scientific reports

OPEN

Quantifying the impact of homophily and influencer networks on song popularity prediction

Niklas Reisz[1], Vito D. P. Servedio[1] & Stefan Thurner[1,2,3]✉

Forecasting the popularity of new songs has become a standard practice in the music industry and provides a comparative advantage for those that do it well. Considerable efforts were put into machine learning prediction models for that purpose. It is known that in these models, relevant predictive parameters include intrinsic lyrical and acoustic characteristics, extrinsic factors (e.g., publisher influence and support), and the previous popularity of the artists. Much less attention was given to the social components of the spreading of song popularity. Recently, evidence for musical homophily—the tendency that people who are socially linked also share musical tastes—was reported. Here we determine how musical homophily can be used to predict song popularity. The study is based on an extensive dataset from the *last.fm* online music platform from which we can extract social links between listeners and their listening patterns. To quantify the importance of networks in the spreading of songs that eventually determines their popularity, we use musical homophily to design a predictive *influence* parameter and show that its inclusion in state-of-the-art machine learning models enhances predictions of song popularity. The influence parameter improves the prediction precision (TP/(TP + FP)) by about 50% from 0.14 to 0.21, indicating that the social component in the spreading of music plays at least as significant a role as the artist's popularity or the impact of the genre.

While you read this paper's abstract, more than 50 new songs were released worldwide. Global music production has long reached such high levels that it is no longer possible to listen to every new song. On the world's largest music streaming platform *Spotify*, roughly 136 days worth of music are published daily[1,2]. This means that more music is produced in a year than one could listen to in an entire lifetime. In this ocean of new songs, there is a growing need for efficient selection and filtering, and the markets for attention have become increasingly competitive. This becomes apparent in increasingly imbalanced distributions of song popularity. While the majority of songs receive little to no attention, a small fraction become hits that are listened to billions of times[3]. Predicting which songs have the potential to become one of these hits has become an increasingly important task for music publishers[4]. The issue has sparked substantial commercial interest as publishers focus their marketing activities on high-potential songs and artists[5]. For quantitative predictions, a multitude of different approaches has been explored. The vast volume of data made available by streaming platforms in the past decade triggered a boost of data-driven approaches.

In the early 2000s, a discussion started on the possibility of quantitatively predicting song popularity. Using traditional machine learning classifiers on lyrical and acoustic song attributes, it was attempted to identify hits prospectively[6]. It was concluded that predictions were significantly better than random, with the lyrical features slightly outperforming the acoustic ones. In[7], it was claimed that the science of predicting hits, the so-called *hit song science*, was not yet a science as they demonstrated that the usual machine learning methods were not able to forecast the success of songs. This assertion was soon challenged in[8], where the authors argued that, given a relevant set of acoustic song features, forecasting song popularity was possible with more specific machine learning approaches. In these early works, the emphasis was primarily on *intrinsic* acoustic and lyrics features. Later, other studies improved outcomes with more sophisticated methods and expansive datasets. Specific song attributes such as "happiness", "partyness" and repetitiveness were found to increase the chances

[1]Complexity Science Hub Vienna, Josefstädter Strasse 39, 1080 Vienna, Austria. [2]Section for Science of Complex Systems, CeMSIIS, Medical University of Vienna, Spitalgasse 23, 1090 Vienna, Austria. [3]Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 85701, USA. ✉email: stefan.thurner@meduniwien.ac.at

of songs becoming hits[9,10]. Deep convolutional structures and machine learning regressions were convincingly shown to have predictive power[11–13].

While these studies focused exclusively on *intrinsic* properties of songs, newer studies have included *extrinsic* features such as metadata or artist popularity. It was claimed that extrinsic factors often carry increased predicting power, the most predictive feature being the previous popularity of the artist[5]. Askin and Mauskapf and Shin and Park [14,15] confirmed this result by finding that both, the previous popularity of the artist and the support of the publisher, have a significant impact on the success chances of songs. In addition to having increased chances of making it into the charts, Im et al.[16] found that songs of well-known artists and big publishers also stay there longer. Reference [17] demonstrated a significant dominance of superstars in the market share and identified a positive correlation between song success and the artist's prior release count. For *extrinsic* song attributes, Yu et al.[18] found that Support Vector Regression (SVR) performs slightly better than neural networks.

Predictive indicators were also found in Twitter data[19] where music-related hashtags such as *#nowplaying* were counted. The number of daily tweets about specific songs and artists is highly correlated with the listening trend of that song. Counting the same tags, Tsiara et al.[20] found a moderate correlation between the number of tweets and the chart position of a song, as well as a weak correlation with the sentiment of the tweets. These Twitter-based correlations hint at a social component to the success of songs and that this information might be encoded in social networks. A similar thought was followed in[21], where the spread of song popularity was modeled with a SIR disease spreading model. There, it is argued that social processes that lead to the spread of music are similar to those of disease spreading. For some genres, for which social connectivity is higher, songs appear to spread faster.

