Society of Song (SOS):

Web Application and Lyrics Classification

Felipe de Oliveira[1]

[1] University of Memphis, Memphis TN 38152, USA

Abstract

Now a days people are listening to their songs without stopping to think what the lyrics or the singer themselves want to say, making songs only a matter of fun but not a matter of discussion and engagement. There is an enormous difference between the current musical and lyrical scene and the one 20 years ago where people would gather to talk and discuss the lyrics' meaning and mirror these meanings to their own life. In addition, in todays world people are listening to music 24 hours a day, figuratively speaking, and consuming their words without even noticing. This consumption of unintentional and obscure words can be mentally harmful in some cases if not well addressed. As a result, this current project comes to fulfill this void and bring back the old times where people can relate their social issues to lyrics and discuss about them with other people.

CONTENTS

1. Introduction

There are different applications using machine-learning algorithms that categorize songs using different kinds of approaches, such as by instruments [1, 2] artist similarity [3, 4], emotion [5, 6] or genre [7]. Many researches and studies have shown that the habit of listening to music is one of the most popular activities, including leisure time and at work, and that it has an enhancing effect on the social cohesion, emotional state, and mood of the listeners [8, 9]. With the amount of people listening to music, the availability of song lyrics for free on the Internet has also grown making it easier to create a valuable dataset and use it in a machine-learning project. Nevertheless, this project aims to build a classification system that can classify lyrics into different social issue topics. The main contributions of this project are as follows:

1. *Creation of a new dataset that can provide the basis of future studies on music and mood.*
2. *Different classification model for topics prediction of music based on lyrics analysis.*
3. *An online web application to perform lyrics classification.*

Different algorithms were used to test the model and the results of each one will be shown below, including Naïve Baines, K Nearest Neighbor, Decision Tree and others. All of these were studied and researched accordingly with the parameters given and the special considerations that this project requires.

The amount of free datasets containing songs is either limited to audio feature datasets or requires manual retrieval from on-line music platforms of Creative Commons-licensed music or public domain recordings. For this project, specifically, the second approach was the one used. A web scraper was developed to search for lyrics, artists and song titles in the MetroLyrics.com website. This scraper retrieved 200,000 links for songs web pages. Of these 200,000 only ¼ was actually retrieved into a dataset and recorded into a CSV file.

Whereas labeling songs with their correspondent genres can be done unambiguously in many occasions, labeling of songs with their respective social issues they talk about is a more challenging task. The perception of social issues and their philosophies is for sure a very subjective work. There are different datasets that divide songs by mood and can classify them [6] as well; however, datasets that are providing ground truth for such mood labels for music are usually covering very specific moods, such as: sad and happy. The work done here is to open the discussion broadly to areas such as: Racism, Love, Relationship, God, Servitude, Depression, etc. This set of topics cannot be transferred to a binary categorization and it must be done as a multi-label classification.

Section 2 provides a formal statement of the web application developed as a structure to this project. Section 3 will cover the infrastructure used in the preprocess level and all data related work. Section 4 summarizes all the setup and models used in the search for the best results. Section 5 will cover the results achieved by the models developed. Finally, section 6 will layout the conclusion and future work.

2. Application

The application developed over the past months is called the Society of Song (SOS) and it brings to the spotlight to the different feelings that a song's lyrics might carry to

different listeners. Through the SOS, the user might go back to the old times where they could get reunited with friends and talk about songs and how they relate their own experiences to them. In addition, the users can connect and discuss the songs and what kind of sentiments each lyric brings to their mind. Afterwards, the researches that have been brought to light have shown that the kind of songs you consume every day can affect deeply your psychological. The Society of Song is for users that believe lyrics are not only for personal enjoyment but also for creating awareness and engagement in a society that might not know what is going through their ears.

