CMPT 353: Song Predictions

Brian Le, Emmanuel Okonkwo, Mila Kobayashi

1. Introduction

As a group we listed out topics that we were interested in researching. The common interest among our team members was music. With this in mind, we looked at the Spotify API due to the fact they provided numeric values that described songs.

While we had a dataset, we did not have a question in mind. Eventually we came up with the initial question: "Given a user's playlist, can we predict some songs that the user may also like?". We further gathered and explored lyrical features of the songs to discover any relationships between lyrics and Spotify playlist data including audio features and popularity.

2. Data Extraction and Transformation

2.1. Extraction

2.1.1. Spotify Playlist

To extract data from each song in the Spotify playlist, we used the Spotify API along with spotipy to retrieve the following information:

- Track features:
    - Danceability, Energy, Key, Loudness, Mode, Speechiness, Acousticness, Instrumentalness, Liveness, Valence, Tempo, Type, Uri, Track href, Analysis URL, Duration_ms, Time_signature
- Song information:
    - Song Title, Artist, Album, Id, Artist URI, Artist Genres, Popularity

2.1.2. Playlist Selections

We chose some pre-made playlists provided by Spotify for popular genres such as rock, country, pop, classical, and rap, in addition to our own personal playlists that we listen to regularly. The pre-made playlists were chosen to see how our model behaves with different genres, while the personal playlists were chosen to serve as an evaluation metric for our model discussed later in the paper.

2.1.3. Lyrics

As the Spotify API is unable to retrieve lyrics we resorted to a different API. Two of our choices were lyricsgenius and swaglyrics API. Upon testing both we found swaglyrics to be significantly faster, thus for a greater performance swaglyrics was used to gather the lyrics for the Spotify playlists.

The lyrics returned by swaglyrics contained components such as “[Verse 1]” or “[Chorus: Reggie Green &amp; Sweet Franklin, 2Pac]” throughout the lyrics. These components were removed along with some punctuations and stopwords using the NLTK stopwords dictionary.

2.2. Transformation

The artist genres extracted from the Spotify API consist of several genres in a list such as ['dance pop', 'europop', 'girl group', 'pop'] for example. To convert the genres to a numerical value, we first created a list of all 5833 spotify genres retrieved from Every Noise at Once. Using the index of this list as the numerical value for each genre, the genres were first quantified then one-hot encoded.

3. Representation of Lyrical Features

In this section we will explore two ways to quantify lyrics to use in the subsequent sections for statistical tests and as a feature for recommendation models.

3.1. TFIDF Score

To better compute a score for lyrics pertaining to a specific song in a playlist as a part of an entire playlist's lyrics dataset, we introduce the Term Frequency-Inverse Document Frequency (or TFIDF for short). The TFIDF score depicts the number of occurrences of a word of a lyric in relation to an entire list of words throughout a lyrics document [2].

We begin by determining the TF score for each word within a particular lyric using the formula below:

$$TF_{i,j} = \frac{|number\ of\ occurrences\ of\ word\ i\ within\ a\ lyric\ j|}{|lyric\ j|} \quad [1]$$

$$IDF_{i} = \frac{|total\ number\ of\ lyrics\ in\ dataset\ j|}{|number\ of\ lyrics\ containing\ word\ i|} \quad [1]$$

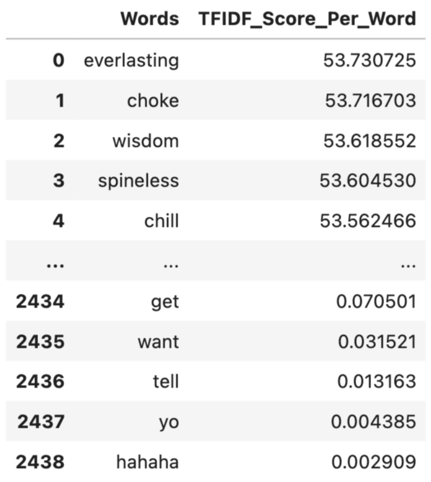
Next, we calculate the inverse document frequency score for each lyric within the document
The resulting TFIDF score from the computations becomes:

$$TFIDF_{i,j} = TF_{i,j} \times IDF_{i} \quad [1]$$

Upon performing the computation in the lyrics_tfidf.py file, we store the output composed each word in all song lyrics within the playlist alongside their corresponding TFIDF scores. This is accomplished by running the following commands:

```
i. python3 lyrics.py ../playlists/spotifyTopHits.csv
ii. python3 lyrics_tfidf.py ../playlists/spotifyTopHits.csv
```

A clip of the output as can be seen in the Lyrics_TFIDF_Score file is below:



	Words	TFIDF_Score_Per_Word
0	everlasting	53.730725
1	choke	53.716703
2	wisdom	53.618552
3	spineless	53.604530
4	chill	53.562466
...	...	...
2434	get	0.070501
2435	want	0.031521
2436	tell	0.013163
2437	yo	0.004385
2438	hahaha	0.002909

3.2. Measuring Valence, Arousal, and Dominance Scores from Lyrics

We quantified the mood of the song by estimating valence, arousal, and dominance scores from the lyrics. Negative or low valence scores are associated with depressed/frustrated emotions while positive or high valence scores relate to happy/pleased emotions. For arousal, negative or low scores are associated with tired/sleepy emotions while excited/tense emotions will result in positive or high scores. Dominance emotions range from fear with a low score to admiration with a high score.