The fact that social networks co-create homophily—the tendency that people that share common treats are more likely to form social ties—has been observed in a variety of contexts, ranging from obesity[22] to performance in schools[23]. Also, in music listening behavior, the concept of homophily has been studied[24]. Recently, homophily's importance in relation to music preference was confirmed in a study on 1144 early adolescents, where music preference plays a significant role in selecting friends[25]. The online music-listening platform and social network *last.fm* https://www.last.fm/ offers an excellent ground for studying homophily as it simultaneously provides access to both social links and music listening records of users[26]. In[27], the authors study *last.fm* data to predict friendship links. They find that music preference alone rarely leads to friendships. In most cases, sharing friends is the best predictor of future friendships, i.e. triadic closure that has been predicted in[28] and explains basic structures of social multilayer networks[29]. Reference [30] confirmed that people with similar music preferences tend to cluster, indicating that friends tend to listen to similar music. Homophily in music listening was found in explorative behavior[31]. By estimating how mainstream, novel, or diverse listening records of users are, they find that highly explorative people tend to look for friends that are similar. Reference[32] confirmed that result and designed a model that explains the finding, stressing that highly explorative users tend to be friends with users with similar discovery rates. While[31] use homophily to predict friendship links on *last.fm*, in[33,34] it is investigated how information about new artists spreads through social links and quantified to which extent user behavior is copied. It is demonstrated how homophily can be leveraged to improve song recommendations by recommending new songs to users shortly after close friends are listening to them.

While extensive research has focused on acoustic and metadata-based factors for predicting song popularity, the influence of social interactions has received comparatively less attention, despite its significant impact on listening behavior. In this study, we quantify the influence of social interactions, particularly homophily and measures derived from homophily, on song popularity. Our work reveals how homophily is self-reinforced as users influence their friends to engage with specific songs, leading to the emergence and popularity of new songs through cascades of recommendations. We quantify this user influence through state-of-the-art machine-learning predictions of song popularity, assessing the extent to which our predictions can be enhanced. Our approach seeks to harness the intertwined dynamics of user homophily and influence, rather than disentangling these closely related factors, with the ultimate goal of optimizing prediction accuracy in the context of music recommendation systems.

Homophily

To estimate homophily, we derived a dataset from the online music-listening platform and social network *last.fm*. Our data includes 300 million individual listening events of 10 million songs. It is enriched by the (undirected and unweighted) friendship network, where nodes represent users and links represent bidirectional friendships. It consists of 2.7 million nodes and 15 million links with an average degree of 11.6. The network is connected with a diameter of 6. This dataset allows us to link friendships to music-listening histories. For more details, see methods.

We first determine the music preferences of every user. Music preferences can be assessed in *last.fm* through user-defined tags. Users define and add tags such as *Rock*, *80s*, or *Acoustic* to songs, artists, or albums. Here, we focus on artist tags because they are more abundant than song or album tags and offer more reliable statistics. Every artist, a, can have multiple tags, t, which are represented as weights, w_{at}, ranging between 1 and 100, based on their frequency of appearance (100 corresponds to the most frequent tag for that artist). We then compute a music preference matrix, m_{ut}, by summing over the weights, w_{at}, for each tag, t, of each artist, a, for each time a user, u, is listening to a song by that artist. The music preference matrix, M, with elements $m_{u,t}$ is hence defined as

$$m_{ut} = \sum_{a,s} l_{us} i_{sa} w_{at} \,, \tag{1}$$

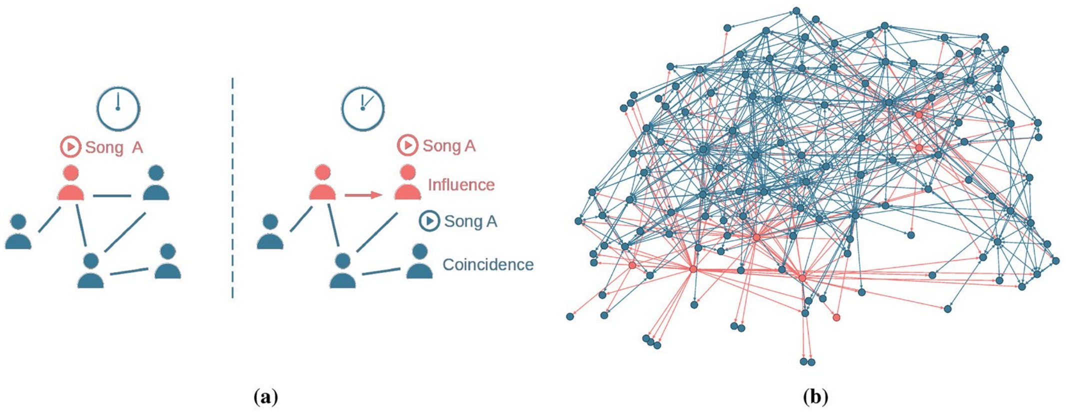
where l_{us} is the number of times user, u, listened to song, s. $i_{sa} = 1$ if artist, a, interpreted song, s, and is zero otherwise. In the music preference matrix M, each row corresponds to a vector of music preference of one user, expressed by the weighted artist tags. To compare the music preference vectors of users *i* and *j*, we use the cosine-similarity, defined as