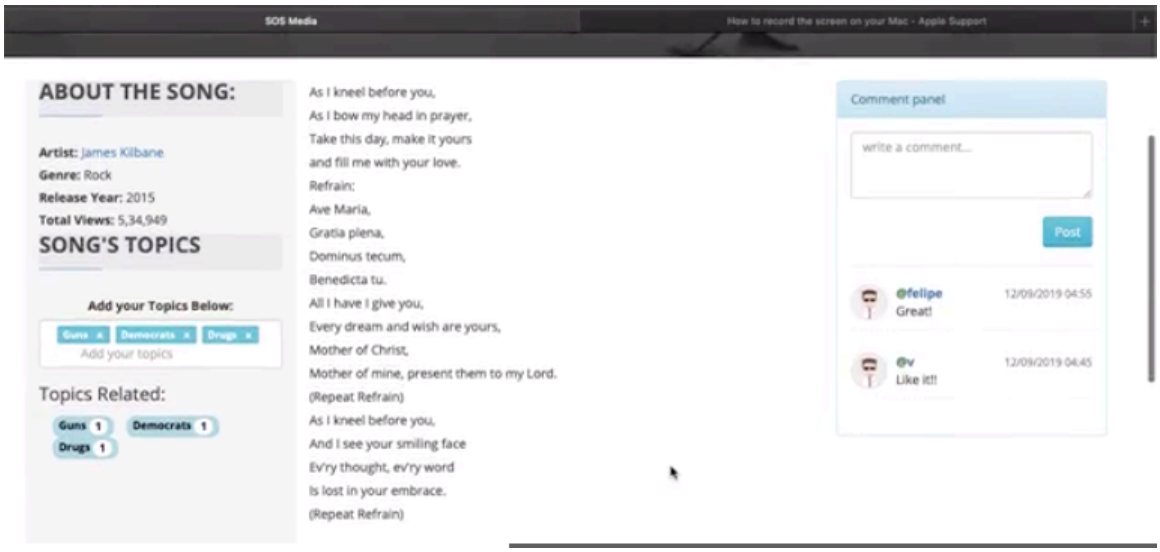

***Figure 1**. Songs lyrics and labels on SOS (2020)*

2.1 Tools and Technologies

Throughout the process of developing the given web application, different technologies were used and embedded into the stack. These techs were chosen using metrics such as: previous knowledge, community support, innovation and personal interest. We can start the conversation with Django: the most widely used python web framework. As suggested by the documentation [10], Django is a web framework for deadline perfectionists. It is a robust web framework that comes with many built-in functions and modules ready to fit most web application use cases. These are the features that make this framework a good choice for web development:

- **It is complete**: it provides the programmer with almost everything needed as soon as you start the project.
- **It is safe**: it is designed to always meet a correct standard to automatically protect your website.
- **It is scalable**: it uses a shared-nothing architecture-based component (each part of the architecture is independent of the others).
- **It is sustainable**: Django code is written with design principles and standards to foster easy-to-maintain reusable code creation.

- **It is portable**: it is written in Python, which runs on many platforms such as Linux, Windows and Mac OS
- **It is thriving**: it has an enormous active community, a large archive, and many free and paid support options. In addition, Django has the support of many web hosting providers.
- **Database ORM**: Django's accompanying object-relational mapper makes it easy to create database tables, create database queries, and other database-related functions without having to actually write any SQL queries.
- **Template Engine**: The template engine that comes with Django is fantastic and comes with many features such as template logic and filters that would quicken your development.
- **Robust Toolbox**: From sending emails for custom authentication, the power of Django is unlimited, with its ready-made functions just waiting for you to use them.

The next technology used was Docker, which came in handy for this project. As a web developer for the last four years, I know how hard it can be to deploy and create an environment either for production or development, a process that can turn to be a headache sometimes. We start by defining what is not Docker. Docker is not a traditional virtualization system: whereas in a traditional virtualization environment we have a complete and isolated S.O., inside Docker we have isolated resources (containers) that use common kernel libraries that are only possible because Docker uses a known Linux component, called LXC, as a backend. Thus, it is important to note that the containers are located on top of a physical server and the host OS. Each of them shares the host OS kernel and usually shares libraries and binaries as well. Therefore, Docker enables the packaging of an entire application or environment within a container, from which point the entire environment becomes portable to any other Host that contains Docker installed. Another difference is that shared items are read-only, which makes the container very lightweight, especially compared to the VM. To get an idea, the size of the first is Mb and so it starts in a few seconds. The virtual machine already contains several Gb and takes minutes to execute. The Figure 2 below shows the division of labor in virtual machine and containers environments.