The valence, arousal, and dominance scores were measured using the Affective Norms for English Words (ANEW) which provides emotional word ratings for the three categories for 13,915 words. The following 3D scatterplot shows 30 English words from ANEW and their valence(x), arousal(y), and dominance(z) scores. As expected, words such as "relaxed" are given high valence and dominance scores but a low arousal score. See lyrics_mood.ipynb in our repository to view a range of words from high to low valence/arousal/dominance scores and to interact with the following plot.

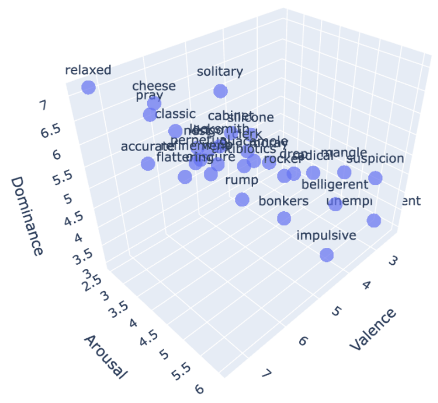


Fig. 3-dimensional valence, arousal, and dominance scatterplot

For each song in the playlist we will use the following formula to find the valence, arousal, and dominance scores:

$$Score \ = \ \frac{1}{n}\sum_{i=1}^{n} rating(w^{i})$$

n is the number of words in the song and w^i is the ith word in the song. $rating(w^i)$ finds the ANEW rating of the word. When the word is not found in ANEW, a rating of 0 will be returned. The ratings for the individual words in the song will be summed and divided by the number of words in the whole song. The division is done to scale the score so that it is not skewed for songs with more words. We will use this formula for all three scores with $rating(w^i)$ corresponding to either valence, arousal, or dominance.

Using the formula above, songs in the spotifyTopHits.csv such as "Bam Bam" received a low score for all three features as expected due to words such as "hate" appearing in the song. On the other hand, "Infinity" with words such as "love" and "paradise" gave high scores, especially for valence resulting in 3.5 compared to most other songs with a score < 3.

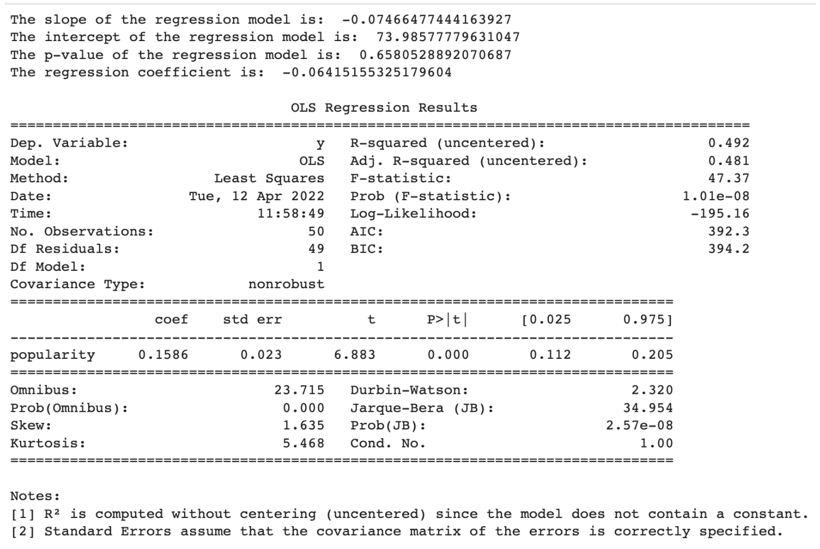
4. Data Analysis

4.1. Statistical Tests Using TFIDF

4.1.1. Linear Regression Model

In the following section, we would discuss the results of using statistical tests to determine whether there is any correlation between words with the highest TFIDF scores and the popularity of songs in a playlist. We firstly begin by computing the average tfidf scores of the words in a playlist and testing it alongside the popularity of each song in the playlist to probe the existence of some form of correlation using the linear regression model.