$$\cos(\theta_{ij}) = \frac{\sum_t m_{u=i,t} m_{u=j,t}}{\sqrt{\sum_t m^2_{u=i,t}}\sqrt{\sum_t m^2_{u=j,t}}} . \tag{2}$$

The idea will be to compare the alignment of music preference vectors for users that are friends to the alignment between randomly picked users (that are very likely not friends).

Influencer network

As a next step, we define the influencer network. Influence is the number of times a user's friends copied the listening behavior of that user within a specified time window. We define it algorithmically. Whenever a friend, u_j, of a user, u_i, listens to a song for the first time at time t_j shortly after user, u_i, listened to that song at time t_i, the influence, I_{ij}, of user, *i*, on user, *j*, increases by one if $t_j - t_i < \Delta$ is smaller than some threshold Δ. A sketch of the concept and the derived influencer network are shown in Fig. 1. The influencer network (orange) is a directed network where users are nodes, and a weighted link represents the strength of a user's influence over another. In our case, it consists of 9200 nodes and 190,000 links, with an average degree of 21. The influencer network is connected and has a diameter of 11. Since friendship is a requirement for influence, the influencer network is a sub-network of the friendship network (blue), all nodes and links present in the influencer network also exist in the friendship network. Influence on *last.fm* is possible because users can see which songs their friends are listening to and can give specific song recommendations to them.



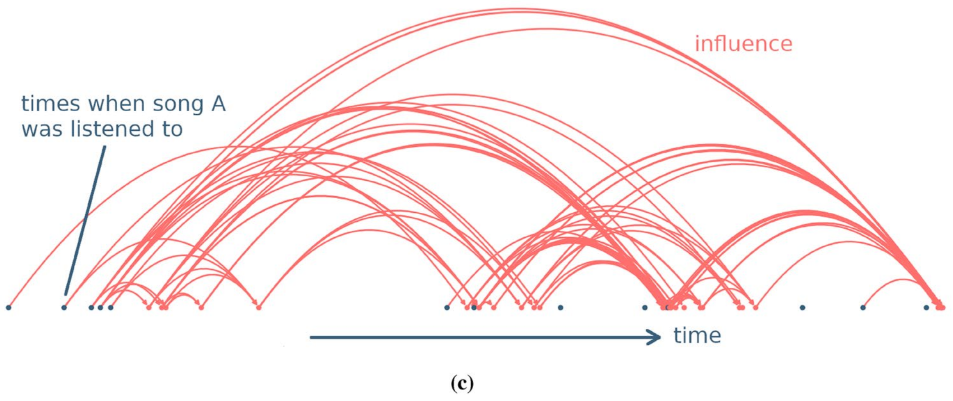


Figure 1. (**a**) Schematic depiction of user influence. If a user listens to a song at time t_1 and a friend of that user listens to the same song for the first time at time t_2 with $t_2 - t_1 < \Delta$ smaller than some threshold Δ, the second user is said to have been influenced by the first with respect to that song. The influencer network is shown in orange, the friendship network in blue. The influencer network is a sub-network of the friendship network. (**b**) A fraction of the influencer network of *last.fm* from a 24 h time window. Several strong influencers are visible (influencers and influencing links in orange) within the friendship network (blue). (**c**) Example of a timeline of a single song. Dots along the x-axis represent different users that listen to a song for the first time (location of dot). Dots are ordered in time, from left to right. Blue dots are users that found the song on their own, while orange dots represent users that were influenced by their friends. Arrows mark influencing events, where one user influences another into listening to a song. Panel (**b**) of this figure was created using the open source tool Gephy v0.10 https://gephi.org/.

Prediction strategy

To simplify the prediction task of the future success of a song, we only classify songs into average songs and future hit songs. We define hit songs as songs with at least 1000 listenings in any one month, which corresponds to the top 1% of songs. We do this classification based on an initial sample of features or parameters derived from the first 200 times a new song has been listened to. That is, when calculating a parameter associated with user attributes for a song, we first compute this value for all users contributing to the first 200 listenings of the song, and subsequently, we obtain the song's predictive parameter by averaging these values.