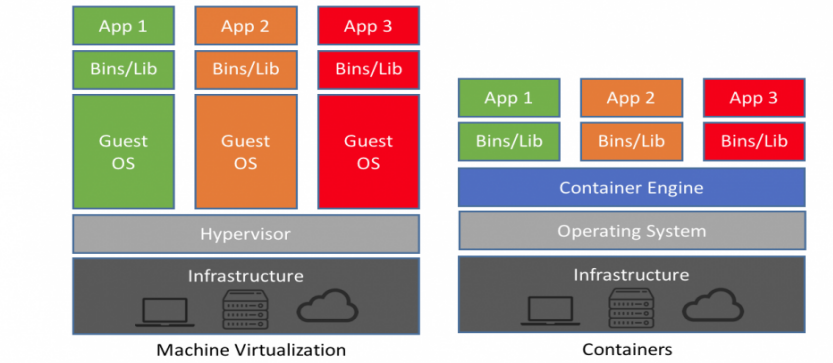

***Figure 2**. Difference between process execution in virtual machine and containers (Brey, 2019).*

A better explanation of the above figure and what it represents is that containers are a form of operating system-level virtualization that allows you to run multiple "systems" on a single real operating system. These isolated systems can be effectively isolated and limited in use of disk, RAM and CPU. Containers use a Kernel share trick to save resources, and hence they are a form of virtualization at the level of operational system.

There are different containers created for this project. The first one is the reverse proxy and static files handler where it uses Nginx. The second container is the Python WSGI HTTP Server that is Gunicorn. The figure below will show the step-by-step how these containers connect to each other in the project environment:

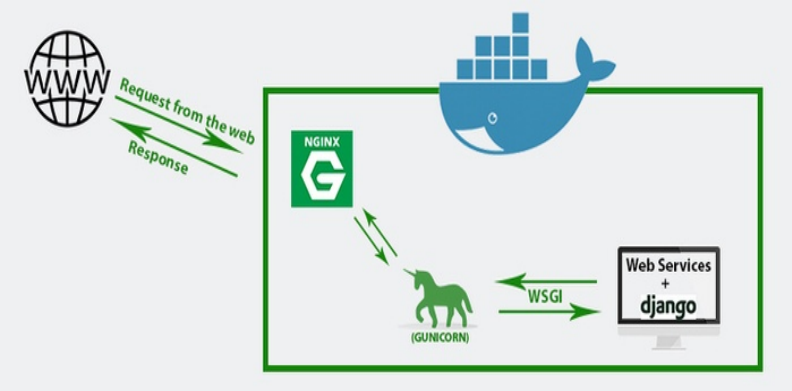

***Figure 3**. Representation of Docker containers present in the project (James, 2019).*

Figure 3 shows the workflow of a request in this project. It starts with the client (Browser) making a request that heads directly to Nginx (Proxy Server). Nginx does its work and sends that request to Gunicorn (python Http server). Gunicorn receives and communicates with the web application via an interface (WSGI). After the work is done by the application, it responds back to the client using the same path which the request came through.

3. Methods

3.1 Data Acquisition

A random subsample of 30,000 songs was crawled and scraped from MetroLyrics.com [13] into TXT format, with each song in a different file. Only songs that have more than 50 words were scraped and used. The next steps consisted in transforming and uniting the data into one TXT file to then upload the information into the Postgres database. A script was developed to generate such a file recording the data as an SQL file. The choice of getting the lyrics in an unprocessed format over the MetroLyrics.com was necessary for comparing different feature extraction and preprocessing steps. Custom code based on the Python NLTK library [14] was written to identify non-English lyrics and remove these songs from the dataset using majority

support based on the counts of English words vs. non-English words in the lyrics. After applying those filtering rules, the remaining dataset of 10,560 songs was uploaded to the Society of Song database. In addition, due to the nonexistence of labels for social issues for the lyrics inserted, the lyrics were then interpreted manually and assigned based on human interpretation of the words. From the 10,000 songs in the database, only 140 songs were read and manually labeled, as the workload to interpret the lyrics was exhaustive and time consuming for only one person. As a result, the current test dataset contains only 140 labeled songs with the most diverse topics chosen, summing a total of 40 different topics. The topics are listed below:

