Classic analysis on historical hittest songs in the USA via XGBoost

Shi Haosheng

May 6, 2022

1 Introduction

With the rapid development of the Internet and streaming media, the range of music people have access to has been largely expanded. On platforms like Spotify, we can easily find a track released even 50 years ago and instantly enjoy it. In this project, I investigate the historical hit songs and expect to design a simple method to identify those that are still popular on Spotify. In fact, these songs are often called classics because they are not buried in oblivion over time.

Many investigations have been carried out to research music popularity on music platforms. However, most works maintain their focus on the time when songs firstly came out, for example, (3) and (5) study which kind of songs is more likely to be packaged as hit songs, while (7) and (6) study how to predict weekly streams or other popularity measurements for songs on boards like Spotify Top200 Charts. To the best

of my knowledge, there are few works concerning prediction of the current popularity for historical hit songs.

The specific question to solve in this report is to decide whether a previous hit song is forgotten, normal or classical at present. I am also interested in which attributes take effect in classification. To crystallize the "hit songs", I consider those having entered the Billboard and find their current popularity on Spotify. I choose the efficient XGBoost for prediction and make an analysis on songs that are misclassified repeatedly. Data and code to reproduce the results are available in https://github.com/bluesun2019/popularity-of-historically-hit-songs/edit/main/README.md.

2 Data Description and Preprocessing

To identify the historical hittest songs, I find the US Billboard Hot100 ranking data (8) from 1970 to 2009. This dataset includes track names, artist names, years of first entry, genre clusters and track characteristics (harmony topics and timbre topics). To simplify the analysis, I pick out part of songs in every decade of last century, specifically in 1970-1972, 1980-1982, 1990-1992 and 1974-1976. The first three datasets are used for training and the last one for testing.

I use the R package "Spotifyr"[1] and "Billboard" to extract some new features for songs appearing in (8), such as loudness, speechiness, danceability of these songs, the artist popularity and the track popularity at present, whether the song hits the year

[1]Spotifyr requires an account in Spotify. For confidentiality I cannot disclose my account and keywords in the Github, so you may use your own account. The whole process to acquire data takes about 6 hours or more. You can also use the data I have downloaded in the "covering" data set.

board, the list of songs with duplicated names and so on. The popularity of a song is then determined as the highest one with the same name and from the same singer.

Most of the newly added characteristics are also commonly used accoustic features in hit song prediction projects like (5) and (4). However, I note that there has been a criticism of popularity prediction merely with these traditional musical properties (9). My preliminary EDA also supports this judgement (see Figure 1). Therefore some non-musical but easily available properties are also included as predictors:

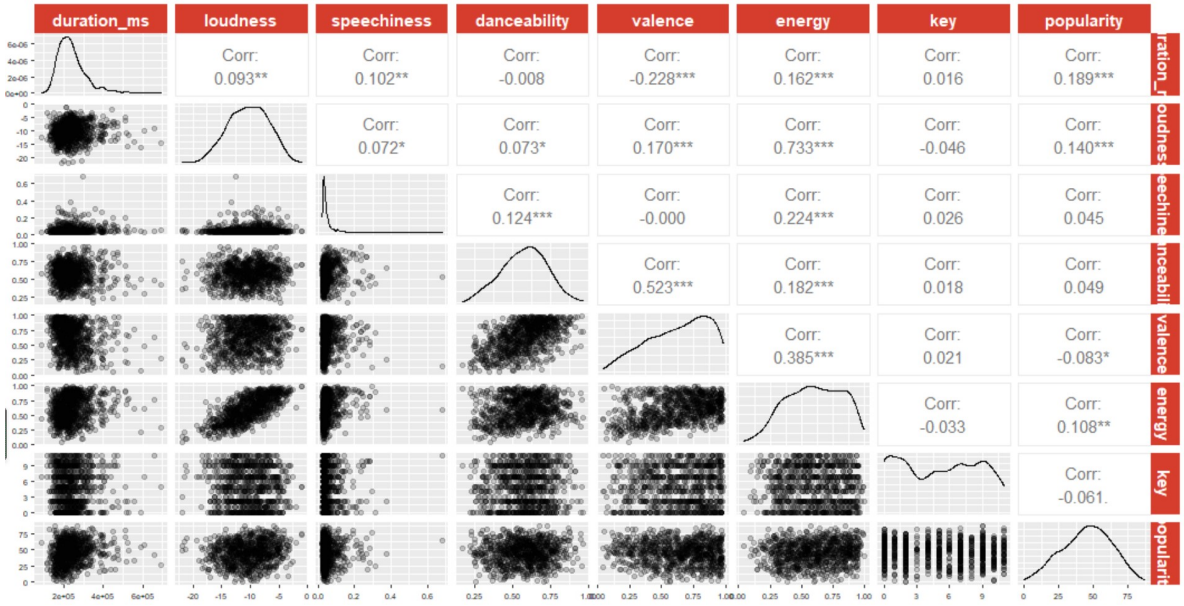


Figure 1: correlation plot of commonly used predictors in the used data

To better use the covering data and exploit the potential following pattern of these tracks, I regard the appearance of cover versions after the original hit song as an inhomogeneous Poisson process and estimate its intensity by a Haar wavelet-based Multiresolution analysis (10). I take the scale parameter $J = 2$ and acquire 4 intensity–related parameters for each original song.

