Predicting Music Track Peak Chart Positions: Enhancing Robustness through a Two-Stage Classifier

Beyonce Obazee
Dublin City University
Dublin, Ireland
beyonce.obazee2@mail.dcu.ie

Darragh McGonigle
Dublin City University
Dublin, Ireland
darragh.mcgonigle3@mail.dcu.ie

Mark Reid
Dublin City University
Dublin, Ireland
mark.reid28@mail.dcu.ie

Mohammadou Boubakary Wadjiri
Dublin City University
Dublin, Ireland
mohammadou.boubakarywadjiri2@mail.dcu.ie

***Abstract*—**

***Index Terms*—machine learning, music, spotify, classification, charts, billboard**

I. Introduction

Music is a massive industry across the world and it carries not just a high cultural impact but an economic one as well. A report in 2020 from the Recording Industry Association of America (RIAA) [1] shows that in 2019 revenues from U.S. recorded music totaled $7.36 billion. This shows a $2.5 billion increase from 4 years prior. The majority of this growth can be attributed to the continued rise in digital streaming platforms like Spotify, Apple Music, etc. [2].

With such large economic incentives at stake, research in the field of hit song prediction aims to create accurate models that can predict songs with a high likelihood of achieving widespread popularity. This has the potential to assist in guiding factors like digital streaming platform (DSP) recommendations, record label marketing spend and the musical direction of artists [3] [4].

Over the years researchers have considered many different facets of this problem, such as, what demographic, socioeconomic and acoustic factors may influence a song's success [5]. Or how to quantify success and the various advantages and disadvantages of different measures [3]. In doing so, researchers have also applied a wide range of machine learning (ML) and deep learning (DL) techniques to the problem in an attempt to maximize predictive performance.

An area that is often ignored in the current research, is the inherent problems with the data itself. Namely, the stark class imbalance between popular and unpopular songs, and the diversity of different tracks that often manifest as outliers in the data. Both of these factors stand to play a huge part in the efficacy of any classifier, and the volume of new music being published on digital streaming platforms (DSP) only serves to exacerbate these issues.

This work seeks to address these challenges by proposing a novel approach to the prediction of a track's peak rank in the Billboard Hot 100 charts [6]. By leveraging a two-stage classifier, this research aims to enhance the robustness of predictive models, thereby mitigating the impact of outliers. Additionally, we will explore the effects of data imbalance on a wide range of machine learning models and examine the effectiveness of common balancing techniques when applied to this domain. Through this, we aim to shine a light on what we view as a fundamental challenge to the problem at hand and to help improve the accuracy of models for music popularity prediction going forward.

The paper will be structured as follows. It will begin with an analytical evaluation of existing works in the field, highlighting the current state of the research and identifying gaps. Next, we will then give a detailed look at our methodology and justifications for key decisions. Following this will be an evaluation of our experimental results. Finally, we will wrap up with a conclusion of our thoughts and recommendations for future work.

II. Related Work

The field of hit song prediction is often framed as the prediction of a song's popularity. Though due to the abstract and immeasurable nature of the concept of popularity, much work has been done in defining proxy measurements for a song's success.

This topic is explored in detail in Lee & Lee's work [3], in which they put forward 8 different measures of popularity based on a song's performance in the Billboard Hot 100 charts. They detail the distributions of each of the metrics and explore what it means for songs to perform highly in one metric vs another. They concluded their work by extracting acoustic features from charting songs and training Support Vector Machines (SVM) to perform binary classification on whether or not a song would be above the mean in a given metric. Though their analysis of each metric is well thought out, their use of only one classifier leaves room for doubt as to whether other models could have found better results. Additionally, as they only analyzed charting songs it is hard to know how well their approach would perform on a wider subset of music.

A more generalized approach to popularity measurement can be seen in the work of Shulman et al. [7]. Their approach involves using social media to track the adoption and spread of various 'items' such as songs, photos and books. Their method involves 'peeking' at an item's performance during its first k weeks and then predicting if the item will have above or below the median social media engagement after n days. Their findings interestingly show that 'temporal' features, such as looking at the early adoption rate, have the highest impact on a model's performance. This is substantiated by Lee & Lee [3] who showed that a song's debut chart rank is correlated with its peak chart rank. Additionally, Shulman et al. [7] also found that these temporal features were the only ones that generalized cross-domain.

