CMPE 351: Song Popularity Prediction

Oliver Austin - 20011458, Allison Christensen - 10211533 , Jenna Malone - 20005119,
Chris Masloub - 20052223, Matthieu Roux - 20013052

April 2021

Abstract - Spotify is a major streaming service with its track, audio, and artist data publicly available via its Web Application Programming Interface (API). The goal of this project is to use audio feature data and design a binary classification model for making predictions on new songs to determine if the song will be on the Billboard Top 100 charts. The result is intended to be used by artists to anticipate the popularity of their song before it is released. The feasibility of incorporating lyrical data was examined but determined to be out of the scope for a classification problem. This analysis uses the Python library, PyCaret, which facilitated the testing of a multitude of models in parallel, while cross-validating against different features and tuning hyper-parameters. Synthetic Minority Oversampling Technique (SMOTE) enhancement was performed on the data in order to extrapolate more data samples and equalize the number of popular and unpopular songs in the dataset. The optimal model derived from this process is an Extra Trees Classifier that obtained an accuracy of 90.82%, a precision of 88.59%, a recall of 93.91%, and a Matthew's Correlation Coefficient (MCC) of 0.8179. These results demonstrated a significant improvement to existing solutions; the baseline for comparison was extracted from [**4**] which obtained an accuracy of 85.11%, a precision of 96.50%, a recall of only 77.06% and an MCC of 0.7253. As future work, it is recommended to further expand the model to predict how many weeks a song will appear on the Billboard Hot 100 Hits to better assist artists in evaluating their new tracks.

Index Terms - Team 5, Music, Artists, Song Prediction, Popularity, Spotify, Top100, Extra Trees Classifier, Billboard Hits

1 Introduction

With over 345 million monthly active users, Spotify is one of the most widely adopted streaming services. There is a vast amount of data available from Spotify's Web API about artists, tracks, and playlists [**1**].

The motivation behind this project is to use Spotify track data to provide artists with a model that predicts whether or not a song is going to be popular, before it is released. Anticipating the popularity of a track is of value particularly to artists and their labels, as popular songs secure the bulk of the revenue in the music industry.

While this analysis focuses specifically on Spotify which has its own popularity metric, a binary classification metric derived from Billboard Hot 100 Hits data is used to ensure that the results are transferable and that the proposed model could be used across other streaming platforms.

Artists are expected to be able to supply their own audio values as inputs to the model. While this may

be easy to measure for features such as *key* and *tempo*, it may be difficult for artists to determine what value to assign to features such as *danceability* for their track. Note that *danceability* is one of the audio features that Spotify captures and the algorithm used to determine this feature is not publicly available. For an artist to utilize the model, they must first upload their song to Spotify. Upon uploading a song, Spotify will calculate the audio feature data, which can then be scraped from Spotify through their web API. After gathering the audio feature data of the song, this can be passed as an input to the model to predict whether the new song will make it to the Billboard Hot 100 Charts.

The data in this analysis will be run against multiple classification models, while cross-validating with different feature engineering techniques to determine the best possible model and parameters. The proposed solution will be the classifier that obtains the best results, compared to the baseline metrics defined in Section 2.1.

2 Related Work

Song popularity is widely regarded as a challenging problem that a number of researchers have attempted with varying approaches. Relevant research works were examined to better understand the problem background and to determine a baseline.

In an IEEE paper published in 2020, the authors propose a multimodal end-to-end deep learning architecture for music popularity prediction, creating a dataset comprised of text and audio attributes. Their architecture was composed of two main stages: a deep autoencoder and a deep neural network. Popularity is predicted from a regression perspective and the response variable is based on Spotify's internal measure of popularity. While the proposed architecture combined with the multi-modal feature vector obtains a high F1-score of 83%, there is a trade-off between the level of compression and the cost. Therefore in applications where technical resources are limited, a higher level of compression may be needed, which would yield a sub-optimal solution in terms of the mean absolute error [**2**].

