Hit Song Science: Modeling Weekly Billboard 100 Chart with Twitter Data

SI 699 WN 2021 | Midterm Report
Team 6 | John Lee, Hsin-Yuan Wu, Melody Chang

1. INTRODUCTION AND MOTIVATION

1.1 Motivation

Every year end, many songs enter the market, some become timeless tracks that we will still listen to 10 to 20 years from now, some fall under momentary spotlights but faded as the audience's taste evolves, others just never made it to the spotlights. While such life cycles of songs are inevitable, they are yet identifiable. This means that characteristics that help determine the song trends are available somewhere. In fact, recent literature pointed to a handful of such sources - Spotify Streaming API, Twitter Decahose Data, and the Million Songs Dataset from Echo Nest, to name a few. These sources acquire a glut of media data, especially during the COVID-19 pandemic, when media consumption rallied astronomically [1]. These data help the entertainment industry understand the primary motives behind the streaming activity and record sales of the public, and most importantly steel up for target marketing and advertising. In the first week of January, 2020, Mariah Carey's holiday hit "All I Want for Christmas is You" fell from Billboard's number one spot to completely off the top 100 chart, but rebounded back to the top spot again over the next few weeks [2]. Never had anyone seen such a sharp drift in song ranking on the Billboard chart before. Yet, if we are able to identify the trend features before a song makes a bold move on chart ranking, this may help primary stakeholders such as the record company and marketers strategize their operations under different popularity trends, especially bumpy ones.

1.2 Introduction

From sensations like "Taylor Swift's new album made history with more than 30 billion copies sold", to empirical rankings like "Maroon 5's Sugar album lying amongst the top 100 hits for the 49th consecutive week", our team aims to seek how textual components and popularity patterns of these sorts may help predict rankings of the Hot 100 songs on Billboard. The previous work by Tsiara and Tjortjis [3] attempted to answer this using VADER's sentiment analysis for consolidating text parameters and regression analysis (e.g., Support Vector Regression) for classifying ranks of the top 10 Billboard songs. We extend their study to include more comprehensively-ranked data of the top 100 Billboard songs. Our study starts with querying 3-month worth of data from the Billboard chart and Twitter Decahose. Tweet records from the Decahose database are combined with chart data to include attributes like song titles, artist names, rankings, and tweet information related to the top 100 songs each week from Billboard. At the middle stage of our study, we evaluated two naive classifiers, Dummy Classifier and Support Vector Classifier (SVC), in particular, as our baseline models. In addition, a Multilayer Perceptron (MLP) classifier is implemented to draw comparison with the baselines. This is just a

preliminary stage of our study. At the later stage of our study, we hope to incorporate more literature reviews, comprehensive data, and validation frameworks to fine-tune the proposed models.

2. DATASETS

To build a prediction model for the music chart, we utilized the following two sets of data.

2.1 Billboard Dataset

We obtained a subset of weekly Billboard Hot 100 singles charts from data.world provided by Sean Miller [4]. The original dataset covers every weekly chart from August 23, 1958 to December 26, 2020. For the scope of this project, we selected data from September 2020 to November 2020, which accounted for 13 weeks of data - 1,300 records in total. Each row of data represents a song and the corresponding position on that week's chart. Available columns include:

- url: The Billboard Chart URL.
- weekid: The releasing date of the weekly chart. We renamed the column as “date”.
- week_position: The position of the song on the chart current week. We renamed the column as “rank”.
- song: The name of the song on chart.
- performer: The performer’s name of the song on chart. We renamed the column as “artist”.
- songid: The concatenation of song and performer.
- instance: The overall number of times a song has appeared on the chart. This is used to separate breaks on the chart for a given song.
- previous_week_position: The position of the song on the chart previous week. We renamed the column as “rank_last_week”.
- peak_position: The highest position that song has been on the chart as of the corresponding week. We renamed the column as “peak_rank”.
- weeks_on_chart: The corresponding week the song has been on chart.

2.2 Twitter Dataset

We accessed the Twitter Decahose Stream via the Michigan Institute For Data Science (MIDAS) at the University of Michigan [5]. It contains a compilation of tweets known as the “Decahose” - a 10% sample of all tweets - from 2012 to current. Based on the time interval selected from the Billboard Dataset above, we only obtained data from August 29, 2020 to November 27, 2020 for this dataset.

Each row in the dataset represents a tweet. The dataset contains 38 main columns, where the majority of them are a nested array or a nested struct as shown in Figure 1 below. Therefore, we ended up having more than 150 variables. For the purpose of this project, we pre-selected the

