possess a massive number of features and we want to express it into just a few number of features for simplicity. After applying normalization and PCA, we have reduced the number of features into 10. Testing the newly improved classifier yielded a better accuracy of 72%.

In addition to improving accuracy, PCA has helped us in visualizing our data. By reducing our features into 2 components, we can visualize our data into a 2D graph and map the components into the x and y axes. In figure 4, the blue points represent the positive songs, while the red points represent the negative songs. We can see that in a 2D graph, the points overlap with each other, making it harder to imagine a decision line separating the positive and negative songs, signifying a need to represent the data in a more complex manner. Originally, the data cannot be separated by an optimal linear separating hyperplane but with the reduction to 2-3 components, linear separation becomes possible.

B. Spotify Audio Feature Importance

Out of the 6 Spotify Audio Features used in this study, 4 of them have a seemingly important correlation to musical valence. These features are acousticness, danceability, energy, and loudness.

In figure 5, we see the correlation of the four features to valence. In acousticness, we can observe that as acousticness increases, the values for valence become smaller. This means that acoustic songs tend to have a valence score that is classified as negative (remember: if valence < 0.5, then song is negative). We also see this trend with the other 3 features.

Danceability, energy, and loudness have somewhat the same pattern and/or relationship with valence. In their respective graphs, it is observed that as their values decrease, the values for valence also decrease and are limited in such a way that

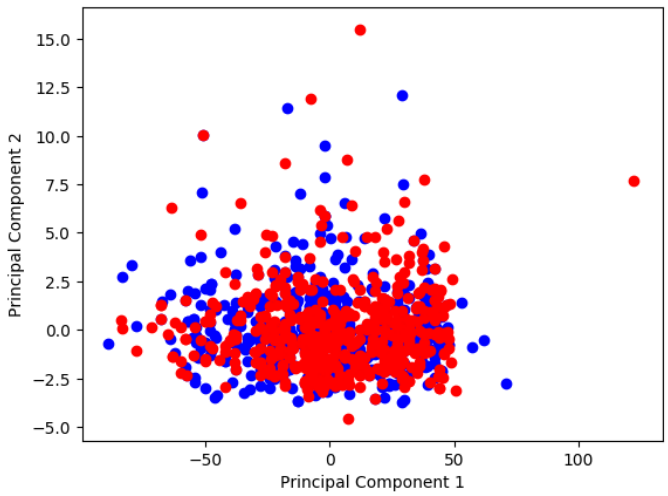


Fig. 4: PCA applied on the dataset where $n\_components = 2$

they only include values that will classify them as negative. For example, in energy we can see that values less than 0.32 yield an equivalent valence less than or equal to 0.5. In summary, the less danceable, energetic, and loud a certain track is, the more likely for it to have less valence. A high score in danceability, energy, or loudness, however, does not guarantee an increase in valence. It only means that as these values increase, the probability of being positive or negative also increase.

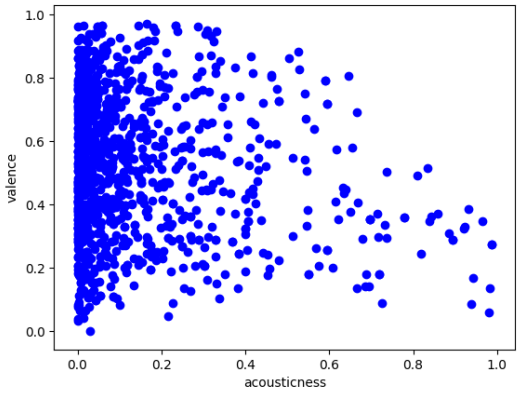
Through the graphs illustrated, we can say that these features have a correlation to valence. Inclusion of these features possibly have a big impact in training an SVM classifier as seen in the results in Chapter 4, where the SVM w/ AF has outperformed all of its competing classifiers.

