Music Recommendation Capstone Project

Ruben Matell and Noah Covarrubias

Dr. Samara

Dat 490

March 31, 2023

Table of Contents

Executive Summary

The music industry has the ability to generate billions of dollars of revenue each year. Thus thousands of people including labels, artists, producers, and the services that make them possible all want to understand and drive themselves forward. One of these companies is Spotify, a digital music streaming platform that allows users to listen to their favorite music while also offering a vast repository of music, podcasts, audio books, and more. Today, listeners rely heavily on machine learning technologies to help them find new music, generate radio stations based on their favorite artist, and more.

This project seeks to build a music recommendation system inspired by Spotify's algorithm. Using data from the Million Song Dataset (MSD) and a variety of tools such as collaborative filtering and content filtering (including genre prediction, popularity prediction, etc.), we produce song recommendations for users based upon their tastes.

Project Plan

Organization Profile - Spotify:

- Founded in 2006 by Daniel Ek and Martin Lorentzon in Stockholm, Sweden
- Launched to the public in October 2008
- A leading music streaming service, allowing users to listen to music on-demand and discover new songs and artists

- As of September 2022, Spotify has 456 million active users and 195 million subscribers
- Revenue in 2022 was $10.5 billion
- Headquartered in Stockholm, with offices in major cities around the world, including London, New York, and Sydney.

<u>Business Description:</u>

Spotify is a digital music streaming platform that allows users to listen to their favorite songs, albums, and playlists on-demand. The platform offers a vast library of music and podcasts, with a wide range of genres and artists to choose from. Spotify lets users create and share their own playlists. It also lets users discover new music through personalized recommendations. The platform also allows users to follow their friends and see what they're listening to.

Spotify has a combination business models, using both advertising and subscription services. The platform offers a free version, which includes ads, and a paid version, called Spotify Premium, which is ad-free and includes extra features such as offline listening. The paid version is available as a subscription service costing about $10 per month. Additionally, Spotify has partnerships with companies such as Uber, which allows users to control their music through the Uber app and also includes an ad for Spotify.

<u>Key Executives</u>

Deniel Ek - Chief executive officer, founder, chairman

Paul Vogel - Chief financial officer

Gustav Soderstrom - Chief research and development officer

Alex Norstom - Chief freemium business officer

<u>Competitors:</u>

- Apple Music
- Pandora
- Tidal
- Amazon Music
- YouTube Music
- Deezer
- Soundcloud
- iHeartRadio
- Google Play Music

Data Set Opportunity

Using both data scraped from Spotify and the Million Song Dataset (MSD), we aim to create recommender systems and content detection algorithms to improve the user experience of Spotify. We hope to be able to improve recommendations to users using our models, which will generate new customers and increase conversions of free to paid users.

The MSD is a large dataset of audio features and metadata for a million popular songs. It was created by the LabROSA research group at Columbia University in New York, in collaboration with The Echo Nest, a music intelligence company that was acquired by Spotify in 2014. The MSD was created to provide a large, publicly available dataset for researchers and developers to use in developing music information retrieval systems and applications.

It includes a wide range of data for each song, such as the artist name, song title, album, release year, and genre. Additionally, it contains a variety of audio features, such as tempo, pitch, timbre, and loudness, which are extracted from the audio files using sophisticated algorithms. This data is all incredibly useful for building both the content-aware models and recommender systems we hope to create.

Research Questions

Research Question 1 - Can we accurately classify songs into one or more of Spotify's many genres?

Currently, Spotify classifies songs into specific genres based on data in listener playlists and through the use of 'music curation teams' which are presumably groups of people manually classifying songs into various genres. There is a large business case for automating this process. If we can accurately predict the genre of a song using machine learning techniques, this will greatly reduce costs for Spotify and increase the amount of recommendations the platform can make to its users.

Research Question 2 - Can we accurately predict what will make a song popular?

Currently, Spotify recommends songs to users based on other users' tastes, meaning that the songs that are recommended to people are likely somewhat well known already. However, there are 60,000 new tracks uploaded to Spotify every day. What if there was an algorithm that could predict a new song's likelihood of being popular and recommend that to users? This could present some bias problems, but if done correctly, could make Spotify's recommendation system much better.

Research Question 3 - Can we accurately recommend songs to users using a combination of content (such as the research questions above) and collaborative filtering methods?

Generating the predictions above is useless unless we are actually able to recommend users new songs. We can use two methods to do this: content and collaborative filtering. Content filtering examines an individual user’s taste and recommends a song similar in content to their favorite music. Collaborative filtering finds a few users which are similar to the target user, and then recommends songs based on what those similar users like. Using a combination of these methods, we can generate the best recommendation system for Spotify users to increase their listening.

Hypotheses

Hypothesis 1 - Song specific attributes will be able to indicate a potential genre classification.

By analyzing certain aspects of a song such as the artist, tempo, pitch, melody, and more a potential genre for the music can be identified. By finding and identifying these aspects of a song it could better identify the genre that it belongs to. Once genre defining aspects are identified on a whole then the characteristics could be searched for and tested on the data to test this hypothesis.

Hypothesis 2 - Spotify users will more likely enjoy already popular songs that users with similar taste listen to.

The more times a song is streamed, the more it is able to increase its popularity. Similar to identifying characteristics that can sort a song into certain genres, are we able to use the same method to identify what makes songs popular? By analyzing songs in the top 100 certain patterns among the songs may be identified to show what builds up a popular song. This hypothesis is intended to predict if songs that are coming out will become popular or not by looking at the musical aspects of the song itself.

Hypothesis Question - Should content analysis or collaborative filtering have more impact on predicting a target user's tastes.

This hypothesis intends to investigate both collaborative filtering and content classification for predicting songs that a target user might enjoy. By analyzing the playlists that a target user has already made and continues to add to, while also comparing to other users with similar interests new song recommendations could be made for the target user. However, which should have more weight and considered a better metric while making new recommendations for these users.

Output Summaries

Research Question 1 - Can we accurately classify songs into one or more of Spotify's many genres?

The analysis here will be able to identify what genre of music a song belongs to while also allowing a company to automate the process of identifying the song. A table will be included showing the top genres and relevant examples of some of the songs that would fit those genres. This will greatly reduce the cost of needing to employ a team while also expanding all genres with the amount of songs they have.

Research Question 2 - Can we accurately predict what will make a song popular?

This analysis is aimed at predicting if certain uploaded daily songs have potential to be popular. By applying the same idea of identifying a genre to identify popular characteristics of a song, Spotify would be able to greatly recommend new users that have attributes of other popular artists. Charts and other data visualization tools could be used to show how many streams a song is predicted to have, or around where it could be ranked for its genre popularity.

Research Question 3 - Can we accurately recommend songs to users using a combination of content (such as the research questions above) and collaborative filtering methods?

This final analysis will be used to identify the best metric and or series of metrics used to recommend songs to target users. By evaluating multiple methods used to recommend to users, it can be found which method has the best outcome as well as moving forward what improvements could be made to the process and methods used to recommend songs.

Literature review

Predictive analysis as well as machine learning have supported and continue to back recommendation systems in multiple fields. One field that continues to expand and allow for growth in the system is music, podcast, and streaming. At this point recommendation systems have become crucial for modern media companies to maximize user retention and engagement. These systems use machine learning algorithms to analyze a users' habits and preferences in order to suggest new content likely to engage the user. The success of music streaming services like Spotify is highly dependent on the effectiveness of their music chosen recommendation systems, as it drives user engagement and retention by suggesting music that they are likely to enjoy. In this literature review, we will explore the current methodology and gaps in knowledge surrounding Spotify's and its competitors' recommendation systems and the various methods and algorithms used to create a personalized and effective music recommendation experience for its users. In order to provide insights and make accurate recommendations to its users, Spotify collects vast amounts of data in a variety of fields. These would include user activity, such as the songs, albums, and playlists a user listens to as well as their engagement with each song (skips, play history, repetitions, e.g.) (Spotify 2023). The company also collects information regarding user tastes, such as any playlists a user makes and which artists they follow. In addition to all the personal data collected, Spotify collects metadata about the songs listed on their platform using advanced analytical tools. Examples of such data include a variety of different musical attributes like genre, tempo, key, danceability and energy levels. Spotify additionally recruits certain employees to categorize songs into different genres, many of which are incredibly niche (many listeners have never heard of them with genres such as Bubblegrunge and Weirdcore). Using this vast amount of stored data, Spotify is able to build a comprehensive profile for each user's musical preferences and song content, therefore making it possible to make informed recommendations based on multiple data points. An example of Spotify's recommendation system would best be seen in its Discover Weekly. Each week, Spotify generates a playlist for each of its users with personalized recommendations for new music using both content and collaborative filtering. These are the two of the main methods which all recommendation systems use to generate new content. Content filtering generates recommendations by focusing on finding similar songs based on song metadata only, in turn putting less weight on user tastes to identify new content. Collaborative filtering, on the other hand, analyzes a certain user's taste profile, comparing it to other similar users, and then making a recommendation based on those users' tastes (Madathil 2017). Through a combination of these two methods, Spotify is able to generate the Discover Weekly playlist as shown in Fig. 1. While this process has been overall very effective (Spotify is one of the most popular streaming platforms), there are limitations compromising the profitability and efficiency of this system. For example, Spotify employs people to categorize music song by song. Employing humans to sort the songs into each category introduces chances for errors that a song would be miscategorized. Instead, if the company could determine a song's genre using machine learning and content detection algorithms, they could much more efficiently create taste profiles for users and decrease spending on salaries. One paper attempts to automate this process by using a k-Nearest Neighbor classification system (Flexer et al. 2005). They were able to achieve an accuracy score of over 90% when attempting to classify songs into 22 different genres. This is a very impressive feat, but there are some caveats. First of all, the authors mention that this accuracy is based on potentially inaccurate data, since the songs were originally categorized by humans. Thus there may be some inaccuracies in a human perception of genre and their subsequent categorizations. However, genres are entirely subjective in the first place making it nearly impossible to rigorously define each musical genre. Also, the researchers only used about 22 genres and ~2500 tracks. The million song database is 400% bigger than this original database while also having thousands of genres instead of 22. Having more access to data and sorting into more genres, it may prove challenging for an algorithm to distinguish between all of these nuanced subgenres. An example of the challenges of sorting the new subset of songs into these categories could be found in the paper written by both Columbia University and EchoNest (Bertin-Mahieux et al., 2011). Here there are two different sets of genre identification tags being used. The first of which is musicbrainz which contains 2,321 unique and possible genres a song could be listed and identified as. The second set of possible genres is one made by EchoNest, their identification tag list include 7,643 unique and possible genres for the music to be classified as. Here it could be seen that the sheer amount of genres to group by could greatly affect where such songs are placed. One thing to note here is that the top three tags for each system do not always identify the music as belonging to a similar three genres. They also include that certain other attributes of a song, not necessarily the tempo, harmony, key, or energy level, are able to help pre-classify a band/songs genre, like the group's name. One huge drawback also made apparent here is lack of global diversity; most of the songs do not come from different ethnic backgrounds. However, the methods that are used in the paper are relevant as they may help in creating a better recommendation system. One of the methods mentioned and used was the Vowpal Wabbit method. It is a regression run on the linear combination of a certain attribute combined with the gradient descent of another set attribute. This was used to try and identify the year certain songs came out first. Moving forward it could better help understand certain attribute combinations to identify a genre. Previously mentioned as a potential issue was the size of the dataset and the amount of potential genres identified. Here the entire million song dataset holds a total unique observation size of 10^6 songs. Training and testing the model with more than 10^5 observations could render the observations severely biased and unable to provide an accurate prediction if a user will enjoy the content. In order to combat this issue, one study details some of the methods they used in order to produce an efficient and accurate large scale regression model (Wang and Zhou, 2021). The main method here was using a combination of two factors, the holistic value of the generalized product with a robust bayesian machine. Here the two would combine the better parts of each model to produce a rating system for the information provided. By combining the methods in the second paper on large datasets with the degree of rating needed it may help form a successful recommendation system for identifying a genre, and then recommending songs from said genre to a user based on the song qualities.