following 28 columns to include in our analysis: “created_at”, “id”, “truncated”, “lang”, “user.id”, “user.followers_count”, “user.friends_count”, “user.listed_count”, “text”, “entities.hashtags.text”, “entities.urls.display_url”, “entities.urls.expanded_url”, “entities.urls.url”, “entities.user_mentions.screen_name”, “extended_tweet.full_text”, “extended_tweet.entities.hashtags.text”, “extended_tweet.entities.urls.display_url”, “extended_tweet.entities.urls.expanded_url”, “extended_tweet.entities.urls.url”, “extended_tweet.entities.user_mentions.screen_name", “place.country”, “place.country_code”, “place.name”, “place.place_type”, “favorite_count”, “reply_count”, “retweet_count”, and “quote_count”.

```
root
 |-- contributors: string (nullable = true)
 |-- coordinates: struct (nullable = true)
 |    |-- coordinates: array (nullable = true)
 |    |    |-- element: double (containsNull = true)
 |    |-- type: string (nullable = true)
 |-- created_at: string (nullable = true)
 |-- display_text_range: array (nullable = true)
 |    |-- element: long (containsNull = true)
 |-- entities: struct (nullable = true)
 |    |-- hashtags: array (nullable = true)
 |    |    |-- element: struct (containsNull = true)
 |    |    |    |-- indices: array (nullable = true)
 |    |    |    |    |-- element: long (containsNull = true)
 |    |    |    |-- text: string (nullable = true)
 |    |-- media: array (nullable = true)
 |    |    |-- element: struct (containsNull = true)
 |    |    |    |-- additional_media_info: struct (nullable = true)
 |    |    |    |    |-- description: string (nullable = true)
 |    |    |    |    |-- embeddable: boolean (nullable = true)
 |    |    |    |    |-- monetizable: boolean (nullable = true)
 |    |    |    |    |-- title: string (nullable = true)
 |    |    |    |-- description: string (nullable = true)
 |    |    |    |-- display_url: string (nullable = true)
 |    |    |    |-- expanded_url: string (nullable = true)
 |    |    |    |-- id: long (nullable = true)
 |    |    |    |-- id_str: string (nullable = true)
 |    |    |    |-- indices: array (nullable = true)
 |    |    |    |    |-- element: long (containsNull = true)
 |    |    |    |-- media_url: string (nullable = true)
 |    |    |    |-- media_url_https: string (nullable = true)
 |    |    |    |-- sizes: struct (nullable = true)
 |    |    |    |    |-- large: struct (nullable = true)
 |    |    |    |    |    |-- h: long (nullable = true)
 |    |    |    |    |    |-- resize: string (nullable = true)
 |    |    |    |    |    |-- w: long (nullable = true)
 |    |    |    |    |-- medium: struct (nullable = true)
```

Figure 1. Partial schema of the Twitter Dataset.

3. METHODS

3.1 Previous Work

In Tsiara and Tjortjis’s research [3], they aimed to use Twitter data to predict the top 10 chart positions for songs on the Billboard Hot 100 charts. They used Twitter Search API to collect more than one million tweets during October and November in 2018.

To obtain features from the Twitter dataset, they first conducted sentiment analysis with VADER [6] and SentiWordNet [7], adding sentimental scores of positive, negative, and neutral scores as features. Next, they mined the tweets to identify features related to songs and artists. The final attributes extracted were as follows:

- Song play counts: Number of tweets, including retweets, related to the song with #nowplaying keywords at the same time.
- Artist play counts: Number of tweets, including retweets, related to artists of the song with #nowplaying keywords, which is irrelevant to the song but the artists.

- Artist tweets: Number of tweets, including retweets, related to artists of the song.
- Artist sentiment score by VADER : Average value of the score by VADER.
- Artist sentiment score by SentiWordNet : Average value of the score by SentiWordNet.

In addition, they used the information from Billboard, including the song position one week before, the number of weeks the song on the chart, the number of weeks the song was in top1. To conduct the classification problem, they performed several regression analysis with different algorithms: Support Vector Regression, Random Forest, and Bagging classifier. The results showed that Logistic Model Tree (LMT) and Simple Logistic classifiers could achieve high accuracy and F1 values.

3.2 Proposed Approach

We shadowed the procedures suggested by Tsiara and Tjortjis to extract variable features, implement prediction models, and fine-tune model performances. Firstly, twitter features with a 7-day lookback period are retrieved from the Decahose database. These are tweet variables related to each song 7 days before it is ranked on the Billboard chart. Secondly, we assess a variant of classification designs for predicting song rankings in the upcoming week. This process includes employing tree-based models and other classifiers to quantify the error results. The specific models and methods are described in the subsections below. Lastly, techniques such as regularization and cross-validation are used to optimize the predictive power of the proposed models while balancing their balance-variance trade-offs.

Feature Engineering

For the feature extraction from the Twitter dataset, we would first obtain basic statistics like the counts of tweets related to the songs and to the artists of each song. The lists of songs and artists were obtained from the Billboard dataset. Also, we would take a look at columns that might be indicators of the popularity of tweets, such as counts of likes and retweets. Furthermore, we would like to explore features that required in-depth data mining. For example, whether urls or emojis in tweets could affect viewers' perception regarding the songs. As of the tweet contents, we would like to retrieve features such as the average length of the tweets for each song and conduct TF-IDF analysis on the text.

Machine Learning Techniques

After adding features extracted from Twitter dataset to the Billboard dataset, we would rely on several machine learning techniques to build the prediction model. In the first step, we would utilize tree-based models, like Random Forest and Extremely Randomized Trees, to help us narrow down and select relevant features. We would then apply these extracted features to the models. The combined dataset would be split into a training set and a test set with a 80:20 ratio. Compared to the baseline models, we proposed using models such as Logistic Regression,

Gradient Boosting classifier, Multi-layer Perceptron (MLP) classifier, and other potential neural network models to approach this multi-class classification problem.