A group of Stanford students took a different approach and performed an investigation to predict whether a song will become a Billboard Hot 100 hit over any time frame, based on its audio features. The paper covers various models such as Expectation Maximization (EM) clustering, Support Vector Machines, Decision Trees, and Neural Networks. The article treated this as a classification problem. Both accuracy and precision were recorded for the models as success metrics. Due in part to the small dataset used, the model exhibited significant overfitting [**3**].

The most successful of the approaches researched was published as an article in the Theoretical and Applied Informatics Magazine, RITA. The authors approach this as a classification problem to predict if a song will be in the platform's Top 50 charts. The audio features collected from Spotify in this study were binarized which served to generalize the model during training and allow the user to more easily provide inputs. Spotify does not disclose how it determines the metrics for all of their audio features therefore an artist cannot accurately estimate the float value for the features in their song ahead of time [**4**].

2.1 Success Metrics

The result of the latter study [**4**] is a more robust model that achieved the following results:

- Accuracy: 85.11%

- Precision: 96.50%
- Recall 77.06%
- MCC: 0.7253

While the accuracy of the model is high, there is room for improvement as it is demonstrated to erroneously predict positive instances as negative in 23% of the instances. The above metrics will be used as the baseline for comparison in the following analysis [**4**].

Note that the MCC in particular is of importance as it provides an effective measure of the quality of binary classification like that which is being performed in this analysis. While accuracy is sensitive to class imbalance and both precision and recall are asymmetric, MCC is perfectly symmetric so no class is more important than the other. Meaning if the positive and negative classes (on or off the billboard charts) were switched then the same MCC value would still be obtained. MCC also accounts for all four values of the confusion matrix meaning its value will be representative of whether both classes are predicted well even if one class is over or under represented in the dataset (see Equation 1 below).

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (1)$$

3 Dataset

For the project, three data files were used. First, we accessed Spotify audio features data that was already mined from Kaggle called Spotify Tracks DB containing 232,725 tracks. Additionally, a supplementary audio features file containing all of the same features as in the Kaggle dataset was used containing 28,493 tracks. Finally, a Billboard Hits file identifying songs that were on the Billboard Hot 100 Hits was used. Through the following process, 23,580 songs were matched from the Billboard Hits file to songs in a combined audio features file.

The audio feature files contained the following fields: *genre, artist_name, track_name, track_id, popularity, acousticness, danceability, duration, energy, instrument, key, liveness, loudness, mode, speechiness, tempo, time_signature*, and *valence* [**5**]. Certain features such as *track_id* and *artist_name* were dropped from the dataset, and the performance of the model was tested with different subsets of the other features. The features used within the Billboard Hits file for matching were the *Song* and *Performer* [**6**].

To prepare the dataset for input into the models, first, the audio feature files were combined. This involved, renaming the columns of the supplementary audio features file to allow for an easy appending of the two files. The genre of the supplementary audio features file needed to be adjusted as it previously contained a list of genres and the Kaggle data contained just one genre per track row. Next, the matching between the combined audio feature file and the Billboard Hits file was completed using the *artist_name* and *track_name* from the combined audio features file and the *Song* and *Performer* columns of the Billboard Hits file, producing the 23,580 tracks matched.

When further exploring the data, it was identified that within the original Kaggle audio features dataset, there were songs that were in the dataset in multiple different rows that were not an exact match. For example, the song Dynamite by Taio Cruz was present in three rows for the genre Dance, Pop, and Rap. However, only one of the rows was match-

ing with the Billboard Hits file, since the matching just assigned a 1 to the first instance of the popular song that it found. The initial models created had performed poorly when classifying unpopular songs. However, once these duplicates were removed, and all songs were properly classified in the training set, this error no longer occurred.

Initially, the idea was to use the Spotify popularity metric, a value between 0 and 100, as the output for the model. However, this would be a complex regression problem, and it was unclear at which threshold a song was to be considered popular. It was decided that the measure of song popularity would be if it had occurred on the Billboard Hot 100 Hits or not. A 1 was assigned to a song that appeared on the Billboard Hot 100 Hits charts, and a 0 was assigned otherwise. This method of measuring popularity creates a classification problem and allows a concrete understanding of the model output. For example, if a song is predicted to be popular, that means it is predicted that the song will make it onto the Billboard Hot 100 Hits chart.