I also conjecture that the track name complexity may have an effect because a

simpler one may be remembered more easily. Using character numbers in its name can be a simple but reasonable choice.

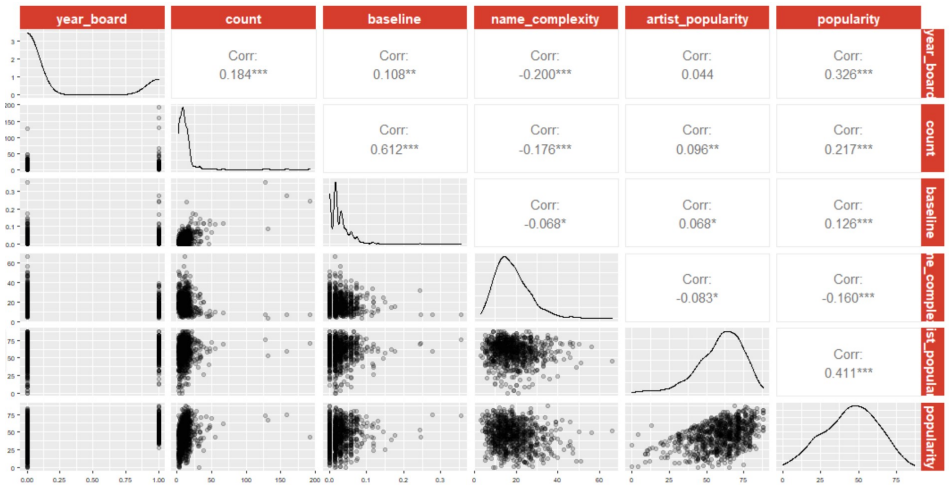


Figure 2: correlation plot of newly added predictors in the used data

Recall that this project aims to classify songs according to their classic levels. Now from the distribution of popularity (Figure 3), it seems a reasonable choice to divide the whole range into three intervals $[0, 30]$, $(30, 60]$ and $(60, 100]$ as a definition of the forgotten, ordinary and classic songs respectively.

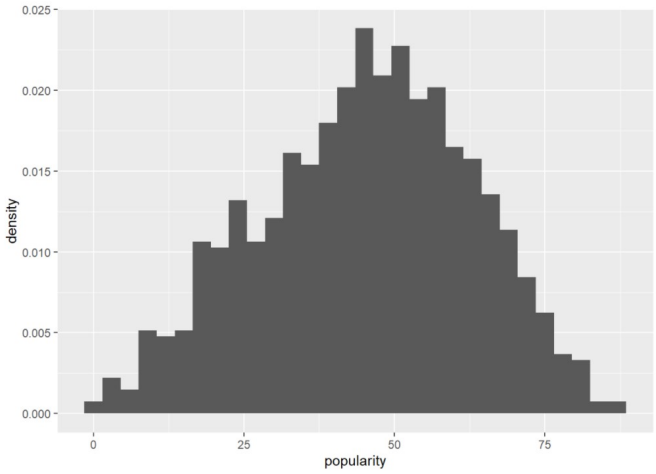


Figure 3: popularity distribution for the data

3 a Brief Introduction of XGBoost

XGBoost(eXtreme Gradient BOOSTing) is one of the most popular machine learning algorithms. As its name, this method seeks to bring the extreme learning efficiency of the boosting technique into play.

Suppose we have N data points $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$. Just like other members of the boosting family, XGBoost aims to find an integrated clssifier with the additive form $F(\boldsymbol{x}) = \sum_{m=1}^{M} f(\boldsymbol{x}; \alpha)$, where $f(x)$ is a simple and perhaps weak classifier parameterized by α. In this way, the process for XGBoost to learn the target classifier boils down to an optimization problem:

$$\min_{\{\beta_m, \gamma_m\}_1^M} \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{M} \beta_m f\left(\boldsymbol{x}_i; \alpha_m\right)\right)$$

L is a predetermined error function.