We define three classes of predictive parameters: preferential-attachment-based, time-based, and homophily-based. The preferential-attachment-based parameters are (i) the previous popularity of the artist and (ii) the genre's popularity. The time-based parameters are (iii) the time needed to reach 200 listenings, (iv) the tendency of users to re-listen to a song, (v) the number of users that listen to the song at least twice within a week, and (vi) the temporal trend quantified by the area under the curve of the cumulative listening count as a function of time. We also include two variations of these parameters, (vii) a normalized variation of (vi), where we take the average y-value instead of the area, and (viii) a variation of (vi) where we subtract the y-value from the diagonal and then compute the area. These constitute the basic eight parameters of the model. In addition to these parameters, we define homophily-based parameters and compute them for every song. For these, we identify the users that contributed to the first 200 times a song has been listened to. We use (ix–xi) the influence scores for three different time windows, as defined above, as well as (xii) the average cosine similarity between users and their friends as homophily-based parameters. Finally, we compute (xiii) the degree, (xiv) the PageRank, (xv) the nearest neighbor degree, and (xvi) the clustering coefficient both on the friendship network and the influencer network (xvii–xx). All parameters are described in detail in the methods section. We use these parameters as input for a machine learning ensemble to predict hit songs.

To determine whether the prediction of hit songs is best approached as a classification or regression task, we also explore this problem as a regression task, aiming to predict the exact number of listenings each song will receive. To establish comparability, we mapped the predicted listenings back to a binary classification of hit songs or average songs, using the 1000-listening threshold for hit songs. More details on this comparative approach can be found in the SI text C.

Results

Influencer network

Analyzing the triad statistics of the influencer networks, we find high levels of reciprocity in the influencer network, see Fig. 2. This is seen by the fact that triangles that include reciprocal links are strongly over-represented (blue bars) with respect to a random graph with the same number of nodes and (directed) links (shuffled network, configuration model). Triangles with no reciprocity are under-represented (red). This finding indicates that, generally, influence goes in both directions. Users who discover a new song from a friend (getting influenced) often influence other friends into listening to the song, leading to cascades, such as shown in Fig. 1c. In the SI text A, we identify the "influencers" and "followers" by comparing the number of times users got influenced with how often they influenced others. Overall, we find that 32.5% of new song listens on *last.fm* qualify as being influenced by friends.

Homophily results

We confirm the presence of strong homophily among users that are friends on *last.fm* by comparing the musical preferences of 1000 randomly picked users to those of their friends. We find an average cosine-similarity of musical-preference vectors $\vec{m}_u$ of $\cos(\theta) = 0.58$. In contrast, when comparing the musical preference vectors to an equal number of randomly picked users, the average cosine-similarity drops to $\cos(\theta) = 0.25$. A histogram of the respective cosine-similarities is shown in Fig. 3. The *t*-test for independent samples has a test statistic of 32 and a *p*-value of $p \leq 10^{-200}$. If the entries of the music preference vectors of the randomly picked users are shuffled in a random fashion, similarity decreases to levels below 10^{-4} and essentially disappears.

We find that both the user's influence and the tendency to get influenced correlate with the average similarity in musical preferences. When comparing users to their friends, this correlation is highly significant with a $p \leq 10^{-4}$. In contrast, when comparing random users, we do not find any correlation. For more details, see SI text B.

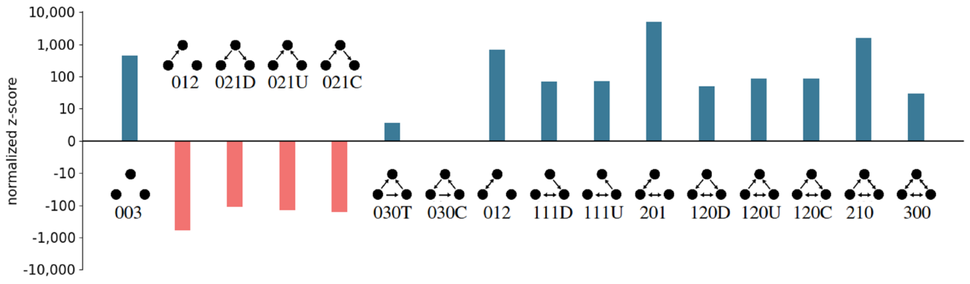


Figure 2. Triadic census of the influencer network. Comparison to a shuffled network (configuration model), averaged over 100 random shufflings. Triad names follow the convention of[35].