Many researchers in hit song prediction have taken similar approaches and looked to social media platforms like Last.FM to create features based on social context. An example of this is the work of Ren & Kauffman [5] who identified 3 categories of features; music semantics, social context and artist reputation. They also devised two popularity metrics, how long it took from release for a song to reach the charts (Time2TopRank) and the length of time the song remained in the charts (Duration). They attempted to predict Duration using Support Vector Regression, Bagging and Random Forest (RF) with RF performing the best in this task. Following this they go on to state 6 popularity patterns that they observed in their data based on the 2 prior metrics. They went on to test 3 classifier models of which RF performed the best again. This paper does a great job of exploring the diverse range of features that may impact hit song prediction. Furthermore, its evaluation of how Synthetic Minority Oversampling Technique (SMOTE) [8] can help boost the model's performance on under-represented classes is well presented. The major problem we see in this paper is it does not address songs that do not reach the charts as all 6 of its patterns involve charting.

In line with the work of Ren & Kauffman [5], Kim et al. [9] used the number of tweets about a song in Twitter hashtags to predict a song's popularity. To begin with, they trained regression models to predict a song's peak Billboard chart rank. Following this they used an RF to perform binary classification on whether or not a song was a hit based on its peak rank. With 'hit' being defined as being within the top n positions of the chart for varying n values. Like Ren & Kauffman [5]; they too account for the class imbalance, but opt for random undersampling. Their definition of 'hit' is arguable as a song that came 11th on the charts would be deemed a non-hit. This definition of 'hit' is also seen in the works, namely, Bischoff et al. [10] and Herremans et al. [11]. The former of which also used social media-derived features, in this case from LastFM. It also used minority undersampling to address the class imbalance and experimented with even narrower ranges for its 'hit' definition, at the most extreme being any charting song that wasn't number 1 was deemed a 'non-hit'. Both Bischoff et al. [10] and Kim et al. [9] lacked consideration of acoustic features and relied entirely on social media for their feature sets which seems unwise as other papers have shown that acoustic features can aid in the prediction of hit songs [5] [11].

As previously mentioned, Herremans et al. [11] took the same range-based approach to defining hits as Bischoff et al. [10] and Kim et al. [9]. However, in their case, they used entirely acoustic features and a wider range of classifiers to make their predictions. Of note, they did acknowledge an imbalance in their dataset but opted not to correct it. This likely limited the performance of their classifier as demonstrated by Ren & Kauffman [5] who, as mentioned, showed an increase in minority class prediction accuracy using SMOTE.

Interiano et al.12 defined popularity as simply having appeared in the Official UK charts. Their work exploring 1/2 a million songs released in the UK between 1985 to 2015, does a great job highlighting trends in successful music across the years. Of note, is the conclusion that adding a feature denoting whether or not the artist has had recent success has a large impact on classification accuracy.

Araujo et al. [4] also opted to denote popularity as having appeared in the charts, in their case they used Spotify's Global Top 50. They gathered 180 weeks of chart data and used lagged features to build a model that aimed to predict if a song would be on the charts in n days based on acoustic features. Though they achieved good results, their deliberate exclusion of time series models for what is admittedly a time series problem leaves room for doubt as to whether their lagged feature approach is optimal . Additionally, it is not clear how well this model will generalize as it only used songs that had appeared in charts at least once.

A true time-series approach to the problem can be found in the work of Karydis et al. [13] In which they discuss the creation of a broad dataset for music popularity prediction. In it, they utilize a Non-linear Auto Regressive (NAR) model and a Non-linear Auto Regressive with eXternal input (NARX) model to make time-series-based predictions using their proposed dataset. Interestingly given the large size of their feature set they experimented with using Principal Component Analysis (PCA) to reduce the feature set and this increased their accuracy significantly. The paper does a great job of explaining its decision making and its examination of other paper's approaches is very in-depth and insightful.

A different approach to feature selection was employed by Khan et al. [14] namely filter methods. With this, they were able to reduce their feature set from 13 down to 8 without any loss of predictive performance. However, their broad removal of all outliers and the extremely high results relative to other papers casts doubt on the effectiveness of their approach overall.