What set this algorithm apart are the techniques designed for overfitting prevention and efficiency orientation (2). Except for the normal error term $\sum_{i=1}^{n} L(y_i, F(\boldsymbol{x}_i))$, another penalization term $\sum_{k=1}^{m} \Omega(f) = \sum_{k=1}^{m} (\gamma T + \frac{1}{2}\lambda \|\boldsymbol{w}\|^2)$ is added to the goal function when classification or regression trees are used. Specifically, T is the number of leaf nodes in the classifier $f(\boldsymbol{x})$ and $\boldsymbol{w}$ is the score assigned to $f(\boldsymbol{x})$. To improve efficiency, XGBoost takes the second–order approximation of L for gradient descent instead of the commonly used first–order approximation:

$$\begin{aligned}\mathcal{L}^{(t)} &= \sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)) + \sum_{k=1}^{m} \Omega(f_t) \\ &\simeq \sum_{i=1}^{n} \left[L\left(y_i, \hat{y}_i^{(t-1)}\right) + g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i) \right] + \Omega(f_t)\end{aligned}$$

Using the approximation established above, we can easily update $\widehat{y}_i, g_i, h_i, f_t(\boldsymbol{x}_i)$ with stepwise regression. More algorithmic details are provided in (2).

Apart from the advantages discussed above, the authors also designed parallel, distributed and out-of-core computation which makes the learning process very fast. Based on these works, a package named xgboost has been developed and publicly available for R users to implement the algorithm.

4 Method: Main Challenges and Solutions

To explain the difficulties in classification, I combine the 1970-1972, 1980-1982, 1990-1992 datasets and divide them into training data (80%) and testing data (20%). For training parameters used in XGBoost, I set the learning rate $\eta = 0.3$, subsample ratio used in splitting the tree 0.7, features sampled in splitting the tree 0.4, number of trees $n = 2000$ (and a further early stopping criterion is set). Direct training and predicting based on the random division of datasets for 10 times leads to an unsatisfactory result (Figure 4).

The bad classification result motivates me to inspect the data again. I plot the number of songs in three classes (Figure 5a) to find two main challenges:

1. This is a highly unbalanced classification problem. Specifically, We care more

about which songs are in [0,30] and [60,100], while those in (30,60] take account almost half of the population. Then most of the data points will be classified into this category and lead to a poor precision behaviour.

2. Borders between classes are actually not clear in that there are many points on the boundary. For this reason, common criteria for judging the accuracy may not be suitable enough.

To overcome these challenges, we make an important assumption that we can get access to the full popularity information in the training data instead of its popularity class only. The assumption is practical because in reality, people are more likely to get the popularity value rather than the artificial classification label.

With this assumption, some techniques can be applied to improve the accuracy.

1. A weighted oversampling technique can be used on the training data. More detailedly, the focus should be laid on those with extreme popularity instead of those on the boundary, so I design two weights highlighting the extreme ones for oversampling. (Figure 6)

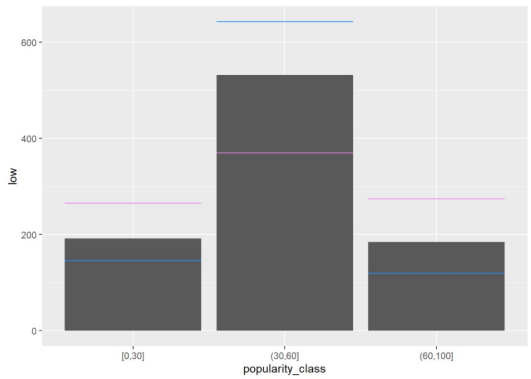
```
kable(indicator$table/10)
```

	[0,30]	(30,60]	(60,100]
[0,30]	12.3	7.9	0.8
(30,60]	24.8	90.5	25.2
(60,100]	0.9	7.6	11.0

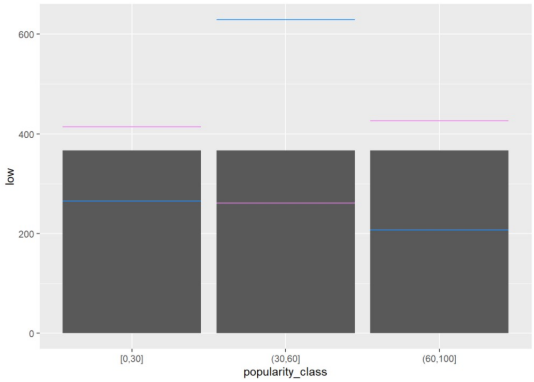
```
kable(t(indicator$byClass[,c(1,2,5,11)]))
```

	Class: [0,30]	Class: (30,60]	Class: (60,100]
Sensitivity	0.3236842	0.8537736	0.2972973
Specificity	0.9391608	0.3333333	0.9409722
Precision	0.5857143	0.6441281	0.5641026
Balanced Accuracy	0.6314225	0.5935535	0.6191348

Figure 4: Results of XGBoost prediction without oversampling



(a) Numbers of songs without oversampling; I also plotted the divisional levels of $[0, 25]$, $(25, 65]$, $(65, 100]$ in blue and $[0, 35]$, $(35, 55]$, $(55, 100]$ in red.