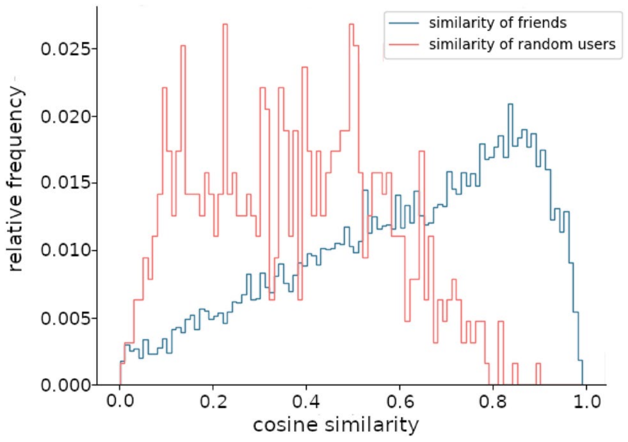


Figure 3. Cosine similarity distribution for users and their friends (blue), and between random users (red) that are typically not befriended. From the distribution, it becomes apparent that very similar people are almost certainly friends on *last.fm*..

Improving predictions

Both the homophily score and the influence score correlate with song popularity. Parameters that correlate with popularity can be tested if they also predict it. The Pearson correlation coefficients, r, between all parameters and the song popularity are found in Table 1. The strongest (anti) correlation for song popularity we find for the area under the temporal listening trend with $r = -0.22$. Other time-based parameters have comparable correlations. The influence score correlates better ($r = 0.17$) with song popularity than the previous popularity of the artist ($r = 0.14$). Also, influencer network based parameters correlate similarly as artist popularity.

Table 2 shows the prediction results for three different models quantified by accuracy, precision, and recall. We predict if a song becomes a hit-song or an average song based on information from the 200 first listenings and the structure of the influencer and friendship networks. A hit song is defined as a song that is listened to at least 1000 times in its best month, putting it approximately in the top 1% of all songs. For the classification task, we use a machine learning ensemble including classifiers based on Support Vector Classification, Random Forest, Ada Boost, Gradient Boost, K-Neighbors, and a multilayer perceptron neural network, see Methods. Based on the input parameters, the machine learning models try to classify each song into one of two classes: hit songs or regular songs. In the first model, which we refer to as the combined model (a), we use the combined parameter sets with preferential-attachment-, time-, and homophily-based parameters (i–xx). In the second, the social networks-only scenario (b), we use the homophily-based parameters (ix–xx) only. In the third, we combine the preferential-attachment and time-based parameters (i–viii) without using the homophily-based parameters and refer to it as the baseline model (c). The baseline model is based on commonly used parameters from the literature and excludes any social-network-based parameters. It forms the baseline against which we want to compare the results of models (a) and (b).

Parameter/feature	r
Degree (friendship network)	0.090
Degree (influencer network)	0.132
PageRank (influencer network)	0.110
Influence	**0.174**
Homophily	0.081
Time to reach 200 listenings	−0.201
Genre average	0.069
Time between repeated listenings	−0.193
Area under the trend curve	**−0.223**
Previous popularity of artist	0.135

Table 1. Pearson correlation coefficients between prediction parameters and song popularity. Bold numbers show the strongest correlations for homophily-based and baseline parameters. The prediction parameters were computed on the first 200 listenings for each song. Song popularity was defined as the maximum number of times a song was listened to *last.fm* in its best month. All p-values are below 10^{-200}. Only the best-performing variations of parameters are listed here. All other variations can be found in the "Methods" section.

Model	Prediction category	
(a) Combined model	Accuracy	0.98
	Hit precision	0.21
	Non-hit precision	1.0
	Hit recall	0.60
	Non-hit recall	0.99
(b) Social network only	Accuracy	0.95
	Hit precision	0.05
	Non-hit precision	1.0
	Hit recall	0.40
	Non-hit recall	0.96
(c) Without social network	Accuracy	0.97
	Hit precision	0.14
	Non-hit precision	1.0
	Hit recall	0.70
	Non-hit recall	0.98

Table 2. Classification results for three different cases: (a) the combined model (preferential attachment, time, and homophily), (b) the model with only homophily-based social network parameters, and (c) the baseline model without explicit social network information. We see that a combined model (a) performs best, with the highest accuracy, hit precision, and non-hit recall. The homophily-based model (b) performs well on its own but is outclassed by the baseline model.

The strongest result is the improvement of hit-precision by 50% in the combined model (a) when compared to the baseline model (c). In addition, non-hit recall and accuracy improved by one percentage point at an already high level. On the downside, hit-recall decreased by around 14%. Adding the homophily-based influence score as a predictive parameter significantly improves accuracy and precision, as well as non-hit recall, at the cost of hit-recall. This suggests that the combined model (a) is able to highly reduce the number of false positives at the cost of missing a small number of true positives. The homophily-based model (b) on its own manages to already identify 40% of all hit songs correctly and reaches an accuracy of 95%. We see that while conventional models like our baseline model (c) are superior to the homophily-based model (b), the homophily-based model performs already on a high level, and a combination of both leads to the best results. In our evaluation, t-tests comparing the results of the three different models yielded p-values below 10^{-4}, indicating statistically significant differences in performance, while all variances were in the range between 10^{-4} and 10^{-5}.