Model Optimization
In order to optimize the performance on the above mentioned models, we would explore regularization and/or normalization, detection and removal of outliers, imputation of missing values, feature selection, and cross-validation to avoid overfitting or underfitting the models. We would also perform hyperparameters tuning via Grid Search to determine the ideal model architecture. For example, for neural network models, we would tune parameters like learning rate, hidden layer size, and regularization.

Evaluation Approach
For model selection, we would use the Mean Absolute Error (MAE), which is the average of all absolute errors, to understand the amount of error in our measurements. Models with a lower MAE value on the test set may generate a better prediction. Then, we would evaluate the prediction accuracy of models with confusion matrix (macro averaged precision, recall, and F1 score) and AUC-ROC curve; the top performing model would be used for further prediction.

4. SUMMARY OF ANALYSIS PERFORMED THUS FAR
We spent most of our time setting up the environment on Cavium to access the Twitter Decahose Data and exploring the information we have.

4.1 Data Preprocessing And Exploratory Data Analysis
4.1.1 Billboard Dataset
To build the prediction model, we included the following columns for further analysis: “date”, “rank” “artist”, “song”, “rank_last_week”, “peak_rank”, and “weeks_on_chart”. To be able to feed these features into the model, we preprocessed the data by transforming “date” from string to timestamp and factorized “song” and “artist” so that they could be fed into the model directly.

The original Billboard dataset ranks the chart from 1 to 100. In order to simplify the problem to fit the scope of a semester-long course project, we transformed the problem into a multi-class classification. We defined the chart ranking into five tiers as follows:

- Tier 1: rank 1 to rank 20
- Tier 2: rank 21 to rank 40
- Tier 3: rank 41 to rank 60
- Tier 4: rank 61 to rank 80
- Tier 5: rank 81 to rank 100

Furthermore, we obtained a list of unique song names and artist names in the three month interval to support preprocessing of the twitter data. The results contained 241 unique songs, and these songs were released by 191 unique artists.

4.1.2 Twitter Dataset

Due to the scale of the dataset, we selected the data in October for the exploratory analysis to help us better understand the available columns and what features we can potentially extract. There are approximately 1.3 billion records available in October. With a careful observation, we found the data contains a lot of noises.

In order to reduce the noise of the dataset and focus our analysis on tweets relevant to the research questions, we filtered the dataset to only include records where the tweet content contains the song name or artist name from the list extracted from the Billboard dataset mentioned above. As expected, not every tweet mentioned both song name and artist name simultaneously; as shown in Figure 2, many of the filtered records only mentioned one of them. By doing so, we ended up with 22,350,893 of records for the exploratory data analysis.

weeks	song_name	count
0.0	None	5771126
1.0	None	8966001
2.0	None	982936
3.0	None	5226490

weeks	artist_name	count
0.0	None	158947
1.0	None	180147
2.0	None	34389
3.0	None	165067

Figure 2. The count of tweets that mentioned artist name but not song name (left) and the count of tweets that mentioned song name but not artist name (right). Count by week.

Among these records (tweets), there were a total of 219 unique songs. The most mentioned song was *Dynamite*, covered by a total of 645,162 tweets (Figure 3). In terms of artists, 107 artists extracted from the Billboard dataset were mentioned in this filtered dataset. The most mentioned artist is *BTS*, which is the performer of *Dynamite*, appeared in 9,621,433 tweets in October (Figure 4).

	song_name	count
1	Dynamite	645162
2	ily	89810
3	24	58196
4	21	48210
5	Savage	39389
6	Golden	25534
7	Positions	24436
8	Better	23908
9	Lovesick Girls	20334
10	All In	17487

Figure 3. The top 10 songs in the list of songs extracted from the Billboard dataset mentioned in the Twitter dataset in October.

	artist_name	count
107	BTS	9621433
106	CJ	2600736
105	BLACKPINK	1395014
104	None	352116
103	Zayn	199011
102	VEDO	195579
101	Harry Styles	155263
100	AJR	137398
99	Drake	121734
98	Taylor Swift	99249

Figure 4. The top 10 artists in the list of artists extracted from the Billboard dataset mentioned in the Twitter dataset in October.

We also took a look at the distribution of the language used for the tweets in the filtered dataset. Overall, there are 66 distinct languages in the dataset and English is the most used language. However, we found that there are a lot of tweets where the language used was labeled as “undefined”; they accounted for the third most tweeted language in October (Figure 5.). To visualize the distribution, we selected languages with more than 10,000 tweets, which are also the top 25 languages in the filtered dataset and plot as below.

	lang_long	count
0	English	11142732
1	Korean	3375060
2	undefined	1696127
3	Spanish	1380127
4	Portuguese	1119565
5	Japanese	938579
6	Thai	790811
7	Indonesian	444500
8	Polish	287494
9	Arabic	184727
10	French	157967
11	Tagalog	145265
12	Turkish	117104
13	Italian	77170
14	German	60933
15	Vietnamese	46736
16	Estonian	45440
17	Russian	40094
18	Hindi	36833
19	Haitian Creole	32792