Exploratory Data Analysis

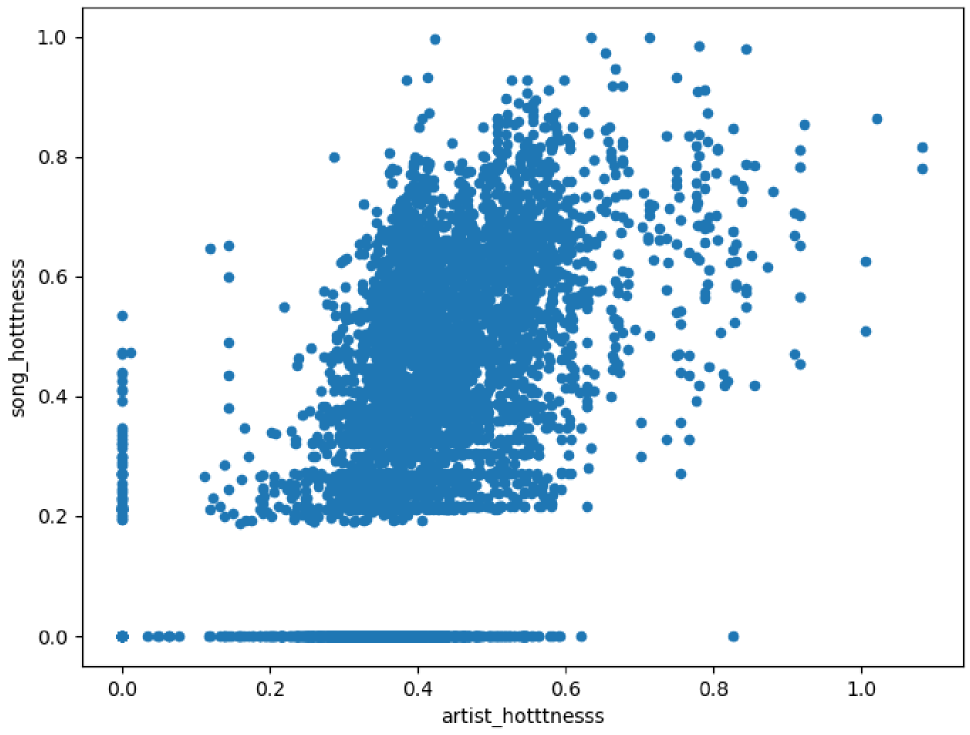
Exploratory data analysis (EDA) is a fundamental step in building a music recommendation system using the Million Song Dataset (MSD). EDA enables us to identify significant features that influence song popularity and user preferences. By analyzing various features of the dataset, we can discern underlying patterns and correlations, facilitating more accurate song recommendations. When building a recommendation system, the objective is to suggest new songs that align with a user's musical tastes. EDA provides valuable insights into the most important audio features that make a song popular or likely to be enjoyed by a user, which can be used to build a more effective recommendation engine. For instance, if the EDA indicates that certain audio features such as tempo, energy, and loudness are correlated with song popularity, the recommendation system can use this information to recommend songs with similar audio features and thus hopefully more enjoyable songs.

I had quite a difficult time getting this dataset. I have to account for two major design requirements while coding. One, the data from the MSD is only available in HDF5 (Hierarchical Data Format) files, which are incredibly difficult to work with. They are useful because they take up very little memory, but are troublesome because there is little documentation about them. My second requirement is that the code I write must be scalable. The following EDA is being done on ~10,000 observations, but the MSD has, well, a million. I am using unfamiliar packages to keep operation complexity low in order to run the same code when we access the full dataset on the AWS cloud server. Additionally, the MSD is missing data for a few columns such as ‘danceability’ and ‘energy’. These are features that are analyzed by EchoNest using their proprietary algorithms. We can get this data from the Spotify API using the SpotiPy Python package, but that will require much more work and for the sake of time and simplicity, it is being left out in this EDA. We hope to use it for the final recommendation system however.

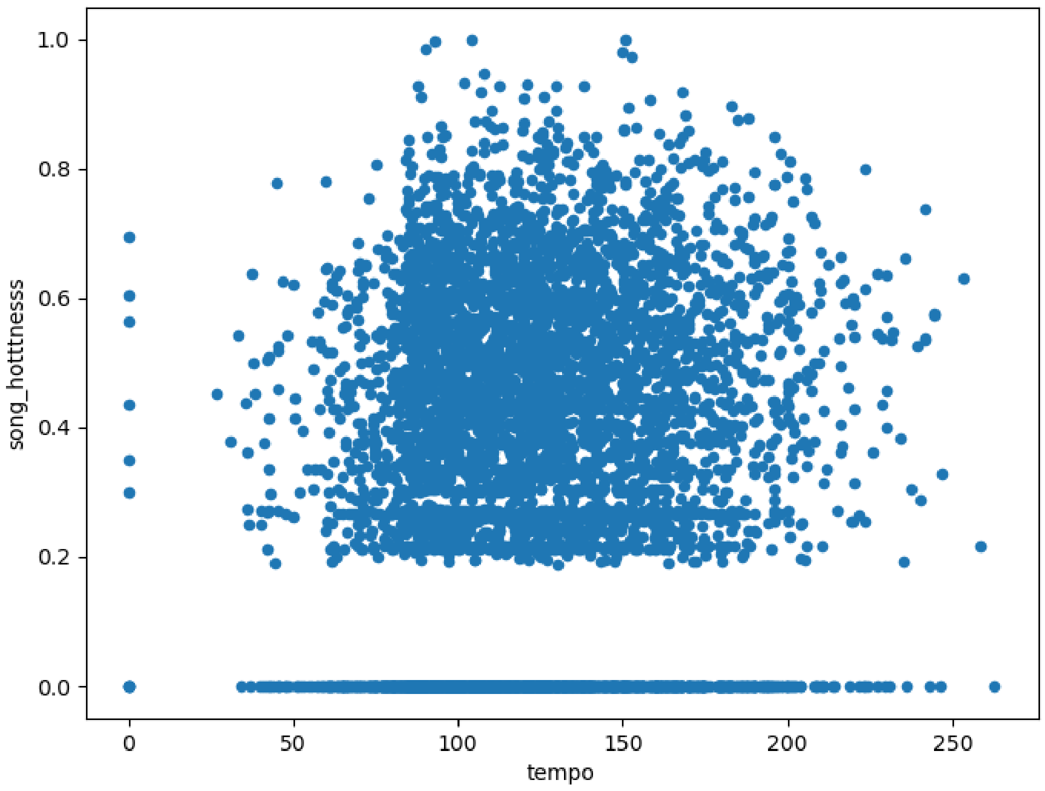
The data used in this project come from a huge data set with many things observed about the songs themselves. When first loading the data into a data frame there were 53 variables however not all of them are noteworthy. Some of the most important variables in the set are artist, which corresponds to the group or singular musician that released the music. Artist popularity, which looks at how popular an artist is in the respective music community by their audience and their peers. Artist tags, if they were to work with another producer or editor in order to make their final song better than it was at the recording. In regards to the music itself there were some important variables tracked. Some of the most relevant being the title, time signature, tempo, key, and finally the duration of how long it lasted. Some other variables included in the entire data set are the artist's individual id, they have both an alpha numeric code as well as a seven or less digit code to identify them. The same holds true for the songs themselves, they had been given track ids as well as the release album name and release album code itself. Other variables that were included in the final set were numeric variables representative of certain composition pieces of a song. It is important to note that the data set that was being looked at contains multiple null points as well as regions where the data was non-numeric. After looking into the variables and understanding what makes them vital to the understanding of what makes a song popular as well as if it matches other songs we are able to move on to the portion of data plotting to investigate some basic relationships among the data.

Moving on is looking at the data and then investigating any patterns that are revealed. Our goal here is to identify a few influential trends that are crucial and able to help us build a stronger and more efficient recommendation system. Therefore, we are starting by looking for features that may predict a song's popularity. A cursory look at the data and the relationships within are important as they will be used in the future system to predict what will help make a song more or less popular.

First, let's look at some plots using song_hottness, a measure of how popular a song is (as measured by EchoNest in 2010) when plotted with other vital measurements and data points.

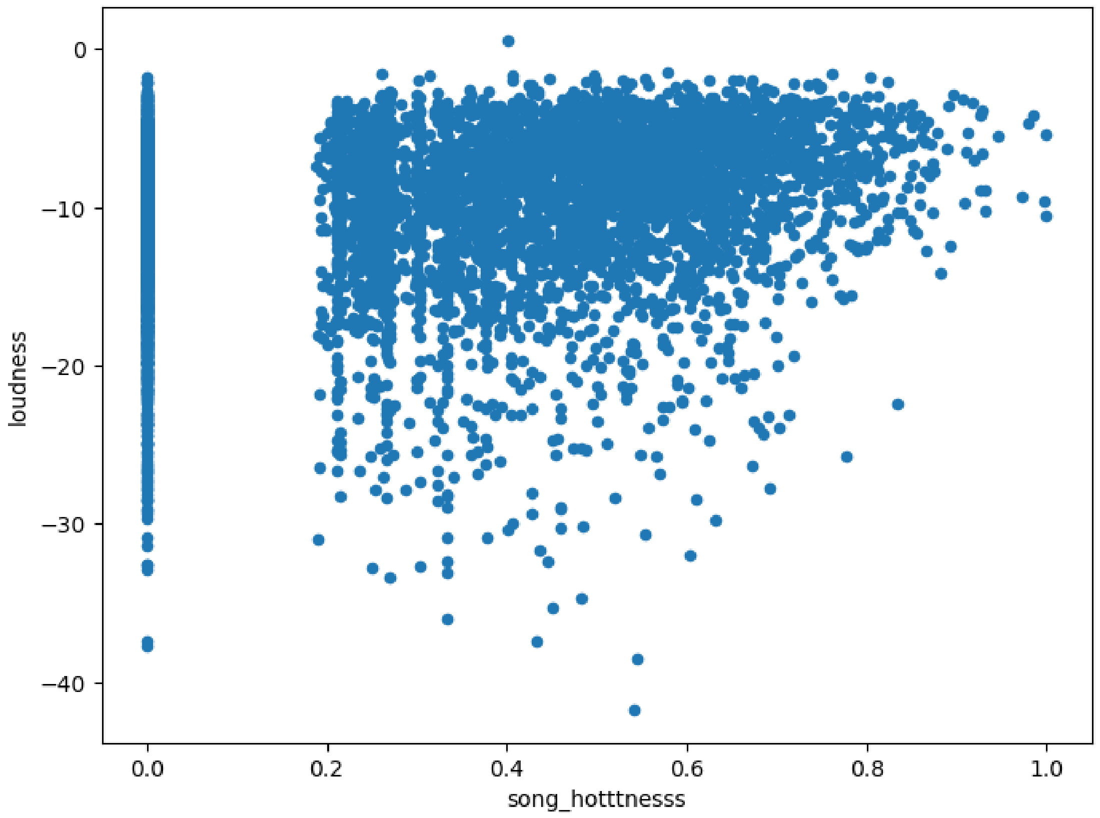


Starting off here, we see that there is a positive correlation between song hotness and artist hotness, again how popular the artist is. This suggests that there is a relationship between popular artists and the songs they tend to release becoming more popular, and vice versa. It is quite important to note that there are a few 0 values on both axes, which indicates missing data. Again it is not just missing values in one area but both the artist rating and where the song landed in the community. This is something that should be accepted now and that will have to be excluded/cleaned as we move forward and build our training model.