When comparing the classification and regression approaches, we do not observe statistically significant differences in the averages of accuracy, precision, or recall. This suggests that both methods are equally viable, and the choice between depends on the task at hand. For more details; see SI text C.

Discussion

We demonstrated how social interactions can be used to enhance song popularity predictions using a large dataset collected from the online platform *last.fm*. From an influencer network we derive an influence score for every user that captures their tendency to influence others to imitate their listening behavior. Based on a small sample of first listenings, we compute several metrics on the influencer network and use these as the predictive parameters in a machine learning classifier ensemble to categorize songs into potential hits and average songs. Our model exhibits up to 50% improved precision over a baseline model that uses common preferential attachment and time-based parameters.

The integration of social network-based parameters empowers our model to prioritize songs with a heightened likelihood of achieving rapid popularity in the current music landscape. This emphasis on the potential for swift growth enhances the model's precision by making its predictions more selective and precise. However, the model's slightly decreased recall may be attributed to its reduced sensitivity to songs that eventually become hits but do not exhibit strong social network signals during the prediction window. Some songs may take time to gain traction or might follow unconventional paths to success. The model's emphasis on current trends and interactions might lead it to overlook these less conventional hit trajectories, causing a decrease in recall. This trade-off between precision and recall has significant implications for the music industry and those seeking to invest in potential hit songs. The heightened precision of the new model provides more confidence in the songs it does predict as hits, thereby enhancing the decision-making process for investors looking to allocate resources to promising music ventures, ultimately leading to more informed and potentially more lucrative investment decisions in the dynamic and competitive music market.

Our analysis indicates that influence plays a significant role in shaping music listening behavior on *last.fm*, as approximately one-third of new song listens can be attributed to influence. Results from[33] suggest that the spread of music listening behavior occurs on relatively short time scales, with influence having a stronger effect in the first day, but decreasing over subsequent days and weeks. The used concept of influencing is closely related to homophily. Using the music preference vectors we estimate the similarity of tastes and find that people that are related through friendship links tend to have aligned tastes – i.e. strong homophily is confirmed, consistent with earlier findings of[30,33,36,37]. Influence represents the degree to which users can make their friends listening to the same music. This increases the similarity in the music preference vectors and drives homophily through adaptation. Research by[27,31] suggests that the main driver of homophily in music listening behavior is this adaptation of users' music listening profiles to be more similar to those of their friends, rather than changes in their friendship network. In that regard, *last.fm* differs from other systems such as for instance academic performance, where changes in network connections are more prevalent[23]. Additionally, we show that while some people tend to pioneer new music tastes and others tend to follow these pioneers, in most cases, influencer interactions go in both directions. This means that users that get influenced by their friends can in turn influence multiple other friends, leading to network structures that enable cascading spreading of new songs, fueling the new song's popularity.

Influence between users contributes to preferential attachment. The more people listen to a song, the more likely it gets recommended to friends, thus the probability of it being listened to increases. Parameters that relate to either preferential attachment, such as the previous popularity of the artist, or to forgetting, such as the listening trend, have been widely used in previous works to predict the popularity of songs[5,14–16]. In this study, we focus on these extrinsic properties and contribute new parameters that have the potential to improve the performance of existing models.

There are several severe limitations. Obviously, *last.fm* only provides a partial view of what is going on in music listening and recommendations. There are many other channels where people listen to music and exchange information about new songs which leads to a "cold-start" problem: a song that is new on the *last.fm* platform doesn't need to be new to its users. Hence, song popularity predictions might be offset by events that are beyond the scope of the available data. This is in part related also to the issue of comparability. A multitude of different datasets is used across the literature, each of them limited in some aspect, for instance, to a specific platform or a geographic region[5,9,11,12]. This is coupled with an apparent lack of a consistent definition of hit songs. In addition to that, in models with many parameters and hyperparameters, performance might be optimized with regard to different metrics. Altogether, specific results are difficult to compare across different studies. For this reason, we chose to use our own baseline model as a benchmark and aim for precise and intuitive definitions of popularity and hit songs, such that different machine learning models might be compared in a consistent way. In order to provide a realistic point of comparison, we aimed to optimize the performance of the aforementioned baseline model. Ultimately, the best-performing model we identified was an ensemble model. Our objective was to demonstrate that even the most effective models can be improved by incorporating social network information. Alternatively, if interpretability is the primary focus, one may choose to conduct the analysis using a single model, such as a random forest model.