3.0.1 Lyric Data Feasibility

After statically preprocessing the data, it became evident that it would be unfeasible to incorporate lyrics through sentiment analysis due to memory restrictions and limitations when scraping the lyrical data for songs in the dataset. Utilizing vectorized natural language processing (NLP) data as the features of a classification problem would require us to treat the vectorized words in themselves as a set of vectorized features. We would then need to reduce this set of features to a single feature which was ultimately determined to be outside the scope of this analysis.

3.0.2 Correlation

The correlation of the various fields in the data was investigated by computing and plotting a correlation matrix. This matrix was used to identify which features are most highly correlated with the prediction target (whether or not a song has landed on the Billboard Hot 100 Hits chart). The resulting correlation matrix is shown in Figure 1.

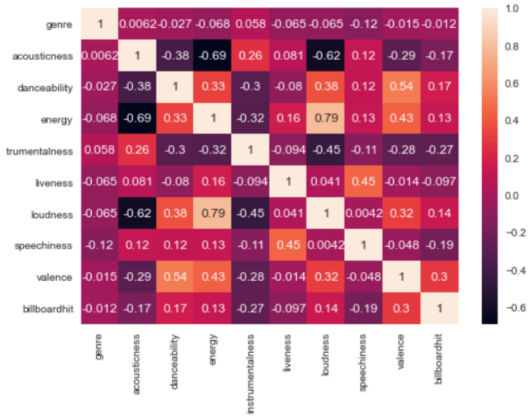

Figure 1: Audio feature correlation mix

The main column of interest in the above correlation matrix is the *billboardhit* column, which depicts the correlation of the various fields with the response variable. As can be observed from the correlation matrix, there are several fields which have a low correlation with *billboardhit*, namely *acousticness*, *valence*, and *loudness*. While these features did not correlate very well initially, they were later demonstrated to improve the performance of the proposed model.

4 Methodology

After reducing the feature list in the exploratory analysis of the dataset, it became apparent that it would be inefficient to design multiple models and tune parameters through trial and error in order to obtain the best result. As a solution, the PyCaret library was

used, which provides the ability to run data against many classification models at once with 10-fold cross-validation, as well as hyper parameter tuning. After running the models, PyCaret displays each model's performance metrics, including all of the baseline metrics defined in Section 2.1.

In addition to this library, the training data was cross-validated against a multitude of classification features and feature engineering techniques. This included:

- binarizing the numerical float data (as it ranged from 0.0 to 1.0), and
- running the models against the less correlated features to examine how they contributed to the overall model generalization.

It was noted in this analysis that the data in our set was very imbalanced given that only a limited number of tracks can be on the Billboard Hot 100 Hits charts. In response, SMOTE enhancement was performed on the data in order to extrapolate more data samples to equalize the number of popular and unpopular songs.

Using this feature engineering in conjunction with PyCaret's capabilities to validate a host of classification algorithms combined to determine the optimal model architecture for the nature of this problem.