Hence, incorporating linear regression using the *stats.linregress* python module and evaluating the average TFIDF scores across all playlists elicits the output below:



```
The slope of the regression model is:  -0.07466477444163927
The intercept of the regression model is:  73.98577779631047
The p-value of the regression model is:  0.6580528892070687
The regression coefficient is:  -0.06415155325179604

                                 OLS Regression Results                                
=======================================================================================
Dep. Variable:                      y   R-squared (uncentered):                   0.492
Model:                            OLS   Adj. R-squared (uncentered):              0.481
Method:                 Least Squares   F-statistic:                              47.37
Date:                Tue, 12 Apr 2022   Prob (F-statistic):                    1.01e-08
Time:                        11:58:49   Log-Likelihood:                         -195.16
No. Observations:                  50   AIC:                                      392.3
Df Residuals:                      49   BIC:                                      394.2
Df Model:                           1                                                  
Covariance Type:            nonrobust                                                  
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
popularity     0.1586      0.023      6.883      0.000       0.112       0.205
==============================================================================
Omnibus:                       23.715   Durbin-Watson:                   2.320
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               34.954
Skew:                           1.635   Prob(JB):                     2.57e-08
Kurtosis:                       5.468   Cond. No.                         1.00
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

4.1.2. Formulating word frequency based on each word's TFIDF score

To conceptualize the relevancy of the TFIDF scores of each word, we would integrate the use of a wordcloud.
The wordcloud works by filtering through each word in a lyric while storing each new occurrence in a list. This classifies the words by the number of times they appear in all the lyrics of an entire playlist which is a measure of each word's TFIDF score.

The result from running the wordcloud.py file with the spotifyTopHits.csv file as an input is below:

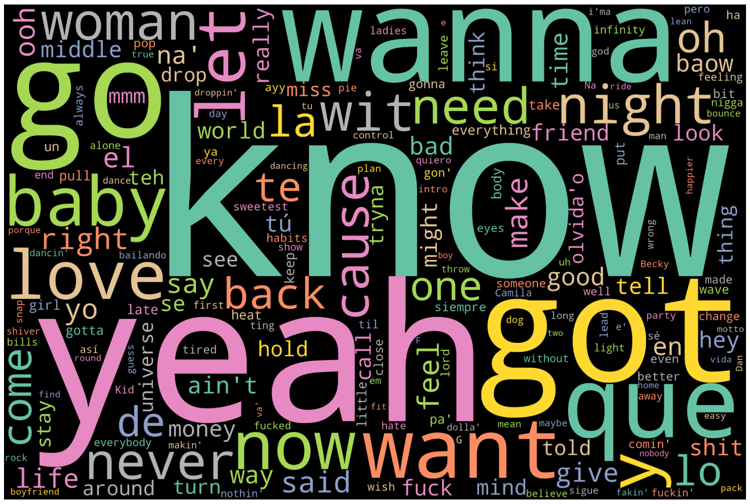


Fig. Wordcloud Image depicting the frequency of the words in a playlist

4.2. Statistical Correlation between Valence/Arousal/Dominance and Audio Features

Using the features valence, arousal, and dominance which quantify the mood of the song from the lyrics, we would like to determine if there exists any relationship between the lyrics/mood and the audio features by calculating the correlation coefficient. Before performing the correlation analysis we plotted valence, arousal, and dominance against the audio features to reveal any patterns in the data to decide which correlation method to use.

Before plotting the data we removed any songs from the playlist that we were unable to retrieve the lyrics for or any songs with valence/arousal/dominance score < 0.5 such as "Var Ska Vi Sova I Natt". Songs with scores < 0.5 had non-English lyrics and were unable to correctly calculate the values.

The following graph shows valence found from lyrics against the audio features (from spotifyTopHits.csv). From observing the individual plots, there seem to be no apparent patterns between the audio features and valence, despite our expectation for features such as danceability and energy to display linearity. However, arousal and dominance display a positive linear relationship. The scatterplots for arousal and dominance produced similar results, showing linearity with the lyrical features but none with the audio.

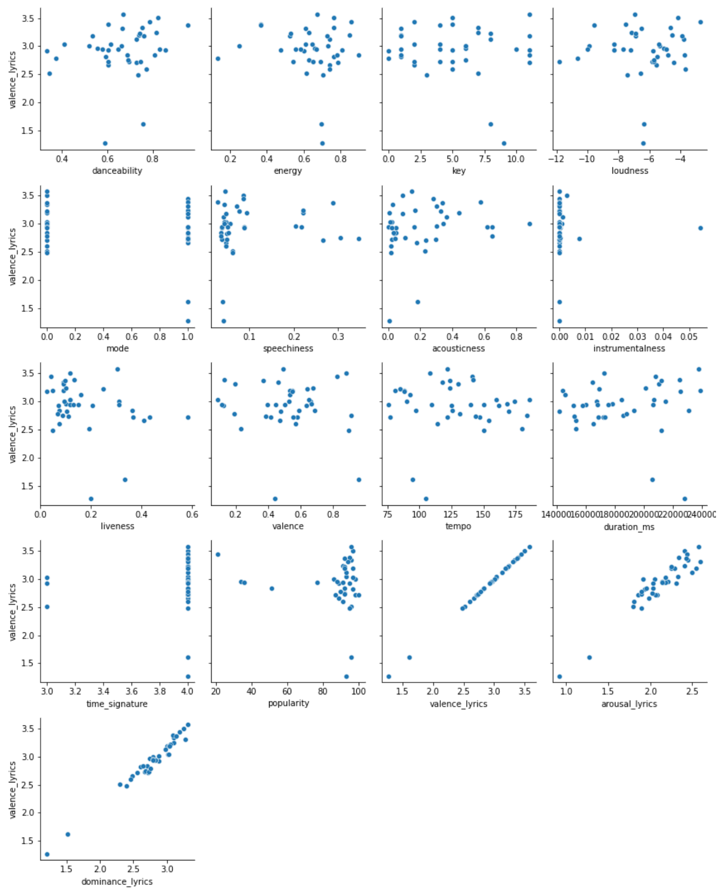


Fig. valence against audio/lyrical features

4.2.1. Spearman's Correlation Coefficient