(b) Numbers of songs with simple oversampling; Songs in $(25, 30]$ and $(60, 65]$ have been replicated while those in $(30, 35]$ and $(50, 55]$ remain unchanged, which is unreasonable.

Figure 5: Number of songs in each class in the training data.

The probability for oversampling takes the form of piecewise functions like

$$w_1(x) = \begin{cases} a & 0 \le x \le 20 \\ 5a - \frac{a}{5}x & 20 < x \le 25 \\ 0 & 25 < x \le 65 \\ -13b + \frac{b}{5}x & 65 < x \le 70 \\ b & 70 < x \le 100 \end{cases} \quad w_2(x) = \begin{cases} c & 0 \le x \le 20 \\ 5c - \frac{c}{5}x & 20 < x \le 25 \\ 0 & 25 < x \le 35 \\ \frac{1}{10}x - \frac{7}{2} & 35 < x \le 40 \\ \frac{1}{2} & 40 < x \le 50 \\ -\frac{1}{10}x + \frac{11}{2} & 50 < x \le 55 \\ 0 & 55 < x \le 65 \\ -13d + \frac{d}{5}x & 65 < x \le 70 \\ d & 70 < x \le 100 \end{cases}$$

where a, b, c, d are undetermined parameters to balance the three classes.

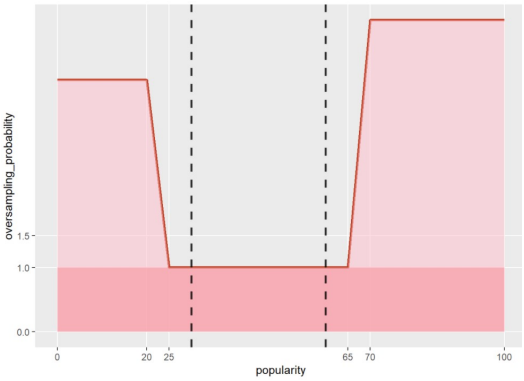
2. Another technique is to increase information in the phase of training. One choice is firstly classifying songs into 4 classes and then merging two classes of them into one. The other choice is firstly training a regression model and then classifying songs according to the regression results.

To match the strategy adopted in 1, I also define some new indicators to help evaluate the classification performance with a removal of border effects.

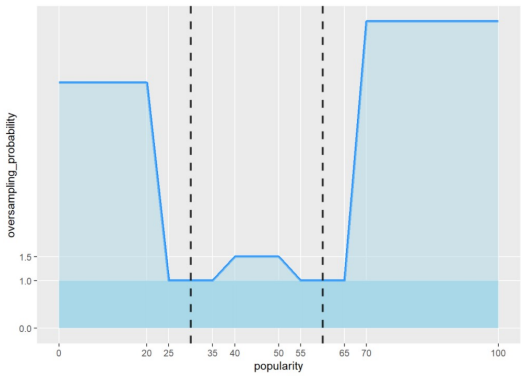
$$\begin{aligned}
\text{Extreme Sensitivity for } [0,30] &= \frac{\#\{\text{songs in } [0,20] \text{ predicted as}[0,30]\}}{\#\{\text{songs in } [0,20]\}} \\
\text{Extreme Sensitivity for } (60,100] &= \frac{\#\{\text{songs in } (70,100] \text{ predicted as}(60,100]\}}{\#\{\text{songs in } (70,100]\}} \\
\text{Extreme Error for } [0,30] &= \frac{\#\{\text{songs in } (60,100] \text{ predicted as } [0,30]\}}{\text{songs predicted as } [0,30]\}} \\
\text{Extreme Error for } (60,100] &= \frac{\#\{\text{songs in } [0,30] \text{ predicted as } (60,100]\}}{\#\{\text{songs predicted as } (60,100]\}} \\
\text{Adjusted Precision for } [0,30] &= \frac{\#\{\text{songs in } [0,45] \text{ predicted as}[0,30]\}}{\#\{\text{songs predicted as } [0,30]\}} \\
\text{Adjusted Precision for } (60,100] &= \frac{\#\{\text{songs in } (45,100] \text{ predicted as}(60,100]\}}{\#\{\text{songs predicted as } (60,100]\}}
\end{aligned}$$

Table 1 shows the experiment results of methods proposed above on the combined data. "Balanced 4 classes" means I directly balance four classes when training the data, while "adjusted 4 classes" means I balance three classes first and then divide the middle class into 2 subclasses for training.

From this table, I find that methods classifying songs into 4 classes and combining 2 of them perform the best overall. Although methods using regression lift the performance of adjusted precision and extreme error, they give a terribly poor performance on extreme sensitivity, indicating that the pattern of popularity may be disguised by noise and not yet learned. Besides, the weight 1 almost always performs better than the weight 2.