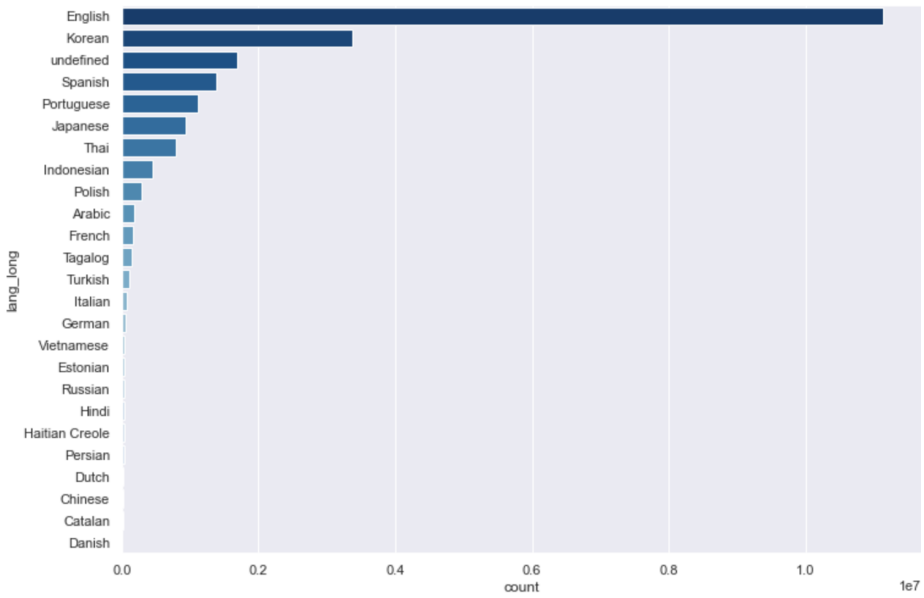

Figure 5. Distribution of top 25 languages used in the October filtered dataset (right). Detailed tweet counts of the top 20 languages are shown in the table on the left.

Furthermore, since we will merge the features from the Twitter dataset to the Billboard dataset on a weekly basis, we also ranked the songs and artists based on tweet counts by weeks. We listed the top 10 most popular songs across 4 weeks in October as shown in Figure 6. We found that most of the songs were on the ranking for more than a week. The song *Dynamite* dominated the ranking throughout October, and the songs *ily* and *24* remained second and third place respectively on the ranking for the last three weeks of October. We also looked at the weekly tweet counts by artists (Figure 7). In October, *BTS*, *CJ* and *Blackpink* were among the top three most frequently mentioned artists, and *BTS* ranked first for the entire four weeks.

weeks	song_name	count	weeks	song_name	count	weeks	song_name	count	weeks	song_name	count
0.0	Dynamite	196202	1.0	Dynamite	187237	2.0	Dynamite	177255	3.0	Dynamite	84468
0.0	Savage	14257	1.0	ily	31286	2.0	ily	17602	3.0	ily	26807
0.0	ily	14115	1.0	24	24660	2.0	24	8409	3.0	24	19556
0.0	All In	13052	1.0	21	20380	2.0	Lonely	5613	3.0	Positions	18505
0.0	Lovesick Girls	5857	1.0	Savage	18978	2.0	Lovesick Girls	5527	3.0	Golden	17645
0.0	24	5571	1.0	Better	8920	2.0	21	5439	3.0	21	17433
0.0	21	4958	1.0	Lovesick Girls	7743	2.0	Savage	4036	3.0	Better	10741
0.0	Ice Cream	4148	1.0	Shut Up	7088	2.0	Dreams	3729	3.0	Cardigan	10354
0.0	Franchise	3264	1.0	Wonder	7029	2.0	Better	2550	3.0	Done	6683
0.0	Like That	2898	1.0	Done	6189	2.0	Done	2397	3.0	Wonder	5449

Figure 6. The top 10 mentioned songs in October by weeks.

weeks	artist_name	count	weeks	artist_name	count	weeks	artist_name	count	weeks	artist_name	count
0.0	BTS	245120	1.0	BTS	5613243	2.0	BTS	736910	3.0	BTS	3026160
0.0	BLACKPINK	23243	1.0	CJ	1231647	2.0	CJ	230158	3.0	CJ	1137065
0.0	CJ	1866	1.0	BLACKPINK	794167	2.0	BLACKPINK	115072	3.0	BLACKPINK	462532
0.0	Zayn	1456	1.0	VEDO	92458	2.0	Zayn	22862	3.0	Zayn	103460
0.0	Harry Styles	909	1.0	Zayn	71233	2.0	Drake	18488	3.0	Harry Styles	88370
0.0	DaBaby	468	1.0	Harry Styles	59230	2.0	VEDO	16970	3.0	VEDO	86142
0.0	Drake	359	1.0	AJR	54970	2.0	AJR	16606	3.0	Taylor Swift	77046
0.0	Why Don't We	357	1.0	Drake	35480	2.0	Harry Styles	6754	3.0	Drake	67407
0.0	H.E.R.	353	1.0	Ariana Grande	23707	2.0	Clairo	5959	3.0	AJR	65743
0.0	Travis Scott	278	1.0	Taylor Swift	18627	2.0	Ariana Grande	4162	3.0	Ariana Grande	48103

Figure 7. The top 10 mentioned artists in October by weeks.