As the previous scatterplot revealed there to be little to no linearity between the lyrical and audio features, we used Spearman's correlation which does not have a linearity assumption. Spearman's correlation measures the degree of a monotonic relationship between two variables. We found Spearman's correlation coefficient as shown below:

	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	duration_ms	time_signature	popularity	valence_lyrics	arousal_lyrics	dominance_lyrics
valence_lyrics	0.236	-0.118	-0.008	0.037	-0.132	0.141	0.254	0.156	-0.278	-0.029	-0.179	0.237	0.128	0.022	1.000	0.887	0.962
arousal_lyrics	0.177	-0.092	-0.042	0.165	0.009	0.098	0.216	0.148	-0.207	-0.100	-0.220	0.211	0.060	-0.013	0.887	1.000	0.913
dominance_lyrics	0.215	-0.191	-0.021	-0.034	-0.105	0.130	0.259	0.168	-0.262	-0.118	-0.208	0.251	0.060	0.015	0.962	0.913	1.000

As shown in the table above, most of the correlation coefficients values are below 0.03, indicating weak to no correlation between the lyrical and audio features. This was expected from the absence of any patterns in the scatterplots. From these results, we can conclude that lyrical features (valence/arousal/dominance) and audio features do not entail a monotonic relationship or have any patterns we can observe from the scatterplots. Hence, certain word choices such as “happy” and "peace" with high valence scores do not consistently map to a high/low audio value, similarly for arousal and dominance. In comparison to the audio features, the valence, arousal, and dominance scores display a high degree of positive linearity with coefficients all greater than 0.88.

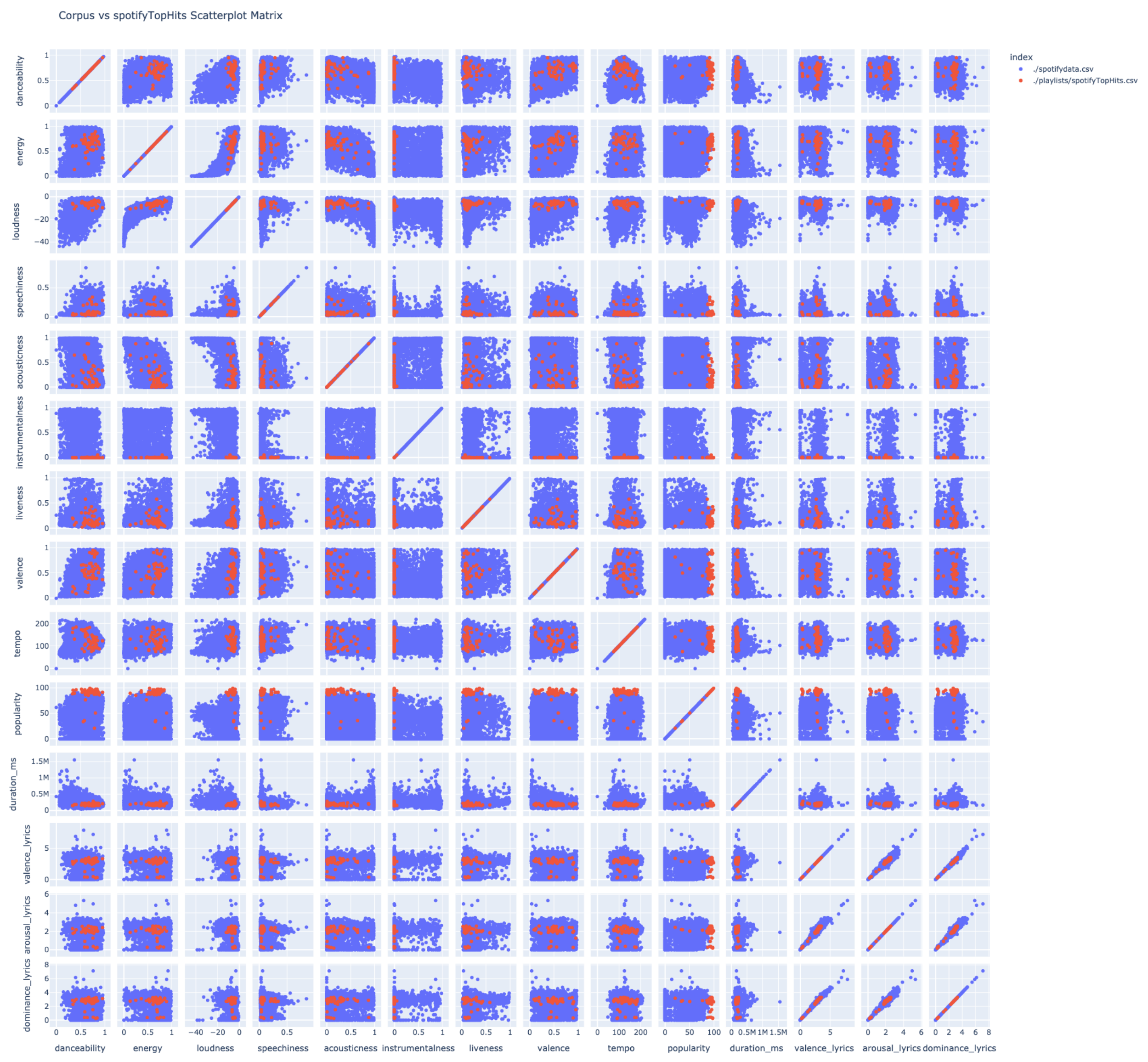
As the correlation coefficient reveals a lack of relationship between the lyrics and existing features, we believe that the valence, arousal, and dominance scores add a new dimension to the data, and possibly improve the performance of the classification models.

4.3. Numerical Feature Analysis

Here we want to analyze the feature space given to us by Spotify.

First we want to visually examine the feature space taken up by a certain playlist and then the feature space taken up by the entire corpus.

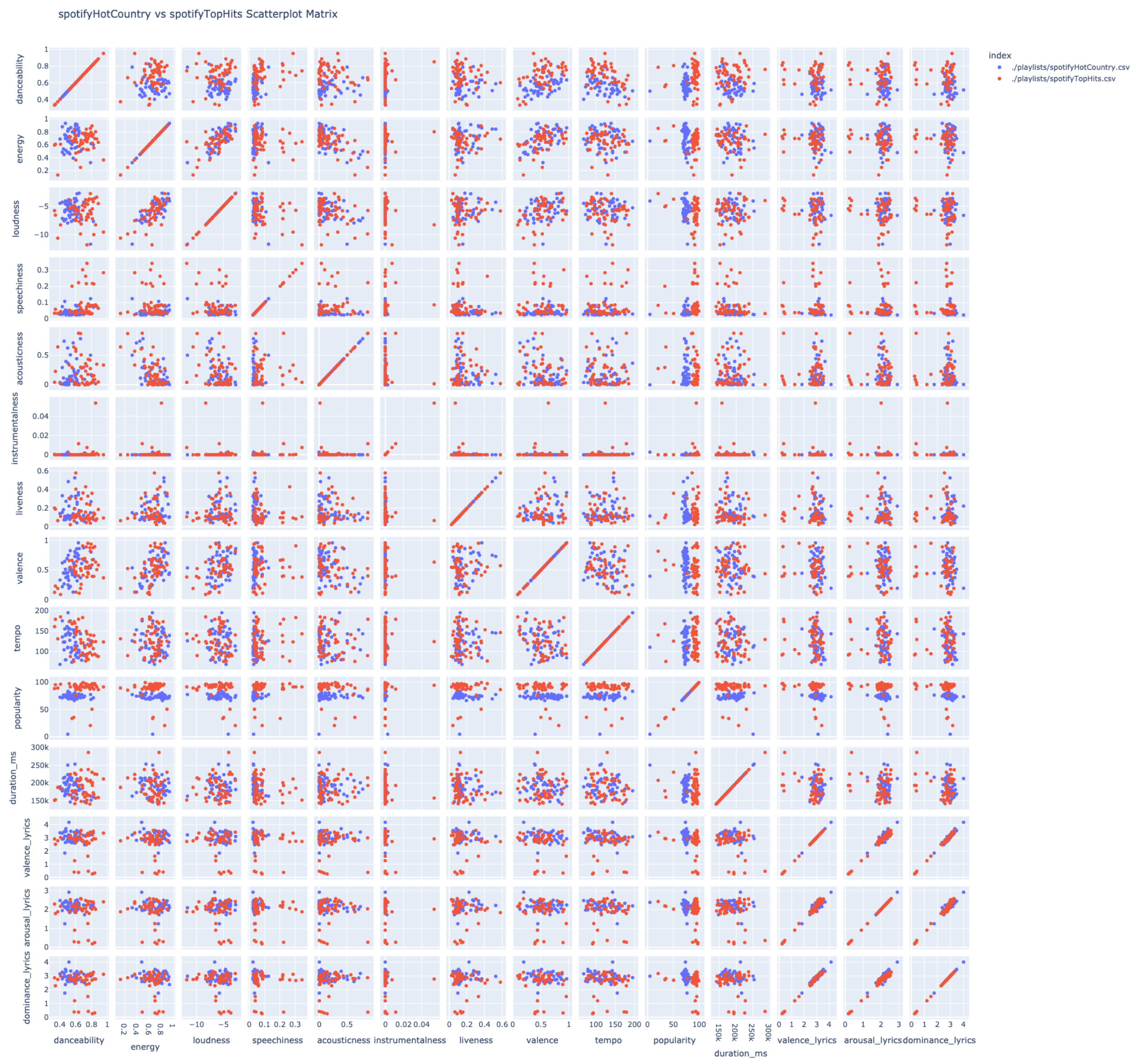
Below we have the pair plot of numerical features between a pop playlist and our corpus:



Upon inspection, this plot gives us some indication that some features such as instrumentalness and loudness occupy a relatively dense section of the corpus which we can hopefully use to narrow in which section of the corpus to draw recommendations from.

Repeating this with all other playlists we were able to find some features that distinguished them from the corpus.

Our next question was whether or not playlists were different from each other. For example, does the feature space of a pop playlist differ from a country playlist?



Our results are less than optimistic. It seems that the only clearly different aspect is popularity. In the optimal situation there would be some musical features such as energy that would clearly distinguish the two playlists. In general, only the most drastically different genres such as rock and classical had some clear variables that were separated.

5. Recommendations

The general outline of our recommendation system was to

1) train a classification model to predict songs given numeric features
2) vectorize our playlist so that it could be represented as a "row" of numeric features
3) ask our model to classify the vectorized playlist as one of the songs

5.1. Classification Models

We tried three classification models provided by the scikit learn library.

- K-Nearest Neighbours: finds the nearest K points and have them vote on which song to recommend
    - Nearest in this context is the Euclidean distance between numerical columns
- Decision Tree: construct a tree that classifies songs by drawing boundaries based on ranges of numerical values
    - The purpose of including this is to have a simple classification system that doesn't strictly rely on Euclidean distance