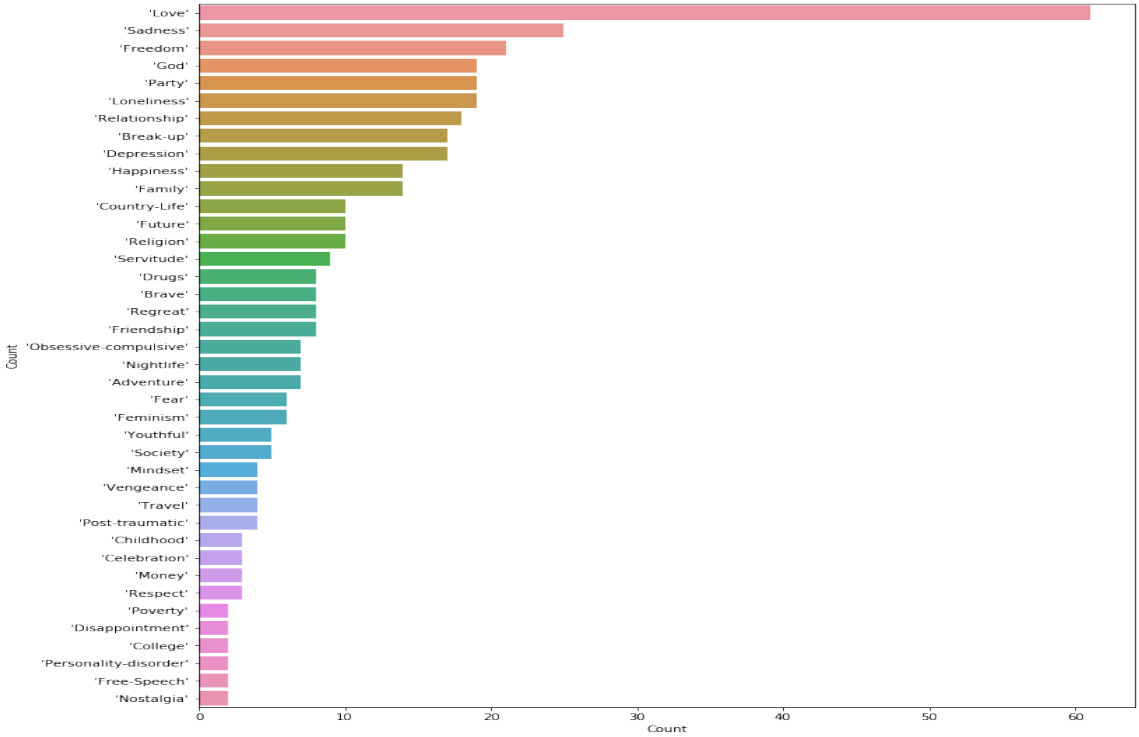

Figure 4. Topics related to songs' lyrics and its division between the dataset.

3.2 Feature Extraction

Before splitting the lyrics into tokens, a bag of words model [15] (a fixed-size multiset where the order of words has no significance) was used to transform the lyrics into feature vectors. Further processing of the feature vectors included stop word removal based on a stop word list from the Python NLTK library [5], and usage of the Porter stemming algorithm [20] for suffix stripping. Also, different representations of the word count in the feature vectors for each song text were used, such as binarization, term frequency (tf) computation, and term frequency-inverse document frequency (tf-idf) computation. It was used the 6,000 most frequently repeated words in the data as its features.

The term frequency-inverse document frequency was calculated based on the normalized term frequency tf-idf(t, d), which is computed as the number of occurrences of a term t in a song text d divided by the total number of lyrics that contain term t

$$\text{tf-idf(t, d)} = \text{tf(t, d)} \times \text{idf(t)}. \quad (1)$$

Let tf-idf(t, d) be the normalized term frequency and idf(t) be the inverse document frequency

$$idf(t) = \log \frac{1 + nd}{1 + df(d,t)} + 1$$

where nd is the total number of lyrics and $df(d,t)$ the number of lyrics that contain the term t.

3.3 Model Selection

Model performances using different combinations of the feature, as mentioned earlier, and preprocessing techniques including hyper parameter optimization of the models were evaluated using grid search on the 140-song training set to optimize the F1-score. The F1-score was computed as the harmonic mean of precision and recall

$$F - 1 = 2\times \frac{precision \times recall}{precision \ + \ recall}$$

Where

$$precision = \frac{TP}{TP + FP}$$

and

$$recall = \frac{TP}{TP + FN}$$

(TP = number of true positives, FP = number of false positves, and FN = number of false negatives).

3.4 Software

The Python libraries NumPy [16] and scikit-learn [17] were used for model training and model evaluation; the libraries seaborn [18] and matplotlib [19] were used for visualization. All data, code for model training and evaluation, and the final web app have been made available at https://github.com/byllynho/sos-music.