Zhang [15] took a different approach to representing music for machine learning, in their paper on genre classification they used a convolutional neural network (CNN) to classify a track into 1 of 10 genres based on an image of the waveform of a song. This approach was expanded upon by Yang et al. [16] by utilizing a traditional CNN in combination with JYnet which is a pre-trained music tagging system to predict a song's play counts. This concept of using a CNN to extract features from the song's waveforms is a very unique and modern approach to the problem. However, they only compared the model against Linear Regression, the inclusion of more complex or non-linear models would have helped contextualize the model's performance.

III. Methodology

A. Dataset Sources

1) Spotify Dataset: Spotify is a digital streaming platform launched in 2008, since then they have grown to become the largest subscription-based digital music platform with over 100 million songs and 602 million registered users [17]. Spotify makes their data on songs public for others to use through APIs. For our research, we will be using data from their track's audio features API [20] which gives us traditional acoustic features of a song such as tempo, key and time signature, Along with, more advanced acoustic features created using proprietary algorithms like danceability, valence and energy. We will not be making use of the API directly but instead relying on an accumulated dataset of over 1 million different song's acoustic features [18].

2) BillBoard Hot 100 Dataset: The Billboard Hot 100 chart is one of the most popular U.S-based music charts. It has been in operation since August of 1958, publishing the top 100 songs from the past week. To create this list they aggregate data from physical and digital sales, radio play and digital streaming to rank the songs. Again, we are not accessing this data directly, but instead, we are using a pre-compiled dataset [19] that contains chart-related data for each song occurring in each week of the charts such as; what week the song appeared in; how many weeks that song has been in the charts up until that point; and what is the peak rank the song had achieved in the charts up until that point. Additionally, the dataset has Spotify acoustic features for some of the songs, which will be used to help conflate songs and fill in missing songs from the Spotify dataset.

B. Data Preparation

1) Data Integration: To create our dataset we started by integrating the Billboard dataset and Spotify dataset. To do this we conflated the two datasets using the ”track id” value assigned to each song by Spotify. For instances where this ID was not present in the Billboard dataset, we relied on the artist name and track name to identify the songs in each dataset. If the song was not found in the Spotify dataset but did contain sufficient Spotify data in the Billboard dataset, the song was added to create a more complete dataset.

2) Data Cleaning: Examining the dataset we identified 3 key exclusion criteria for tracks in the dataset. The first was the removal of any track with the genre of 'sleep', as this genre accounted for a large percentage of outliers in the dataset and is comprised of white noise and other sleep aids which for our model are not being considered as songs.

The second was the removal of any song with a tempo of 0 beats per minute (BPM) as this is not a possible tempo value for a song and is most likely just an error in Spotify's tempo estimation algorithm. [20] We opted not to impute the erroneous tempo values, as tempo affects other acoustic features like danceability. However, its level of influence as well as exactly what features it influences are not known due to the proprietary nature of the algorithms used to create the complex features.

Finally, we removed all songs released after 2020 as this was the last complete year from the Billboard dataset and failing to do so would have led to songs that charted post 2020 being labeled as uncharted which could have a negative impact on model performance.

3) Categorical Encoding: Within our dataset, we identified 3 categorical features, those being genre, key, and mode. The key was already encoded using integer pitch class notation which we opted to leave unchanged. Mode, which represents whether or not a song is in a major or minor key was a binary value and thus no encoding was needed. This left genre, which contained 81 unique classes. We identified 4 potential encoding strategies and tested each by building a logistic regression model and measuring accuracy, Akaike information criterion (AIC) and Bayesian Information Criterion (BIC). The first method was One-Hot encoding, with this method the logistic regression model failed to converge. We believe this is due to the increase in dimensionality brought on by adding 81 sparse columns. The second method was label encoding, which performed the worst of the tested encoding methods. This is likely because of this method's poor suitability for non-ordinal data, as by nature it implies a category can be greater than another (e.g. rock ¡ jazz) which doesn't make sense in non-ordinal scenarios. The third approach was count encoding, where a portion of data (5% in this case) is withheld from the test and train dataset and the labels are replaced with the occurrence of each label in the withheld portion of data. Finally, target encoding which is similar to count encoding, in that a portion of the data is withheld. But in this case, the label is replaced with the probability of the target given the label.

Of the tested methods, target encoding performed the best across all tested metrics. The full results can be seen in Table I.