- Neural Network: use a multi-layer perceptron to fit a playlist to songs.
    - Overcomes disadvantage of K-nearest neighbours and decision trees by having a true probability weighting of recommended songs

Here we use classification models in an unusual way. The standard formulation of a classification problem is to have a series of features in a matrix X and we try to classify rows of X to labels y. However, in the pure sense of our problem we have a corpus of songs X_A and some playlist X_B and we want to find some songs from X_A that would fit with X_B. It was clear we needed to come up with a way to reframe our problem to make it fit into a more standard formulation of a classification problem.

In terms of input features we chose the following features based on our prior analysis: danceability, energy, loudness, speechiness, acousticness, instrumentalness, liveness, valence, tempo, popularity, duration_ms, valence_lyrics, arousal_lyrics, dominance_lyrics.

5.2. Vectorizing the Playlist

Our solution was to turn each playlist we wanted to predict songs into a row of numerical features and attempt to find labels that were songs in our corpus X_A. By doing so, we express our playlist X_B as only a single row.

Here we decided to consider two methods of vectorizing the playlist. We could either take the mean or the median of the numeric features.

From our understanding of the two methods, the mean would be more sensitive to outliers in the playlist and result in songs that were less similar to most of the songs in the playlist, whereas the median would find songs that were more homogenous with the rest of the playlist. Since our question involves finding songs that “fit” into the playlist, we chose to use the mean.

5.3. Results

The final step was to extract song recommendations from the classification algorithms.

Each algorithm was able to output a single song that it thought the vectorized playlist was describing, however, we wanted to push our question further. Having one song was good but it would be better if we could get as many songs as we wanted to.

For the neural network, it was quite straightforward. The final layer of neurons simply outputted probabilities of each song, so extracting *n* recommendations involved taking the top *n* highest probability neurons.

K-Nearest Neighbours was a bit trickier. It did not have a true sense of probabilities of songs like the neural network. Instead, we piggyback off of the K-Nearest in K-Nearest Neighbours. For example, if the user requested *n* songs then we would train the model with the parameter the *n* nearest neighbours to vote on the classification. After passing in the vectorized playlist, we would ask for the *n* nearest neighbours that voted on the classifications and those neighbours would be the songs we recommended.

To coerce the *n* other recommendations out of the decision tree, we had to reduce the minimum samples per leaf to the number of recommendations. This ensured that wherever our playlist vector ended up, it would have *n* songs in the same leaf as itself. Those other songs would be the recommendations given by the algorithm.

6. Evaluation

Now that we got our results, from our recommendation system, the question arose: how do we know if our recommendations are any good? How can we define what a “good fit” of a song to a playlist is? We can’t use a typical train/validation split where we hide labels of known rows and measure how accurate the predictions are. This is because we do not know whether or not a song would belong in the playlist.

The initial method we tried to evaluate this was to take songs away from existing playlists, run our recommendation system, and then see what percentage of the songs we took away were added back. The main issue with this approach was that our recommendation system almost never added songs back. However, this did not mean that our recommendation system was poor as it was entirely possible that the songs we added could have been good recommendations for the playlist. The issue was that there were too many songs in our corpus that could plausibly fit, at least from an initial subjective judgment.

