UCLA Data Science Union
Jun 1 · 18 min read · Listen

Exploring K-pop and BTS Song Popularity with Spotify Audio Features & Song Lyrics

Project Leads: Emily Gong, Tiffany Le & Project Members: Audrey Tey, Chelsea Chen, Julia Martin, Justin Kaufman, Sarah Kosic, Sheridan Ohlinger

Photo Credit

Introduction

This project is for all the K-pop fans, ARMY's, and music lovers (with taste). We will be using specific features of K-pop songs, with some focus on BTS songs, to predict each song's popularity through machine learning and natural language processing.

We were first inspired by the BTS 147 Songs Audio Features (Spotify) dataset from Kaggle, but to conduct our analysis, we referred to the following two datasets: the Kpop Artists and Full Spotify Discography and BTS Lyrics dataset (more details in later sections).

Before proceeding, we extracted the "Popularity" scores Spotify generated for each song we observed through the Spotify API and appended it to the datasets mentioned above. This way, we had a fixed metric we could use to determine the accuracy of our results produced by our models. In addition, we extracted the corresponding audio features and popularity scores based on the songs IDs listed in the "BTS Lyrics" data set.

. . .

Machine Learning: Predicting Popularity of K-pop Songs

By Audrey, Chelsea, Julia

> *The machine learning models utilized for this project, including Bagging, Random forests (single tree and pruned tree), Boosting, Ridge Regression and Lasso Regression, were largely focused on reporting the popularity scores predicted from observing the audio features Spotify provided.*

Description of Dataset

The original dataset we used was downloaded from Kaggle and has 16,329 K-pop songs by 273 different solo and group artists released from the 1990s to 2021. We narrowed our selection to the 100 most popular K-pop artists, which left us with 8,528 songs.

Search this file…

1		Artist	Artist_Id	Album_Name	Album_Id	Track_Title

3	1	H.O.T.	5JrfgZAgqAMywJpLpJM0eS	FOREVER 2001 LIVE CONCERT	697cPaD568S5Zt4bgo4cQf	I yah! - Live
4	2	H.O.T.	5JrfgZAgqAMywJpLpJM0eS	FOREVER 2001 LIVE CONCERT	697cPaD568S5Zt4bgo4cQf	Git It Up! / ᄌ
5	3	H.O.T.	5JrfgZAgqAMywJpLpJM0eS	FOREVER 2001 LIVE CONCERT	697cPaD568S5Zt4bgo4cQf	Opening Me
6	4	H.O.T.	5JrfgZAgqAMywJpLpJM0eS	FOREVER 2001 LIVE CONCERT	697cPaD568S5Zt4bgo4cQf	It's Been Ra
7	5	H.O.T.	5JrfgZAgqAMywJpLpJM0eS	FOREVER 2001 LIVE CONCERT	697cPaD568S5Zt4bgo4cQf	N.B.K.(Natu

Kpop_Artists_and_Full_Spotify_Discography.csv hosted with ❤ by GitHub view raw

The first 5 rows of the original dataset

The dataset includes the artist, song, and album name, as well as the Spotify track ID. From the Spotify track IDs, we were able to use the Spotify Web API to pull the Spotify popularity score for each song. Spotify **popularity scores** on a scale from 1 to 100 (1 being the least popular, 100 the most popular) based on an algorithm that takes into account streams, shares, likes, adds to playlists, and how recent all of those actions took place.

In addition, there are 13 variables that describe aspects of the songs: danceability, energy, key, loudness, mode, speechiness, acousticness, instrulmentalness, liveness, valence, tempo, duration, and time signature. Spotify provides more information here about what each of these variables mean and their units. We decided to focus on these 13 variables for our modeling.

The 8,528 songs of the dataset were randomly split into three different datasets: “train” (5970 songs), “validation” (1705 songs), and “test” (853 songs). As the name implies, the “train” dataset is for training the models we wish to use. The “validation” dataset is to check the prediction error of the trained models and fine-tune performance. The “test” dataset is to test the final accuracy of each model.

Description of Models

As our goal was to predict popularity from a list of other variables, we focused mainly on methods for regression analysis. We selected the following **supervised machine learning** models to be trained on our dataset:

- Bagging
- Random forests, including a single tree and a pruned tree
- Boosting
- Ridge regression
- Lasso regression

Bagging, random forests and boosting required the tuning of parameters to improve the model. For bagging and random forests, we created 4 models for each method with 500, 1000, 1500, and 2000 trees, although we disregarded the models with 2000 trees because the small decrease in RMSE compared to the models with fewer trees was not worth the overfitting present in the model with 2000 trees.