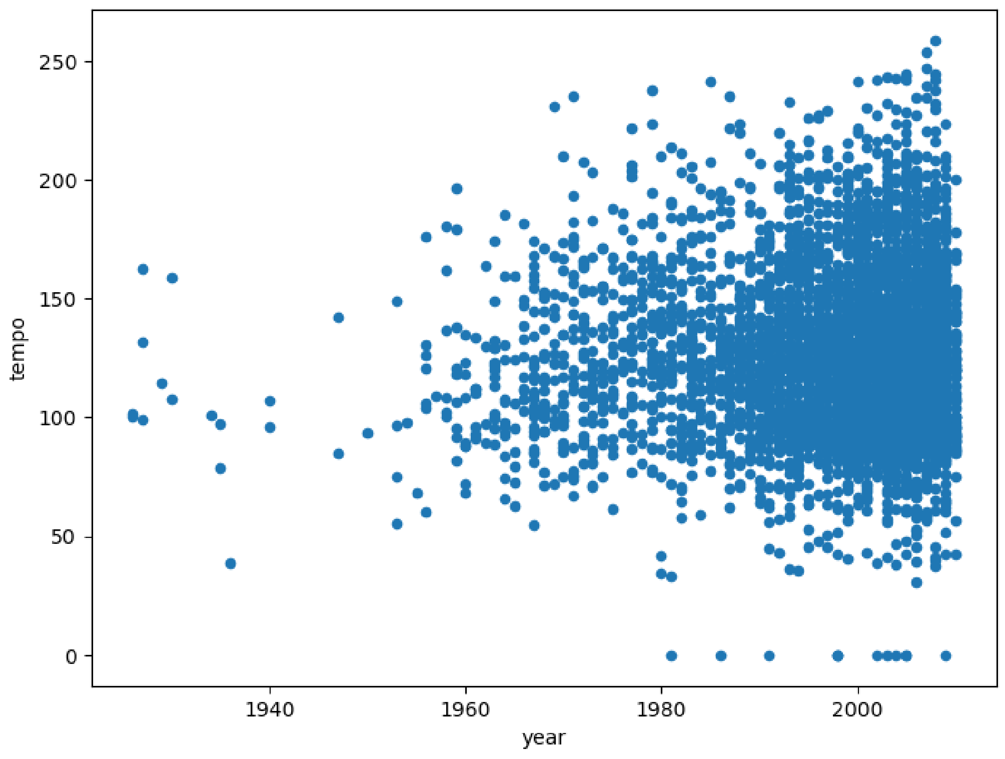


Additionally, the scatterplot of tempo vs song hotness shows that the most popular songs have a moderate tempo around 120 bpm. There is high variance in the hotness, but it looks to be approximately normally distributed. Therefore, it is very unlikely that a song that is quite fast or quite slow becomes popular. There is some interesting science about perceptions and preferences for tempo. The range of perceptible tempo is between 40 - 300 BPM. At 40 BPM, the beats are so far apart that we have trouble anticipating the next beat and discerning a musical pattern. At 300 BPM, the beats are so quick that they begin to sound like tones (A metronome playing at 300 BPM would produce a tone of 5 Hertz). So, then why is the distribution we see slightly skewed right and centered at 120 BPM if the middle of the range would be 170 BPM? In a paper titled *Marching to the beat of the same drummer: the spontaneous tempo of human locomotion,* some scientists discovered that when asking people to perform rhythmic tasks at their own tempo such as walking, tapping their fingers, cycling, and so on, the mean tempo recorded was ~120

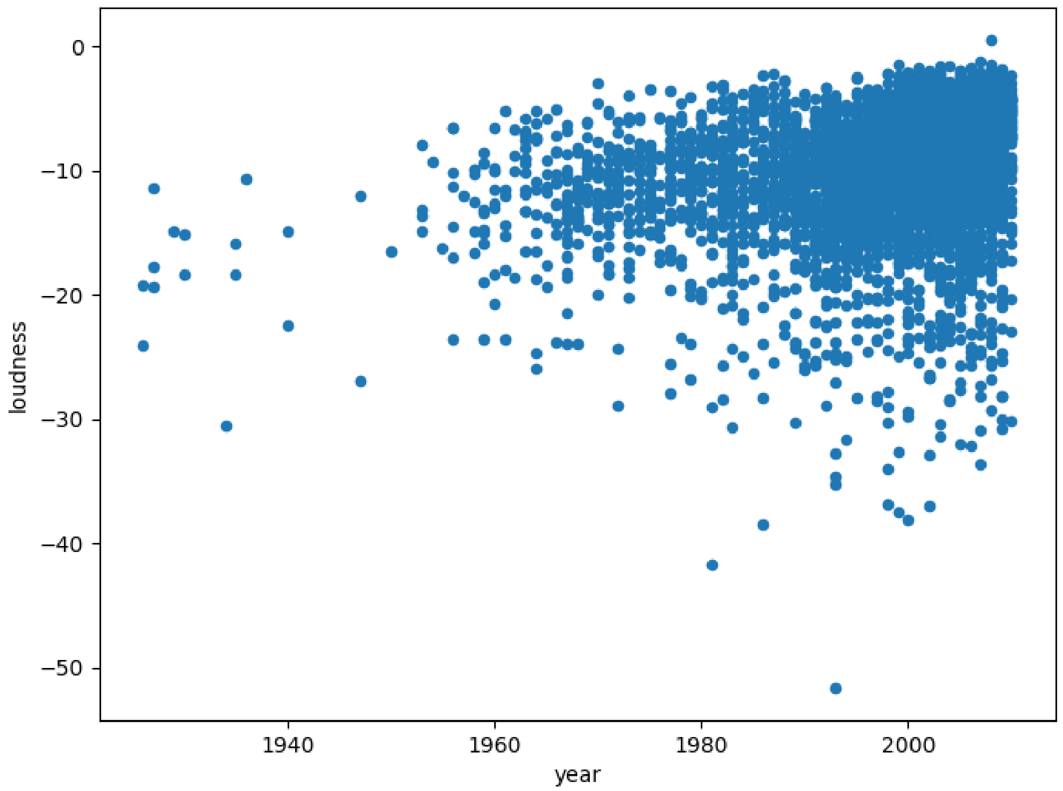
BPM. This suggests that humans naturally gravitate towards music of that speed, as it is somehow our ‘resonant frequency’.



Another noteworthy finding is the relationship between loudness and song popularity. The scatterplot indicates that the loudest songs tend to be the most popular, which could be attributed to the ability of loud music to capture and hold the listener's attention. Something that may not be as apparent here is that the year could affect how loud a song is and the popularity of the song. For reference some songs today would not have been viewed positively in prior decades. Therefore it is important to be mindful of changes as we scale the project up from this 10,000 to the full 1,000,000.



Further analysis reveals another interesting trend over time. The scatterplot above of tempo versus time shows that the variance in tempo of songs has been increasing, which could be related to the rise of electronic music and the diversity of genres in the industry. This is important due to how we would go about trying to predict the popularity of a song. Tempo having a nonconstant variance is a real issue for regression analysis. In order to find out some of the other patterns it may be beneficial to use tools such as KMeans clustering but moving forward after that tempo could make it hard for a linear regression equation to be produced.



The scatterplot of loudness vs time also shows an upward trend, which may be attributed to technological advancements in music production that enable songs to be mastered at higher volumes. As time has moved on the ways that music was stored and made has vastly changed. In the modern age producers are able to incorporate the use of digital tools to make better music and then spread it around the world. Where it once had to be recorded and transferred to a physical tape or disc before it could be shared around the world. All of these plots help understand some of what will be necessary in the k-means clustering as well as getting an idea of what will be present when considering what to include for recommending it to another user. Here the exploratory analysis of the dataset has revealed multiple paramount variables as well as the relationships they have to other necessary parts of the data.

Ethical Considerations

There are a multitude of ethical issues to consider while building a music recommendation system for a streaming platform like Spotify. While a good recommendation system may have many benefits to both the user and the business, it can also have negative knock-on effects that are unintended by the developers. First of all, something that is currently present in music recommendation systems is their tendency to over-recommend certain songs. The machine learning algorithm that chooses songs may place itself into a vicious cycle accidentally. This is because when it effectively recommends a song to users with similar music preferences, the song's popularity will increase and the algorithm may be more likely to continue recommending that song. This is good because it will improve listening time in the short term, but in the long run, users will not only get bored of the same song, but it can prevent new music from being discovered as there is no incentive for the algorithm to take risks on new music. This is a shame from a listener's perspective- while it is nice to never be shown a song you don't like, the point of a recommendation system is not to reinforce a user's tastes but recommend novel music that it thinks the user will like.

Additionally, avoiding the recommendation of new music may have some negative effects in the music industry. The vicious cycle of recommending the same music over and over will boost certain artists' sales, but make it very difficult for new musicians to enter the space. This may not only make it difficult for aspiring artists to make a living, but also discourage artists from pushing the boundaries of their art and creating something new and inspiring. Additionally, artists may try to make music that will get picked up by the algorithm instead of music that is inspiring to their fans and listeners. This is likely not good for the future of music and should be avoided when designing the recommendation system.

Something else to consider is that music is incredibly subjective. An algorithm may be fairly good at classifying music into various genres, but will it fair well against new genres, or be able to distinguish between various nuances that a human would be able to? Again, this could lead to the suppression of certain genres while boosting others, only through the model's own naivety.

When building the model, we must use listener taste data to develop a collaborative filtering system, which means that we need consent from the users to use their data. This is not a huge ethical concern since data is available publicly online through Kaggle, but it is something to consider while building the model.

In conclusion, we must be vigilant while building our recommendation system to make sure that it does not have negative effects on the state of music and treat artists unfairly.

Methodology

Our primary objective here is greatly related to which set of genres is used, in order to identify patterns relating to songs of the same genre. The data set itself gives necessary information such as the tempo, key, harmony, loudness, and time signature. Thus a song is built out of the different elements that could relate to different genres. One way to order by genre would be to examine examples of the genre and only allow songs matching in a combination of tempo, key, harmony, loudness, or time signature. This way smaller, more strict groups of songs could be formed before opening up the k-means clustering to the entire data set. In order to prepare the data we would need to take a subset of the songs as well as the genres that could be associated with it. For the genres it is important to understand and associate the different aspects of a song with how it relates to the genre. Starting from one genre and determining key values for its loudness or tempo to the next for an acceptable threshold for what could be part of the genre. Going forward with the music it is important to compare and ensure that the songs are only labeled as that genre if they satisfy at least a certain minimum for the genre classification. Finally once small and strict groups have been formed then run the whole analysis on all of the data to gauge how well the model worked. From there it would be to work on the restrictions and possibly be more liberal in the approach to allow for songs with less qualifications to be marked as that genre.

Other Preparations

It is important to note that from the exploratory data analysis there are some variables and columns that do not have specific values attached to them. This would mean that all the values for the column were the exact same at 0. Currently due to this, the genre classification system will not be able to utilize the variable columns that either have non-numeric entries or have null entries. However, there are more than enough other valuable terms in the set that should allow us to identify patterns and build a functional model.

The data set that is being used has 1,000,000 labeled songs that come with varying amounts of input such as their name, artist, year produced and more. Due to the set containing one million unique entries there are multiple viable methods available to use to analyze the data. However, in order to determine what the best model would be and to prevent overfitting the data, cross validation will also be used. To start, about 10 fold will be used so that the set can be broken up into ten equal portions, then one portion will be used to train while the other nine will be used to test and validate the model. Once that has run it will be compared to the actual labels of the songs and data in the whole set to examine just how accurate the overall model was. This is paired with a correlation matrix to see the exact numbers as well as the true ratio between what was sorted correctly or close to correct.

Another way to generate song recommendations would be to use collaborative filtering. To do this, we need a database of users with information about what music they like. For example, this could be each user's top five favorite songs. To perform collaborative filtering, we need to determine which users' tastes are similar to each other. Then, to generate recommendations, we can find the target user's closest few users by similarity and recommend them a song that those similar users like that the target user has not yet heard. We could use a few different machine learning algorithms to generate a similarity model and to pick songs.

Modeling Techniques:

K-Nearest Neighbor-

- K-Means clustering is an unsupervised learning algorithm that identifies the patterns present in the data. Now having those patterns to reference k-nearest neighbors algorithm would estimate what group that a certain data point would belong to. Thus going off multiple different aspects of the song just how likely it would be to fit the genre that it was being sorted into. Running multiple iterations of k-nearest neighbors would allow for the model to become more and more trained thus resulting in stronger clustering and the most accurate prediction. Additionally, the nearest neighbor algorithm is great for collaborative filtering because it is easy to determine each user's *nth* nearest neighbors and then pull recommendations from them.