Another limitation with our system is that we can not directly ask the owner of a playlist whether they liked our recommendations with a thumbs up or thumbs down like on the Spotify app. As such we devised three methods to evaluate our results.

1) Use statistical tests to determine if the numeric features of the recommendations were "different" from our input playlist
2) Use the t-SNE dimension reduction to cluster songs together and see if our recommendations were close to the playlist songs
3) Use our group members' own playlists and ask them for their opinion.

6.1. Statistical Tests

One measure of success we decided was to see if the set of recommended songs had a different means for its numeric columns than the original playlist. To check this we decided to use a Mann-Whitney U Test on each numeric column as we did not know if the underlying distribution was normal and we did not have enough recommended songs to appeal to the central limit theorem.

Counts of Numeric Columns with Different Estimated Population Means Between Recommendation and Playlists

Playlist	Decision Tree	KNN	Neural Network
90s Mix	0	0	1
Mila JPOP	1	0	0
Brian EDM	2	0	1
Classical	3	2	2
Christian	4	3	0
Country	4	2	3
Just Hits (Pop)	8	4	3
R&B	5	0	0
Rap	8	0	0
Classic Rock	2	1	0
Sad Pop	2	1	1
Top Hits (Pop)	6	2	1
Average	3.75	1.25	1

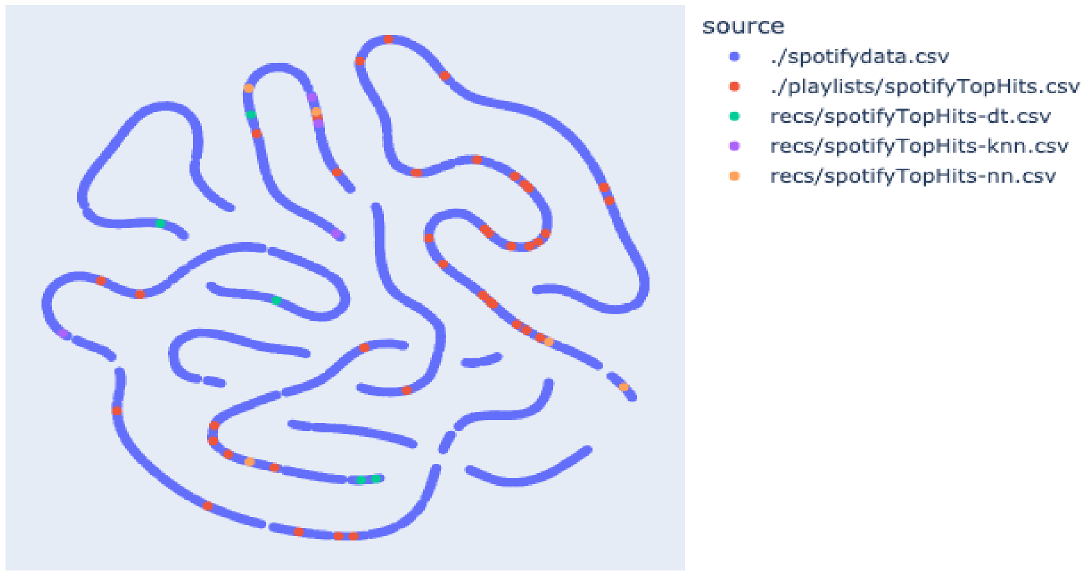
From this test we conclude that the decision tree is the worst basis for the recommendation system while K-Nearest Neighbours and the Neural Network performed better with only an average of 1.25 and 1 columns differing from the original playlist estimated population mean respectively.

6.2. t-SNE

Next we use a visual method to inspect the output. We use t-SNE to plot the entire feature space onto two dimensions. The importance of the algorithm is depicted by the clusters that it creates. If the recommended songs are in different clusters, then it is a sign that the recommended songs are in a different category or group of songs. As such, to evaluate our model, we need to see how many points are on separate clusters than our original playlist.

Below we see a figure of the t-SNE algorithm run on the Top Hits playlist.

t-SNE Plot of Corpus, Top Hits Playlist, and Recommendations



Here we see that by far, the decision tree (green) does the worst with the majority of its points landing on clusters separate than those of the original playlist (red). K-Nearest Neighbours (purple) and the Neural Network (orange) seem to do much better with the majority of their points landing close to original playlist points.

In general for most of the playlists we analyzed, this trend held true with the decision tree performing the worst and K-Nearest Neighbours and the Neural Network being more difficult.

6.3. Group Member Opinions

This method of evaluation is the most subject to bias but is the most important result as we want our recommendation system to work for real people. Please see our appendix to see the evaluator's direct feedback.

Overall, in the limited sample size, the Neural Network did the best all three times. K-Nearest Neighbours was always second, and the Decision Tree was last, recommending only one song that fit a given playlist among all evaluators.

The common consensus among rejections was the genre didn't fit correctly and that the feeling of the songs did not match the feelings of the playlist. The recommendation system did not seem to take into account instrument preferences. Among all types of user created playlists: EDM, Japanese Pop (JPOP), and Christian music, the recommendation systems as a whole had the most difficulty with Christian music. This could be attributed to the fact that Christian music is not musically different from other genres, other than the lyrics have a more religious connotation.

6.3. Conclusion

Overall, for our recommendation system, using a Neural Network produced the best results. This is from looking at the t-SNE plot, Mann-Whitney U tests, and our own personal evaluations as a team.