Ridge and lasso regression did not require the tuning of parameters as they are linear model regularizations, where a normal linear model is first trained before applying regularization. Regularization is training a model by adding a penalty in the loss function, so that coefficients would not take on extreme values.

While some models are better than others, we decided to evaluate the performance of all of them side-by-side and compile the results to better understand how the models were predicting popularity.

Predictions and Accuracy of Models

We decided to use **root mean squared error (RMSE)** to evaluate the performance of our models, and to compare the models with one another. RMSE is the square root of the average of all squared differences between the actual value of the response variable (popularity score, in this case) and the value predicted by the regression model.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Photo Credit

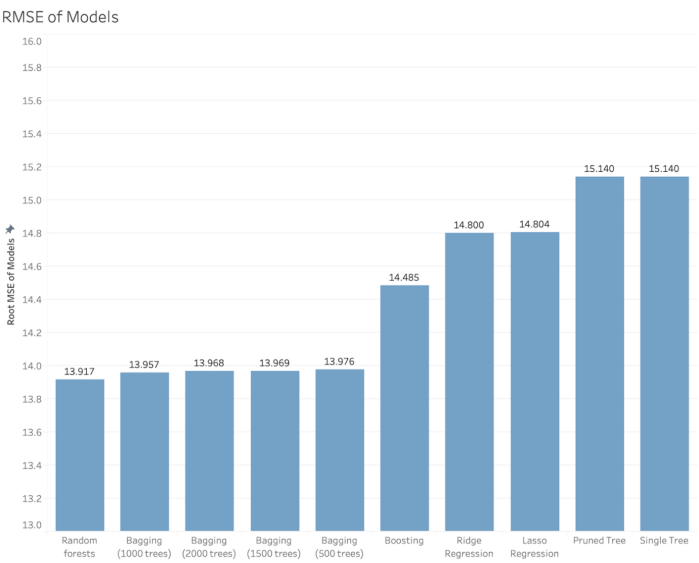
While mean squared error (MSE) is one of the most popular performance metrics used, we decided to use RMSE as it can serve as a good estimator for the standard deviation of the distribution of errors. As such, it could give us insight into how far off our predictions from each model were. Furthermore, RMSE does not penalize smaller errors like MSE does.

Below is a table listing the machine learning methods used and their respective RMSEs as calculated from the validation and testing datasets. A bar graph of the testing RMSEs are also plotted below.

Search this file…

Methods	Validation RMSE	Test RMSE
Bagging	500 trees, 14.01276	500 tree: 13.97592
	1000 trees, 14.01264	1000 tree: 13.95713
	2000 trees, 14.00195	
	5000 trees, 14.00494	
Boosting - manual grid search	14.59892	14.48544
Single & pruned regression tree	Single: 15.21865	Single: 15.14024
	Pruned: 15.21865	Pruned: 15.14024
Random forests	500 trees, m = 3, 13.9288 (lowest RMSE)	500 tree: 13.9169
	1000 trees: m = 4; rmse = 13.9394943633929	
	1500 trees: m = 3; rmse = 13.9393947737853	
	2000 trees: m = 3; m = 3; rmse = 13.9222589054928	
Ridge regression	14.87354	14.80356
Lasso regression	14.87477	14.79958

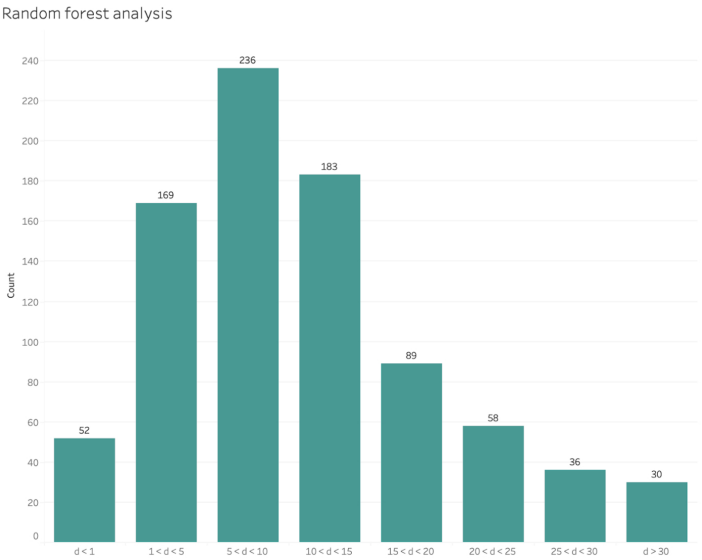
rmse.csv hosted with ❤ by GitHub view raw