4. Multi-Label Classification Techniques

Most traditional learning algorithms are developed for single-label classification problems. Therefore, a lot of approaches in the literature transform the multi-label problem into multiple single-label problems, so that the existing single-label algorithms can be used. Different approaches were tested during the process of finding the best solution for the problem. Let's go through some of them.

4.1 OneVsRest

The One-Vs-The-Rest [20] classifier strategy consists of fitting one binary classifier per class. We associate a set of positive examples for a given class and a set of negative examples, which represent all the other classes. The main assumption here is that the labels are *mutually exclusive*. You do not consider any underlying correlation between the classes in this method. Although this strategy is popular, its heuristic suffers from several problems. First, the scale of the confidence values may differ between the binary classifiers. Second, even if the class distribution is balanced in the training set, the binary classification learners see unbalanced distributions because the set of negatives they see typically is much larger than the set of positives.

4.2 Classifier Chain

A classifier chain is an alternative method for transforming a multi-label classification problem into several binary classification problems. It differs from binary relevance in that labels are predicted sequentially, and the output of all previous classifiers (i.e. positive or negative for a particular label) is inputted as features to subsequent classifiers [22]. A chain of binary classifiers C0, C1, . . . , Cn is constructed, where a classifier Ci uses the predictions of all the classifier Cj , where j < i. This way the method, also called classifier chains (CC) [21], can take into account label correlations. The total number of classifiers needed for this approach is equal to the number of classes, but the training of the classifiers is more involved. In Ensemble of Classifier Chains (ECC) several CC classifiers can be trained with random order of chains (i.e. random order of labels) on a random subset of data set. Each classifier predicts labels of a new instance separately. After that, the total number of predictions or "votes" is counted for each label. The label is accepted if it was predicted by a percentage of classifiers that is bigger than some threshold value.

4.3 Label Powerset

The label powerset (LP) [23] transformation creates one binary classifier for every label combination present in the training set. This method needs worst case (2^|C|) classifiers, and has a high computational complexity. When the number of classes increases, however, the number of distinct label combinations can grow exponentially. This easily leads to combinatorial explosion and thus computational infeasibility. Furthermore, some label combinations will have very few positive examples.

4.4 K-Nearest Neighbor

The K-nearest neighbor is a very common algorithm used in machine learning and pattern recognitions. In this project we are using a slightly different form of the same algorithm called Multi Label K-Nearest Neighbor (MLKnn). The k-nearest neighbors algorithm (k-NN) is used for classification and regression [24]. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression. The output expected in this project was the classification where the object is assigned to the class most common among its k nearest neighbors.

5. Results

Using bootstrapped data and the setup above, I trained and tested models with various parameterizations by performing a 80/20 partition of the original dataset into both a training set and testing set for calculating the F1-Scores. Broadly speaking, batch size describes the number of words included in each training group, and the dropout rate specifies the probability of ignoring any given entry in the matrix of weights—this helps to prevent the model from overfitting on any specific word when making predictions. The result metrics were given using F1 – Score as this metric returns a more accurate value over unbalanced data.

Results are summarized in Table 1 below:

Approach/Classifier	F1 - Score
OneVsRest/Logistic Regression	0.031
OneVsRest/LinearSVC	0.454
Binary Relevance/ GaussianNB	0.816
OneVsRest /MultinomialNB	0.031
MLkNN (1 fold validation)	0.171
Label Powerset /GaussianNB	0.197
Decision Tree /ClassifierChain	0.225

Table 1. *Classifiers results based on F-1 Score*

In terms of score, we can see by the table above that based on the classifiers and multi label approaches, the Decision Tree with Classifier Chain has brought the best results. Even though a 0.225 cannot be considered an excellent outcome, we need to consider that the data is fairly unbalanced and the size of the training data is very small, as it was inserted manually by one single person. In addition, just F-1 Score is not sufficient to show the qualitative accuracy of the model. The best way to do so is by showing the outcomes and the labels generated for each song and compare them with the labels chosen by the user.