C. Data Analysis

1) Analysis of Distributions: By looking a distributions in the data between the subset of songs that have charted and the subset that have not charted, we can get an understanding of the characteristics that may help differentiate songs that reach the charts and those that do not. Figure 1 shows the

TABLE I
GENRE ENCODING RESULTS

Encoding Method	Accuracy	AIC	BIC
One-Hot Encoding	n/a[a]	n/a[a]	n/a[a]
Label Encoding	0.74	14089	14194
Count Encoding	0.72	13576	13680
Target Encoding	**0.84**	**9318**	**9423**

[a]One-Hot Encoding failed to converge due to singular matrix.

distributions for 6 acoustic features in the dataset which highlight certain aspects of charting music. For instance, we can see that the subset of charted songs skews higher in terms of 'danceability' and 'valence' which indicates that they tend to be happier and easier to dance to. These two traits would suggest that charted songs would have a higher 'energy' and that does appear to be true as the 'energy' distribution is skewed further right for charted songs. However, there is a steep drop off for extremely high 'energy' songs (>0.9) in the charted distribution, despite that being the most represented group in non-charted songs. This illustrates that songs with extreme energy values (<0.2 or >0.9) have a lower representation in the charts.

Another interesting trend can be seen in 'instrumentalness' which in both categories has the vast majority of songs with values below 0.1. However, in the non-charted case, we can a higher representation of the other values especially towards the higher end. This is in stark contrast to charted songs which are composed of almost exclusively songs with below 0.1 'instumentalness' except for a few outliers. This trend can also be seen in 'acousticness' but to a lesser extent.

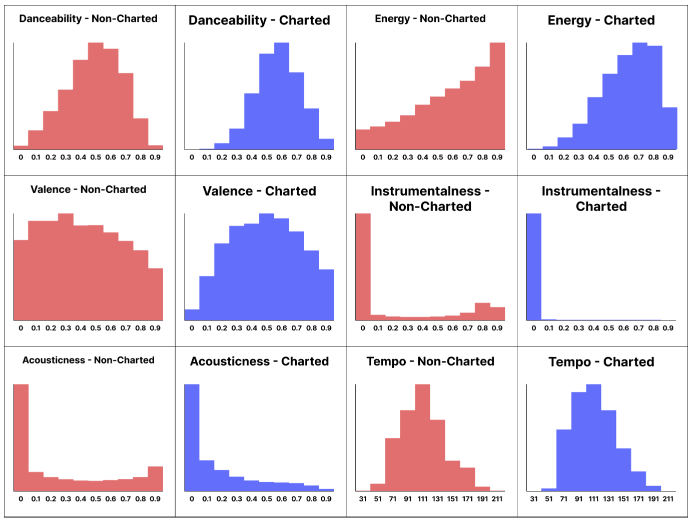

Fig. 1. Charted verse Non-Charted distributions for acoustic features

2) Outliers and Anomalies: The dataset contains a large number of outliers, as is to be expected with the diverse range of songs present in the dataset. Features like 'duration' and 'liveness' are particularly outlier heavy as one may expect due to Spotify containing a fair number of live performances, movie soundtracks, and short album interludes. Even ignoring these fringe cases, outliers can be seen even among songs that reached the charts. This highlights how diversity will be present in any subset of music and the models used in this domain must account for that fact and must be designed in a way that is robust to outliers.

3) Correlation Analysis: Looking at correlation in our dataset we see some high correlation between certain features. For instance, 'energy' and 'loudness' are positively correlated with a correlation factor of 0.79. 'Acousticness' has a high negative correlation with both 'energy' and 'loudness' (-0.74 & -0.61 respectively). Finally, 'danceability' has a positive correlation factor of 0.5 with 'valence'.

To ensure this degree of correlation was not a sign of high multicollinearity in our dataset we calculated the variance inflation factor (VIF) for each feature. We found there were some relatively high VIF scores for 'energy' with 4.68 and 'loudness' with 3.51. However, both were below the 10 limit which would indicate problems. To verify, we trained a logistic regression model with and without each of these values and saw an increase in both AIC and BIC with their exclusion. For these reasons, we opted to keep both of these features in our dataset.