We can see that there is ***not*** a big difference in the performance of all models, with the best performing model being random forest with a RMSE of 13.917, and the worst performing models being the single and pruned trees with a RMSE of 15.140. This is expected because a single or pruned tree is a building block of the random forest ML method but is not good enough to serve as a model for prediction or regression on its own.

The random forest model performed the best on the testing dataset. Diving further into that model, we wanted to view the distribution of errors, or the differences between the actual popularity scores and the scores predicted by the model. The bar chart below shows that distribution, with counts of songs

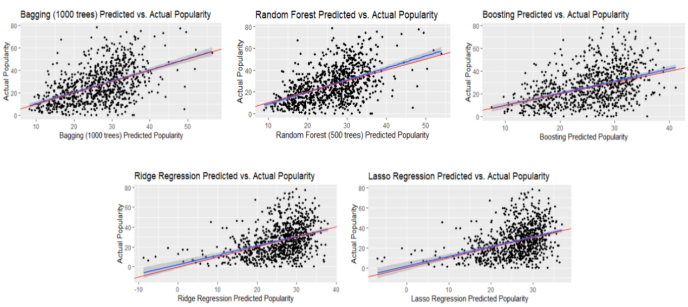
grouped based on their errors (d, difference between actual and predicted popularity scores).



The plot shows a right-skewed Normal distribution, with the most number of songs having a popularity score difference of 5 to 10. 52 out of 853 songs (or roughly 6% of songs) were extremely accurately predicted, and had predicted scores within +/- 1 of their actual scores. There were also predictions that did not fare well, with differences of more than 30.

Comparing Model Predicted Popularity and Actual Popularity Scores

The following graphs depict scatterplots of predicted popularity scores, from the various models we explored, against the actual popularity scores of the songs from the "test" dataset. The blue line plotted on top of the scatterplot is the line of best fit or regression line, with the grey shaded region indicating the 95% confidence interval of the range the regression line would be drawn. The red line is the line of equality, the y = x line that we ideally want all of our points to fall along if our predictions were completely accurate.



From these graphs, we can see that none of our models predicted above a popularity score of 60 even though actual popularity scores reached up to 78.

Despite the regression lines falling closely along the equality line, this does not necessarily indicate good predictions. We can clearly see variation of dispersed predicted vs actual scores across the graphs.

However, as previously indicated with the lowest RMSE score, on the "Random Forest Predicted vs. Actual Popularity" we can see slightly more points gathered around the equality line compared to the other graphs. Bagging with 1000 trees also showed similar trends.

Insights & Case Studies

To explore the "test" dataset further, we compared the predicted popularity scores across all the models run with the actual score and calculated the RMSE.

The table below shows a list of interesting cases that we picked out from the dataset. The last column on the right explains why the case study was picked out. The first three songs had the lowest RMSE, meaning that they were the most accurately predicted songs out of the testing dataset of 853 songs. However, we were not able to find relevant reasons to account for this.

The fourth song had the highest RMSE, meaning that its popularity score was the least accurately predicted, and it is also classified as an unexpectedly popular song. Unexpectedly popular songs are songs that had the highest actual popularity scores, but also fairly high RMSEs, and were not predicted to be popular.

On the flipside, the last song is categorized as an unexpectedly unpopular song as it had the lowest actual popularity score and a fairly high RMSE, and was predicted to be popular.

Actual_Popularity <int>	rmse <dbl>	Artist <chr>	Track_Title <chr>	Case_studies <chr>
32	0.8263566	Super Junior	Rock Your Body	Low RMSE
31	0.8362071	BoA	Cloud	Low RMSE
18	0.8385307	BIGBANG	Wings - Live	Low RMSE
78	50.4400276	TXT	LOSER=LO♡ER	Highest RMSE and Unexpectedly Popular
75	40.7512126	TXT	0X1=LOVESONG (I Know I Love You) feat. Seori	Unexpectedly Popular
77	37.7535003	BTS	DNA	Unexpectedly Popular
0	34.5854670	MAMAMOO	AYA	Unexpectedly Unpopular

Curiously, the two songs with the highest RMSEs were "0X1=LOVESONG" and "LO$ER=LO♡ER," both by the artist TXT (TOMORROW X TOGETHER). These two songs had an actual popularity score of 75 and 78 respectively, and are amongst the highest scores in the "test" dataset; however the predicted popularity scores from our models underscored them in the 30s range.