Song	Topics	MLknn	Decision Tree	GaussianNB
Darling	Love, Friendship	Drugs, Loneliness, Party, Sadness	Love, Party	Nightlife, Party
Always On My Mind	Break-up, Sadness, Relationship, Regret, Love	Break-up, Regret	Break-up, Childhood, Love, Sadness	Nightlife, Party
Forevermore	Loneliness, Love, Happiness	Depression, Love, Obsessive-compulsive	Family, Love, Obsessive-compulsive	Love, Relationship
Drunken Sailor	Freedom, Drugs	Freedom, Happiness, Society	Love, Party	Nightlife, Party
Darling	Love, Friendship	Drugs, Loneliness, Party, Sadness	Love, Party	Nightlife, Party
Orange Colored Sky	Love, Country-Life	Adventure, Family, Future	Disappointment, Free-Speech, Loneliness, Love, Relationship	Love, Relationship
Life Changes	Freedom, Youthful, College	Drugs, Loneliness, Party, Sadness	Depression, Love, Money, Obsessive-compulsive	Nightlife, Party
Over The Rainbow	Future, God	Brave, Freedom, Happiness	Love	God, Religion, Servitude
Maybe	Future, Sadness, Love, Obsessive-compulsive	Loneliness, Party	Depression, Love, Sadness	Nightlife, Party
Desperate Man	God, Depression, Sadness, Religion	Drugs, Loneliness, Party, Sadness	Love	God, Religion, Servitude
I'm Ready	Respect, Happiness, Love	Depression, Love, Sadness	Disappointment, Loneliness, Love, Party, Sadness	Nightlife, Party
Old Time Religion	God, Religion, Love	Freedom, Happiness, Society	Family, Freedom, Future	God, Religion, Servitude
A Mistake	Freedom, Party, Regret	Feminism, Love, Mindset, Relationship, Youthful	Freedom, Obsessive-compulsive	Nightlife, Party
If I Die Young	Depression, Future, Sadness, Loneliness	Drugs, Loneliness, Party, Sadness	Depression, Fear	Nightlife, Party

Table 2. *Actual labels generated by the classifiers.*

Using the information stated in Table 2 above, we can easily see that there are many divergences between the classifiers, especially when compared with the Gaussian Naïve Baines. The Naïve Baines classifier tends to label the lyrics as “Nightlife and Party” in 9 of 14 cases. Also, we can consider it normal as the training dataset is small and unbalanced towards love and party songs.

If you look at the data outputted by the Decision Tree classifier we can observe the accuracy is higher than the other two classifiers. In addition, the topics are very subjective and they can interpolate with other topics, such as: Drugs and Party, Love and Happiness, Depression and Sadness. With that being said, we can consider the results above to be accurate enough due to the lack of information and the difficulty in labeling the data correctly.

6. Conclusion and Future Work

The current application is just a prototype of what the SOS has the potential to become one day. Throughout the development of this prototype I was able to foresee different paths that could improve the application and increase the number of users. I read a variety of articles containing many innovative ideas that would be useful in

accomplishing these goals. One of those ideas was the connection with an outside music source (YouTube, Spotify, Apple Music, etc) through available APIs that would provide a larger amount of available information. Also, this connection would allow the application to have information about the user's taste in music through their playlists, most-listened-to songs and genres, favorite artists, etc. Another feature that will also be available is the opportunity for the users to share their posts with bigger platforms such as Medium.com. With this, the users might take our platform that is based only in music matters and share it with other people from different backgrounds.

A very important and innovative idea that was also contemplated during the process was to make the platform suggest to its users new songs based on the topics that they engage with most. That would be a very interesting feature generating a lot of data and bringing content to our users in a more effective way. More studies need to be done when it comes to creating an algorithm such as this one. However, that could be a game changer for the application in the future.