To validate whether feature extraction is possible for this dataset, we tried to generate four features related to popularity measures that we are interested in with the filtered dataset. They were the total number of retweet counts of these relevant tweets (i.e., sum(retweet_count)), the total number of reply counts of these relevant tweets (i.e., sum(reply_count)), the total number of favorite counts of these relevant tweets (i.e., sum(favorite_count), and the total number of quote counts of these relevant tweets (i.e., sum(quote_count)). Looking at Figure 8, we can see that all songs in the filtered dataset have a value of 0 for all four features, which make them sparse features that may not be helpful for optimizing the performance of the prediction model. We looked into Twitter documentation and found that some of the attributes such as “favorite_count” may only show accurately when the authenticating user is the owner of the object. Knowing this, we think adding a manual step to judge whether the generated feature is useful is essential.

song_name	sum(retweet_count)	sum(reply_count)	sum(favorite_count)	sum(quote_count)
Why We Drink	0	0	0	0
Why Would I Stop?	0	0	0	0
Throat Baby (Go B...	0	0	0	0
More Than My Home...	0	0	0	0
Hate The Other Side	0	0	0	0
Beers And Sunshine	0	0	0	0
Mr. Right Now	0	0	0	0
Steppin On N*ggas	0	0	0	0
One Of Them Girls	0	0	0	0
Whats Poppin	0	0	0	0
RIP Luv	0	0	0	0
Pretty Heart	0	0	0	0
Big, Big Plans	0	0	0	0
Party Girl	0	0	0	0
Brand New Draco	0	0	0	0
I Called Mama	0	0	0	0
I Love My Country	0	0	0	0
La Toxica	0	0	0	0
Take You Dancing	0	0	0	0
Money Over Fallouts	0	0	0	0

Figure 8. The total number of retweet count, reply count, favorite count, and quote count of tweets in the filtered dataset for 20 songs.

Besides the features explored above, we also identified the following features that we could potentially extract that would optimize the performance of our prediction model:

- count_unique_user: Number of unique users of these relevant tweets. Identified by user_id.
- sum_unique_user_followers_count: Total number of followers these unique users' accounts currently has.
- sum_unique_user_friends_count: Total number of Twitter users these unique users' accounts is following.
- sum_unique_user_listed_count: Total number of public lists these unique users is a member of.
- count_emoji: Number of emojis used in the tweets. We could also identify top used emoji and obtain the count of each emoji.
- count_hashtags: Number of hashtags used in the tweets. We could also focus on specific hashtag such as "#nowplaying" in these relevant tweets [3].
- count_url: Number of urls used in the tweets. We would categorize tweets based on whether urls are included and count tweets with and without urls respectively.
- text-based features: Average length of the tweets and TF-IDF analysis.

Depending on the features that we will be extracting, additional data preprocessing such as removing stopwords, expanding contractions, and tokenization may be required.

4.2 Baselines

This is a multi-class classification where each record is assigned to one and only one tier in the chart. To find the best performing classifier for this dataset, we built various models available in

scikit-learn. In order to give an intuitive performance comparison against all the models built, two baselines were defined. We used only the attributes in the Billboard dataset to build the baselines. The performance of the baseline models are shown in Table 1.

Baseline 1. We trained a Dummy Classifier with 'uniform' as the strategy. The classifier generated predictions uniformly at random. The random state of the classifier was set to 0 to ensure the results are reproducible. See Figure 9 for the confusion matrix and Figure 10 for the AUC-ROC curves displaying the performance of the classifier.

Baseline 2. Taking into account the size of the test set, we think a Support Vector Machine (SVM) algorithm that directly optimizes the classification accuracy can be an indicative baseline. It uses a kernel trick to transform data and then finds an optimal boundary between the possible outputs based on the transformations. It has a tendency not to overfit but still perform effectively in many cases, especially in high-dimensional spaces. We trained a SVC classifier with the default parameters: 'rbf' as the kernel and 'ovr' as the decision function of shape which trained the classifier with one-vs-rest strategy. The random state was set to 0 to ensure the results are reproducible. The performance of the classifier is shown in Figure 9 and Figure 11.

Table 1. Evaluations of baseline models.

	MAE	**Macro Precision**	**Macro Recall**	**Macro F1 Score**
Baseline 1: Dummy	1.46	0.21	0.21	0.21
Baseline 2: SVC	0.319	0.73	0.74	0.73