These two songs were the title tracks of TXT's last two comebacks in 2021, with "0X1=LOVESONG" released in May 2021 and "LO$ER=LO♡ER" released in August 2021; hence they are part of the more recent songs included in our dataset. It is therefore not surprising that they have high Spotify popularity scores since these songs were released more recently, combined with the fact that TXT has a fairly large, dedicated fanbase.

Photo Credit

Besides being released in the same year, some interesting similarities between these two songs are that they are part of the genre of Pop-Rock, and played off the characteristics of Rock to invoke the theme of teen angst. Although TXT has been popular since their debut, these two songs were incredibly popular upon their release because of their focus on teen angst, garnering them a larger fanbase and greater exposure during the time periods of the songs release dates.

Therefore, across their discography that dates back to 2019, both of these songs are among TXT's most streamed songs, despite these songs being released

around a year ago.

Our models can not account for many of these outside factors to what makes a song popular, highlighting the shortcomings of attempting to predict song popularity based solely on spotify song characteristics.

Limitations

Another limitation of our case studies is that all of the case study songs came from the test dataset because that was the data we used to evaluate our model. Therefore, there may be songs from the training dataset that would show more extreme RMSEs and make for more interesting and standout case studies.

In regards to the actual popularity scores, Spotify calculates these scores with an algorithm that takes into account several factors like streams, likes, adds to playlists, and more. However, Spotify favors songs that are currently or recently played a lot. Songs that were played a lot in the past may have lower popularity scores than songs played a lot currently, even if they were both played around the same number of times during their respective eras. This is a limitation of our model because the actual popularity variable may not be very accurate and may be an underestimate of how popular the song really is.

Lastly, the dataset we used for this project originally had K-pop songs from 273 artists and had songs from bands and artists that have not come out with new music in several years. We decided to refine this dataset but handpicked 100 artists that are generally more well-known and have at least 200,000 monthly listeners in order to arrive at a dataset that represents more current K-pop music to better align with the original goal of the project, predicting the popularity of BTS songs. Therefore, the handpicked selection of artists is a limitation of this project in that we started with a selection of songs from artists that are already considered popular, so in the future it would be interesting to make a model for the entire dataset.

. . .

Exploring BTS song lyrics with NLP: Sentiment Analysis, KNN Classification & Type-Token Ratio

By Justin, Sarah, Sheridan

> *We explored Natural Language Processing topics to evaluate how repetitive a song is and conduct sentiment analysis on BTS song lyrics. This way, we could develop insights into why certain BTS songs may be more popular than others based on texts in addition to the song audio features we explored earlier.*

Description of Dataset

The original dataset was downloaded from Kaggle and was collated as a casual project using data pertaining to 19 Korean and English albums from Genius and Big Hit.

Search this file…

	id	album_title	eng_album_title	album_rd	album_seq	track_title	kor_track_title
1	0	2 Cool 4 Skool	2 Cool 4 Skool	2013-06-12	1	Intro: 2 Cool 4 Skool (ft. DJ Friz)	
2	1	2 Cool 4 Skool	2 Cool 4 Skool	2013-06-12	2	We Are Bulletproof Pt.2	
3	2	2 Cool 4 Skool	2 Cool 4 Skool	2013-06-12	3	Skit: Circle Room Talk	
4	3	2 Cool 4 Skool	2 Cool 4 Skool	2013-06-12	4	No More Dream	
5	4	2 Cool 4 Skool	2 Cool 4 Skool	2013-06-12	5	Interlude	
6	5	2 Cool 4 Skool	2 Cool 4 Skool	2013-06-12	6	좋아요 (I Like It)	좋아요
7							

BTS_Lyrics.csv hosted with ❤ by GitHub view raw

The first 5 rows of the orginial dataset

With 230 songs in total, we mainly focused on the lyrics column, which contains the text corpora of each song, all translated into English. To connect the lyrics with popularity, we also extracted the corresponding data from the Spotify API. Another issue we encountered was having different popularity scores for the

same song. For such incidents, we decided to only keep a copy with the highest score and proceed with our analysis.