In addition to these limitations, to some degree, popularity predictions might be self-fulfilling prophecies. It has been observed that people tend to reproduce perceived song popularity that is presented to them, even if these have been heavily modified[38–40]. Finally, there is also an ongoing discussion on whether artists should focus on improving popularity over other goals. Most popular doesn't necessarily imply most enjoyable or most relevant[41]. Rather, it indicates high commercial relevance, which might not necessarily be the top priority. However, even given these shortcomings, in this work, we were able to compare the predictions within a closed framework. Our main result is that we are able to find a substantial relative increase in predictability by including social information. The concept presented here can straightforwardly be transferred to other domains such as movies, books, posts, or even physical goods. Given the growing availability of social network data, comparable approaches might further uncover the social underpinnings of consumer behavior in our society.

Methods
Data
Our analysis is based on data collected from last.fm. Table 3 shows the number of songs, listenings, albums, artists, and users in the dataset that we collected for this study. Users are further broken down into users for which the full friendship data i.e. the account names of all their friends are known and users for which we collected the full listen history. Figure 4 shows the timeline of last.fm and marks the time periods for which data was fetched. The bootstrap phase was used to bootstrap the popularity of artists.

Songs	10,000,000
Listenings	300,000,000
Albums	200,000
Artists	1,000,000
Tags	95,000
Users	2,500,000
With friendship data	100,000
With listen history	17,500

Table 3. Available data that was collected on the last.fm for the purpose of this study. Numbers are rounded down.

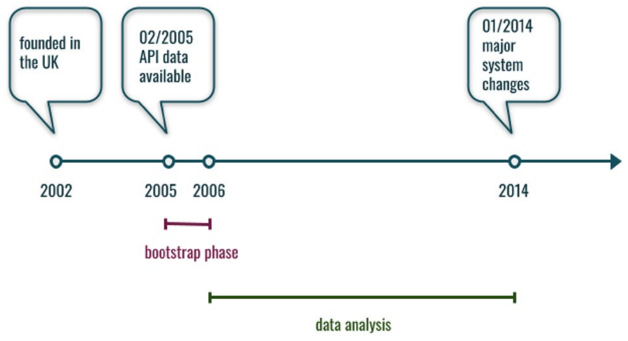


Figure 4. last.fm timeline and data availability. The company was founded in 2002 in the UK. User data is available on the API starting from February 2005. In 2014 there was a major change to the system—users were not able anymore to listen to music directly on last.fm but rather could connect their last.fm account to other streaming services such as Spotify.

Data collection
For data acquisition, we employed the Python interface *pylast*[42] and initiated crawling by selecting seed users from online communities. The crawl commenced by querying for the friends of these selected users and expanded to their friends' friends and so forth, using a breadth-first search strategy. This approach aimed to establish a core set of users with complete friend lists as well as second-order friends, providing a comprehensive understanding of their links in the friendship network. While we captured a significant portion of the friendship network, including all friendship links of 100,000 users, we selectively collected listening histories due to the high number of required API requests (averaging approximately 20,000 listenings per user). This focused data collection involved a core set of 17,500 users, striking a balance between computational efficiency and maintaining a thorough representation of the listening history and friendship network.

Data pre-processing
In preprocessing Last.fm tags, we strategically addressed the inherent noise and potential inaccuracies associated with user-generated tags. Instead of relying solely on song tags, which can be prone to errors, we opted for a more robust approach by utilizing artist tags. This decision leveraged the much larger number of user votes for artists, enhancing the reliability of the tags used in our analysis. Last.fm utilizes a weighted tagging system, where the influence of tags is determined by the number of votes received, enabling us to mitigate the impact of potentially incorrect or less popular tags. To enhance the quality of our dataset, we introduced a manual blacklist for tags deemed misleading, harmful, or derogatory. This step ensured the exclusion of specific tags, irrespective of their popularity, contributing to a more accurate representation of user preferences. Additionally, our database incorporates merging rules to address instances where identical items might appear separately due to variations in spelling or the addition of extraneous information. The Python scripts for downloading, cleaning, and merging the data are accessible on the GitHub repository at https://github.com/NiklasRz/lastfm-data-downloader.

Prediction parameters
We use the data to train a prediction model that is based on several social- and metadata-based parameters. For the social parameters, we use two networks: the friendship network and the influencer network. The friendship network of *last.fm* consists of nodes that represent users and links that represent bidirectional friendships. The influencer network is a directed network where users are nodes, and a weighted link represents the strength of a user's influence over another. On these two networks, we define the following user metrics:

(ix–xi) user influence with time windows of 6h, 12h and 24h. The influence score of each user is divided by the number of songs where a user influenced another user thus giving us the number of influencing events per song. This is equivalent to the out-degree of the user in the influencer network divided by the number of songs.
(xii) homophily as defined here by the average cosine similarity of the music preference vectors between a user and their friends
(xiii) the degree of each user, computed on the friendship network
(xiv) the PageRank of each user (node), computed on the friendship network
(xv) the nearest neighbor degree of each user, computed on the friendship network
(xvi) the clustering coefficient of each user, computed on the friendship network
(xvii) the degree of each user, computed on the influencer network
(xviii) the PageRank of each user, computed on the influencer network
(xix) the nearest neighbor degree of each user, computed on the influencer network
(xx) the clustering coefficient of each user, computed on the influencer network

Each of these user metrics is computed for the users that contribute the first 200 listenings to a song and averaged per song. The average is then the value of the predictive parameter for that song.