Decision Tree/Forest-

- Decision trees or in this case a random forest of decision trees would work to find paths that would split up and then be used to narrow down the possible classifications that the data point could be identified as. Here the decision tree would work with certain input variables and find the best value to split at thus where the two classifiers break up into their distinct groups then doing it for the subgroups to more accurately classify the data point. This would work extremely well for certain genres as some are subsets or offshoots of the same genre. Hence it could find the necessary value for the parent genre, then as it continues it would find the best value to split for the sub genre classification. Alternatively there could be an issue if the song is incorrectly identified early on and is then fed further and further down the decision tree. However, using a random forest this would decrease the odds that a tree is chosen with a high amount of misclassified data points.

Naive Bayes

- Naive Bayes is a classification algorithm which relies on the assumption that each of the features it is passed for analysis are independent. This is not necessarily true, which is why it is called ‘naive’. However, it performs quite well at classification despite this often erroneous assumption. It could be very useful for genre classification, since it can work well with high-dimensional data. However, it can be tricky when working with both discrete and continuous data, like we have in the MSD. It would likely be necessary to perform a lot of data cleaning to transform the discrete features into continuous so it can be understood by the algorithm. If successful, it could prove to be a very accurate method for classification of songs into genres.

Combinations of the above

- We will most likely use a combination of the above techniques. For example, a possible case would be to use Naive Bayes to classify the songs into various genres. From there, the next possible step could be to use k-nearest-neighbors to generate a ‘map’ of all of the different users tastes and then determine any similarities between users. In order to recommend new songs, we can figure out which users like which genres, as determined by the naive bayes algorithm, and then use that information to inform the recommendations we generate by pulling songs from similar users. Another possible start would be to use the decision trees on the bigger more broken up data to then find all the sub genre classifications. Then again using the above mentioned naive bayes algorithm it could be used to make new recommendations to users based on their tastes as well as other users who listen to and enjoy other songs. All types of hybrid and combinations models used will likely lead to better results than the use of just one. Therefore the most possible outcome would be to apply all of these classification algorithms to the problems they are best at, then apply the next to make the best model. That would relate to both the genre classification as well as the recommendation systems which would benefit from multiple methods.

Analysis

Our first task was to use a machine learning algorithm to generate genre classifications for songs based on a number of features. We decided to use a Random Forest classification model because it performs well with high feature data and is able to perform multi-class classification (as opposed to binary classification). We didn’t use Naive Bayes since it is not as good at multi-class classification problems. We downloaded genre classification data from the MSD and EchoNest and we were able to match this data with about 75% of our subset. The genre data included 25 different genres: Big Band, Blues Contemporary, Country Traditional, Dance, Electronica, Experimental, Folk International, Gospel, Grunge Emo, Hip Hop Rap, Jazz Classic, Metal Alternative, Metal Death, Metal Heavy, Pop Contemporary, Pop Indie, Pop Latin, Punk, Reggae, RnB Soul, Rock Alternative, Rock College, Rock Contemporary, Rock Hard, and Rock Neo Psychedelia. We would have liked to also use the thousands of different musicbrainz genres mentioned in the literature review, but unfortunately Spotify does not provide them and I was unable to locate them anywhere else online. I then connected the Spotipy library, which is a package for Python that connects to the Spotify Web Development API. This allowed me to obtain various audio features that have been algorithmically generated by Spotify (and were missing from the MSD) such as 'danceability', 'key', 'liveness', 'instrumentalness', and so on.

Unfortunately, the Random Forest model was only about 26% accurate. This is interesting because it does about 5 times better than picking at random (1/25 chance), it is not nearly accurate enough to be useful in making strong recommendations. I did some quick tuning with little success, and did not spend much more time and energy tuning this model, since performance was so poor and gains would be marginal. In all, I decided to not use the results from this model for any content filtering since it would provide an erroneous prediction well over half of the time.

Our second task was the use of regression and model selection techniques to predict if certain songs would become popular. Here the main variable that would determine if a song was popular was the song hotttnesss variable, which describes how popular a song was on the music charts. For this portion of the analysis a subset of the full one million songs was created, then the ten thousand songs data was saved to a csv file. It is important to note here that there were more than sixty different variables available for the model selection process. However, not all of the variables were continuous, many of the variables were categorical which could lead into more categorization and or logistical regression to see if a song would be popular. Loading the data set into SAS, some simple correlation tables and exploratory plots were able to be made to visualize what the data indicated. Here the correlation table indicated that many of the variables had either no correlation or very weak correlations with song hotness, having values close to 0.0 or less than 0.5. However, that does not mean that a predictive model could not be made using the variables and data in the data set. Moving forward there were twodifferent models made for the subset, the first model was a stepwise regression and the second a mallow cp selection model.

The stepwise regression model starts out with a model that contains no variables selected, from there it then selects the best variable based on t-test statistics and its corresponding p-value, this is called the step forward. This would ensure that the true best fit model could be made, if a variable that was good was chosen then resulted in a poor output then the stepwise regression would remove it from the overall model. That is the strength of the model, with simple forward and backward model building the regression types do not allow for data to be taken out once in the model. What came out of this first part was a ten variable model, this included the variables year, artist hotness, energy, liveness, release 7 digital id, valence, mode confidence, instrumentalness, loudness, and artist latitude. The regression equation for this model was $SH = -0.1197 + Y * 0.00007399 + AH * 0.84793 + E * 0.13337 - LV * 0.05966 - R7 * 0.00000000523 - V * 0.0459 + MC * 0.04068 + I * 0.03332 + LO * 0.00321 + AL * 0.0046643$. Diving into the results from this regression were less than favorable, the overall model had a r-square value of 39.45% and an adjusted r-squared value of 39.08%, meaning that the model with these ten variables was only able to identify a weak trend and produce some accurate results(Appendix Table 1). The result of the coefficient of variance was also 53.864, which would indicate that the model requires a lot of changes to become more descriptive and fit to the data. However, although this model was weak, it did have very strong F-test and f-values, The F-test was 107.70 meaning that there was a significant pattern identified among the dataset and that this model was able to follow it. The f-value was less than 0.0001 which would suggest that the model indeed was statistically significant. It is also important to reference the residual plots from the regressions to identify what problems they may encounter in the future. The first set of residual plots (Appendix Plots Cluster-1) showed that only mode, energy, and release id, would be variables with random distributions. The other variables in the model were following some sort of pattern that would mean that the variance in that portion is nonconstant meaning that it would not be smart to continue to run regressions using it. On the other hand, it is still vital to the model so it is included but going forward it might need to be dropped or used differently to produce better results. Another issue that may have had an impact on how well the model turned out was transformations. A box cox transformation would be able to suggest if a logarithmic transformation or exponential or even root transformation need to be done on the set. This would help minimize the residuals in the model and lead to a stronger overall model.

The second modeling technique that was used for prediction was still a stepwise method, however the change here was the metric was used to determine if a variable was good to keep in the overall model. The metric used is called the CP Mallow, the measure for Cp is the total expected square bias by using expected values. So this would include using actual value subtracted from the expected value, squared, then all of that over either the standard deviation squared or the mean square error terms. This essentially includes more variable input and makes the overall more effective in predicting whether a song would be popular or not. When it was run this model included thirteen different variables including artist familiarity, artist hotness, artist latitude, duration, energy, instrumentalness, liveness, loudness, mode confidence, release 7 digital id, start of fade out, tempo, and year. The regression equation for this model would be

$$SH = -0.22067 + AF * 0.54050 + AH * 0.35236 + E * 0.00046743 + DR * 0.00201 + E * 0.1024 + I * 0.04079 - LV * 0.05238 + LO * 0.00337 + MC * 0.03714 - R7 * 0.0000000411 - SFO * 0.00206 - TM * 0.00023737 + Y * 0.00006896$$

The results from this regression were still less than favorable even after using a different measuring system, the overall model had a r-square value of 42.59% and an adjusted r-squared value of 42.14%. In comparison to the other model this is a good step in the right direction to find even stronger models for predicting if a song would become popular. Having the adjusted r-squared and regular r-squared values would mean that the model with these ten variables was able to identify a weak trend and produce somewhat more accurate results than the regular stepwise model(Appendix Table 2). It is interesting to note that one thing that came out of this model was that duration and the start of the fade out were dependent on other variables. When looking at the variable inflation rates, whether or not a variable has relations to another, their values greatly exceeded a value of 10. Anything less than or about equal to 10 would mean that there could be some relationships among the data however not enough to completely require a new equation. Thus in order for the best model out of these to be selected with these 10 variables would be to bring in and incorporate some second order terms using both duration and start of the fade would have to be made and then tested to see which one contributes the most. Although the base model was somewhat weak, it did have very strong F-test and f-values similar to the model before. However these values were slightly less than before, the F-test was 94.18 meaning that there was a significant pattern identified among the dataset and that this model was able to follow it. The f-value was less than 0.0001 which would suggest that the model indeed was statistically significant. Overall, the takeaway from the regression was that models are able to be constructed that could identify these trends and make predictions based on it, however the models are not accurate enough to show more than strong correlations. Going forward it would be necessary to include more values in certain areas and then check if second order terms are needed. Similar to the first song popularity modeling portion it is important to reference the residual plots from the Cp model regression to identify what problems it may encounter as it continues to go forward. The set of residual plots for the Cp mallow stepwise regression (Appendix Plots Cluster-2) showed that like the original stepwise regression only mode, energy, and release id, would be variables with random distributions. When looking at the subplots many of them have clear patterns or biases toward certain values. The other variable residual plots in the model were following some sort of pattern that sometimes appeared to be almost logistic. A key suggestion here would be to run some transformation functions on the data to then find the best set to use. It may still need higher order terms in order for the best model to be produced however this would greatly help in the long run. From there if the transformations did not help then maybe shrinking the dataset and then testing it using logistic regression could be better. On the other hand, it is still vital to the model as they are numeric variables so it is important that they are included but going forward it might need to be used differently to yield a better model.

Lastly, we built a collaborative filtering model using the k-Nearest-Neighbors algorithm. This model uses user taste data, also from MSD. It lists a few hundred thousand songs, matched with user IDs and the amount of times each user listened to each song. To run the model, I had to transform the data into a pivot table, where rows are users and columns are songs, and the values in each cell of the matrix are the number of plays. From there, I ran k-Nearest-Neighbors to find the 5 most similar users for each user. Then, I selected the most played song from each of the similar users that the target user had not yet heard, and recommended those 5 songs back to the target user (Appendix, Table 3). This code is ran using the 10,000 song subset and not the MSD. There is little overlap between the songs in the taste data and the songs in the subset data, so I do not have titles for the recommendations. Unfortunately, we were not able to use the full dataset for analysis. We continued to run into time-out errors using AWS while trying to open the HDF5 files from the server. Since this is an unsupervised algorithm, there is not a way to determine accuracy.

Challenges

We have encountered many challenges throughout this project. In fact, most of the challenges were in the beginning. Sadly, we were not able to access the full dataset on AWS. We are running into time-out errors since some of the code takes very long to run and were not able to fix the issue. Our first step was to transform the data from .HDF5 files into a readable format (I decided to use .CSV files and Pandas dataframes). This proved to be incredibly challenging as there is very little documentation about .HDF5 files. While it ended up only being a few lines of code, it took weeks of trial and error to find a solution that worked. In the end, I just used the Dask package to open and read the files, but I had tried countless other ways before which was incredibly frustrating.

Another challenge I encountered was code complexity. I found that some of the code I wrote would take upwards of 20 minutes to run since the datasets were quite large and I was doing complex operations to properly transform the data. I had to make sure to write code that was scalable and fast, which proved to be quite challenging.

Additionally, connecting all of our code to AWS proved to be difficult as well. To begin AWS offers a service called ec2. In ec2 users are able to create server instances which would make a server available to store data, run code, or a combination of other things. When setting up the instance in ec2 there were multiple options on the size of the server and what it would have available like storage, amount of computing cores, ram, and where the server was stored. For the purpose of out project we had to use a server based in Virginia. It was also classified as a t2.micro server. This would lead into the first challenge we had which was trying to run a bigger portion of the million songs but the server would run out of memory and then have to terminate the operation. The second challenge presented in the server was that it was in linux. This meant that in order to begin the project we had to make new directories and then attach the dataset there. However, in the ec2 service we had to find the snapshot of the whole original dataset, then make a copy of it, make it available, and then finally attach that copy of it to our server. With this we could then install python and be able to run our code. A smaller challenge was setting up a file system that could connect to the server and allow us to upload our python files to it. In order to access the server we needed to have external putty keys which were ssh. This would be given to a tool called filezilla which could then take the public dns code for the server and connect to it using the key. Once that was accomplished it the python files could be loaded onto the server and then run on it. It was a complicated process that took a couple of hours to fully understand and not all challenges were presented at the beginning leading to some redundant steps.