7. Limitations

7.1. Data Extraction

There were several restrictions we encountered during the data extraction stage. One of the major limitations was due to the Spotify API only providing artists' genres, therefore, we were unable to identify the exact genre for each song and they may even be associated with the incorrect genre. Lyric extraction also had its own limitations which were mainly due to the language of the songs. The swaglyrics API can find lyrics for most English songs, however, when asked for the lyrics of non-English songs it will most likely fail.

7.2. Lyrical Features

To estimate the valence, arousal, and dominance scores the ratings from the ANEW dictionary was used. As this dictionary only consists of English words, non-English songs were mostly discarded from the data. Enabling the lyrical features to be computed for multiple languages could be one of the future works to consider. We also face limitations in the accuracy of the lyrical feature scores as our method/formula only considers individual words when finding the ANEW rating. Maybe using n-grams to find negated words or phrases may result in higher accuracy for score estimations.

7.3. Collaborative Filtering

The current method that most recommendation systems use is called Collaborative Filtering. This involves finding playlists similar to the current one, and then finding songs in the similar playlist that aren't in the current playlist to recommend. Unfortunately due to our inability to query by similarity among users and playlists, we did not reach a sufficient scale to use this method. An interesting comparison would be to see whether Collaborative Filtering outperformed our best system.

7.4. User Based Evaluation Methods

Although we had our own group members evaluate the recommendation systems, it was heavily subject to bias and the sample size of three was too small. Given more time and potentially funding, further surveys could be conducted to provide a better picture of the effectiveness of our recommendation system.

8. Accomplishment Statements

Brian:

- Implemented Machine Learning based recommendation systems through Scikit-Learn with an 60% average satisfaction rating among users.
- Devised method to evaluate Machine Learning recommendation system with Mann-Whitney U Test and t-SNE graphs.
- Implemented Bash scripts to automate the generation and collection of API data, graphs, statistical tests, and Machine Learning outputs.

Emmanuel:

- Implemented a script to compute lyrics TFIDF scores for each word in all lyrics of a playlist based on a formula.
- Customized statistical tests to determine the correlation between the average TFIDF score in a lyrics and the words with the highest popularity in the overall lyrics dataset.
- Integrated the use of a wordcloud to characterize the words in a playlist based on their number of occurrences throughout the entirety of the playlist.

Mila:

- Created scripts to retrieve data for a user-specified Spotify playlist by utilizing Spotify and Swaglyrics API, allowing access to song information and lyrics
- Cleaned lyrics data by removing some punctuations and components that are not part of the lyrics such that analysis can be performed on them with a minimal amount of undesired words/components.
- Performed correlation analysis on the lyrical and audio features by estimating valence, arousal, and dominance scores from the lyrics and using Spearman's correlation coefficient to discover any existence of a monotonical relationship between the two.

9. Appendix

Brian's EDM Playlist

K-Nearest Neighbours

Song Name	Artist	Did it fit?	Why/Why Not
Ledisi	Piece of Me	No	A bit too slow and relaxed for the feeling of the playlist.
Freal Luv	Far East Movement & Marshmello	Yes	Definitely an electronically influenced song that fits the flow of the other songs.
SOS	Johnny Ashby	No	Too acoustic/country-like for an EDM playlist.
You Give it All Away	Dar Williams	No	Very acoustic and folk sounding.