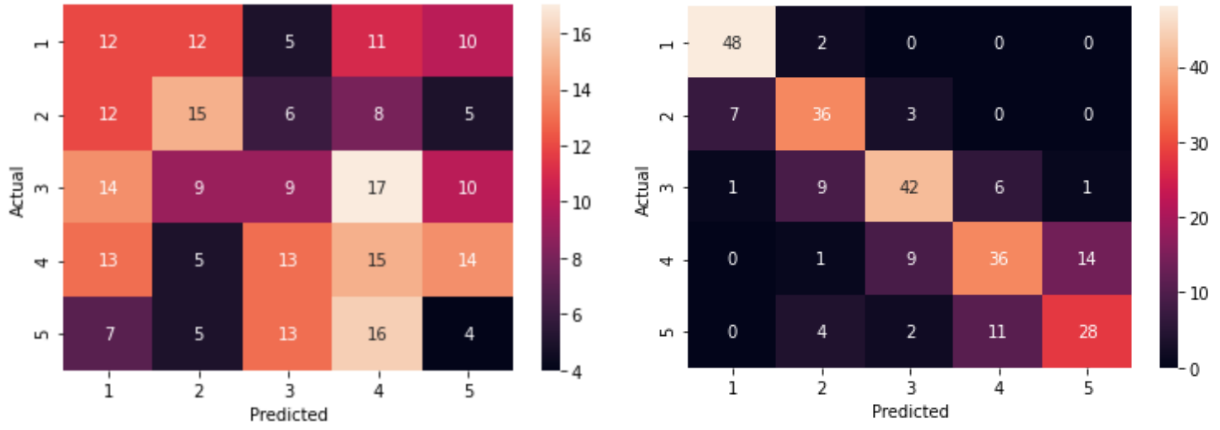

Figure 9. Confusion matrices showing the performance of the baseline models: Baseline 1 (left) and Baseline 2 (right).

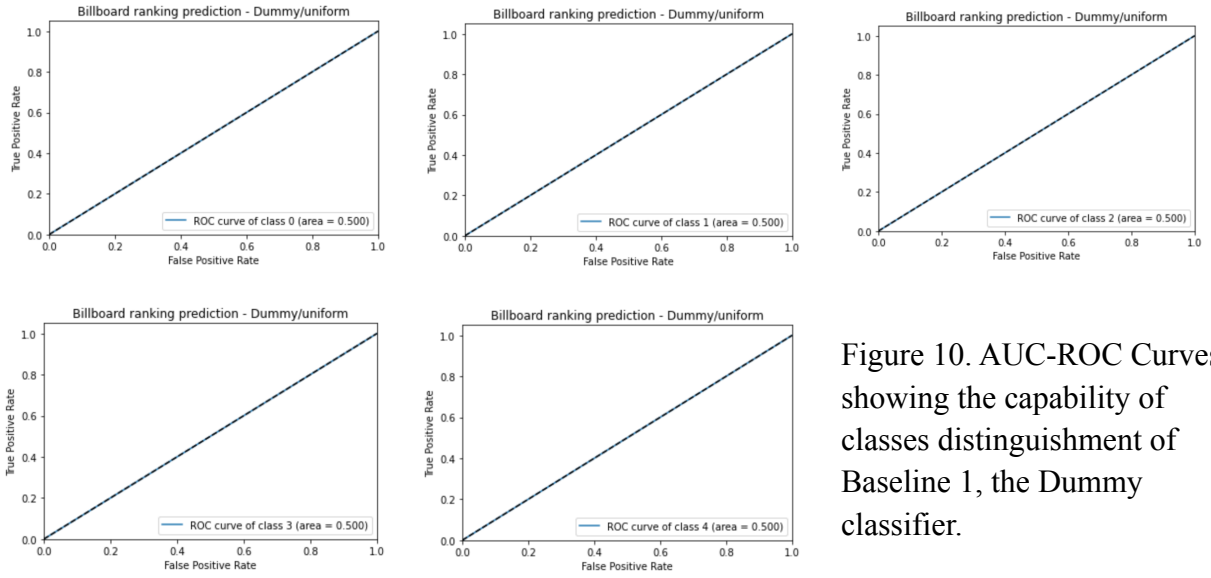

Figure 10. AUC-ROC Curves showing the capability of classes distinguishment of Baseline 1, the Dummy classifier.

