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

Yin Kwong (John) Lee
LSA, Department of Statistics
University of Michigan, Ann Arbor
lykjohn@umich.edu

Hsin-Yuan Wu
UMSI, School of Information
University of Michigan, Ann Arbor
wusean@umich.edu

Tz-Huei (Melody) Chang
UMSI, School of Information
University of Michigan, Ann Arbor
tzhueic@umich.edu

EXECUTIVE SUMMARY

In this project, we aim to explore whether tweet features from Twitter data can help predict rankings of the top 100 songs on Billboard. Our work is an extension of the previous work by Tsiara and Tjortjis [1]. In order to better represent the music charts prediction problem in reality, we also incorporated songs that were popular on Spotify. After querying 3-month worth of data, we gathered 16 features from the Twitter Decahose Data related to popularity measures (e.g., counts of song-related tweets), network patterns (e.g., average number of friends), and textual components (e.g., average sentiment score). Combining this with the weekly song rankings from the Billboard and Spotify charts, we compiled a complete list of predictors for the songs' future rankings. Considering the dimension of the training data, this is a multi-class classification problem. To represent song popularity, we grouped rankings into 6 classes, with class 0 indicating songs that were popular on the Spotify chart but not on the Billboard chart.

Following some data exploration, we fit an array of training algorithms, where the performance of each was evaluated with MAE, confusion matrix (macro F1 score), skewness, and kurtosis. We first fit and trained two baseline models using Naive Bayes and Support Vector Machine (SVC). The latter performs better among the two with a MAE score of 0.742 and a F1 score of 0.498. We then built a variant of classification designs with 11 other models, including Logistic Regression, Random Forest, Quadratic Discriminant Analysis (QDA) Classification, Multilayer Perceptron (MLP), Gradient Boosting, and many more. To optimize these models, we further applied normalization, feature selection, feature expansion, and hyperparameter tuning. It turns out that the Random Forest model with normalization and tree-based feature selection yielded the best results, having a MAE score of 0.387 and a F1 score of 0.719.

In general, we were impressed with the fit and evaluation scores for the ensemble methods. The results showed that discussions and activities on social media could imply song popularity, that predicting chart rankings with features extracted from social media is possible. Among all the features fed into the models, the popularity measure- counts of relevant tweets- is of the most important. Therefore, we believed that if we were able to incorporate other popularity measures such as the counts of retweets and likes, it would be possible for us to further enhance the model performance in the future. Some of the challenges that we faced include the real-time streaming nature of Decahose, which prevented us from obtaining accurate data on some important popularity measures, as well as the limited resources on Cavium, which led to longer data processing and feature extraction runtimes.

1. INTRODUCTION AND MOTIVATION

Every year end, many songs enter the market, some become timeless tracks, some fall under momentary spotlights, while others just never make it to the spotlights. These cycles throughout the market are identifiable to us. As media consumption rallied astronomically during the COVID-19 pandemic [2], our team has been active in sourcing popularity data such as song ranking patterns from Spotify and Billboard, as well as tweet features from Twitter. With this information, we are in a rich place to understand constituents of common trends, or even abnormal ones in the music industry. From sensations like "Taylor Swift's new album made history with more than 30 billion copies sold", to sudden shocker like "Mariah Carey's holiday hit "All I Want for Christmas is You" fell from Billboard's number one spot to completely off the chart", our team aims to seek how the features we gathered from the Twitter Decahose Data can help predict rankings of the Hot 100 songs on Billboard.

The previous work by Tsiara and Tjortjis [1] attempted to answer this using VADER's sentiment analysis for consolidating text parameters and regression analysis for classifying ranks of the top 10 Billboard songs. We extended their study to include more comprehensively-ranked data of the top 100 Billboard and Spotify songs. Our study started with querying 3-month worth of data from Billboard charts, Spotify charts, and the Twitter Decahose database. Tweet records from the Decahose database were combined with charts data to include features like song titles, artist names, rankings, and tweet information related to popularity measures, network patterns, and textual components. The next stage is to understand how well the features extracted from the Twitter dataset can help explain and predict song rankings on Billboard. Our team experimented over 25 machine learning models, including Logistic Regression, Random Forest, Quadratic Discriminant Analysis (QDA) Classification, Multilayer Perceptron (MLP), Gradient Boosting, and many more.The primary goal is to identify a training framework that can not only minimize the testing errors, but also regulate the balance-variance trade-off possessed by modern day learning models. The results from our study could potentially imply whether discussion on social media could predict the popularity of a song on a music chart.

2. RELATED WORK

Although it is not highly anticipated, the Institute for Language & Speech Processing indicates that the music information research field does have continuous development. Track popularity prediction, or "Hit Song Science", aims to apply machine learning techniques to capture some information from musical data that would explain the popularity of the respective musical tracks [3]. One example that we found interesting was a study by Kim et al. in 2014 [4]. They utilized music-related hashtags such as #nowplaying to collect users' tweets on Twitter, and built a predictive model to forecast the Billboard rankings. The researchers found that Billboard popularity could be predicted by the numbers of daily tweets about specific songs and artists. It was also suggested that "users' music listening behavior on Twitter is highly correlated with

general music trends and could play an important role in understanding consumers' music consumption patterns" [4].

Tsiara and Tjortjis's research [1] is the one that inspired us the most. They utilized Twitter data accessed via the Twitter Search API to predict the top 10 chart positions for songs on the Billboard Hot 100 charts. To obtain features from the Twitter dataset, they conducted sentiment analysis with Valence Aware Dictionary and Sentiment Reasoner (VADER) [5] and SentiWordNet [6] and added sentimental scores of positive, negative, and neutral scores as features. They also mined the tweets to identify features related to songs and artists. In addition, they included information from Billboard such as the song position from prior week and the accumulated number of weeks the song was on the chart. They approached the problem via regression analysis and implemented several algorithms, including 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. DATASETS

To build a prediction model for the music chart, we utilized the following three 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 [7]. 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. The dataset has a total of 10 columns, but we only utilized the following four attributes:

- 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".