4) Data Imbalance: Another characteristic of the dataset is the data imbalance, not just between the binary classes of charted and non-charted but also between the peak ranks of charted songs. In the binary case, the ratio is 148 non-charted songs for every charted song. With roughly 1 million non-charted songs and nearly 7000 charted songs.

In terms of peak ranks among charting songs, the most common peak rank is 1st with 203 songs having this value and the least common is 74th with only 45 songs peaking at that rank.

Addressing this imbalance is at the core of this work and is part of the motivation behind the two-stage approach. Instead of having 101 classes with 45 members of the smallest class and a million members of the largest. By breaking the problem into two distinct stages we have a much more manageable balance of 7000 to 1 million and 45 to 203.

Breaking the problem into two stages is still not enough to fully correct for the imbalance so in addition we will be exploring 4 different class balancing techniques.

The first is random undersampling, where a random sample of the majority class is taken equal in size to the the minority class. This is the approach taken in the works of Kim et al. [9]and Interiano et al. [12].

The second approach is a variation on random undersampling called near-miss undersampling. With this approach, a sample equal to the number in the minority class is still taken, but the key difference is the data points are not picked at random but instead picked based on their average distance to points in the minority class. This leads to the points most similar to the minority class being selected from the majority class.

The third approach is random oversampling, with this approach random members of the minority class are replicated to increase the size of the minority class. In our case, for the binary classification, it would be too extreme to replicate each member 148 times to achieve balance. So instead, we have opted to use random oversampling in conjunction with random undersampling to double the minority class size and reduce the majority class size to match.

Finally, SMOTE increases the class size of the minority class by creating synthetic data points between the existing class members. In a similar manner to random oversampling, for our purposes, we will be using it in conjunction with random undersampling to avoid over-inflating the minority class size.

D. Model Architecture

The model we are proposing is a novel two-stage classifier, the first stage is a binary classifier that discriminates between charting and non-charting songs and and second is a multi-class classifier that classifies charting songs based on their peak rank achieved in the Billboard charts. See Figure 2.

We hypothesize that with this approach we will have two key advantages over a traditional single-stage classifier. This first, as mentioned previously, is it helps with class balance by considering all charted songs as one class for the first stage. This will still be an imbalanced problem, but far less than if we were considering the full 101 different classes at once. The second advantage of this approach is robustness, as the dataset contains a lot of noise, by employing a robust binary classifier to filter out non-charting data, our multi-class classifier can concern itself solely with the less noisy charted data. This should hopefully lead to more accurate predictions overall.

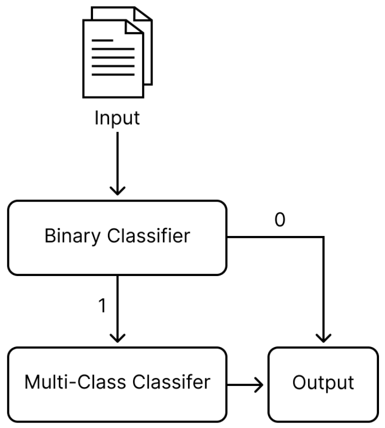

Fig. 2. High level two-stage model overview

1) Binary Classifier: For our binary classifier, we will experiment with 6 different models, Logistic Regression (Logit), Support Vector Machine (SVM), Random Forest (RF), eXtreme Gradient Boosting (XGBoost), Voting Ensemble and Stacking Ensemble. Both ensemble models will use Logit, Random Forest and XGBoost as base classifiers, as this was the optimal combination found through our testing. The voting classifier takes the output of the base models and selects a final output using a soft voting strategy that accounts for each model's predicted probability. The stacking classifier is similar, but instead of voting, the base models' output is passed to a final estimator (Logit in our case) which makes the final classification. See Figure 3.

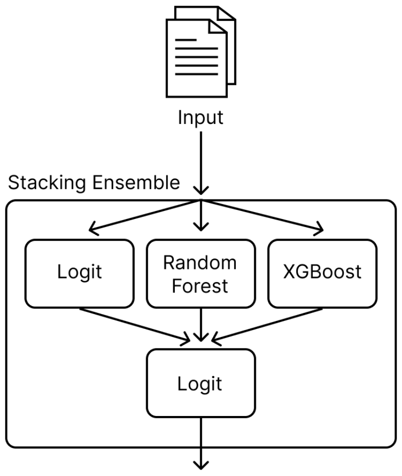

Fig. 3. Stacking Ensemble Binary Classifier

In addition, each model will be tested using 5 different data balancing techniques, random undersampling, near-miss undersampling, random oversampling and SMOTE. As mentioned previously in the data imbalance section, both oversampling techniques will be used to double the minority class size and then random undersampling will reduce the majority class to match the newly enlarged minority class. Other ratios of synthetic/replicated data in the minority class were explored but 1:1 proved the most effective.