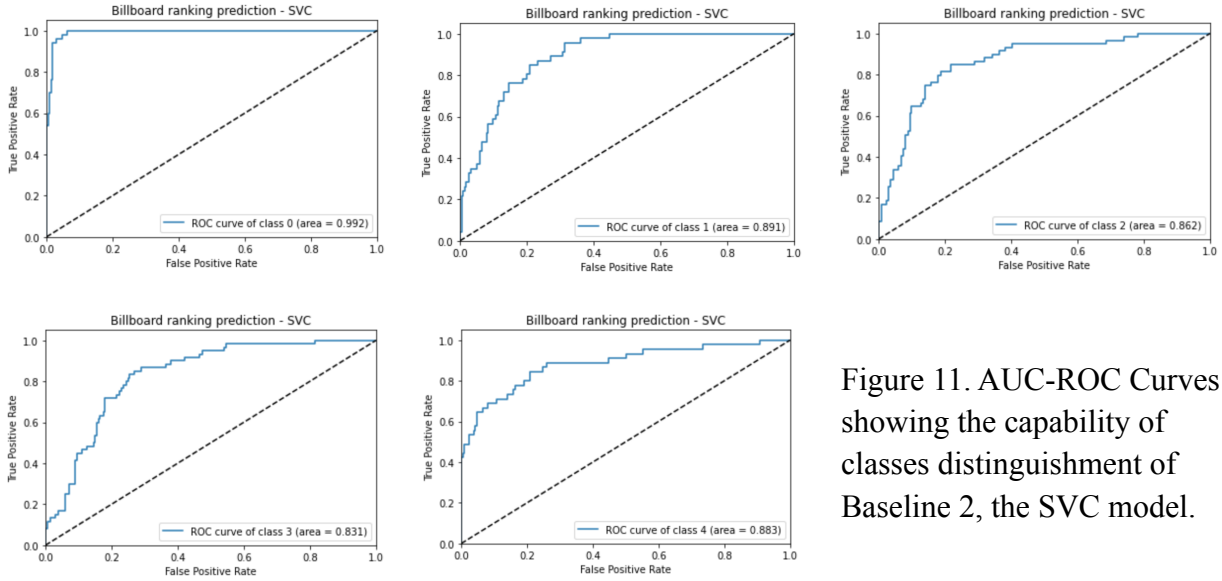

Figure 11. AUC-ROC Curves showing the capability of classes distinguishment of Baseline 2, the SVC model.

4.3 Analysis

In addition to the two baselines models, we trained a Multilayer Perceptron (MLP) classifier to validate the possibility of optimizing the prediction model. MLP is a classical type of neural network that comprises one or more layers of neurons; it is often considered to be very flexible. We trained a MLPClassifier with default parameters where ‘relu’ was the activation function. The random state was set to 0 to ensure the results are reproducible. The feature vectors were fed to the input layer and predictions of the chart ranking tier were made on the output layer; there might be hidden layers in between which allow the neural network to learn classifications that were not linearly separable. According to Table 2, the MAE, macro averaged precision, recall, and F1 score all performed better than both baseline models.

Table 2. Evaluation of the MLP model with only attributes from the Billboard dataset.

	MAE	Macro Precision	Macro Recall	Macro F1 Score
MLP Classifier	0.3	0.74	0.75	0.75

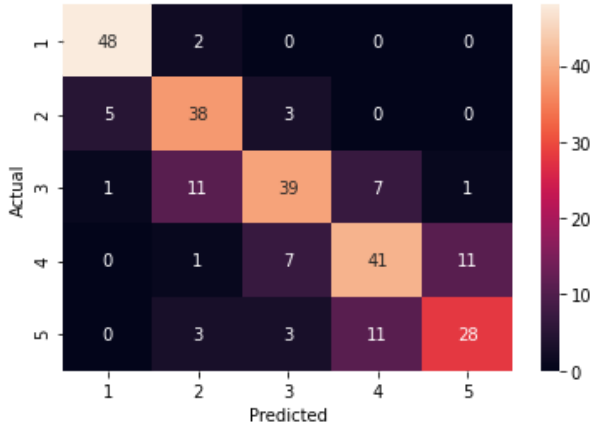

Figure 12. Confusion matrices showing the performance of the MLP model.

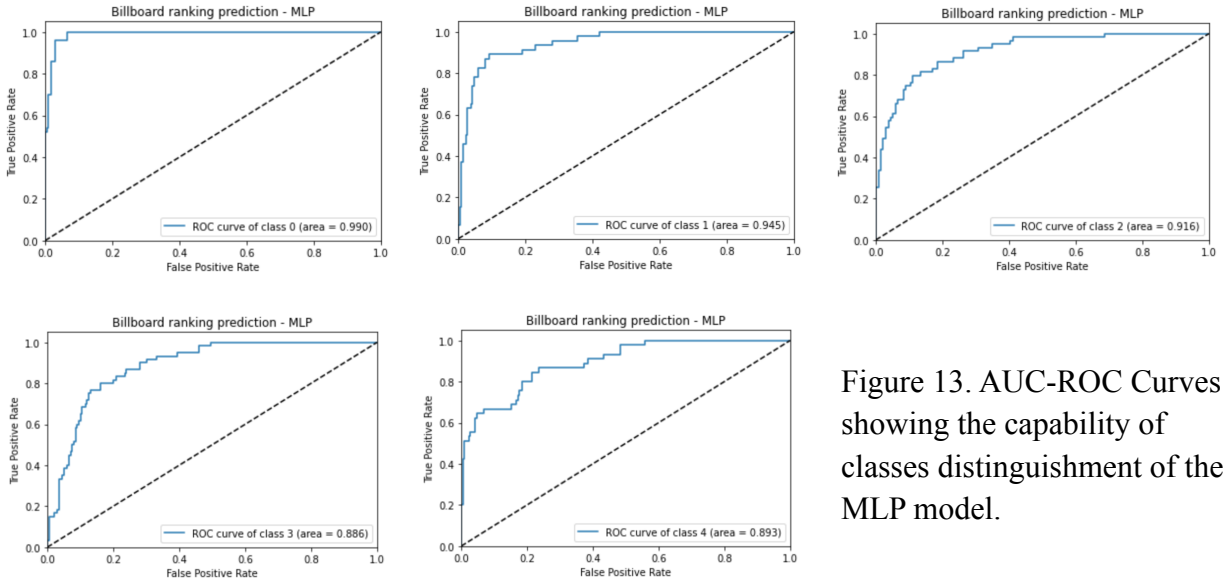

Figure 13. AUC-ROC Curves showing the capability of classes distinguishment of the MLP model.