2.2 Twitter Dataset

We accessed the Twitter Decahose Stream via the Michigan Institute For Data Science (MIDAS) at the University of Michigan [8]. 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. Therefore, we ended up having more than 150 variables. For the purpose of this project, we pre-selected 28 columns to include in our analysis. See Appendix A for the columns and description of these columns.

2.3 Spotify Dataset

In order to better represent the music charts prediction problem in reality, we wanted to include songs that were popular even though they were not on the Billboard chart. We further collected the top 100 Spotify Weekly Charts [9]. We targeted only the chart in the United States for the same 13 weeks defined for the Billboard dataset. For the purpose of enriching the list of popular songs for feature extraction with Twitter dataset, we only included three columns: 'week', 'song_name', and 'artist_name'.

4. METHODS

We shadowed the procedures suggested by Tsiara and Tjortjis [1] to extract variable features, implement prediction models, and fine-tune model performances.

4.1 Feature Engineering

To avoid data leakage, twitter features with a 7-day lookback period were retrieved from the Decahose database. These are tweet variables related to each song 7 days before it is ranked on the Billboard and Spotify charts. We focused on tweets written in English. In our initial exploration of the twitter dataset, we found that it is possible for different artists to have songs with the same name. For example, as shown in Figure 1, the name '21' was a song title for multiple artists; this may be due to the name of the song being a word often used in daily conversations. To ensure the tweets extracted from the dataset were indeed related to the songs, we only included tweets that mentioned both song title and artist name. This might overkill relevant tweets, but we think this could make sure tweets we retrieved were indeed about the songs.

song	artist	
21	AJR	11
	Ariana Grande	11
	Ava Max	4
	BLACKPINK	14
	BLACKPINK X Selena Gomez	2
	BTS	14
	Big Sean	7
	Billie Eilish	9
	Bryson Tiller	10
	CJ	14

song	artist	
21	Chase Rice	1
	Chloe X Halle	4
	Chris Stapleton	1
	Clairo	2
	Conan Gray	1
	DaBaby	11
	Don Toliver	3
	Drake	14
	Drake Featuring Lil Durk	1
	Dua Lipa	7

Figure 1. Example of a song title used by multiple artists.

Feature extraction was performed with PySpark. We first obtained basic statistics like the counts of tweets related to each song. Also, we took a look at columns that might be indicators of the popularity of tweets, such as counts of like, retweet, reply, and quote. Looking at Figure 2, 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. Because Decahose delivers a 10% random sample of the realtime Twitter Firehose through a streaming connection, all the tweets collected are considered "newly created". Therefore, all four popularity measures would have a value of 0 in this data source. Knowing this, we decided to exclude them from the features.

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

Furthermore, we explored features that required in-depth data mining. For example, whether urls or emojis in tweets could affect viewers' perception regarding each song. As of the tweet contents, we retrieved features such as the average length of the tweets for each song and the average counts of hashtags included in relevant tweets for each song. For features related to social patterns, we aggregated the data by unique user ids.

For sentiment analysis, we leveraged VADER, which is a lexicon and parsimonious rule-based sentiment analysis for social media text [5]. We turned the emoji into utf-8 lexicon along with other texts in tweets, computing the scores by mapping to the VADER sentiment score files. The scores were normalized to -1 (most extreme negative) and +1 (most extreme positive).

With careful understanding of the twitter dataset, we found that for tweets that exceed 140 characters, the full text of the tweet content would be placed under the extended tweets object. Therefore, for records where the 'truncated' column equals 'True', we applied logical rules to ensure that we are extracting information from the complete tweet content.

Below is the complete list of features we extracted from the twitter dataset:

- week: The corresponding week that the tweet was created at.
- song: The song title that the tweet is relevant to.
- artist: The artist of the song that the tweet is relevant to.
- tweets_count: The total number of tweets relevant to the song.
- sum_http: The total number of tweets relevant to the song that contain “http”.
- avg_tokens: The average word count of the tweets relevant to the song.
- avg_char: The average character count of the tweets relevant to the song.
- url_avg_combined: The average count of urls included in the tweets relevant to the song.
- display_url_avg: The average count of ‘display_url’ in the tweets relevant to the song.
- hashtag_avg_combined: The average number of hashtags included in the tweets relevant to the song.
- avg_friends: The average number of ‘friends_count’ for tweets relevant to the song.
- avg_listed: The average number of ‘listed_count’ for tweets relevant to the song.
- avg_followers: The average number of ‘followers_count’ for tweets relevant to the song.
- sum_trunc: The total number of tweets relevant to that song that are truncated.
- avg_emoji_count: The average number of emoji used in the tweets relevant to the song.
- avg_senti: The average sentiment score of the tweets relevant to the song. It is a combination of text sentiment scores and emoji scores.

4.2 Data Preprocessing

We combined the Billboard dataset, Spotify dataset, and the features extracted from Twitter dataset by ‘week’, ‘song’, and ‘artist’. We dropped songs that did not have relevant tweets and ended up having 620 records in the combined dataset. To be able to feed these features into the model directly, we preprocessed the data by factorizing string attributes- “song” and “artist”.