(a) Weight1



(b) Weight2

Figure 6: Two weights designed for oversampling probability.

Regions in deep colors means no oversampling parts, for example in 6(b), we reserve all observations with popularity 45 and oversample a further 50% from the original observations.

Table 1: Prediction performance

Indicators	3 classes		balanced 4 classes		adjusted 4 classes		Regression	
	Weight1	Weight2	Weight1	Weight2	Weight1	Weight2	Weight1	Weight2
Mean Balanced Accuracy	0.653	0.667	0.660	0.668	0.663	0.642	0.631	0.632
Mean Sensitivity	0.549	0.570	0.563	0.571	0.570	0.543	0.519	0.520
Mean Extreme Sensitivity	0.674	0.701	0.737	0.724	0.854	0.750	0.540	0.495
Mean Precision	0.529	0.549	0.526	0.537	0.507	0.484	0.586	0.581
Mean Adjusted Precision	0.739	0.778	0.747	0.747	0.710	0.716	0.843	0.823
Mean Extreme Error	0.070	0.051	0.052	0.058	0.073	0.075	0.008	0.010
Mean Specificty	0.756	0.764	0.757	0.763	0.756	0.742	0.743	0.743

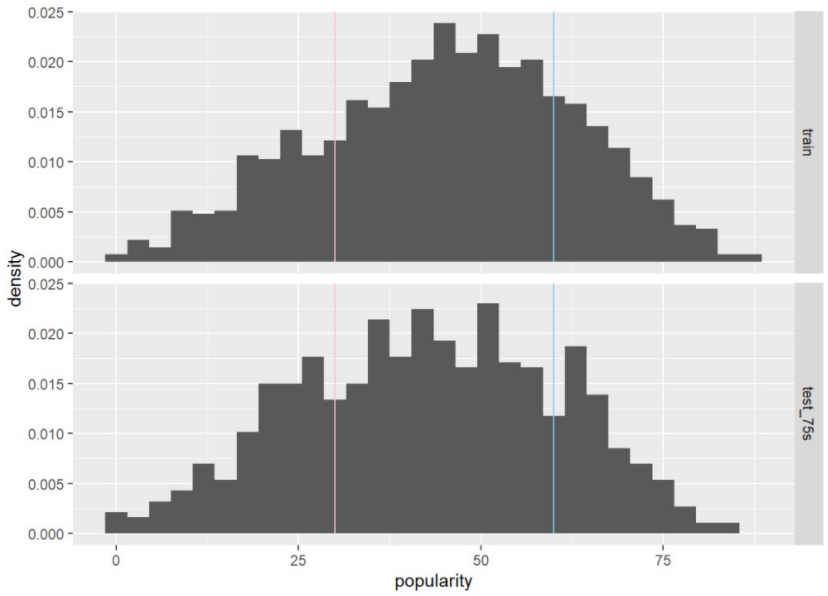


Figure 7: popularity distribution of training and testing data

5 Prediction and Interpretation

Another dataset including the 1974-1976 hit songs is introduced for predictive analysis. I use the whole combined dataset in Section 4 to train the model and predict the result on the new data. Figure 7 shows the popularity distribution of both datasets.

Adopt weight 1 and run XGBoost with balanced 4 classes and adjusted 4 classes for 50 times. The average classification performance is listed in Figure 8.

evaluation measures	Class: [0,30]	Class: (30,60]	Class: (60,100]
Balanced Accuracy	0.659	0.559	0.709
Sensitivity	0.477	0.504	0.645
Extreme Sensitivity	0.557		0.788
Precision	0.529	0.606	0.397
Adjusted Precision	0.801		0.71
Extreme Error	0.034		0.077
Specificty	0.841	0.615	0.774

(a) balanced 4 classes

evaluation measures	Class: [0,30]	Class: (30,60]	Class: (60,100]
Balanced Accuracy	0.689	0.554	0.721
Sensitivity	0.604	0.354	0.726
Extreme Sensitivity	0.697		0.84
Precision	0.501	0.628	0.371
Adjusted Precision	0.788		0.696
Extreme Error	0.04		0.089
Specificty	0.774	0.754	0.716

(b) adjusted 4 classes

Figure 8: Average classification measures using two methods by 50 times.