Recommendations and Next Steps

Overall, we think that our analysis was successful, despite facing some challenges and failures along the way. We are happy with our collaborative filtering system- it is far from perfect, but think it does a good job given the data available. We are disappointed that the content predictive algorithms were largely unsuccessful, however. The ‘better-than-random’ accuracy indicates that our goals are definitely possible to achieve, but we do not have the data, time or skills to build perfect models. In order to drastically improve the performance of these models, we would likely need much more modern data (the MSD is about 10 years old now and is largely outdated), better audio analysis data and more in depth user taste data. Additionally, it would be really interesting to see a model that relies in part on Natural Language Processing of lyrics. We think that an NLP model could drastically improve both genre and popularity predictions. We would also like to see the data improved through a much more in depth analysis of social networks on Spotify through followers, playlists, and other relevant information . There are more than a million different possibilities to build a more perfect recommendation system, and no single right way. However, we think that our attempt was a solid start at the fundamentals and provided confidence that our goals are possible, even if not fully realized in this project. Going forward there are many ways that this project could be enhanced and also ask more questions.

Appendix

Jupyter Notebook Code (Python):

```
# ## The first step is to transform all of the .hdf5 files into a usable dataframe. This first cell iterates over the directory and places the data into a Dask dataframe.
#

# In[1]:
```

```python
import time
import dask.dataframe as dd
import os
import sys
import time
import glob
import datetime
import sqlite3
import numpy as np
import pandas as pd
# we define this very useful function to iterate the files
def list_all_files(filelist,basedir,func=lambda x: x,ext='.h5'):
    """
    From a base directory, go through all subdirectories,
    find all files with the given extension, apply the
    given function 'func' to all of them.
    If no 'func' is passed, we do nothing except counting.
    INPUT
       basedir  - base directory of the dataset
       func     - function to apply to all filenames
       ext      - extension, .h5 by default
    RETURN
       number of files
    """
    cnt = 0
    # iterate over all files in all subdirectories
    for root, dirs, files in os.walk(basedir):
        files = glob.glob(os.path.join(root,'*'+ext))
        # add files to list
        filelist.extend(files)

    return filelist


start = time.time()
mylist = list()
songdir = 'C:/Users/14847/Desktop/college/spring_23/dat490/MillionSongSubset/'
songlist = list_all_files(mylist, songdir)
```

```
del songlist[:3]
print(songlist)
df = dd.read_hdf(songlist,'/metadata/songs')
df2 = dd.read_hdf(songlist, '/analysis/songs')
df4 = dd.read_hdf(songlist, '/musicbrainz/songs')
df3 = dd.multi.concat([df,df2,df4], axis=1)
df3.head()
end = time.time()


## Time - 2.5 minutes - scale to MSD... ~ 4 hours. Not bad


start = time.time()
df =df3.compute()
end = time.time()
print(end-start)
#time - 7.5 minutes - Scale to MSD... ~ 12 hours. Wish this was faster.


# ## Now, we have to use Spotipy to fill in some missing audio analysis features.

import spotipy
from spotipy.oauth2 import SpotifyClientCredentials
cid = 'e313f26451d84498a8f6b63b8041c8f1'
secret = 'e56d54964c2a4eefb5896d24b35364c6'
client_credentials_manager = SpotifyClientCredentials(client_id=cid, client_secret=secret)
sp = spotipy.Spotify(client_credentials_manager
=
client_credentials_manager)


import time
def get_sp_track_id(this_track):
    artist = this_track['artist_name']
    song = this_track['title']
    try:
        track_id = sp.search(q='artist:' + artist + ' track:' + song,type='track')
        return track_id['tracks']['items'][0]['id']
    except:
        print('no such track found')
        return None
start_time = time.time()

# Apply the function to each row of the DataFrame
sp_track_ids = []
for i, row in df.iterrows():
    sp_track_id = get_sp_track_id(row)
    sp_track_ids.append(sp_track_id)
df['sp_track_id'] = sp_track_ids

end_time = time.time()

print("Time elapsed: ", end_time - start_time, "seconds")
#time - 24 minutes - Scale to MSD... ~ 40 hours T_T



dropped_df =
df.drop(['danceability','energy','loudness','mode','tempo','key','key_confidence','time_signature']
,axis=1)
dropped_df = dropped_df[~dropped_df['sp_track_id'].isna()]
import numpy as np
import time
start = time.time()
feature_list =
['danceability','energy','key','loudness','mode','speechiness','acousticness','instrumentalness','live
ness','valence','tempo','time_signature']
features_list = []
track_ids = dropped_df['sp_track_id'] #column of all track_ids
for x in track_ids: # for each track id
    results = sp.audio_features(x) #get its audio features
    feature_values = [] #make an empty list
    try:
        for y in feature_list: #for each of the features i want
            feature_values.append(results[0][y]) #add the key value to the empty list
    except:
        print("No audio features for this track")
    features_list.append(feature_values)
features_df = pd.DataFrame(features_list,columns= feature_list)
```

```
end = time.time()
print(end-start)
# Time -  11 minutes, scaled to MSD... ~ 18 hours

#Total time for all operations -


full_df = pd.merge(features_df, dropped_df, on=dropped_df.index)


full_df.head()

import pandas as pd
df = pd.read_csv('C:/Users/14847/Desktop/college/spring_23/dat490/finished_dataset.csv')
styles = pd.read_csv('C:/Users/14847/Downloads/Styles (1).csv',header=None)

styles.rename(columns = {0:'track_id'}, inplace = True)
styles.rename(columns = {1:'Style'},inplace=True)
styles.head()
df_merged = pd.merge(df,styles, on = 'track_id',how='left').fillna(0)
df_merged


pd.DataFrame(df_merged.Style.unique())


#Let's set up a smaller df and preprocess it for a random forest genre classification model.
import numpy as np
df_with_genre = df_merged[df_merged['Style']!=0]
df_with_genre =
df_with_genre[['title','artist_name','track_id','song_id','sp_track_id','Style','danceability','energy'
,'key','loudness','speechiness','acousticness','instrumentalness','valence','tempo','time_signature',
'year']]


df_with_genre['year'].replace(0, df_with_genre['year'].median(), inplace=True)
df_with_genre['key'].replace(0, df_with_genre['key'].median(), inplace=True)
df_with_genre['instrumentalness'].replace(0, df_with_genre['instrumentalness'].median(),
inplace=True)
df_with_genre.replace(0, np.nan, inplace=True)
```

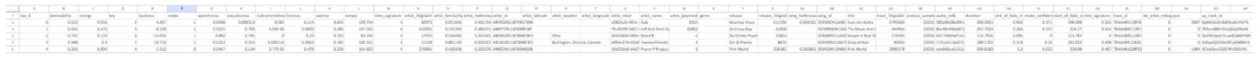
```
df_rf_dropped=df_with_genre.dropna()
kMeansFrame =
df_rf_dropped[['track_id','Style','danceability','energy','key','loudness','speechiness','acousticness','instrumentalness','valence','tempo','time_signature','year']]


df_rf_dropped.head()


from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score,
ConfusionMatrixDisplay
from sklearn.model_selection import RandomizedSearchCV, train_test_split
from scipy.stats import randint

x = df_rf_dropped.drop(['Style','title','artist_name','track_id','song_id','sp_track_id'],axis=1)
y = df_rf_dropped['Style']
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

##this is pretty bad.. maybe it is not possible to detect genre from audio features like this. Let's
try clustering instead, that may provide some better recommendations.


from sklearn.cluster import KMeans
from sklearn.preprocessing import OrdinalEncoder,LabelEncoder # for encoding categorical
features from strings to number arrays

km = KMeans(n_clusters=10, random_state=123)
le = LabelEncoder()
kMeansFrame.Style = le.fit_transform(kMeansFrame.Style)
#normalize the data
kMeansNorm = kMeansFrame.drop('track_id',axis=1)
for col in kMeansNorm.columns:
    kMeansNorm[col] = kMeansNorm[col]  / kMeansNorm[col].abs().max()
display(kMeansNorm)
```

```
#Let's tune our model
import matplotlib.pyplot as plt
wcss = [0] * 23
for i in range(2,25):
    kModel = KMeans(n_clusters=i, random_state=123)
    kModel.fit_predict(kMeansNorm)
    wcss[i-2]=kModel.inertia_
x = list(range(2,25))
plt.plot(x,wcss)
kModel = KMeans(n_clusters=10, random_state=123)
clusters = kModel.fit_predict(kMeansNorm)
kMeansFrame['cluster'] = clusters
kMeansFrame


#Looks like 10 is good, so we will keep it at that.
df_with_clusters = pd.merge(df_merged,kMeansFrame[['track_id','cluster']], on =
'track_id',how='left')
df_with_clusters


userScores = pd.read_csv('C:/Users/14847/Downloads/UserScores.csv',header=None)
#rename columns
userScores.rename(columns = {0:'user_id'}, inplace = True)
userScores.rename(columns = {1:'song_id'},inplace=True)
userScores.rename(columns = {2:'plays'},inplace=True)
userScores.head()
#get array of unique users
uniqueUser = pd.DataFrame(userScores.user_id.unique())
uniqueSongs = pd.DataFrame(userScores.song_id.unique())
uniqueUser
uniqueUser.rename(columns = {0:'user_id'}, inplace = True)
uniqueSongs.rename(columns = {0:'song_id'}, inplace = True)


subsetUserList = pd.DataFrame(userScores['user_id'].value_counts(ascending=True))
subsetUserList_over20 = subsetUserList[subsetUserList['user_id']>=20]
subsetUserList_over20.rename(columns = {'user_id':'count'}, inplace = True)
subsetUserList_over20['user_id']=subsetUserList_over20.index
```

```
subsetUserList_over20


subsetUniqueUsers =
uniqueUser[uniqueUser['user_id'].isin(subsetUserList_over20['user_id'])]
subsetUniqueUsers
subsetUserScores =  userScores[userScores['user_id'].isin(subsetUserList_over20['user_id'])]
subsetUserScores


userMatrix = subsetUserScores.pivot_table(index='user_id', columns='song_id',
values='plays',fill_value = 0)


from scipy.sparse import csr_matrix
from sklearn.neighbors import NearestNeighbors
kNNMatrix = csr_matrix(userMatrix.values)
model_knn = NearestNeighbors(metric = 'cosine', algorithm = 'brute')
model_knn.fit(kNNMatrix)


collab_songRecFrame = pd.DataFrame(subsetUniqueUsers['user_id'])
distances, indices = model_knn.kneighbors(userMatrix.iloc[0,:].values.reshape(1,-1),
n_neighbors=6)
nearNeighbors_idx = pd.DataFrame(indices)
for i, user_indices in enumerate(indices):
   print(f"User {i} nearest neighbors:", user_indices)
nearNeighbors_idx.drop(columns=nearNeighbors_idx.columns[0], axis=1,  inplace=True)
nn_idx_t = nearNeighbors_idx.T
nn_idx_t.rename(columns = {0:'user_idx'}, inplace = True)

sr = []
# for idx in nn_idx_t
for i in nn_idx_t['user_idx']:
   nn1 = userScores[userScores['user_id']==userMatrix.iloc[i,:].name]
   user = userScores[userScores['user_id']==userMatrix.iloc[0,:].name]
   nn_novel = nn1[~nn1['song_id'].isin(user['song_id'])]
   nn_novel.sort_values('plays',ascending=False)
```

```
    sr.append(nn_novel['song_id'].head(1))

sr


all_sr = []
userlist= []
import time
start = time.time()
for user in range(0,10):

    distances, indices = model_knn.kneighbors(userMatrix.iloc[user,:].values.reshape(1,-1),
n_neighbors=6)# get indices for 6 NN
    nearNeighbors_idx = pd.DataFrame(indices) #make a dataframe list of the indices
    nearNeighbors_idx.drop(columns=nearNeighbors_idx.columns[0], axis=1,  inplace=True)
#drop the first one (user themselves)
    nn_idx_t = nearNeighbors_idx.T #transpose
    nn_idx_t.rename(columns = {0:'user_idx'}, inplace = True) #rename column
    sr = []
    for i in nn_idx_t['user_idx']: #for 5 nearest users
        nn1 = userScores[userScores['user_id']==userMatrix.iloc[i,:].name] # get rows from
userScores at index i
        user1 = userScores[userScores['user_id']==userMatrix.iloc[user,:].name] # get rows from
userScores at
        nn_novel = nn1[~nn1['song_id'].isin(user1['song_id'])]
        nn_novel.sort_values('plays',ascending=False)
        sr.append(nn_novel['song_id'].iloc[0])
    all_sr.append(sr)
    userlist.append(userMatrix.iloc[user,:].name)
end= time.time()
print(end-start)

songRecs = pd.DataFrame(all_sr)
songRecs['users'] = userlist
songRecs['song0'] = df_rf_dropped[df_rf_dropped['song_id'].isin(songRecs[0])].title
songRecs['song1'] = df_rf_dropped[df_rf_dropped['song_id'].isin(songRecs[1])].title
songRecs['song2'] = df_rf_dropped[df_rf_dropped['song_id'].isin(songRecs[2])].title
songRecs['song3'] = df_rf_dropped[df_rf_dropped['song_id'].isin(songRecs[3])].title
songRecs['song4'] = df_rf_dropped[df_rf_dropped['song_id'].isin(songRecs[4])].title
#df_match = df_rf_dropped.drop(df_rf_dropped['song_id'].isin(songRecs[0]))
```