C. Web Application for Classifying Musical Valence

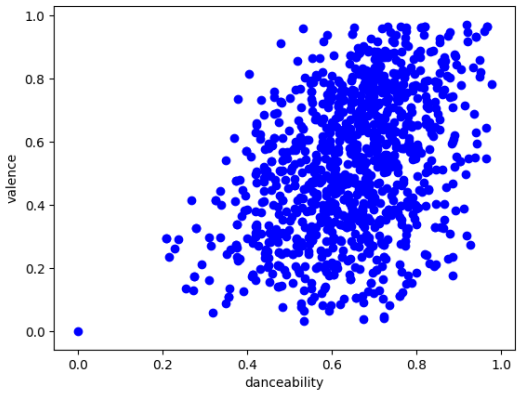
Using Django, Bootstrap, the Genius and Spotify API, and the models we have trained, we created a web application that searches and classifies a track according to valence through the use of its lyrics, and if available, its Spotify audio features. In the event that the features are not available (e.g. track is not in spotify, poor internet connectivity) SVM and MNB modes are available for its lyrics.

In Figure 6, we can see the two modes available in our application for classification. Figure 6a demonstrates lyrics mode, wherein users will manually input the lyrics for classification. Models available in this mode are MNB and SVM w/o Audio Features only. This mode is naturally helpful when a song is not available in Spotify. Figure 6b, however, demonstrates search mode, wherein users are asked for the artist and title of the track. This is particularly recommended for users because the application itself searches for the track's lyrics through Genius' API and searches for its features in Spotify. In this mode, all three classifiers (SVM w/o AF, MNB, and SVM w/ AF) are available for utilization.

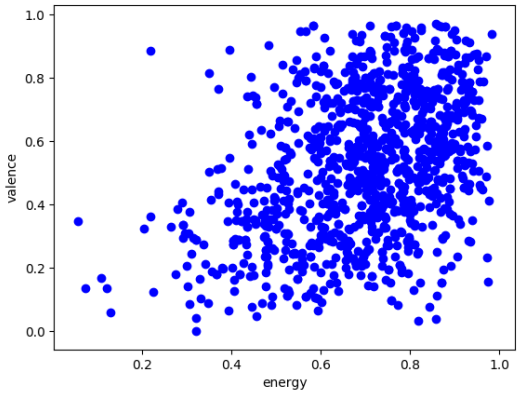
In the example, we searched for a track titled "There is a light that never goes out" by the artist "the smiths". The search view, as shown in Figure 7, displays the first match that Genius encounters and displays its lyrics on the right side.