The original Billboard dataset ranks the chart from 1 to 100. Besides the exact rank positions, we wanted to focus on song popularity, therefore, we transformed the problem into a multi-class classification. We defined the popularity class, with a column name ‘pop’ in the combined dataset, into six tiers as follows:

- Class 0: top 100 on Spotify chart but not ranked on Hot 100 Billboard chart
- Class 1: rank 1 to rank 20 on Billboard chart
- Class 2: rank 21 to rank 40 on Billboard chart
- Class 3: rank 41 to rank 60 on Billboard chart
- Class 4: rank 61 to rank 80 on Billboard chart
- Class 5: rank 81 to rank 100 on Billboard chart

4.3 Exploratory Data Analysis (EDA)

4.3.1 Class Distribution

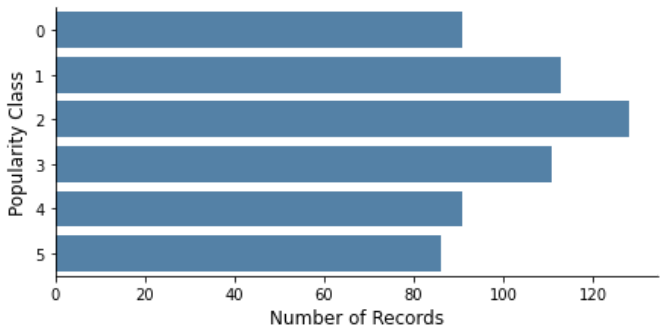

Figure 3. Distribution of the popularity class.

The originally sampled popularity classes are ill-balanced after dropping songs without relevant tweets, with the majority of songs falling in class 0. This would have exposed our predictions to be extremely biased towards the majority class. To cope with this, we down-sampled class 0 to contain only 7 songs each week so that the labels are better balanced (Figure 3).

4.3.2 Feature Relationships

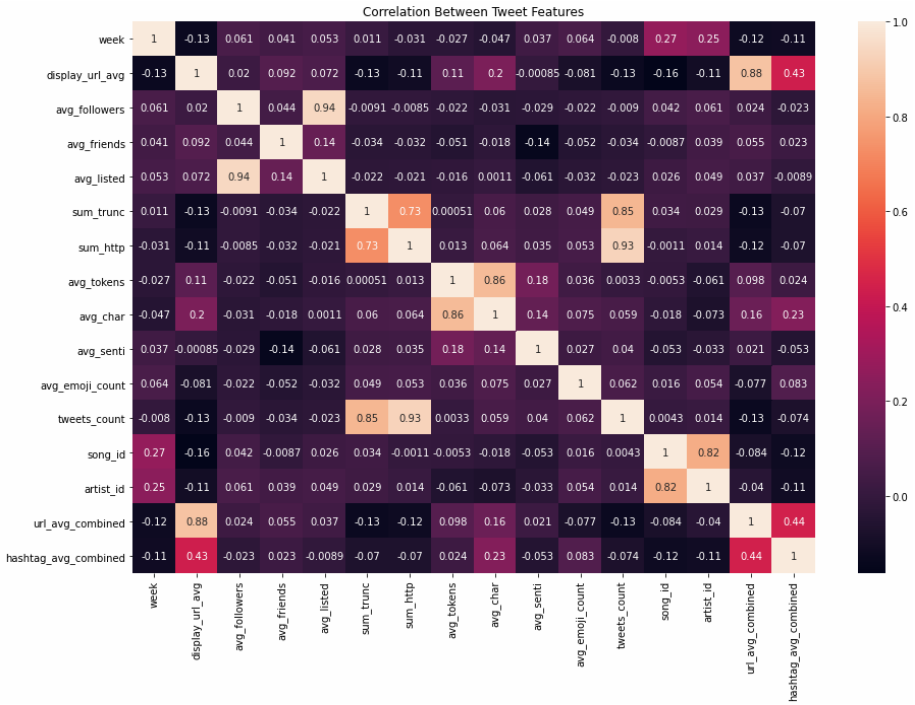

Figure 4. Paired correlations of features extracted from the Twitter dataset.

According to the correlation heatmap (Figure 4), some noticeable correlations are highlighted. From a network perspective, the average number public lists that a user has been added to has a near-perfect positive correlation with the average number of followers the user has at 0.94. This is not surprising because the more groups a user is in the more connection he or she has, hence more expected followers. From a tweet perspective, the relatively high correlations in the 0.8-0.9

range between any pair of content features seems trivial for our interpretation. For example, the average word counts and average number of characters has a 0.86 correlation; of course, more words in a tweet implies more characters.

4.3.3 Random Forest Feature Importance

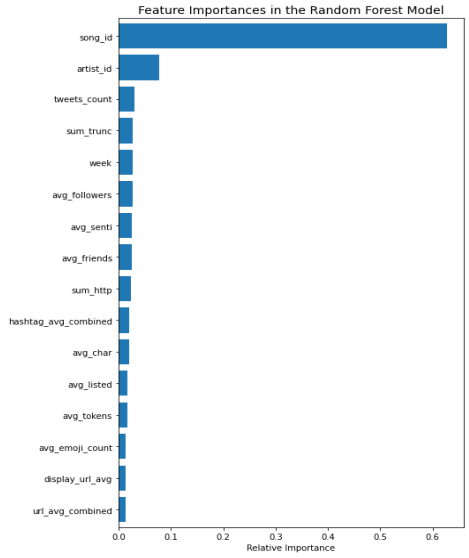

Figure 5. Feature importances in the Random Forest Model.

To get a sense of how the prediction models might work, we ranked all 15 features based on their importance scores from a Random Forest Regression model. According to Figure 5, song title (i.e., 'song_id') and artist name (i.e., 'artist_id') were the two most important features; we believed this is highly related to how we obtained the list of popular songs from two datasets- some of the songs would only appear in class 0. What caught our attention was that the count of tweets relevant to each song (i.e., 'tweets_count') ranked third among all features. This may indicate that popularity measures extracted from Twitter may have a significant impact on predicting chart rankings. In addition, with the assumption that tweets from users with more friends and followers would have higher reach, we did see higher importance of 'avg_followers' and 'avg_friends' comparatively.