2) Ranking Classifier:

IV. Evaluation

A. Binary Classifier

For the evaluation of the 6 different binary class models, it was important that we first tuned each of their hyperparameters.

For Logit and SVM this tuning was done manually using 10-fold cross-validation with random undersampling to test each model's accuracy. With Logit we experimented with varying the solvers and penalties but found many combinations failed to converge. Ultimately, we found the Newton-Cholesky solver with l2 penalties to be the best-performing and most consistent setup. Similarly, for SVM we experimented with varying kernel types and found a linear kernel with l2 penalty to be the best-performing parameters.

Due to Random Forest and XGBoost having more complex parameters, we opted to use GridSearchCV to explore different combinations of hyperparameters. For Random Forest we explored different numbers of estimators, max depth, minimum sample splits and max features. When tuning XGBoost we explored different numbers of estimators, max depth and learning rates. Both models showed a marginal increase in performance when using the tuned hyperparameters.

For the voting and stacking ensembles, we used the optimal hyperparameters found in the previous sections for each of the base models. Initially, we tried a 'hard' voting strategy with the voting ensemble and all 4 base models. We explored different base model compositions and found that dropping SVM had no negative effect on the model's performance. Additionally, we saw a slight performance increase using a soft voting strategy where the predicted probabilities are weighted into the voting step. The stacking ensemble was configured similarly, again SVM proved to not be beneficial to the model's performance. Additionally, we found no significant difference in performance across different final estimator models. We ultimately used Logit for the final estimator as it was less complex than the other 3 and offered the same performance.

Our evaluation was done using 10-fold cross-validation, we performed this 3 times for each model-balancing method pairing and then averaged out the results. These results can be seen in Table II & Table III.

TABLE II
BINARY CLASSIFICATION ACCURACY

Model	RU[a]	NMU[b]	RO[c]	SMOTE
Logit	0.84	0.79	0.85	0.87
SVM	0.84	0.83	0.84	0.86
RF	0.89	0.85	0.89	0.91
XGBoost	0.89	0.86	0.89	0.90
Voting	0.89	0.86	0.89	0.91
Stacking	0.89	0.86	0.87	0.91

[a]Random Undersampling
[b]Near-Miss Undersampling
[b]Random Oversampling

TABLE III
BINARY CLASSIFICATION F1 SCORE

Model	RU[a]		NMU[b]		RO[c]		SMOTE	
Class	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**
Logit	0.83	0.85	0.80	0.79	0.84	0.86	0.86	0.88
SVM	0.82	0.85	0.83	0.82	0.82	0.85	0.85	0.87
RF	0.89	0.89	0.86	0.83	0.89	0.89	0.91	0.91
XGBoost	0.90	0.89	0.87	0.85	0.89	0.89	0.91	0.91
Voting	0.89	0.89	0.87	0.86	0.89	0.89	0.91	0.91
Stacking	0.89	0.89	0.87	0.85	0.87	0.88	0.91	0.91

[a]Random Undersampling
[b]Near-Miss Undersampling
[b]Random Oversampling

Our testing shows a clear divide with XGBoost, Random Forest, Voting Ensemble and Stacking Ensemble performing significantly better than Logistic Regression and SVM. However, among these 4 models there does not appear to be a significant difference in performance as each performed nearly identically across all of the tested balancing methods.

As for balancing methods, we opted to exclude the imbalanced dataset testing results from both tables as the results were identical across all models. Every model learned to solely predict the majority class leading to 0.99 accuracy and 0 F1-score for the positive class (Charted) and 1 F1-score for the negative class (Non-Charted). Between the remaining 4 balancing strategies SMOTE proved to be significantly better than the others, as it had the highest accuracy for every model.