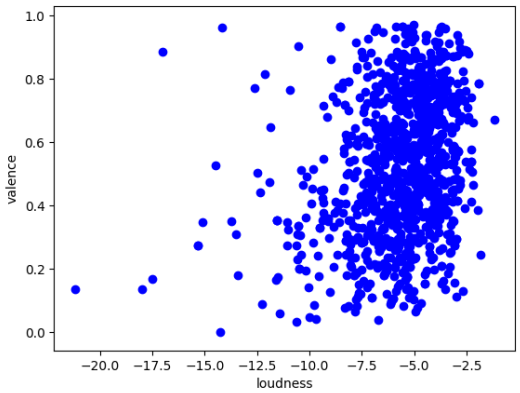
(a) acousticness



(b) danceability

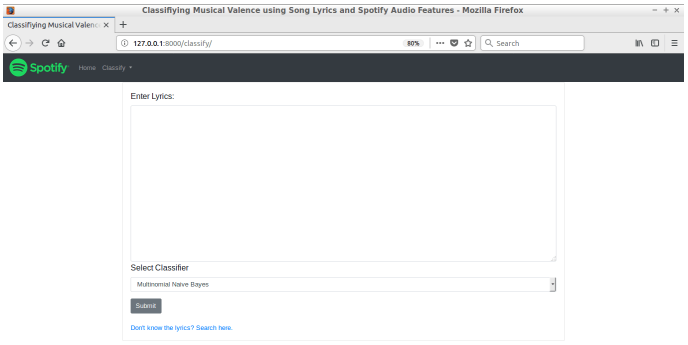


(c) energy

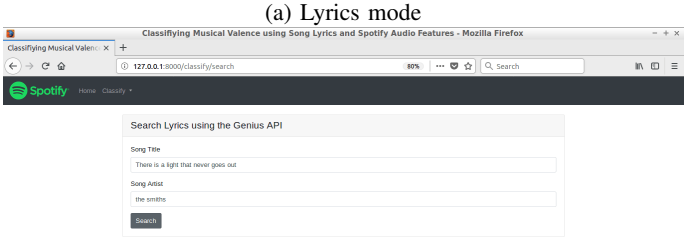


(d) loudness

Fig. 5: Spotify Audio Feature Importance



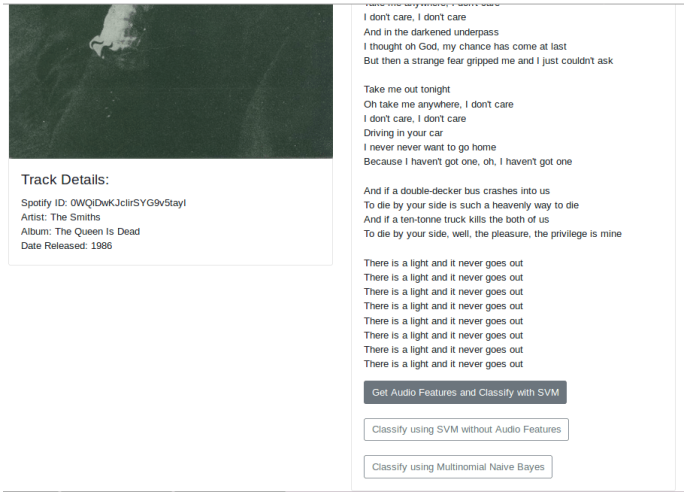
(a) Lyrics mode



(b) Search mode

Fig. 6: Classification modes in the Web Application

(a) Search Results



(b) Track Details, Lyrics, and Classification Options

Fig. 7: Search view in the Web Application

Track details and art retrieved from Spotify are also displayed at the bottom of the search form. Users then have three options for classification: SVM w/ Audio Features, SVM w/o Audio Features, and MNB.

If a user, selects one of the three options, he will be redirected to a page displaying the result of the classification. Figure 8 shows the perspective of the user once SVM w/ AF model is selected. The album artwork is displayed alongside the classification of Negative or Positive. Audio Features of the track will also be displayed as well as its valence to check if the classification decision is correct (Note: A valence score >= 0.5 is positive, while a score < 0.5 is negative).

Misclassification in the application is inevitable since our models have a range of 60-70% of accuracy. While sometimes it may seem like audio features have the biggest impact in classification, we should also consider lyrics' part in the equation, as they make up the feature space almost entirely. In effect, songs that have stress in negativity, even if they possess positive musical features, will most likely be classified as negative.

V. Conclusion and Future Work

The inclusion of Spotify Audio Features has, in no doubt, helped in boosting the performance of the SVM. By identifying which features are important, we eliminate unnecessary features and focus on the ones that directly affect our models. This will help us to be efficient and practical at the same time as we do not waste our resources training models that have lots of unimportant features.

While we only deal with binary classification, it is sufficing to say that this is enough to make an idea out of a song. It is safe to assume that Spotify's valence is a suitable label for classification because it summarizes a song's positivity or negativity based on musicality. It also correlates well with other features Spotify has provided. As we have observed in the usage of the web application, the SVM utilizes the audio features very well in junction with lyrics as its feature space. The problem arises if there is a need for specific emotion classification. Valence is weak in detecting which emotions are projected in a song. It is overgeneralized, but perhaps valence could still be used as a feature for training specific emotion classification models.