4.3.4 Exploratory Factor Analysis (EFA)

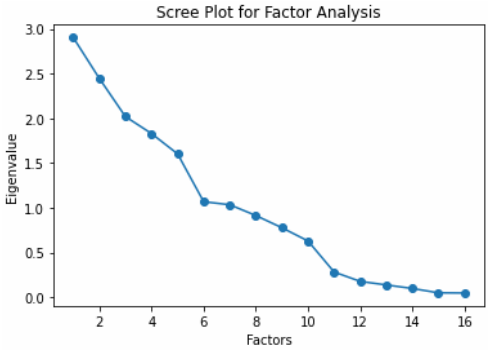

Figure 6. Scree plot for factor analysis.

To consolidate factors that cannot be observed, we produced a scree plot (Figure 6) to display the eigenvalues of each factor, and decided to keep 6 of them because with eigenvalues of at least 1, each factor explains more variance than an observed variable. We produced a clustermap that designates 6 latent factors and their corresponding weights to each of the tweet features. Factor 1 may capture the fact that media results such as contents of a music video or comments on a celebrity image directed from an URL may propagate media attention amongst the twitter community, therefore giving rise to tweets counts. This is well justified by its near-perfect correlations of 0.99 and 1with the number of tweets with URL and the number of overall tweets, respectively.

According to Figure 7, the average number public lists that a user has been added to and the average number of followers the user has are two features with strong association with Factor 2, having loadings of 0.99 and 0.95, respectively. Hence, Factor 2 can represent the networking effect of the twitter community. Factor 3 stands out as a possible indicator for public appetite of lengthy tweets because of its high correlations of 0.87 and 0.99 with average word and character counts in a song-related tweet, respectively. Factor 4 could be an assortment of marketing strategies used to promote artist and song titles because of its high correlations with these features-0.8 and 0.99, respectively. For example, including a capital letter in a performer's name, like "Jay Z", may serve better crowd recognition. We decided to discard Factor 5 because it can be replaced with the average number of users an account is following for their high correlation at 0.99, as well as Factor 6 because there are no features with significant loadings that can give meaningful latent implications.

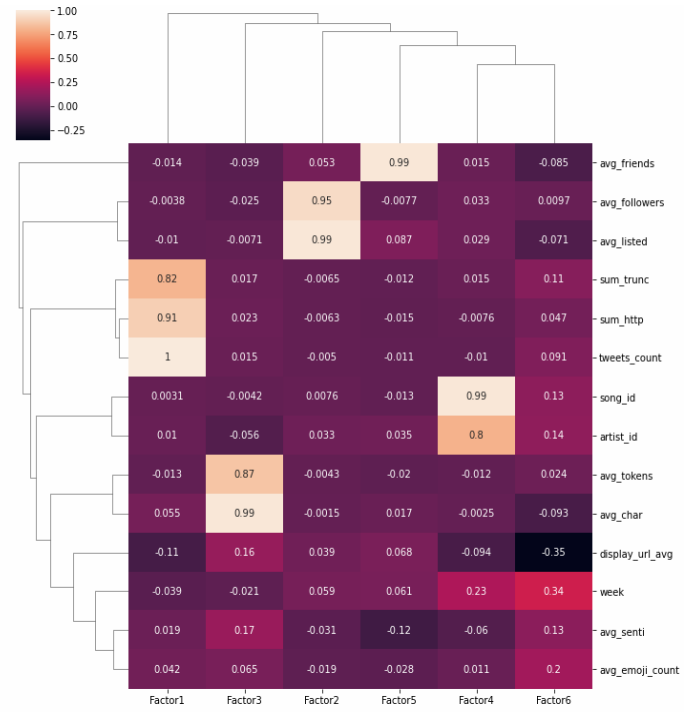

Figure 7. Clustermap comparing original variables vs. 6 latent factors.

4.4 Model Building

The combined dataset was split into a training set and a test set with a 80:20 ratio. In order to predict the popularity of each song on the Billboard chart, we treated it as a multi-class classification problem. Therefore, after splitting the data, We assessed a variant of classification designs for predicting song rankings in the upcoming week. To find the best performing classifier for this dataset, we built various models available using the scikit-learn library in Python.

We first trained two baseline models with Support Vector Machines and Naive Bayes classifiers. Then, we employed neural networks models, ensemble models, and other classifiers to quantify the error results. Models that we explored include Random Forest Classifier, Extra Trees Classifier, Logistic Regression, K-Nearest Neighbors Classifier (KNN), Multilayer Perceptron (MLP), Linear Discriminant Analysis (LDA) Classification, Quadratic Discriminant Analysis (QDA) Classification, Classification and Regression Tree (CART), Bagging Classifier, Gradient Boosting Classifier, and AdaBoost Classifier.

Due to the relatively smaller size of the final combined dataset, we applied cross-validation to avoid overfitting or underfitting the models.

4.5 Model Optimization

In order to optimize the performance on the models mentioned above, different techniques- standardization, regularization, feature selection, and feature expansion- are used to optimize the predictive power of the proposed models while regulating their balance-variance trade-offs. We also performed hyperparameters tuning to determine the ideal model architectures.

4.6 Evaluation Approach

For model selection, we used 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 evaluated the prediction accuracy of models with confusion matrix (macro averaged F1 score), skewness, and kurtosis.