From this, we have opted to use Random Forest and SMOTE as our first stage classifier. Random Forest tied for the highest accuracy with Voting Ensemble and Stacking Ensemble. However, as Random Forest was used as a base classifier in both approaches, by using the standalone model as opposed to the ensembles we reduce the complexity of our model while still maintaining optimal performance.

B. Ranking Classifier

C. Overall Evaluation

V. CONCLUSION AND FUTURE WORK

REFERENCES

[1] R. Stoner and J. Dutra, Recording Industry Association of America (RIAA), Washington, D.C., 2020, pp.10.

[2] "U.S. Music Revenue Database," RIAA, https://www.riaa.com/u-s-sales-database/.

[3] J. Lee and J.-S. Lee, "Music popularity: Metrics, characteristics, and audio-based prediction," IEEE Transactions on Multimedia, vol. 20, no. 11, pp. 3173–3182, Mar. 2018. doi:10.1109/tmm.2018.2820903

[4] C. V. Soares Araujo, M. A. Pinheiro de Cristo, and R. Giusti, "Predicting music popularity using music charts," 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), Dec. 2019. doi:10.1109/icmla.2019.00149

[5] J. Ren and R. J. Kauffman, "25th European Conference on Information Systems ECIS," in Understanding music track popularity in a social network, 2017, pp. 374–388

[6] "Billboard hot 100," Billboard, https://www.billboard.com/charts/hot-100/.

[7] B. Shulman, A. Sharma, and D. Cosley, "Predictability of popularity: Gaps between prediction and understanding," Proceedings of the International AAAI Conference on Web and Social Media, vol. 10, no. 1, pp. 348–357, Aug. 2021. doi:10.1609/icwsm.v10i1.14748

[8] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," Journal of Artificial Intelligence Research, vol. 16, pp. 321–357, Jun. 2002. doi:10.1613/jair.953

[9] Y. Kim, B. Suh, and K. Lee, "#nowplaying the future Billboard," Proceedings of the first international workshop on Social media retrieval and analysis, Jul. 2014. doi:10.1145/2632188.2632206

[10] K. Bischoff, C. S. Firan, M. Georgescu, W. Nejdl, and R. Paiu, "Social knowledge-driven music hit prediction," Advanced Data Mining and Applications, pp. 43–54, 2009. doi:10.1007/978-3-642-03348-3_8

[11] D. Herremans, D. Martens, and K. Sörensen, "Dance hit song prediction," Journal of New Music Research, vol. 43, no. 3, pp. 291–302, Jul. 2014. doi:10.1080/09298215.2014.881888

[12] M. Interiano et al., "Musical trends and predictability of success in contemporary songs in and out of the top charts," Royal Society Open Science, vol. 5, no. 5, p. 171274, May 2018. doi:10.1098/rsos.171274

[13] I. Karydis, A. Gkiokas, V. Katsouros, and L. Iliadis, "Musical Track Popularity Mining Dataset: Extension & Experimentation," Neurocomputing, vol. 280, pp. 76–85, Mar. 2018. doi:10.1016/j.neucom.2017.09.100

[14] F. Khan et al., "Effect of feature selection on the accuracy of music popularity classification using machine learning algorithms," Electronics, vol. 11, no. 21, p. 3518, Oct. 2022. doi:10.3390/electronics11213518

[15] J. Zhang, "Music feature extraction and classification algorithm based on Deep Learning," Scientific Programming, vol. 2021, pp. 1–9, May 2021. doi:10.1155/2021/1651560

[16] L.-C. Yang, S.-Y. Chou, J.-Y. Liu, Y.-H. Yang, and Y.-A. Chen, "Revisiting the problem of audio-based hit song prediction using Convolutional Neural Networks," 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Mar. 2017. doi:10.1109/icassp.2017.7952230

[17] "About Spotify," Spotify, https://newsroom.spotify.com/company-info/.

[18] A. Joshi, "Spotify_1million_tracks," Kaggle, https://www.kaggle.com/datasets/amitanshjoshi/spotify-1million-tracks.

[19] "Billboard Hot Weekly Charts," Kaggle, https://www.kaggle.com/datasets/thedevastator/billboard-hot-100-audio-features.

[20] “Get track’s audio features,” Web API Reference — Spotify for Developers, https://developer.spotify.com/documentation/web-api/reference/get-audio-features.