In the future, a more dissected quantification of other audio features could mean better performances in classification. So far, what Spotify has given us is the overview of some of its musical elements. If quantification arrives with each musical element present in a given track, it could pave the way for a better and suitable classification in music, and perhaps, this could even lead to the classification of multiple emotions.

It is also possible that in the future, other models can be trained for classification such as the well-known neural networks and the random forests. The vastness of machine

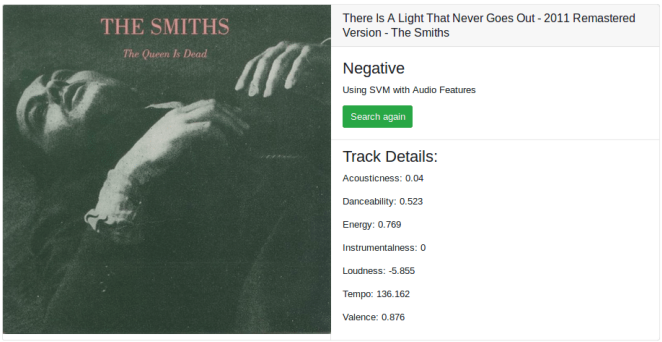


Fig. 8: Classification results when using SVM w/ AF

learning has paved a way for researchers to contrast and compare discoveries and decide which ones work for specific situations. It would be very interesting to see more complicated models dive into the problem that is music classification. The complexity and figurativeness of music makes it an interesting topic tackled by machine learning and sentiment analysis in general.

REFERENCES

[1] M. V. Mäntylä, D. Graziotin, and M. Kuutila, "The evolution of sentiment analysis-a review of research topics, venues, and top cited papers," *arXiv preprint arXiv:1612.01556*, 2016.

[2] Y. E. Kim, E. M. Schmidt, R. Migneco, B. G. Morton, P. Richardson, J. Scott, J. A. Speck, and D. Turnbull, "Music emotion recognition: A state of the art review," in *Proc. ISMIR*, 2010, pp. 255–266.

[3] "Get audio features for a track - spotify developer," https://developer.spotify.com/web-api/get-audio-features/, (Accessed on 01/02/2018).

[4] C. O. Alm, D. Roth, and R. Sproat, "Emotions from text: Machine learning for text-based emotion prediction," in *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, ser. HLT '05. Stroudsburg, PA, USA: Association for Computational Linguistics, 2005, pp. 579–586. [Online]. Available: https://doi.org/10.3115/1220575.1220648

[5] D. Yang and W.-S. Lee, "Music emotion identification from lyrics," in *Multimedia, 2009. ISM'09. 11th IEEE International Symposium on*. IEEE, 2009, pp. 624–629.

[6] A. Jurek, M. D. Mulvenna, and Y. Bi, "Improved lexicon-based sentiment analysis for social media analytics," *Security Informatics*, vol. 4, no. 1, p. 9, 2015.

[7] S. Ahire, "A survey of sentiment lexicons," 2014.

[8] A. M. Oudenne, P. Swarthmore, and S. E. Chasins, "Identifying the emotional polarity of song lyrics through natural language processing."

[9] T. Li and M. Ogihara, "Detecting emotion in music," 2003.

[10] A. Jamdar, J. Abraham, K. Khanna, and R. Dubey, "Emotion analysis of songs based on lyrical and audio features," *arXiv preprint arXiv:1506.05012*, 2015.

[11] "fitter happier | rcharlie," http://www.rcharlie.com/post/fitter-happier/, 2017, (Accessed on 01/03/2018).

[12] "Support vector machines - university of michigan | coursera," https://www.coursera.org/learn/python-text-mining/lecture/e5cEj/support-vector-machines, (Accessed on 01/09/2018).

Kidd Jason C. Evangelista is an undergraduate from the Institute of Computer Science. His hobbies include playing video games and writing music. He is interested in studying machine learning and artificial intelligence. He is also a proud charter member of the Alliance of Computer Science - UPLB founded in 2014.