5. RESULTS

5.1 Baselines

This is a multi-class classification where each record is assigned to one and only one class in the chart. In order to give an intuitive performance comparison against all the models built, two baselines were defined. The performance of the baseline models are shown in Table 1.

Baseline 1. Naive Bayes is a generative classifier that explicitly models the actual distribution of each class. Since the algorithm requires a small amount of training data to estimate the necessary parameters, it fits well for our situation where the training data only contains 496 records. We trained a Multinomial Naive Bayes (MNB) classifier with default parameters. MinmaxScaler was applied to rescale each feature into a range. See Figure 8 for the confusion matrix displaying the performance of the classifier.

Baseline 2. Taking into account the size of the dataset, we believe 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 tuned hyperparameters: 'rbf' as the kernel, 10 as the regularization parameter, 0.5 as the gamma, 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. Each feature was normalized by MinmaxScaler. The performance of the classifier is shown in Figure 8.

Table 1. Evaluations of baseline models.

	MAE	Macro F1 Score	Skewness	Kurtosis
Baseline 1: Naive Bayes	1.177419	0.302438	0.380208	0.788192
Baseline 2: SVC	0.741935	0.49782	0.481523	-0.872659

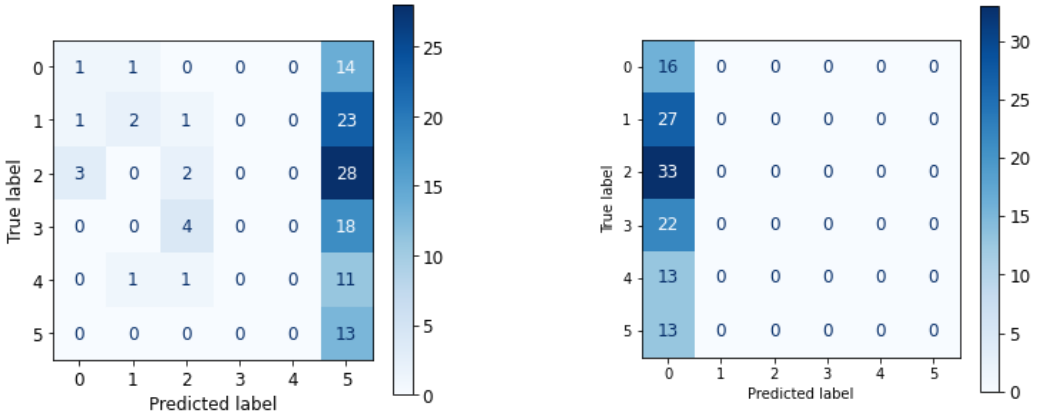

Figure 8. Confusion matrices showing the performance of the baseline models: Baseline 1 (left) and Baseline 2 (right).

5.2 Classification Models

In addition to the two baselines, we built a series of classification models with 11 different machine learning algorithms. The evaluations of all models that we trained can be found in Appendix 2. In this section, we will go into detail the results and analysis of the best performing model for each machine learning algorithm. We compared the prediction results via confusion matrices shown in Figure 9, and found it interesting that Extra Trees and AdaBoost models predicted all songs as class 0, Logistic Regression and KNN models predicted the majority of songs as class 2, and QDA classification classified most of the songs to class 4. Such biases may be due to dropping some important data that shapes our assumptions for model fitting- for example, data that make labels more normally-distributed before fitting the QDA model. From a balancing standpoint, QDA and KNN tend to suggest more extreme rank prediction. Regardless, we found that models using ensemble methods outperformed other types of models overall, in particular, Random Forest Classifier, which reduces the estimation variance substantially. Details for each model are described below.