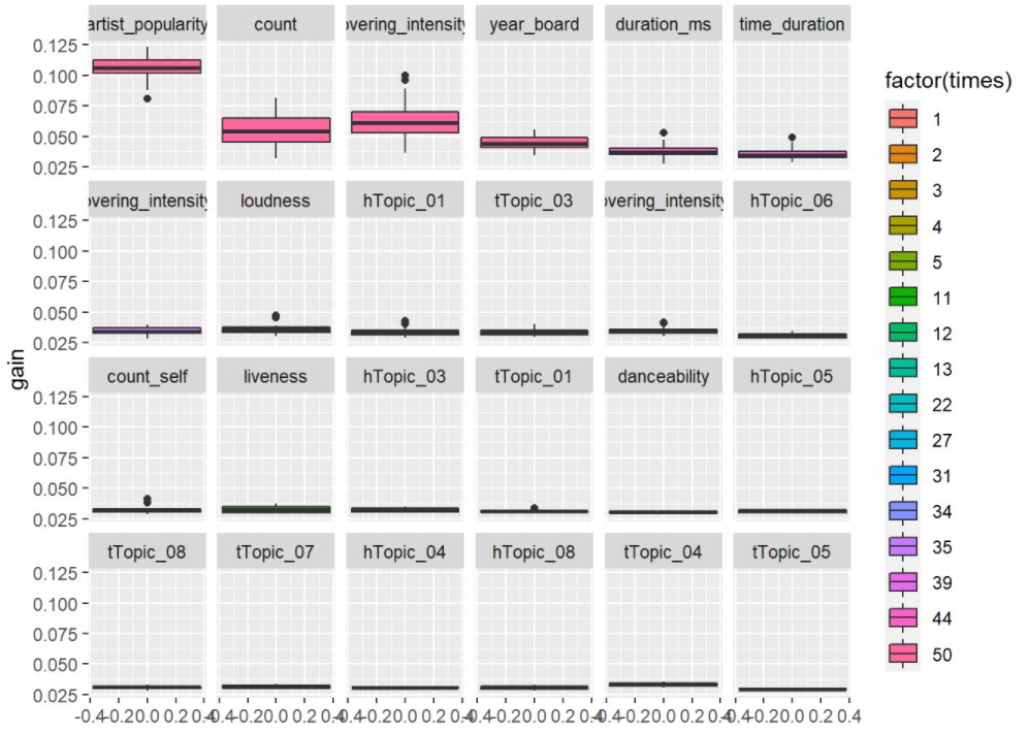


Figure 9: Feature importance given by the adjusted 4 classes classification. I record the most important 10 features every time according to their total contribution of the gain and sort by their appearance times.

There seems a tradeoff between sensitivity and precision, but generally the method via adjusted 4 classes performs better.

Figure 9 shows important features screend by cXGBoost. As expected, the most crucial predictors are the artist popularity of this track, the number of subsequent covering versions, the estimated covering frequency closest to the present (Covering_intensity4) and whether the song entered the year board. To my surprise, the name complexity does not play an important part in classification.

Finally, let us take a close look at the classification results, especially for those with high popularity:

Figure 10 lists the correctly classified songs with high popularity. Among them are

```
a3=testing_data[which(pred2=="(60, 100]"&(reference2=="(70, 100]"))%%624, ]
a4=testing_data[which(pred2=="(60, 100]"&(reference2=="(60, 70]"))%%624, ]
n=(a3%>%group_by(name)%>%count())
kable(full_join(n, unique(a3))%>%filter(n>=30)%>%select(n, popularity, artist_popularity, covering_intensity4, year_board, count)%>%arrange(desc(n)))
```

name	n	popularity	artist_popularity	covering_intensity4	year_board	count
Annie's Song	50	71	72	0.1512290	1	14
Carry On Wayward Son	50	76	69	0.1554550	0	17
Dancing Queen	50	84	82	0.2486229	0	26
Jolene	50	74	75	0.1657559	0	19
Killer Queen	50	72	88	0.2589617	1	20
Mamma Mia	50	79	82	0.1544459	0	43
Night Moves	50	74	71	0.1243115	0	10
Ob-La-Di, Ob-La-Da	50	75	88	0.1552909	0	10
Piano Man	50	80	80	0.1357264	0	13
Rebel Rebel	50	74	81	0.1209832	0	10
Rock And Roll All Nite	50	79	76	0.1833183	0	23
Somebody To Love	50	73	88	0.1708561	0	22
Sweet Home Alabama	50	85	75	0.1060302	0	14
Waterloo	50	72	82	0.1361061	1	15
Bohemian Rhapsody	49	78	88	0.1999468	1	24
I'm Not In Love	49	74	65	0.1222122	0	17
Sara Smile	49	71	74	0.0615731	1	14
50 Ways To Leave Your Lover	48	73	72	0.1537416	0	16
You Sexy Thing	48	73	62	0.1535195	1	14
Free Bird	45	75	75	0.0911948	0	12
One Of These Nights	44	71	79	0.1222621	0	10
Sweet Emotion	44	77	79	0.0917341	0	15
Come And Get Your Love	37	80	65	0.3313802	0	34
Feel Like Makin' Love	34	71	64	0.0305968	1	18
This Will Be	34	75	66	0.2145295	0	18
Show Me The Way	32	74	64	0.0924171	0	19

Figure 10: Correctly classified songs with high popularity.
Correct classification times and important predictors are also listed.

many well-known songs such as *Killer Queen* and *Bohemian Rhapsody* sung by Queen, *Mama Mia* and *Dancing Queen* sung by ABBA, etc.