Sentiment Analysis

By Justin & Sarah

We decided to conduct sentiment analysis on each song to determine if there was a link between a song's sentiment and its popularity. Essentially, sentiment analysis is identifying the "attitude" of text; more specifically, it is the process of categorizing a piece of text as negative, neutral, or positive for our purposes. To carry this out, we used a model called VADER (Valence Aware Dictionary for Sentiment Reasoning), which uses a dictionary from which it calculates the intensity or strength of emotion of text based on lexical features. Utilizing this, we were able to calculate sentiment scores for each BTS song.

```
#VADER sentiment score
sia = SentimentIntensityAnalyzer()
sentiment_scores.append(sia.polarity_scores(df.lyrics[i]))
```

From this, we were able to create a dataframe that contained each BTS song, its corresponding positive, neutral, and negative sentiment score, as well as its popularity. This would be utilized to feed into a model that would attempt to predict a BTS song's popularity based off these scores.

Using word tokenization and ridding of stop words for each BTS song's lyrics, we were able to gather the top 10 most frequent words that appeared in each song as well as their corresponding frequencies. From here, along with the sentiment scores of each song, we produced word clouds of the most negative and positively scored songs.

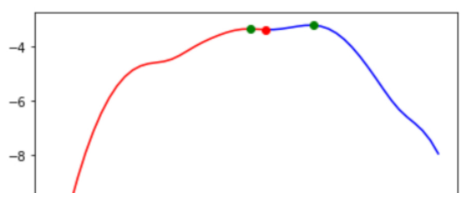
Word Cloud of Most Negative Words (left) & Most Positive Words (right)

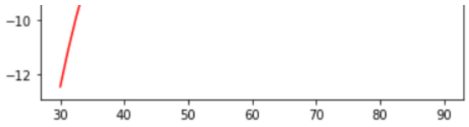
It can be seen that the most negatively scored songs have very negative words like rage, malice, boom, and the most positively scored songs have very positive words like love, baby, dream.

K-Nearest-Neighbors Classification

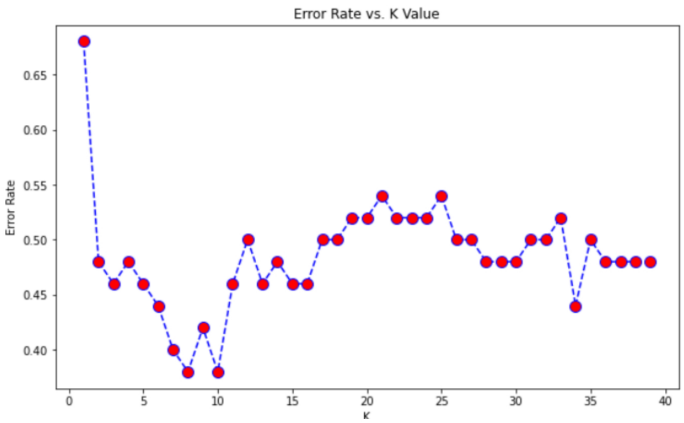
With our sentiment analysis for each song, we wanted to explore if these scores had any predictive power with respect to song popularity. Specifically, we wondered if the sentiment analysis could classify a song correctly as either high or low popularity. To do this, we first partitioned our numeric popularity data into high and low categories using Kernel Density Estimation. This technique allows us to simulate a smooth curve in the shape of our data that we can use to form our high and low classes based on minimum and maximum values.

```
from sklearn.neighbors import KernelDensity
kde = KernelDensity(kernel='gaussian', bandwidth=3).fit(np.array(df_final.popularity[:len(df_final)-1]).reshape(-1,1))
```

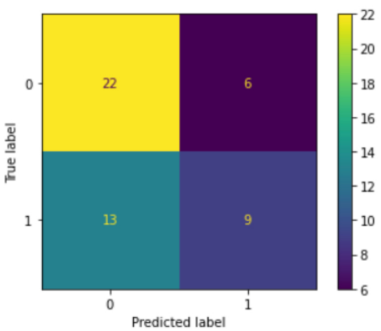




Once we had high and low popularity classes, we employed a K-Nearest-Neighbors classifier on the negative, neutral, and positive sentiment analysis scores to predict popularity class. Using the elbow method, we plotted error vs k to find our optimal K value of 10.