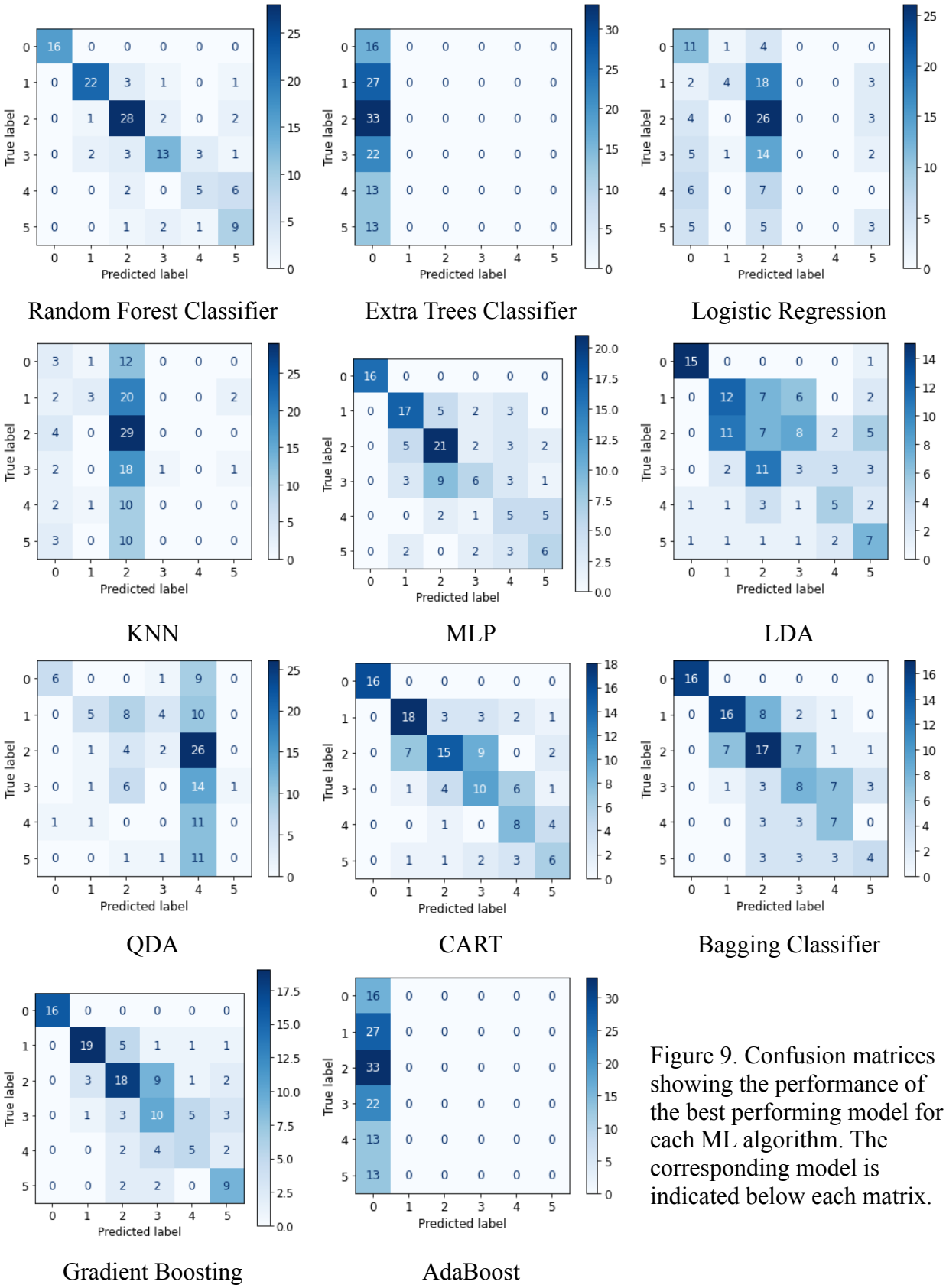

Figure 9. Confusion matrices showing the performance of the best performing model for each ML algorithm. The corresponding model is indicated below each matrix.

Random Forest (RF) Classifier. The best performing RF model is with normalization and tree-based feature selection. Compared to baselines, this model yielded much lower MAE and higher F1 score (Table 2). This is the most optimized model among all that we trained because of its decorrelated bootstrap sample- as the tree structure in the RF algorithm splits under bootstrap sampling, a random subset of predictors is chosen as a split candidate, hence decorrelating the tree features and reducing the variance of the prediction. Skewness and Kurtosis seems about average compared to those of other models.

Extra Trees Classifier. We trained an Extra Trees Classifier with default parameters. Random state was set to 0 to ensure the performance is reproducible. It has a lower MAE and a slightly higher F1 score than the baseline models (Table 2). However, it did not perform better than other tree-based classifiers, as all predicted values are 0, so we did not future optimize the model.

Logistic Regression (LR). We performed hyperparameter tuning and built a LR model with 5 as the inverse of regularization strength value ('C') and l2 penalties. Looking at Table 2, the performance was worse than the baselines both in terms of MAE and F1 score. Prediction seems right-skewed and kurtosis seems to be close to the average of the ensemble classifiers.

K-Nearest Neighbors Classifier (KNN). A KNN model with 7 nearest neighbours was selected because it yielded the highest accuracy score in cross validation. Compared to the baselines, the model had a lower MAE but a slightly lower F1 score (Table 2). This model seems to be relatively right skewed, meaning that the majority of songs are predicted to have higher ranks. A relatively small kurtosis indicates a tail prediction closest to a normal distribution.

Multilayer Perceptron (MLP). The best performing MLP model was when normalization, feature selection, and feature expansion was applied. We utilized MinMaxScaler, SelectFromModel with ExtraTreesClassifier, and PolynomialFeatures with 2 degrees. Based on Table 2, the performance for MAE and F1 score were both better than the baselines. Skewness and Kurtosis seems about average compared to those of other models.

Linear Discriminant Analysis (LDA) Classification. A LDA classification model was trained using the default parameters. It assumes each class is Gaussian and has constant variance. Although the MAE was slightly lower than the baselines, it did not perform better in terms of F1 score (Table 2). Predictions are relatively symmetric and tail values are relatively light compared to those of other models.

Quadratic Discriminant Analysis (QDA) Classification. A QDA classification model was trained with default parameters. It assumes each class is Gaussian. However, according to Table 2, the performance of the QDA model not only did not exceed the baseline models, it was even worse than the LDA model. Looking at skewness and kurtosis, the model seems to be heavily

left skewed, meaning that most songs are predicted to have lower ranks. Furthermore, the predicted labels have heavier tails under the QDA framework, implying that most songs are predicted to rank in the 0 and 5 classes.

Classification and Regression Tree (CART). We trained a Decision Tree Classifier with 'entropy' as the impurity measure, which recursively split the feature space into parts. Although this process is inexpensive to construct, it lacks the global optimality (greediness). Similar to other tree-based models, the model performance was better than the baselines (Table 2) based on its MAE and F1 score. Prediction results are the least skewed, and extreme ranks are the lightest compared to normal.

Bagging Classifier. A Bagging classifier was trained using the default parameters. It allows us to use a random subset of the dataset with replacement (bootstrap) and aggregate each individual prediction. This model performed better than the baselines (Table 2) based on its MAE and F1 score. Symmetry is in sight for the predictions, as kurtosis seems to be close to the average of the ensemble classifiers.

Gradient Boosting Classifier. We then tried to build models with boosting classifiers. The best performing Gradient Boosting model was with normalization (MinMaxScaler) and feature selection (SelectFromModel using ExtraTreesClassifier). Seeing Table 2, the performance turned out to be better than the baselines according to MAE and F1 score. Predictions are more right skewed than the Bagging Classifier as the kurtosis is higher.