What is even more interesting might be the songs with extremely high popularity but always misclassified as listed in Figure 11. I call these songs as atypical classic songs and some reasonable facts can be found to explain their singularity.

Hooked On A Feeling was firstly sung by a Blue Swede, a Swedish band, and made a massive hit, reaching #1 in the US, Holland, Australia, and Canada. As the first Swedish singer to score a No.1 hit in the U.S., Blue Swede paved the way for well-known ABBA. However, they themselves have never again cracked the American market. As you can see, the song was quickly forgotten and did not hit the year board.

```
a2=testing_data[which(pred2=="(60,100]"&(reference2=="[0,20]"))%%624,]
n=(a2%>%group_by(name)%>%count())
kable(full_join(n,unique(a2))%>%filter(n>=20)%>%select(n,popularity,artist_popularity,covering_intensity4,year_board,count)%>%arrange(desc(n)))
```

name	n	popularity	artist_popularity	covering_intensity4	year_board	count
Shotgun Shuffle	50	19	66	0.1379401	0	12
TVC 15	47	16	81	0.1853351	0	12
On And On	40	11	63	0.1814385	0	21
Wake Up Susan	39	13	64	0.1856453	0	14
Good Vibrations	32	0	61	0.3862761	0	33
Put A Little Love Away	32	16	65	0.0301855	0	13
Once You Hit The Road	30	7	66	0.1076859	0	7
Kung Fu	23	0	64	0.3782997	0	32

Figure 11: Misclassified songs with extremely high popularity. Misclassification times and important predictors are also listed.


```
a2=testing_data[which(pred2=="(60,100]"&(reference2=="[0,20]"))%%624,]
n=(a2%>%group_by(name)%>%count())
kable(full_join(n,unique(a2))%>%filter(n>=20)%>%select(n,popularity,artist_popularity,covering_intensity4,year_board,count)%>%arrange(desc(n)))
```

name	n	popularity	artist_popularity	covering_intensity4	year_board	count
Shotgun Shuffle	50	19	66	0.1379401	0	12
TVC 15	47	16	81	0.1853351	0	12
On And On	40	11	63	0.1814385	0	21
Wake Up Susan	39	13	64	0.1856453	0	14
Good Vibrations	32	0	61	0.3862761	0	33
Put A Little Love Away	32	16	65	0.0301855	0	13
Once You Hit The Road	30	7	66	0.1076859	0	7
Kung Fu	23	0	64	0.3782997	0	32

Figure 12: Misclassified songs with extremely low popularity. Misclassification times and important predictors are also listed.

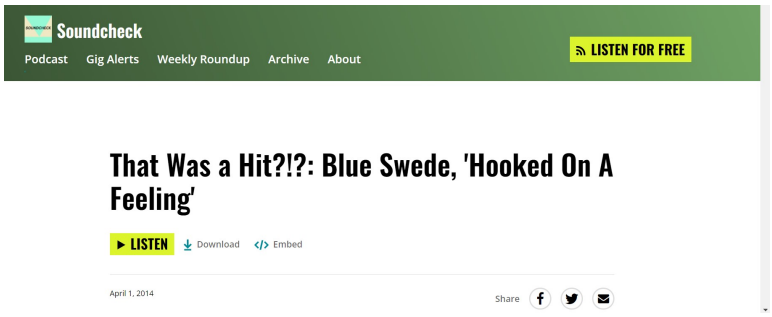


Figure 13: A news report (1) when the famous film *Guardians of the Galaxy* came out.

In 2014, the song was used in the hot film *Guardians of the Galaxy*, which helped it return to #1 in America and revived this song. This may explain why most evidence points out this is not a hot song in a long period while its current popularity is extermely high (Figure 13).