Make It Up to You	Kyden	No	It has the right sounds for EDM but is too slow for this playlist.

Decision Tree

Song Name	Artist	Did it fit?	Why/Why Not
A Girl Like You	Edwyn Collins	No	It feels too dated to fit in the playlist of electronic-heavy music
Havana	Camila Cabello	Yes	Although it is not quite electronic, it fits with the more pop focused songs in the playlist.
Say It Ain't So	Weezer	No	Too slow and feels more rock than EDM.
Last Nite	The Strokes	No	Although it is high energy, it again sounds like a rock song instead of an EDM song.
Wonderwall	Oasis	No	Too acoustic to fit with the EDM songs.

Neural Network

Song Name	Artist	Did it fit?	Why/Why Not
Sign	DEAMN	Yes	Very fitting EDM sound at a good tempo.
Little Tot	Dotter	Yes	Again, instrumentation and vocals fit the rest of the playlist.
Giving In To The Love	AURORA	No	It almost fits in terms of instrumentation but the vocals are a bit too strange.
Arcadia	Hardwell	Yes	Very EDM and vocals are fitting.

SOS	Johnny Ashby	No	Too acoustic/country-like for an EDM playlist.

Mila’s JPOP Playlist

K-Nearest Neighbours

Song Name	Artist	Did it fit?	Why/Why Not
廻廻奇譚	Eve	Yes	JPOP, similar to Vocaloid like songs in playlist.
孤鸟的歌	Di Ma	No	Very slow tempo for the playlist.
凄美地	郭顶	Yes	Sounds similar to some of the songs in the playlist.
十万嬉皮	万能青年旅店	No	Sounds very much like a folk song.
Daisy	Aimer	Yes	JPOP, song by artist in the playlist.

Decision Tree

Song Name	Artist	Did it fit?	Why/Why Not
Lost Stories	Jay Morale	Yes	Sounds similar to some of the songs in the playlist.
Mundian to Bach Ke - Jay Z Remix	Panjabi MC	No	Too much rap and has more of a hip-hop feel, does not fit into the playlist.
Janji Pada Mu	Various Artists	No	Slow tempo and sounds like a 90s song, very outdated.
Está Dañada	Ivan Cornejo	No	Acoustic heavy song and does not match with the songs in the playlist.
Sunset	Various Artists	No	OST piano song not fitting for a JPOP playlist.

Neural Network

Song Name	Artist	Did it fit?	Why/Why Not
One Day	Omoinotake	Yes	JPOP, song by artist in the playlist.
SAKURA	Ikimonogakari	Yes	JPOP, good tempo and vocals.
孤鸟的歌	Di Ma	No	Very slow tempo for the playlist.
廻廻奇譚	Eve	Yes	JPOP, similar to Vocaloid like songs in playlist.
Daisy	Aimer	Yes	JPOP, different song by artist in the playlist.

Emmanuel’s Christian Playlist

K-Nearest Neighbours

Song Name	Artist	Did it fit?	Why/Why Not
You Found Me	The Fray	Yes	It suits appropriately as a song in the playlist.
King and Lionheart	Of Monsters and Men	No	It generally has the taste of a folk kind of music.
Viva La Vida	Coldplay	No	It feels more like a pop kind of music.
You & I	Avant	No	The song is characterized more by romance.
The Only Exception	Paramore	No	It has exceptionally peculiar traits to very folk sort of music.

Decision Tree

Song Name	Artist	Did it fit?	Why/Why Not
Paint it, Black - Mono	Rolling Stones	No	Certainly sounds like a very outdated song and does not match with the playlist.
Y Control	Yeah Yeah Yeahs	No	It has a very rock feel with fast vocals that do not match with the playlist.
A Matter of Time	Ben Böhmer	No	Very classical rock tempo which does not fit with the playlist.
Sweet Child O’ Mine	Guns N’ Roses	No	Has a mostly jazzy feeling which does not fit properly.
Whiskey in The Jar	Metallica	No	Strong presence of folk feeling to the music does not allow it to match the playlist.

Neural Network

Song Name	Artist	Did it fit?	Why/Why Not
I Will Wait	Mumford & Sons	Yes	Although hugely characterized by a country music feeling, it was along the lines of a christian playlist.
King and Lionheart	Of Monsters and Men	No	It generally has the taste like a folk kind of music.
Today was a Fairytale	Taylor Swift	No	The pop and folk music vibe did not allow it to fit in with the christina playlist.
Wait for You	Elliott Yamin	No	There is a more modern pop and romance vibe than

			that which is in the christian playlist.
The Only Exception	Paramore	No	It has exceptionally peculiar traits to very folk sort of music.

10. References

[1] https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.650.7424&rep=rep1&type=pdf