AdaBoost Classifier. We also fit a AdaBoost Classifier with default parameters. The model begins by fitting a classifier on the dataset, then fits another same classifier on the dataset, and adjusts the weights on those incorrect predictions. Even though the MAE score was lower than the baselines, as the F1 score of 0.369 was not better than both baseline models (Table 2). Skewness seems moderate on a positive end, while kurtosis is the smallest amongst the ensemble methods.

Table 2. Evaluations of the best performing model of each machine learning algorithm.

Model	MAE	Macro F1 Score	Skewness	Kurtosis
Random Forest	0.387097	0.718867	0.384444	-0.825835
Extra Trees	0.5	0.590756	0.335892	-0.967174
Logistic Regression	0.927419	0.446445	0.529091	-0.875086
KNN	0.540323	0.431001	0.639962	-0.020552
MLP	0.66129	0.558377	0.357644	-0.879741
LDA Classification	0.604839	0.424988	0.289208	-1.044898
QDA Classification	0.790323	0.199224	-1.28805	0.411757

CART	0.41129	0.595368	0.140396	-1.066185
Bagging	0.451613	0.551487	0.168065	-0.82598
Gradient Boosting	0.564516	0.624683	0.202851	-0.907105
AdaBoost	0.596774	0.369157	0.417929	-0.412642

5.3 Additional Works

We adopted suggestions from the GSI and explored prediction on the Billboard dataset via regression analysis. We looked at the original rankings from Billboard and built models with the same set of features mentioned above. In particular, we trained a Linear Regression model and a Stochastic Gradient Descent (SGD) model, which yields very high MSE scores of 209 and 214, respectively. We are currently seeing low prediction power on the rankings with these features, but if more time is given, we hope to further optimize these models.

6. CONCLUSIONS

6.1 Conclusions

In this project, we built a variant of models to predict the Billboard song popularity with features extracted from Twitter, and we did achieve pretty good accuracy compared to the baseline models. Overall, we were impressed with the fit and evaluation scores for the ensemble methods such as Random Forest and Gradient Boosting. It is not surprising that the ensemble methods outperformed other models we fit and evaluated since they do a better job of handling features and variables through decision trees.

The results showed that discussions and activities on social media could indicate song popularity, and predicting chart rankings with features extracted from social media is possible. Among all the features fed into the models, counts of relevant tweets (i.e., ‘tweets_count’) are an important feature. This feature is considered a popularity measure of the song. Therefore, we believed that if we were able to obtain other popularity measures such as the counts of retweets and likes, it would be possible for us to further enhance the model performance.

6.2 Future Work

As mentioned in section 4.1, the current definition of relevant tweet to a song required the tweet to contain both song title and artist name. However, we found that some of the tweets used partial or abbreviated song titles, and they referred to artist names in the same way. Therefore, we think utilizing a keyword list that includes all variations of song titles and artist names will allow us to filter relevant tweets more accurately.

Seeing the limitations of the Decahose data, we think it will be better to utilize the Twitter Search API next time. Since popularity measures turned out to be important in the model, we will be more likely to obtain accurate data on the counts for popularity features such as likes, retweets, etc. Also, by manipulating timestamps and queries, we will be able to collect relevant tweets for

every song and better-balance the dataset. This can help mitigate persisting biases such as label-imbalance or missing-value imbalance problems encountered in our study.

If time permits, we would like to expand the time interval of our data extraction and look at rankings for a longer period of time, say over a 12-month horizon. By doing so, we will possess a larger training data, which can further enhance model performance and prevent overfitting. Furthermore, as shown in section 5.3, if we were to solve the problem through regression analysis, having more data will enable us to obtain better predictive models

7. CHALLENGES

One challenge is related to the Twitter Decahose Data. As mentioned above, the dataset has a deeply nested structure, which makes it difficult to narrow down features for our analysis, especially similar ones. Due to the reason that Decahose does not have a dedicated documentation, we spent a lot of time mapping each attribute to the general Twitter API documentation to figure out what information each column is providing, especially for the nested columns that have duplicated attribute names. Furthermore, due to the nature of Decahose, we are missing some popularity measures that may be important features in our models. These missing values may also lead to biases since not all distributional properties are captured. Missing value imputation can be an alternative to removing them in further studies.

Other challenges are related to the use of Cavium. First, Cavium has limited resources and only a limited number of users can use Cavium at the same time. Therefore, we needed to "line up" to be able to access those resources. Second, we encountered multiple disruptions on Cavium while working with the large Twitter dataset, which significantly slowed down the running time for our feature engineering process. Last but not least, Cavium does not allow for collaboration and this affected the efficiency of our project execution.

8. REFERENCES

[1] 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

[2] 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

[3] Music Information Research. (n.d.). *Track popularity prediction*. Institute for Language & Speech Processing. http://mir.ilsp.gr/track_popularity.html

[4] Kim, Y., Suh, B., & Lee, K. (2014). #nowplaying the future billboard: mining music listening behaviors of twitter users for hit song prediction. Proceedings of the First International Workshop on Social Media Retrieval and Analysis - SoMeRA '14, 51–56. https://doi.org/10.1145/2632188.2632206

[5] 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
[6] 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
[7] Miller, S. (2021). *Billboard Hot Weekly Charts*. Data.World. https://data.world/kcmillersean/billboard-hot-100-1958-2017
[8] Michigan Institute for Data Science. (n.d.). *Twitter Decahose Data*. https://midas.umich.edu/twitter-decahose-data/
[9] Spotify. (n.d.). *Spotify Charts*. https://spotifycharts.com/regional/us/weekly/latest
[10] Twitter. (n.d.). *Data dictionary: Standard v1.1*. Twitter Developer. https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/overview

9. APPENDIX

Appendix A. Description of the selected columns from Twitter dataset [10].