5 Experiments and Results

5.1 Training and Validation Results

Multiple different preprocessing operations were performed to produce several dataset variations. With each unique dataset that was produced, multiple different models were trained and validated on the variation. Figure 2 shows the results of the training and validation with the dataset variation which achieved the highest accuracy and the accuracy metrics achieved on each of the many models trained.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
et	Extra Trees Classifier	0.8953	0.9358	0.9347	0.8668	0.8995	0.7906	0.7931	8.0080
rf	Random Forest Classifier	0.8899	0.9524	0.9310	0.8606	0.8944	0.7797	0.7823	9.6500
knn	K Neighbors Classifier	0.8856	0.9343	0.9814	0.8239	0.8958	0.7710	0.7856	4.8060
lightgbm	Light Gradient Boosting Machine	0.8615	0.9390	0.9207	0.8238	0.8695	0.7230	0.7281	0.7810
catboost	CatBoost Classifier	0.8592	0.9359	0.9207	0.8203	0.8676	0.7183	0.7237	6.4460
dt	Decision Tree Classifier	0.8406	0.8470	0.8456	0.8378	0.8417	0.6812	0.6812	1.8480
gbc	Gradient Boosting Classifier	0.8362	0.9164	0.8935	0.8022	0.8454	0.6723	0.6767	34.1970
ada	Ada Boost Classifier	0.8003	0.8870	0.8466	0.7757	0.8095	0.6006	0.6032	7.5090
lr	Logistic Regression	0.6651	0.7274	0.7425	0.6438	0.6897	0.3300	0.3339	1.5620
ridge	Ridge Classifier	0.6624	0.0000	0.7566	0.6374	0.6919	0.3244	0.3303	0.0870
lda	Linear Discriminant Analysis	0.6623	0.7264	0.7566	0.6374	0.6919	0.3244	0.3302	0.2690
qda	Quadratic Discriminant Analysis	0.6571	0.7645	0.9073	0.6054	0.7262	0.3135	0.3621	0.1120
svm	SVM - Linear Kernel	0.6554	0.0000	0.7077	0.6455	0.6696	0.3107	0.3205	1.2320
nb	Naive Bayes	0.6511	0.7161	0.8652	0.6064	0.7131	0.3014	0.3335	0.0930
xgboost	Extreme Gradient Boosting	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0780

Figure 2: Training and Validation Results

The accuracies shown in Figure 2 are the averages calculated using k-folds training and validation, with ten folds.

The Extra Trees Classifier achieved the highest accuracy, with 89.60%. Furthermore, when the data was being curated initially after the SMOTE data enhancement, approximately 15% of the dataset was left out to be used as the test set which has not been seen by the model. Testing on this portion of the dataset achieved an accuracy of approximately 91% which was a considerable improvement from the baseline. The best model was achieved using the extremely randomized trees classifier otherwise known as the Extra Trees Classifier.

Extra Trees is an ensemble learning technique which constructs a Random Forest using a number of decorrelated Decision Trees to output a classification. According to [**7**], "Each Decision Tree in the Extra Trees Forest is constructed from the original training sample. Then, at each test node, Each tree is provided with a random sample of k features from the feature-set from which each decision tree must select the best feature to split the data based on some math-

ematical criteria (typically the Gini Index). This random sample of features leads to the creation of multiple decorrelated decision trees."

5.2 Model Configuration

The Extra Trees Classifier, which achieved the highest model performance, used the parameters listed in Table 1.

Table 1: Model Parameters

Function to measure split quality	Gini impurity
Maximum depth	None
Number of estimators	100
Minimum samples to split	2
Features considered when looking for best split	$\sqrt{\text{Number of features}}$
Minimum samples at leaf	1

5.3 Testing Results

The highest accuracy was achieved using the dataset variation with the *valence* and *loudness* features, and non-binarized data. Figure 3 shows the confusion matrix of the model with the highest accuracy.

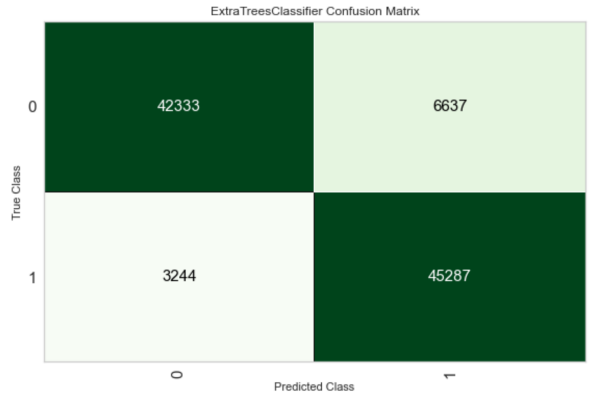

Figure 3: Testing Confusion Matrix

Note in Figure 3 above, the model had a much higher rate of false positives than false negatives, meaning the model was more likely to misclassify a song that did not make the Billboard Hot 100 Hits, than a song that did.