We got a resulting accuracy of **62%** on our test data, which is slightly better than random guessing. We suspect that the small number of variables and small distribution of scores may have played a role in limiting the accuracy of our model. Further, there may just not be a strong correlation between popularity and sentiment that we can leverage for making predictions.



Confusion Matrix with y_test and y_predict, where 0 means high popularity and 1 means low popularity

Overall, although we did not get great predictive results, our sentiment analysis seemed to work well with our intuition of positive and negative lyrics. While a great introduction to the world of NLP, there are many areas in which we could improve in the future. VADER was built and optimized for tweets and social media, so would likely not have the best results when extrapolated to song lyrics. By figuring out ways to better modify and apply current sentiment analysis tools to our task, hopefully our sentiment analysis scores would become more accurate and be able to make more accurate predictions.

Delving into Repetitiveness of Kpop Songs with Type-Token Ratio & Searching its Correlation with Popularity

By Sheridan

Data Processing:

Our variables of interest were Spotify's generated popularity score of each song and each song's type-token ratio. The type-token ratio (TTR) is the number of types — the amount of unique utterances — divided by the token — the total number of utterances in a song — for each song. Specifically, tokens are

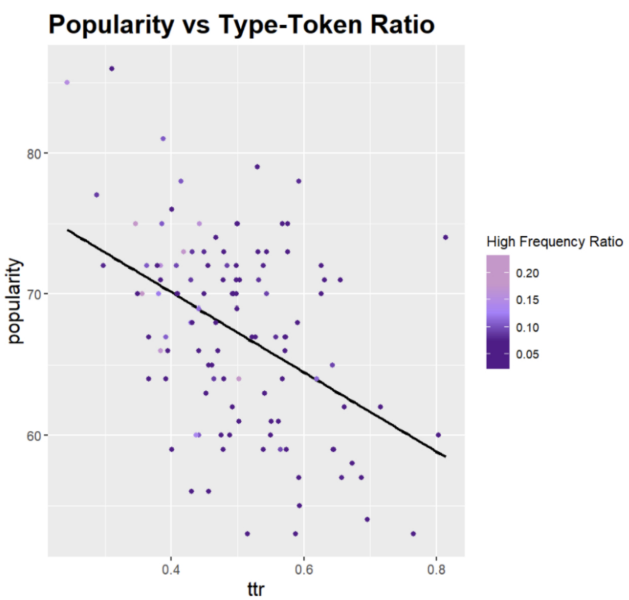
number of utterances in a song — for each song. Specifically, tokens are identified as any individual string of text, separated by a space on either side. These utterances, or tokens, can include actual words and your typical “la la la” and “na na na” sounds often heard in BTS’s songs. Therefore, the type-token ratio is a metric that gives some insight as to how repetitive a song is. If a song has a high number of tokens (individual linguistic utterances) and a low number of types (unique linguistic utterances), it has a low type-token ratio, indicating how the same utterances are repeated multiple times. As listeners, songs with high repetition are often seen as “catchy,” so we wanted to see if a linguistic measure of “catchyness”, or repetitiveness, would correlate with a song’s popularity.

The data for type and token values was gathered using a linguistic software known as AntConc. Original Korean song lyrics gathered from azlyrics.com were individually converted to tokenized text files, through individual copy and pasting into google docs, and downloading as plain text files with the correct unicode format (UTF-8). A tokenized text file refers to one in which individual utterances are separated by a space on each side, with an utterance being a word, or a sound you heard in songs like “la” or “na.” Then, the files were uploaded into AntConc to provide me with each song’s individual type and token count. Using AntConc’s Concordance Tool, we also extracted other metrics including how often the song’s title was repeated and how many times the most frequent token was repeated to understand how different forms of repetition can affect popularity.

After inputting these values into an Excel file, we uploaded everything into R in order to produce our scatter plots. Songs that are considered intros, outros, or were highly instrumental with total token values of less than 100 were removed, as they are not representative of a typical BTS song and thus would affect our findings. Additionally, the ‘skits’ that are often included in BTS’s albums were excluded since they lacked the melodic component songs have which defeats the purpose of looking at “catchyness” or repetition, as they are more of a dialogue than a song.

Findings

We used a simple scatterplot for this approach in order to get the most interpretable results in understanding how lyrics contribute to a song’s popularity. There was a clear negative trend with a small P-value attached to slope, but there was quite a bit of variation in the plot resulting in an R-squared of only around .22.