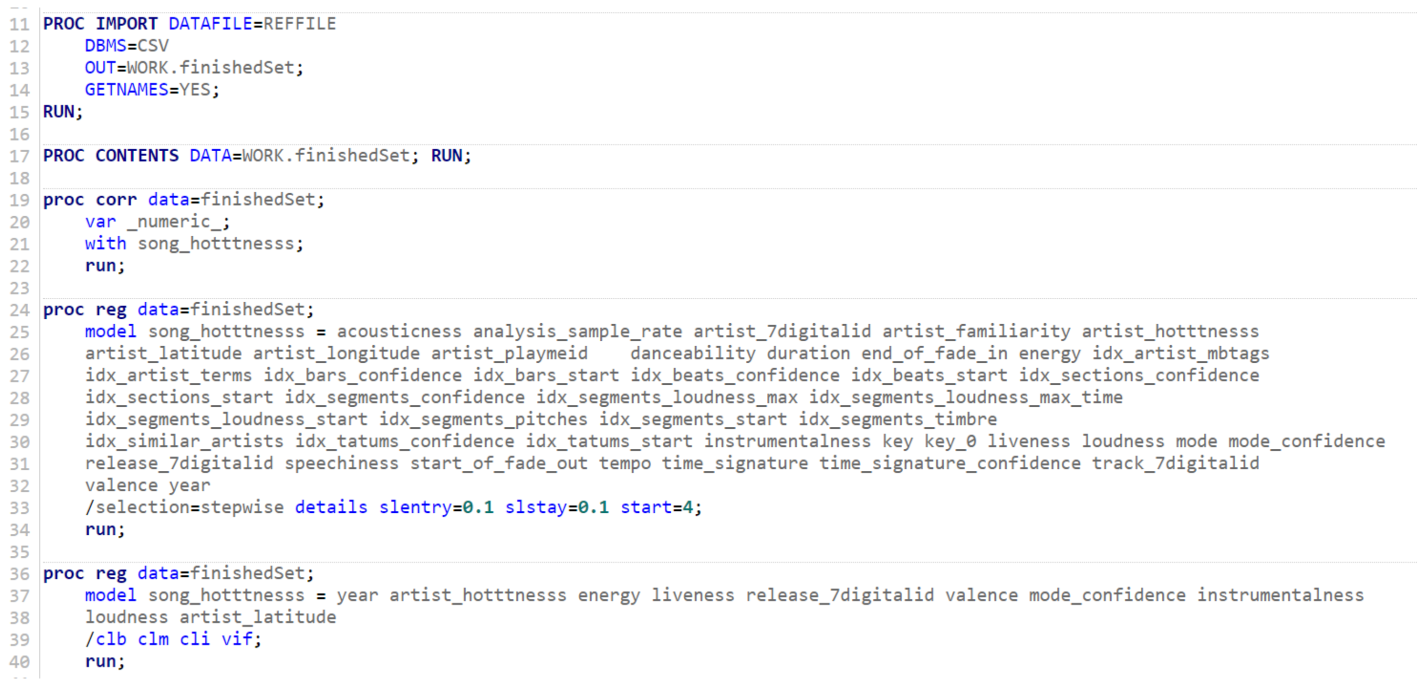
Head of DataFrame: This is very large and does not fit on the Doc very well. I will include the CSV file of the subset in the submission if you would like to look at it.



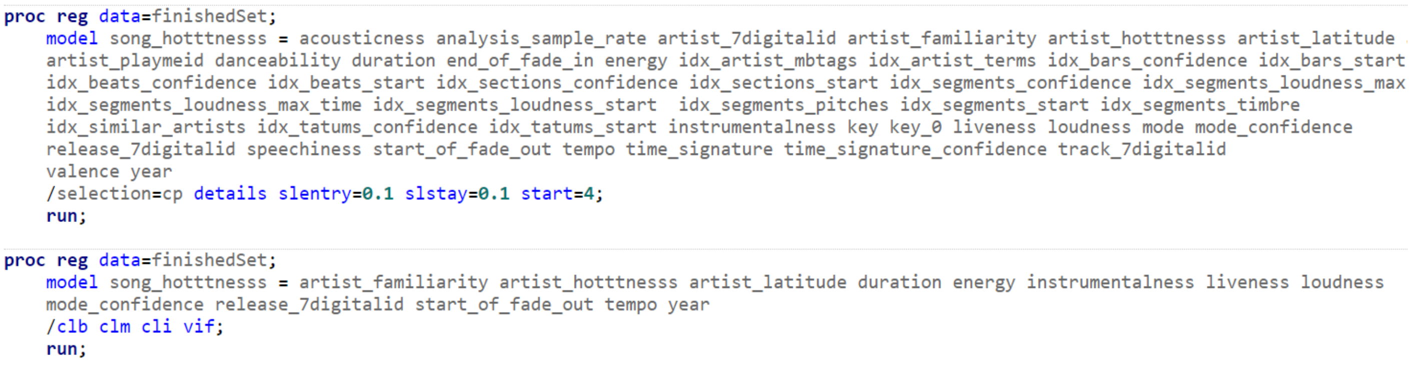
key_0	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalnes	liveness	valence	tempo	time_signature	artist_7digitalid	artist_familiarity	artist_hotttness	artist_id	artist_latitude	artist_location	artist_longitude
0	0.523	0.916	2	-4.667	1	0.0498	0.000313	0.582	0.114	0.665	129.724	4	30973	0.651046	0.401724	ARXR32B1187FB57099			
1	0.456	0.472	5	-8.328	1	0.0325	0.706	4.93E-06	0.0833	0.286	147.565	4	432935	0.535293	0.385471	ARKFYS91187B98E58F			
2	0.742	0.174	3	-16.056	1	0.892	0.795	0	0.23	0.702	82.244	4	17970	0.556496	0.261941	ARD0S291187B9B7BF5		Ohio	
3	0.498	0.3	7	-10.716	1	0.0352	0.553	0.000116	0.0963	0.185	100.315	3	21128	0.801136	0.605507	AR10USD1187B99F3F1		Burlington, Ontario, Canada	
4	0.332	0.834	4	-5.312	0	0.0467	0.134	3.77E-05	0.479	0.328	104.923	4	276891	0.426668	0.332276	ARBZCNI1187B9A069B			

artist_mbid	artist_name	artist_playmeid	genre	release	release_7digitali	song_hotttness	song_id	title	track_7digitalid	analysis_sample	audio_md5	duration	end_of_fade_in	mode_confidenc	start_of_fade_o	time_signature_	track_id	idx_artist_mbtag	year	sp_track_id
c6903a2e-063c-	Gob	8355		Muertos Vivos	611336	0.604501	SOFSOCN12A8C	Face the Ashes	6795666	22050	580a8fe08ef0f1	209.6061	0.066	0.371	198.699	0.562	TRAAAFD128F92	0	2007	0qMOsUBinMDILdStVIUiTL
79c403f9-5467-	Jeff And Sheri Ea	63801		Ordinary Day	41838		SOYMRWW12A(	The Moon And I	444964	22050	8ee90e90bb871	267.7024	2.264	0.557	254.27	0.454	TRAAAMO128F1	0	0	7ERvcsBBYzSHqQQqI9xAN
56503d6d-094e-	Rated R	-1		Da Ghetto Psych	25824		SOMJBYD12A6D	Keepin It Real (S	276593	22050	0e574964bf7a5	114.7816	0.096	0	114.782	0	TRAAAMQ128F1	0	0	0e0hF4Jxyh7cuw82pMYhBh
d89de379-665d-	Tweeterfriendly	-1		Gin & Phonic	8876		SOHKNRJ12A67	Drop of Rain	90004	22050	1741d2c1bd271	189.5702	0.319	0.16	181.023	0.408	TRAAAPK128E0	0	0	6dfppQDZQ4x3fCsGN89kV3
19d232b9-b4d7	Planet P Project	-1		Pink World	358182	0.265861	SOIAZJW12AB01	Pink World	3996579	22050	eda960ea61211	269.8183	5.3	0.652	258.99	0.487	TRAAARJ128F93	0	1984	0CVyGAvr532CYRISDGVRtj

SAS Code:

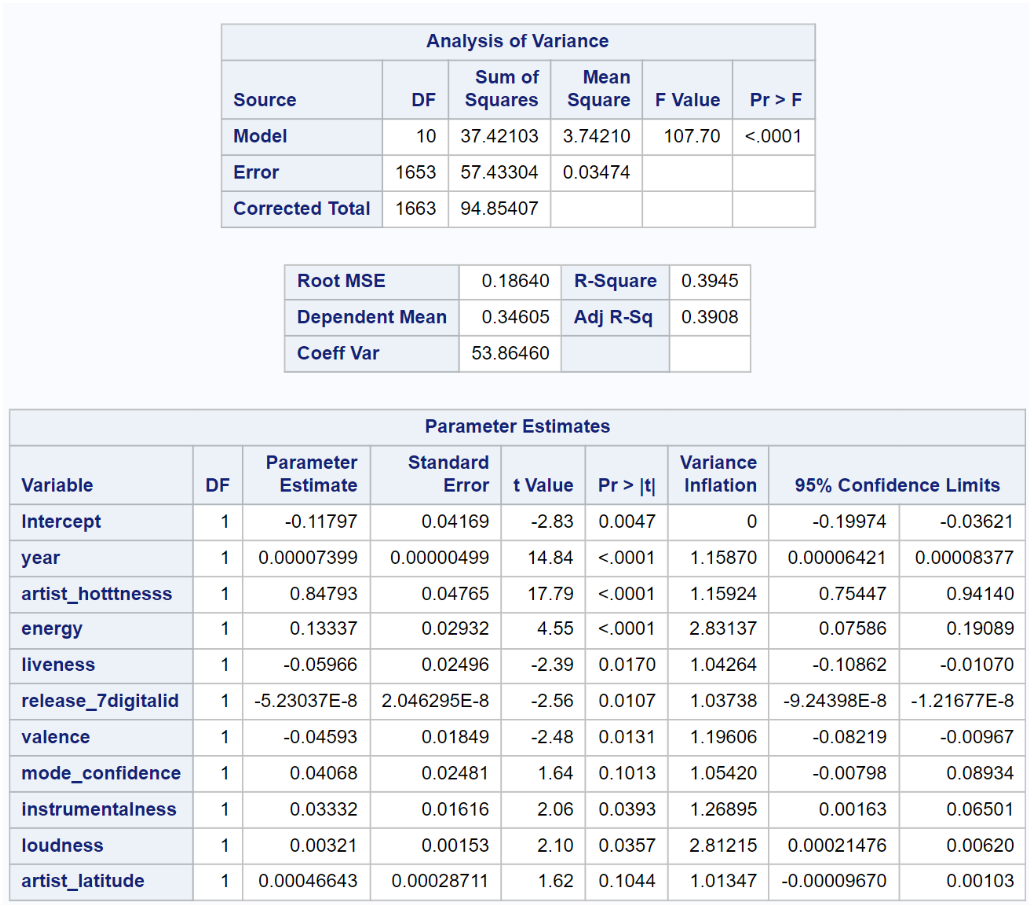


```
PROC IMPORT DATAFILE=REFFILE
    DBMS=CSV
    OUT=WORK.finishedSet;
    GETNAMES=YES;
RUN;

PROC CONTENTS DATA=WORK.finishedSet; RUN;

proc corr data=finishedSet;
    var _numeric_;
    with song_hotttnesss;
    run;

proc reg data=finishedSet;
    model song_hotttnesss = acousticness analysis_sample_rate artist_7digitalid artist_familiarity artist_hotttnesss
    artist_latitude artist_longitude artist_playmeid    danceability duration end_of_fade_in energy idx_artist_mbtags
    idx_artist_terms idx_bars_confidence idx_bars_start idx_beats_confidence idx_beats_start idx_sections_confidence
    idx_sections_start idx_segments_confidence idx_segments_loudness_max idx_segments_loudness_max_time
    idx_segments_loudness_start idx_segments_pitches idx_segments_start idx_segments_timbre
    idx_similar_artists idx_tatums_confidence idx_tatums_start instrumentalness key key_0 liveness loudness mode mode_confidence
    release_7digitalid speechiness start_of_fade_out tempo time_signature time_signature_confidence track_7digitalid
    valence year
    /selection=stepwise details slentry=0.1 slstay=0.1 start=4;
    run;

proc reg data=finishedSet;
    model song_hotttnesss = year artist_hotttnesss energy liveness release_7digitalid valence mode_confidence instrumentalness
    loudness artist_latitude
    /clb clm cli vif;
    run;
```



```
proc reg data=finishedSet;
    model song_hotttnesss = acousticness analysis_sample_rate artist_7digitalid artist_familiarity artist_hotttnesss artist_latitude
    artist_playmeid danceability duration end_of_fade_in energy idx_artist_mbtags idx_artist_terms idx_bars_confidence idx_bars_start
    idx_beats_confidence idx_beats_start idx_sections_confidence idx_sections_start idx_segments_confidence idx_segments_loudness_max
    idx_segments_loudness_max_time idx_segments_loudness_start  idx_segments_pitches idx_segments_start idx_segments_timbre
    idx_similar_artists idx_tatums_confidence idx_tatums_start instrumentalness key key_0 liveness loudness mode mode_confidence
    release_7digitalid speechiness start_of_fade_out tempo time_signature time_signature_confidence track_7digitalid
    valence year
    /selection=cp details slentry=0.1 slstay=0.1 start=4;
    run;

proc reg data=finishedSet;
    model song_hotttnesss = artist_familiarity artist_hotttnesss artist_latitude duration energy instrumentalness liveness loudness
    mode_confidence release_7digitalid start_of_fade_out tempo year
    /clb clm cli vif;
    run;
```

Table 1 - Stepwise Regression Output:

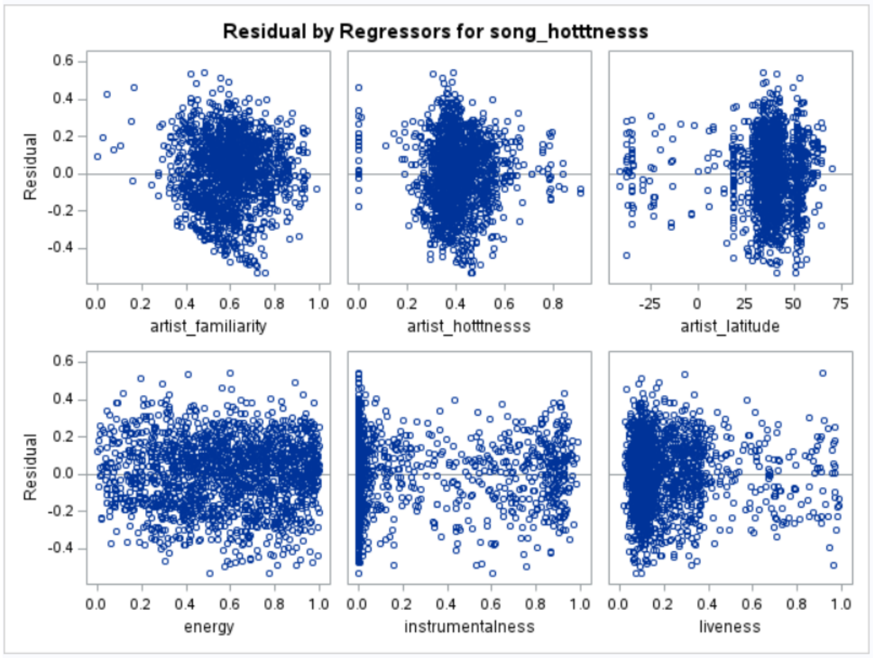


Analysis of Variance					
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	10	37.42103	3.74210	107.70	<.0001
Error	1653	57.43304	0.03474		
Corrected Total	1663	94.85407			

Root MSE	0.18640	R-Square	0.3945
Dependent Mean	0.34605	Adj R-Sq	0.3908
Coeff Var	53.86460		

Parameter Estimates								
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variance Inflation	95% Confidence Limits	
Intercept	1	-0.11797	0.04169	-2.83	0.0047	0	-0.19974	-0.03621
year	1	0.00007399	0.00000499	14.84	<.0001	1.15870	0.00006421	0.00008377
artist_hotttnesss	1	0.84793	0.04765	17.79	<.0001	1.15924	0.75447	0.94140
energy	1	0.13337	0.02932	4.55	<.0001	2.83137	0.07586	0.19089
liveness	1	-0.05966	0.02496	-2.39	0.0170	1.04264	-0.10862	-0.01070
release_7digitalid	1	-5.23037E-8	2.046295E-8	-2.56	0.0107	1.03738	-9.24398E-8	-1.21677E-8
valence	1	-0.04593	0.01849	-2.48	0.0131	1.19606	-0.08219	-0.00967
mode_confidence	1	0.04068	0.02481	1.64	0.1013	1.05420	-0.00798	0.08934
instrumentalness	1	0.03332	0.01616	2.06	0.0393	1.26895	0.00163	0.06501
loudness	1	0.00321	0.00153	2.10	0.0357	2.81215	0.00021476	0.00620
artist_latitude	1	0.00046643	0.00028711	1.62	0.1044	1.01347	-0.00009670	0.00103

Plot Cluster 1 - Stepwise Regression Residual Plots:


Residual by Regressors for song_hotttnesss
Residual
artist_familiarity
artist_hotttnesss
artist_latitude
energy
instrumentalness
liveness

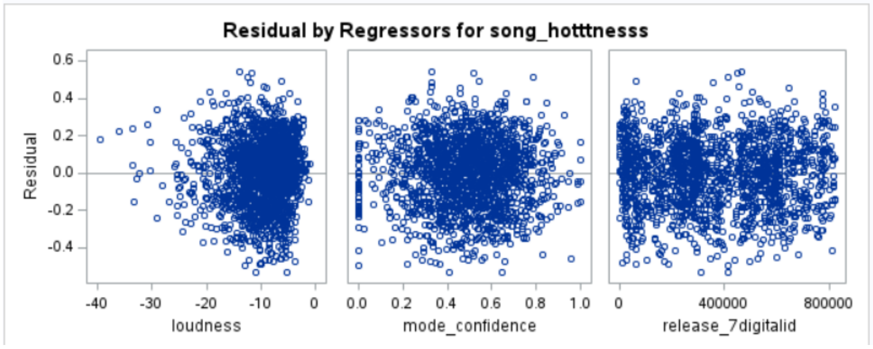
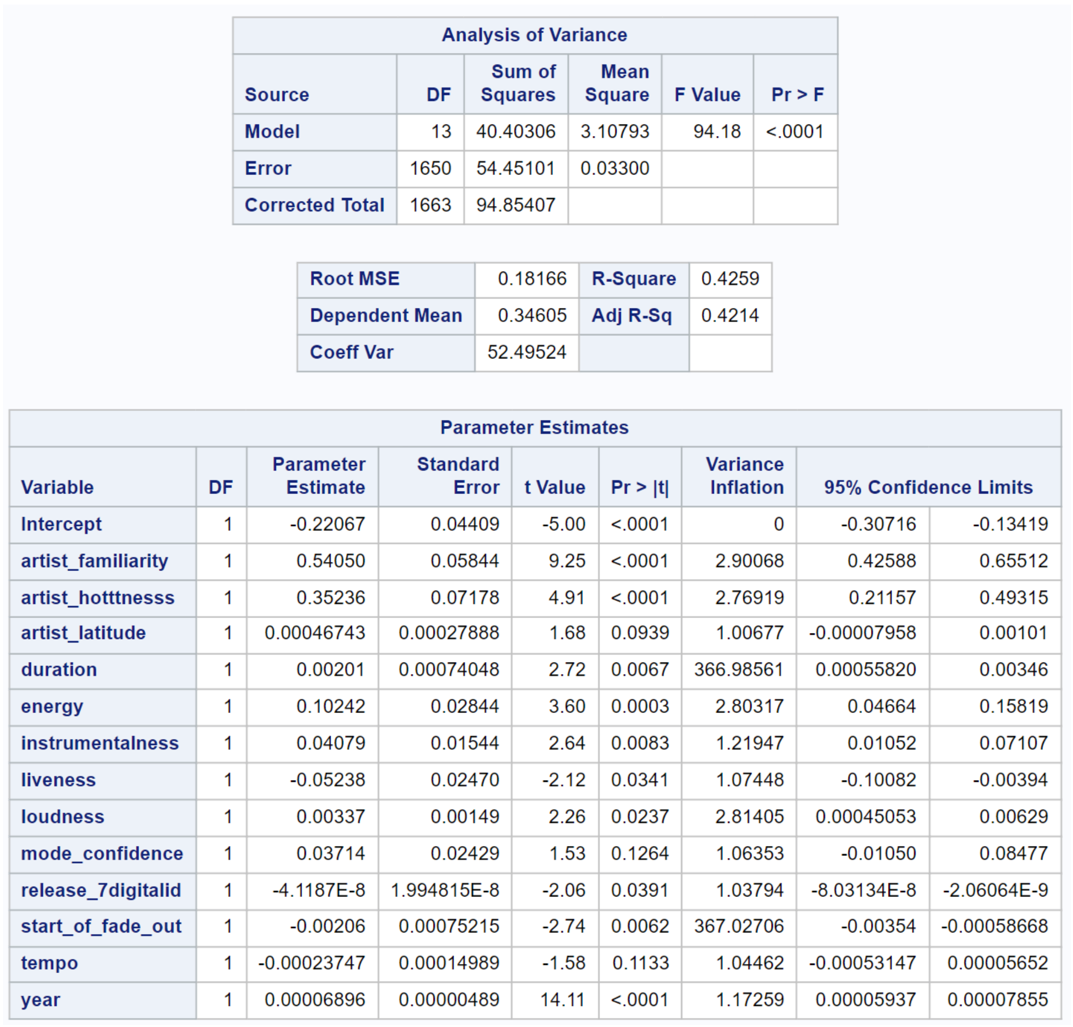

Residual by Regressors for song_hotttnesss
Residual
loudness
mode_confidence
release_7digitalid

Table 2- CP Mallow Output:

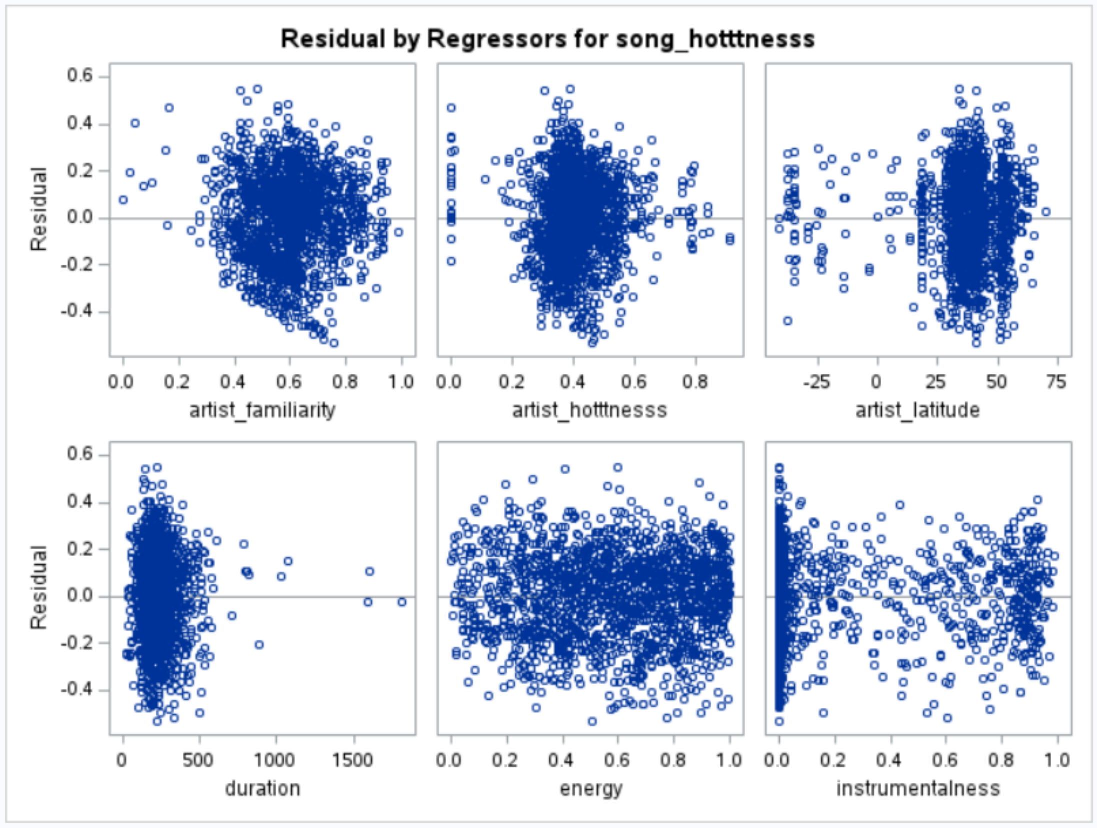


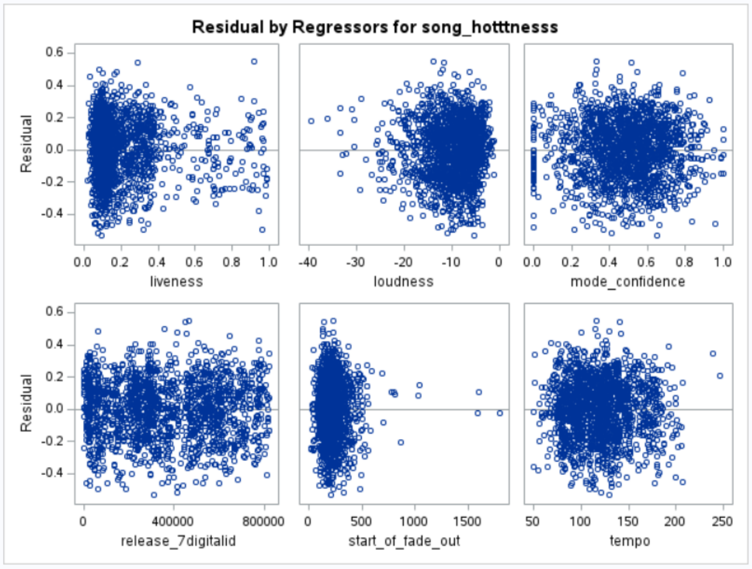
Analysis of Variance					
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	13	40.40306	3.10793	94.18	<.0001
Error	1650	54.45101	0.03300		
Corrected Total	1663	94.85407			

Root MSE	0.18166	R-Square	0.4259
Dependent Mean	0.34605	Adj R-Sq	0.4214
Coeff Var	52.49524		

Parameter Estimates								
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variance Inflation	95% Confidence Limits	
Intercept	1	-0.22067	0.04409	-5.00	<.0001	0	-0.30716	-0.13419
artist_familiarity	1	0.54050	0.05844	9.25	<.0001	2.90068	0.42588	0.65512
artist_hotttnesss	1	0.35236	0.07178	4.91	<.0001	2.76919	0.21157	0.49315
artist_latitude	1	0.00046743	0.00027888	1.68	0.0939	1.00677	-0.00007958	0.00101
duration	1	0.00201	0.00074048	2.72	0.0067	366.98561	0.00055820	0.00346
energy	1	0.10242	0.02844	3.60	0.0003	2.80317	0.04664	0.15819
instrumentalness	1	0.04079	0.01544	2.64	0.0083	1.21947	0.01052	0.07107
liveness	1	-0.05238	0.02470	-2.12	0.0341	1.07448	-0.10082	-0.00394
loudness	1	0.00337	0.00149	2.26	0.0237	2.81405	0.00045053	0.00629
mode_confidence	1	0.03714	0.02429	1.53	0.1264	1.06353	-0.01050	0.08477
release_7digitalid	1	-4.1187E-8	1.994815E-8	-2.06	0.0391	1.03794	-8.03134E-8	-2.06064E-9
start_of_fade_out	1	-0.00206	0.00075215	-2.74	0.0062	367.02706	-0.00354	-0.00058668
tempo	1	-0.00023747	0.00014989	-1.58	0.1133	1.04462	-0.00053147	0.00005652
year	1	0.00006896	0.00000489	14.11	<.0001	1.17259	0.00005937	0.00007855

Plost Cluster 2 - CP Mallow Residual Plots:


Residual by Regressors for song_hotttnesss
Residual
artist_familiarity
artist_hotttnesss
artist_latitude
duration
energy
instrumentalness


Residual by Regressors for song_hotttnesss
Residual
liveness
loudness
mode_confidence
release_7digitalid
start_of_fade_out
tempo

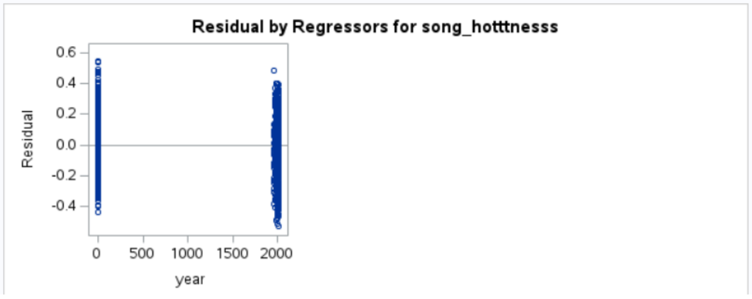
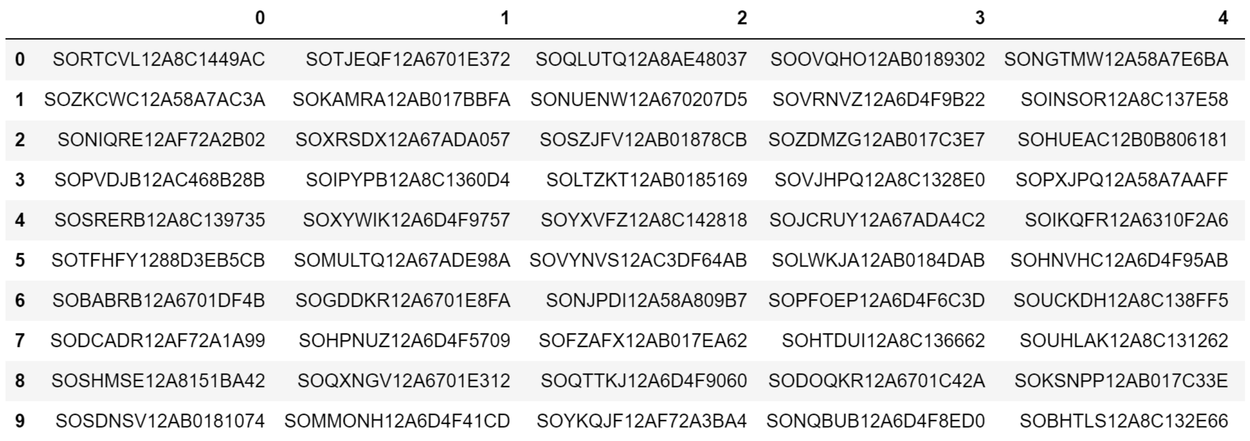

Residual by Regressors for song_hotttnesss
Residual
year

Table 3: Song recommendations (*Still need titles matched using full dataset)*



	0	1	2	3	4
0	SORTCVL12A8C1449AC	SOTJEQF12A6701E372	SOQLUTQ12A8AE48037	SOOVQHO12AB0189302	SONGTMW12A58A7E6BA
1	SOZKCWC12A58A7AC3A	SOKAMRA12AB017BBFA	SONUENW12A670207D5	SOVRNVZ12A6D4F9B22	SOINSOR12A8C137E58
2	SONIQRE12AF72A2B02	SOXRSDX12A67ADA057	SOSZJFV12AB01878CB	SOZDMZG12AB017C3E7	SOHUEAC12B0B806181
3	SOPVDJB12AC468B28B	SOIPYPB12A8C1360D4	SOLTZKT12AB0185169	SOVJHPQ12A8C1328E0	SOPXJPQ12A58A7AAFF
4	SOSRERB12A8C139735	SOXYWIK12A6D4F9757	SOYXVFZ12A8C142818	SOJCRUY12A67ADA4C2	SOIKQFR12A6310F2A6
5	SOTFHFY1288D3EB5CB	SOMULTQ12A67ADE98A	SOVYNVS12AC3DF64AB	SOLWKJA12AB0184DAB	SOHNVHC12A6D4F95AB
6	SOBABRB12A6701DF4B	SOGDDKR12A6701E8FA	SONJPDI12A58A809B7	SOPFOEP12A6D4F6C3D	SOUCKDH12A8C138FF5
7	SODCADR12AF72A1A99	SOHPNUZ12A6D4F5709	SOFZAFX12AB017EA62	SOHTDUI12A8C136662	SOUHLAK12A8C131262
8	SOSHMSE12A8151BA42	SOQXNGV12A6701E312	SOQTTKJ12A6D4F9060	SODOQKR12A6701C42A	SOKSNPP12AB017C33E
9	SOSDNSV12AB0181074	SOMMONH12A6D4F41CD	SOYKQJF12AF72A3BA4	SONQBUB12A6D4F8ED0	SOBHTLS12A8C132E66

Works Cited

Bertin-Mahieux, Thierry, et al. *The Million Song Dataset*. 2011. *DOI.org (Datacite)*, https://doi.org/10.7916/D8NZ8J07.

Fessahaye, Ferdos, et al. “T-RECSYS: A Novel Music Recommendation System Using Deep Learning.” *2019 IEEE International Conference on Consumer Electronics (ICCE)*, IEEE, 2019, pp. 1–6. *DOI.org (Crossref)*, https://doi.org/10.1109/ICCE.2019.8662028.

Flexer, Arthur, Elias Pampalk, and Gerhard Widmer. "Novelty Detection Based on Spectral Similarity of Songs." ISMIR. 2005.

Madathil, Mithun. *Music Recommendation System Spotify-Collaborative Filtering*. July 2017.

Spotify. *Discover Spotify's Features*. 1 Jan. 2023, https://developer.spotify.com/discover/.

Wang, Wengsheng, and Changkai Zhou. “A Two-Layer Aggregation Model with Effective Consistency for Large-Scale Gaussian Process Regression.” *Engineering Applications* *of Artificial Intelligence*, vol. 106, Nov. 2021, p. 104449. *DOI.org (Crossref)*, https://doi.org/10.1016/j.engappai.2021.104449.

Yang, Zhen, et al. "A Systematic Literature Review of Methods and Datasets for Anomaly-Based Network Intrusion Detection." *Computers & Security*, vol. 116, May 2022, p. 102675. *DOI.org (Crossref)*, https://doi.org/10.1016/j.cose.2022.102675.