Some Kind Of Wonderful is also interesting because the popularity for 1999 remastered version is a lot higher than the 1974 version (Figure 14), possibly leading to a misclassification. I review the history of this song and found that it was used as the background music for the famous game *Grand Theft Auto* in 2004 (Figure 15). So maybe the listeners of this song are not the classical music fanciers and cannot use the classical predictors to correctly categorize it.

	track_name	artist_name	release_date	popularity
1	Some Kind of Wonderful	The Drifters	1962-04-23	50
2	Some Kind of Wonderful	Carole King	1971-12-01	37
3	Some Kind Of Wonderful - Remastered	Grand Funk Railroad	1974-12-01	32
4	Some Kind Of Wonderful - Remastered 1999	Grand Funk Railroad	1999-01-01	71
5	Some Kind Of Wonderful - Remastered 1999	Grand Funk Railroad	1999-01-01	69
6	Some Kind Of Wonderful - Remastered 1999	Grand Funk Railroad	1999-01-01	69
7	Some Kind Of Wonderful	Reflection Eternal	2000-10-17	26
8	Some Kind Of Wonderful	Joss Stone	2003-01-01	48
9	Some Kind Of Wonderful	Joss Stone	2003-01-01	48
10	Some Kind Of Wonderful	Peter Cincotti	2004-01-01	38
11	Some Kind Of Wonderful	Grand Funk Railroad	2006-01-01	51
12	Some Kind Of Wonderful - Remastered 2002	Grand Funk Railroad	2008-01-01	30
13	Some Kind Of Wonderful	Grand Funk Railroad	2008-01-01	34
14	Some Kind of Wonderful	Michael Bublé	2009-10-09	33
15	Some Kind of Wonderful	,Douglas Lyons,Alan Wiggins	2014-04-01	30
16	Some Kind Of Wonderful	Rod Stewart	2021-11-12	41
17	Some Kind Of Wonderful	Mean Marc Ash	2021-12-22	23

Figure 14: Covering information for *Some Kind Of Wonderful*.

Thare are also some songs with extremely low popularity predicted as classic songs (Figure 12). It is much more difficult to give a reasonable interpretation, but most of them are misclassified possibly because of missing information. As far as I am concerned, the song *Good Viberations* and *Wake Up Susan* should have been very famous songs.

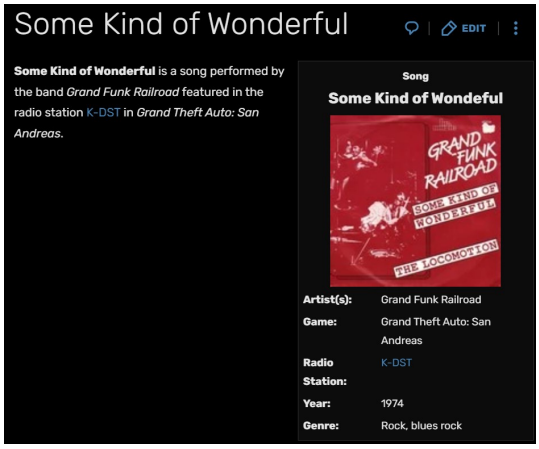


Figure 15: A snapshot of GTA songs Wiki for *Some Kind Of Wonderful*.

6 Conclusion and future work

To conclude, although predicting the popularity of old songs is not an easy task because of the complex unknown affecting factors, our XGboost with designed weighted oversampling method is effective in lifting the performance of predicion. Besides, the artist popularity and the appearance frequency of covering versions are still the gold standards in measuring the classic of the song.

For future work, I may consider adding more predictors such as lyric information if available. Deep learning methods may also be applied for comparison.

References

[1] That was a hit?!?: Blue swede, 'hooked on a feeling'. https://www.wnycstudios.org/podcasts/soundcheck/segments/that-was-a-hit-hooked-feeling-blue-swede.

[2] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system.

ACM, 2016.

[3] Ruth Dhanaraj and Beth Logan. Automatic prediction of hit songs. In *ISMIR*, pages 488–491. Citeseer, 2005.

[4] Jianyu Fan and Michael Casey. Study of chinese and uk hit songs prediction. In *Proceedings of the International Symposium on Computer Music Multidisciplinary Research*, pages 640–652, 2013.

[5] Dorien Herremans, David Martens, and Kenneth Sörensen. Dance hit song prediction. *Journal of New Music Research*, 43(3):291–302, 2014.

[6] Ioannis Karydis, Aggelos Gkiokas, Vassilis Katsouros, and Lazaros Iliadis. Musical track popularity mining dataset: Extension & experimentation. *Neurocomputing*, 280:76–85, 2018.

[7] David Martín-Gutiérrez, Gustavo Hernández Peñaloza, Alberto Belmonte-Hernández, and Federico Álvarez García. A multimodal end-to-end deep learning architecture for music popularity prediction. *IEEE Access*, 8:39361–39374, 2020.

[8] Matthias Mauch. Main dataset for "evolution of popular music: Usa 1960–2010". =https://doi.org/10.6084/m9.figshare.1309953.v1, 2015.

[9] François Pachet and Pierre Roy. Hit song science is not yet a science. In *ISMIR*, pages 355–360, 2008.

[10] Y. Taleb and EaK Cohen. Multiresolution analysis of point processes and statistical

thresholding for haar wavelet-based intensity estimation. *Annals of the Institute of Statistical Mathematics*, 73, 2021.