5.4 Baseline Comparison

The goal of outperforming the accuracy of the baseline model was achieved. The model results had an accuracy 5.71% higher than the baseline, with 7.91% lower precision, 16.85% higher recall, and a 0.0926 higher MCC. Outperforming the baseline model by 5.71% is a significant achievement, considering that the baseline model had a strong accuracy of 85.11%. The objective of this analysis was to outperform the MCC baseline and this was achieved.

6 Group Member Contributions

Group Member	Tasks
Oliver	Data Preprocessing Linear Regression Model Lyric Matching
Alli	Exploratory Data Analysis Technical Writing
Jenna	Data Preprocessing Billboard Hits Matching
Chris	Model Implementation Parameter Tuning
Matthieu	Lyric scraping and analysis Updating Build Files

7 Replication Package

To to view the code and instructions to replicate please refer to the git repository.

8 Overleaf Project

To view the shared overleaf project used to write this report see overleaf project.

9 Conclusion and Future Work

This analysis emphasized the value in the use of binarized feature data in conjunction with PyCaret's validation capabilities. The Extra Trees Classifier demonstrated results that are considerably better than those obtained in previous studies. Using a Billboard Hot 100 Hits based popularity metric, rather than Spotify's internally used popularity metric yielded a more robust prediction model that allows artists to input their own values for their songs to determine if it will be a hit.

Given more time, the prediction model could be improved and expanded to predict not only whether a song will appear on the Billboard Hot 100 Hits but also for how many weeks a song will appear. This offers a more exact measure of how truly popular a song is estimated to be and help artists to further fine tune the arrangement of their music. The model would also benefit from some increased transparency. Offering further insight into what audio characteristics are most often associated with popularity could further help artists in writing popular music.

Additionally, while lyrical analysis was out of the scope for this study, as future work it is recommended to implement word embedding to translate these high-dimensional vectors into a lower dimensional space. This type of automated feature extraction would ensure that words with the same meaning have similar representation. In this way, we may reduce the feature vector such that the use of lyrical data is more feasible and then repeat the experiments conducted in this analysis with this data included.

It should be noted that a number of socio-cultural elements may also have an impact on the potential popularity of a track and in future iterations of this analysis, it would be valuable to conduct more time series analysis to investigate trends and shifts in the importance of different features over time.

References

[1] "Spotify — Company Info," Spotify. https://newsroom.spotify.com/company-info/ (accessed Apr. 14, 2021).

[2] Martin-Gutierrez, D.; Hernandez Penaloza, G.; Belmonte-Hernandez, A.; Alvarez Garcia, F. A Multimodal End-to-End Deep Learning Architecture for Music Popularity Prediction. IEEE (Institute of Electrical and Electronics Engineers) access 2020, 8, 39361–39374. https://doi.org/10.1109/ACCESS.2020.2976033

[3] Georgieva et al. - HITPREDICT PREDICTING HIT SONGS USING SPOTIFY DATA.pdf." Accessed: Apr. 05, 2021. [Online]. Available: http://cs229.stanford.edu/proj2018/report/16.pdf.

[4] C. V. S. Araujo, M. A. P. Cristo, and R. Giusti, "A Model for Predicting Music Popularity on Streaming Platforms," vol. 27, 2020.

[5] "Spotify Tracks DB." https://kaggle.com/zaheenhamidani/ultimate-spotify-tracks-db (accessed Apr. 05, 2021).

[6] "Billboard Hot weekly charts - dataset by kcmillersean," data.world. https://data.world/kcmillersean/billboard-hot-100-1958-2017 (accessed Apr. 05, 2021).

[7] "Extra Tree Classifier for Feature Selection." https://www.geeksforgeeks.org/ml-extra-tree-classifier-for-feature-selection/ (accessed March 22, 20201)