The visualization shows popularity score against type-token ratio, with a clear negative correlation between popularity and type-token ratio. This implies that a

negative correlation between popularity and type-token ratio. This implies that a song with low repetition, i.e., one with a high type-token ratio, is less popular than one with high repetition, or a low type-token ratio. Additionally, individual points were color coded based on their high frequency ratio, a rate that divides each song's most frequent token by its total number of tokens. For example, BTS's Dynamite repeats the string of tokens "na na na" around 90 times, making it the most frequent token in the song which has 500 token total. This gives it a relatively high "high frequency ratio" compared to other songs, around .18. The high this ratio, the lighter the purple color is on the plot, and we see an overall lighter color for the top left section of the graph, where highly repetitive and popular songs are found. Thus, a bit of extra insight can tell us that having one particular lyric repeated throughout a song (or a high frequency ratio), not just an overall high level of repetition (or a high type-token ratio), can play a role in determining popularity as well.

Some explanations for the high variance include:

- Though the software we used was able to understand Korean lyrics, BTS fans aren't always able to! Therefore, it may be hard to correlate popularity with songs sung in Korean when fans may not be too sure of what is being said when they are listening.
- A song's catchiness would be ideally influenced by both its lyrics and instrumentals, so the type-token ratio metric calculated may only capture half of the catchy aspect we were searching for. If all of these songs were purely instrumental (no lyrics at all), we assume that their popularity scores would drastically differ, so it would be difficult to model popularity solely based on linguistic variables.

Advantages & Limitations of the Approach

This model is much more simple than others used in the project since it is meant to be highly interpretable. However, linguistic variables can become quite complex and uninterpretable when transformations are applied to them due to the nature of human-generated language (in comparison to computer-generated language). Thus, this model is meant to focus on analyzing and interpreting this specific set of linguistic data and isn't meant to be highly accurate or applicable to all sets of song lyrics.

While we believe that the models were a good fit, we would have a more holistic idea of how catchyness contributes to song popularity if we could observe how the combination of both 'linguistic' and 'instrumental' variables (bpm or danceability score) influences popularity scores.

The errors encountered while conducting this analysis mostly stemmed from having included different languages in each text file. While the AntConc software can work well in multiple languages, it cannot process multiple at the same time. The software can easily determine sentences and phrase-wide linguistic features if it was only looking at Korean text, but because BTS lyrics include both English and Korean words, the software is unable to compute complex linguistic analysis like identification of relative or complement clauses. Consequently, this limitation can make tokenizing and concordance slightly different from its true counterpart — as identifying strings of tokens when multiple languages are present in a text file, and searching for specific strings of tokens when multiple languages are present in a text file both prove difficult. Finally, the website the lyrics were gathered from weren't always completely tokenized. Thus, we had to tokenize the lyrics manually even though we were not fluent in Korean. Therefore, the tokenization of Korean is likely to be slightly incorrect even though we did our best.

. . .

Conclusion

We performed analysis on both audio features and lyric texts in this project, which are essentially the two main components of every song. While we were

hoping to determine what specific factors strongly contribute to a K-pop song's popularity, we were unable to come to a perfect conclusion by the end of the project due to certain limitations.

Most prominently, we had a working understanding of the meanings and calculations behind Spotify's popularity and audio feature scores to base our analysis on. Therefore, our limited knowledge of these metrics affected our interpretation of our results since our exploration was heavily reliant on these metrics.

Likewise, our lyrics dataset had certain issues we tried to circumvent when conducting our sentiment analysis. First, many Korean songs were translated into English. While it may have helped ease our sentiment analysis, the modification leads to a huge loss in the nuances of the songs, such as the rhymes, the catchiness, and the cultural contexts of certain lyrics. Also, our lyrics dataset represented BTS's entire discography, including several repackaged albums that had repeated songs. Spotify had assigned different popularity scores to these recurring songs, so we had chosen to conduct analysis on the version that had the highest score. By omitting these differences, we may have overlooked additional insights on how Spotify determined a songs' popularity. In addition, the lyrics data set also had discrepancies in the titles of the songs it presented which made it difficult to merge with other datasets.

In conclusion, while we can tentatively suggest that lyrics and a collection of audio features do influence a K-pop song's popularity score, our results may not be as useful in determining the popularity scores of other songs outside our dataset.

If you would like to explore more about what we did in the project, please feel free to visit our Github page. Thank you very much for reading!