REFERENCES

1. Perfecto Herrera, X. Amatriain, E. Batlle, and Xavier Serra. Towards Instrument Segmentation for Music Content Description a Critical Review of Instrument Classification Techniques. *International Conference on Music Information Retrieval*, 2000.
2. TaoLiandMitsunoriOgihara.Musicartiststyleidentificationbysemi-supervisedlearningfrombothlyrics and content. In *International Multimedia Conference: Proceedings of the 12 th annual ACM international conference on Multimedia*, volume 10, pages 364–367, 2004.
3. Janet Marques, Janet Marques, and Pedro J. Moreno. A Study of Musical Instrument Classification Using Gaussian Mixture Models and Support Vector Machines. *COMPAQ CORPORATION, CAMBRIDGE RESEARCH LABORATORY*, 1999.
4. Markus Schedl and David Hauger. Mining microblogs to infer music artist similarity and cultural listening patterns. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 877–886. ACM, 2012.
5. Lie Lu, Dan Liu, and Hong-Jiang Zhang. Automatic mood detection and tracking of music audio signals. *IEEE Transactions on audio, speech, and language processing*, 14(1):5–18, 2006.
6. Pieter Kanters. *Automatic mood classification for music*. PhD thesis, Master's thesis, Tilburg University, Tilburg, the Netherlands, 2009.
7. George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *Speech and Audio Processing, IEEE transactions on*, 10(5):293–302, 2002.
8. Thomas Schäfer, Peter Sedlmeier, Christine Städtler, and David Huron. The psychological functions of music listening. *Frontiers in psychology*, 4, 2013.
9. Daniel Västfjäll. Emotion induction through music: A review of the musical mood induction procedure. *Musicae Scientiae*, 5(1 suppl):173–211, 2002.
10. Django (2019). Documentation. Retrieved Marc 30, 2020, from https://docs.djangoproject.com/en/2.2/#django-documentation.

11. Brey, P. (2019). *Containers vs. Virtual Machines (VMs): What's the Difference? | NetApp Blog*. [online] NetApp Blog. Available at: https://blog.netapp.com/blogs/containers-vs-vms/ [Accessed 19 Nov. 2019].
12. James, S. (2019). *Nginx: Setting Up a Simple Proxy Server Using Docker and Python/Django... | Codementor*. [online] Codementor.io. Available at: https://www.codementor.io/samueljames/nginx-setting-up-a-simple-proxy-server-using-docker-and-python-django-f7hy4e6jv [Accessed 19 Nov. 2019]
13. MetroLyrics.com dataset -\ url{https://www.metrolyrics.com/top100.html}
14. Steven Bird. NLTK: the natural language toolkit. In Proceedings of the COLING/ACL on Interactive presentation sessions, pages 69–72. Association for Computational Linguistics, 2006.
15. Zellig S Harris. Distributional structure. Word, 1954.
16. Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. Computing in Science & Engineering, 13(2):22–30, 2011.
17. FabianPedregosa,GaëlVaroquaux,AlexandreGramfort,VincentMichel,BertrandThirion,OlivierGrisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, and Others. Scikit-learn: Machine learning in Python. The Journal of Machine Learning Research, 12:2825–2830, 2011.
18. MichaelWaskom,OlgaBotvinnik,PaulHobson,JohnBCole,YaroslavHalchenko,StephanHoyer,Alistair Miles, Tom Augspurger, Tal Yarkoni, Tobias Megies, Luis Pedro Coelho, Daniel Wehner, Cynddl, Erik Ziegler, Diego0020, Yury V Zaytsev, Travis Hoppe, Skipper Seabold, Phillip Cloud, Miikka Koskinen, Kyle Meyer, Adel Qalieh, and Dan Allan. seaborn: v0.5.0 (November 2014). nov 2014.
19. J D Hunter. Matplotlib: A 2D graphics environment. Computing In Science & Engineering, 9(3):90–95, 2007.
20. Wu, Wei & Gao, Xiaorong & Gao, Shangkai. (2005). One-Versus-the-Rest(OVR) Algorithm: An Extension of Common Spatial Patterns(CSP) Algorithm to Multi-class Case. Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference. 3. 2387-90. 10.1109/IEMBS.2005.1616947.
21. Read, Jesse & Pfahringer, Bernhard & Holmes, Geoffrey & Frank, Eibe. (2009). Classifier Chains for Multi-label Classification. Machine Learning. 85. 254-269. 10.1007/978-3-642-04174-7_17.
22. Read, Jesse; Martino, Luca; Luengo, David (2014-03-01). "Efficient monte carlo methods for multi-dimensional learning with classifier chains". Pattern Recognition. Handwriting Recognition and other PR Applications. 47 (3): 1535–1546.
23. Spolaôr, Newton; Cherman, Everton Alvares; Monard, Maria Carolina; Lee, Huei Diana (March 2013). "A Comparison of Multi-label Feature Selection Methods using the Problem Transformation Approach". Electronic Notes in Theoretical Computer Science. 292: 135–151.

24. Altman, Naomi S. (1992). "An introduction to kernel and nearest-neighbor nonparametric regression" (PDF). The American Statistician. 46 (3): 175–185.