Since the focus of this project is to explore whether textual components and popularity patterns obtained from Twitter dataset can help predict rankings of the Hot 100 songs on Billboard. At this middle stage of the project, we extracted two features from the Twitter dataset and added to the Billboard dataset for model building. These features were:

- count_tweet_song: Number of tweets relevant to the song in the time interval.
- count_tweet_artist: Number of tweets relevant to the artist in the time interval.

To get a sense of how features obtained from Twitter can affect the performance of the chart rankings prediction model, we built two additional models - a SVC model and a MLP model. The performance of the models built on the dataset with added features is shown in Table 3, and the confusion matrices are displayed in Figure 14. As mentioned in section 4.1.2, we were only exploring records in October for the Twitter dataset, therefore, we were not able to compare the results with the baseline models at this stage. However, the results indicated that our proposed approach is feasible and improving the models with added features is possible.

Table 3. Evaluations of SVC and MLP models on features added dataset.

	MAE	**Macro Precision**	**Macro Recall**	**Macro F1 Score**
SVC Classifier	0.36	0.67	0.67	0.66
MLP Classifier	0.29	0.77	0.78	0.77

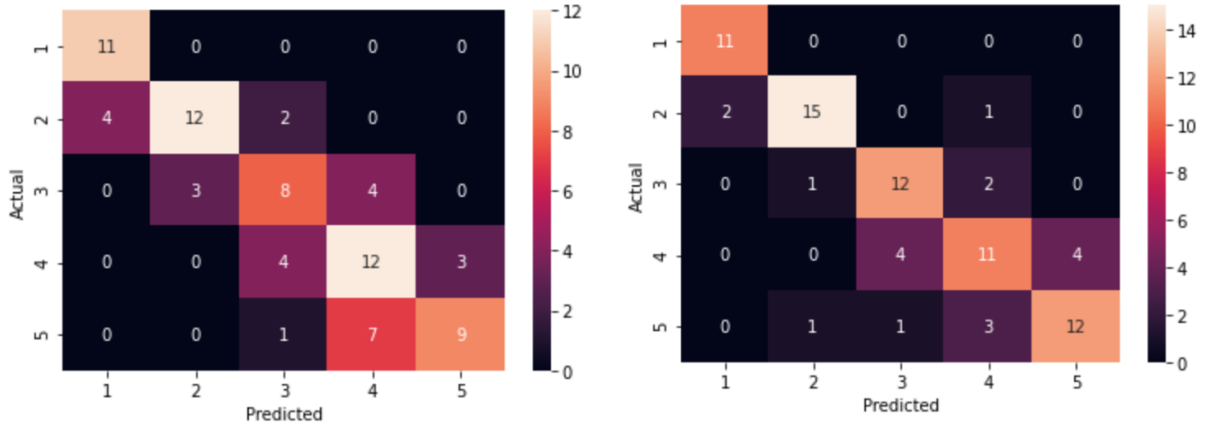

Figure 14. Confusion matrices showing the performance of the models built with added features: SVC (left) and MLP (right).

4.4 Next Steps

Our next steps will focus on feature extraction and models building. We will start by exploring and mining the Twitter Dataset more in-depth to extract features that we identified in section 4.1.2. Then, we will merge those extracted features to the Billboard dataset; this combined dataset will be utilized for building the prediction model. We will rely on feature selection, hyperparameters tuning, and other methods to optimize the performance of the classifiers.

Below is our proposed timeline for the second half of the semester:

Week	**To-do**
8, 9	• Extract features from the Twitter dataset. Explore different methods.

	● Expand literature reviews to find potential helpful techniques on feature extraction. ● Prepare the final dataset for model building - merge extracted features to the Billboard dataset.
10, 11, 12	● Build and optimize the multi-class classification models. ● Explore different machine learning techniques.
13	● Evaluate all the models built and decide on a final optimized model for further predictions. ● Document the results to the final paper.
14	● Finalize final paper and presentation.

5. REFERENCES

[1] Martin, J. (2020, May 13). *Music video streams have also increased by 10 per cent*. NME. https://www.nme.com/news/music/people-are-listening-to-more-new-music-during-coronavirus-pandemic-new-study-says-2667098

[2] Nolfi, J. (2020). *Mariah Carey makes chart history again — this time for dropping off completely*. Entertainment Weekly. https://ew.com/music/2020/01/06/mariah-carey-falls-off-billboard-hot-100/

[3] Tsiara, E., & Tjortjis, C. (2020). Using Twitter to Predict Chart Position for Songs. *IFIP Advances in Information and Communication Technology*, 62–72. https://doi.org/10.1007/978-3-030-49161-1_6

[4] Miller, S. (2021). *Billboard Hot Weekly Charts*. Data.World. https://data.world/kcmillersean/billboard-hot-100-1958-2017

[5] Michigan Institute for Data Science. (n.d.). *Twitter Decahose Data*. https://midas.umich.edu/twitter-decahose-data/

[6] Hutto, C. J., & Gilbert, E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. *Proceedings of the International AAAI Conference on Web and Social Media*, *8*, 216–225. https://www.aaai.org/ocs/index.php/ICWSM/ICWSM14/paper/view/8109

[7] Baccianella, S., Esuli, A., & Sebastiani, F. (2010). SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. *LREC*, *10*, 2200–2204. http://www.lrec-conf.org/proceedings/lrec2010/pdf/769_Paper.pdf