Column	Description
created_at	UTC time when this Tweet was created.
id	The integer representation of the unique identifier for this Tweet.
truncated	Indicates whether the value of the text parameter was truncated, for example, as a result of a retweet exceeding the original Tweet text length limit of 140 characters.
lang	When present, indicates a BCP 47 language identifier corresponding to the machine-detected language of the Tweet text, or “und” if no language could be detected.
user.id	The integer representation of the unique identifier for this User.
user.followers_count	The number of followers this account currently has.
user.friends_count	The number of users this account is following (AKA their “followings”).
user.listed_count	The number of public lists that this user is a member of.
text	The actual UTF-8 text of the status update.
entities.hashtags.text	Name of the hashtag, minus the leading ‘#’ character.
entities.urls.display_url	URL of the media to display to clients.
entities.urls.expanded_url	An expanded version of display_url. Links to the

	media display page
entities.urls.url	Wrapped URL for the media link. This corresponds with the URL embedded directly into the raw Tweet text.
entities.user_mentions.screen_name	Screen name of the referenced user mentioned in the tweet.
extended_tweet.full_text	The entire untruncated text of the Tweet.
extended_tweet.entities.hashtags.text	The name of the hashtag from the entire untruncated text.
extended_tweet.entities.urls.display_url	URL of the media to display to clients in the entire untruncated Tweet text.
extended_tweet.entities.urls.expanded_url	An expanded version of display_url for the entire untruncated Tweet text.
extended_tweet.entities.urls.url	This corresponds with the URL embedded directly into the entire untruncated Tweet text.
extended_tweet.entities.user_mentions.screen_name	Screen name of the referenced user mentioned in the entire untruncated Tweet text.
place.country	Name of the country containing this place.
place.country_code	Shortened country code representing the country containing this place.
place.name	Short human-readable representation of the place's name.
place.place_type	The type of location represented by this place.
favorite_count	Indicates approximately how many times this Tweet has been liked by Twitter users.
reply_count	Number of times this Tweet has been replied to.
retweet_count	Number of times this Tweet has been retweeted.
quote_count	Indicates approximately how many times this Tweet has been quoted by Twitter users.

Appendix B. Results of all models we have fit and trained. Error refers to MAE and score refers to accuracy.

Model	train error	test error	train score	test score	F1-Score	Skewness	Kurtosis
SVC (With scaler) - Baseline	0.639	0.742	0.605	0.492	0.498	0.482	-0.873
SVC (Without scaler)	1.383	1.161	0.214	0.282	0.094	-7.682	57.016
SVC (ExtraTrees)	0.833	0.847	0.510	0.524	0.503	0.542	-1.006

MLP (With scaler)	0.335	0.710	0.778	0.508	0.515	0.537	-0.469
MLP (Without scaler)	1.498	1.250	0.397	0.411	0.115	-2.647	5.202
MLP (SelectPercentile)	0.921	0.927	0.492	0.452	0.427	0.104	-1.308
MLP (ExtraTrees)	0.708	0.758	0.556	0.532	0.503	0.365	-1.003
MLP (Expansion2)	0.034	0.694	0.978	0.516	0.510	0.454	-0.890
MLP (ExtraTrees + Expansion2)	0.361	0.661	0.742	0.573	**0.558**	0.358	-0.880
Random Forest (With Scaler)	0.000	0.573	0.163	0.129	0.560	0.418	-0.789
Random Forest (SelectPercentile)	0.319	0.444	0.742	0.653	0.663	0.201	-0.999
Random Forest (ExtraTrees)	0.000	0.387	1.000	0.750	**0.719**	0.384	-0.826
Random Forest (LR)	0.002	0.734	0.998	0.516	0.503	0.207	-1.020
Random Forest (Expansion2)	0.000	0.516	1.000	0.605	0.599	0.265	-0.961
Random Forest (ExtraTrees + Expansion2)	0.000	0.540	1.000	0.629	0.610	0.276	-0.916
Logistic (With scaler + CV)	0.839	0.927	0.518	0.435	0.446	0.529	-0.875
Extra Trees (With scaler)	0.000	0.500	1.000	0.621	0.591	0.336	-0.967
Gradient Boosting (With scaler)	0.369	0.710	0.758	0.532	0.541	0.064	-1.078
Gradient Boosting (Without scaler)	0.369	2.226	0.758	0.532	0.038	0.000	-3.000
Gradient Boosting (ExtraTrees)	0.413	0.565	0.716	0.621	**0.625**	0.203	-0.907
Gradient Boosting (SelectPercentile)	0.464	0.645	0.675	0.581	0.593	0.282	-1.092
Gradient Boosting (Expansion2)	0.288	0.750	0.819	0.516	0.506	0.120	-1.218
Gradient Boosting (ExtraTrees + Expansion2)	0.381	0.815	0.780	0.508	0.510	0.152	-1.209
Naive Bayes (With scaler) - Baseline	1.319	1.177	0.385	0.395	0.302	0.380	0.788
LDA (With scaler)	0.516	0.605	0.484	0.395	0.425	0.289	-1.045

QDA (With scaler)	0.673	0.790	0.327	0.210	0.199	-1.288	0.412
CART (With scaler)	0.000	0.411	0.161	0.129	0.595	0.140	-1.066
Bagging (With scaler)	0.022	0.452	0.978	0.548	0.551	0.168	-0.826
AdaBoost (With scaler)	0.524	0.597	0.476	0.403	0.369	0.418	-0.413
KNN (K=7)	0.417	0.540	0.583	0.460	0.431	0.640	-0.413