The performance of these parameters is compared to the performance of commonly used parameters that form our baseline. Specifically, we test the following parameters:

(i) previous popularity of the artist. For this, we take a snapshot in time when the new song is released and calculate the average number of listenings that songs of the same artist have received in the past.
(ii) the average number of listenings, that songs of the same genre receive in our dataset. We define the genre of a song, as the value of the the user-defined artist-tag with the largest weight (100).
(iii) the time, Δt_j, it takes a song, j, to reach the first 200 listenings in last.fm, given by

$$\Delta t_j = t_{j,200} - t_{j,1} \tag{3}$$

where $t_{j,i}$ is the timestamp in seconds when song, j, was listened to for the ith time.
(iv) the average time that passes between two listenings of the same user, measured in seconds. Again, we only look at the first 200 listenings here. Whenever the same user listens more than once to the song withing those first 200 listenings, we compute the time that has passed in between. We then average these timespans per song.
(v) the number of users among the first 200 listenings that listen to the song again within a timespan of at most one week. This is similar to (iv), but instead of looking at the time that passes in between, we simply count how many users listen to the song more than once.
(vi) the area, A_j, under the curve if the cumulative listenings of a song, j, up to a number of 200 are plotted vs time. This is given by

$$A_j = \int_{t=t_{j,1}}^{t_{j,200}} l_j(t)dt \tag{4}$$

where $t_{j,i}$ is the timestamp in seconds when song, j, was listened to for the ith time and $l_j(t)$ is the total number of times song, j, has been listened to at time, t.
(vii) the same area as in (vi), but divided by the length of the x-axis (total time passed between the first and the 200th listening). This is given by

$$A_j'' = \frac{\int_{t=t_{j,1}}^{t_{j,200}} l_j(t)dt}{t_{j,200} - t_{j,1}} \tag{5}$$

where $t_{j,i}$ is the timestamp in seconds when song, j, was listened to for the ith time and $l_j(t)$ is the total number of times song, j, has been listened to at time, t.
(viii) the same area as in (vi), but subtracted from the area under the diagonal. This is given by

$$A_j' = \frac{(t_{j,200} - t_{j,1})(l_j(t = t_{j,200}) - l_j(t = t_{j,1}))}{2} - \int_{t=t_{j,1}}^{t_{j,200}} l_j(t)dt \tag{6}$$

where $t_{j,i}$ is the timestamp in seconds when song, j, was listened to for the ith time and $l_j(t)$ is the total number of times song, j, has been listened to at time, t.

Table 4 shows the correlation coefficients for the parameters listed above and the popularity of songs.

Prediction model

These parameters are used in our machine-learning ensemble for song popularity predictions. The ensemble includes classifiers based on Support Vector Classification, Random Forest, Ada Boost, Gradient Boost, K-Neighbors, and a neural network. The different classifiers hold a majority vote on the classification of each song. Each song is classified as either a future hit-song or an average song, where hit-songs are defined as songs in the top 1% of songs, which coincidentally equates to being listened to approximately at least 1000 times in the best month in our dataset. Given the challenge of defining a precise "hit song" and comparing absolute popularity metrics across different systems, we chose this classification approach over regression. By basing our classification on songs included in the last.fm chart of top-performing songs that we defined, we emphasize the intuitive and comparative nature of the model results. In the following, we give a brief overview of the machine learning models used, the specific implementation, and their hyperparameter settings.

Support vector classification is a supervised learning model that tries to map training data into a higher dimensional space with the aim of maximizing the gap between points of different classes in that space[43]. In this study, we use the Python sklearn implementation[44] with balanced class weights and a value of $C = 1$ for the regularization hyperparameter.

Random Forest classification is a supervised learning model that builds multiple randomized decision trees based on the training set[45]. The classification outcome is then the majority vote of the individual trees. In this study, we use the Python sklearn implementation[46] with 100 estimators and balanced class weights.

Ada Boost, short for adaptive boosting, is a classifier that iteratively learns from the mistakes of weak classifiers, turning them into strong classifiers[47]. In this study, we use the Python sklearn implementation[48] with 100 estimators and a decision tree classifier.

Gradient Boost classifier is a classifier that works as an ensemble of weak classifiers, typically decision trees. These weak classifiers are gradually added during the learning process while aiming for maximum correlation with the negative gradient of the loss function[49]. In this study, we use the Python sklearn implementation[50] with 100 estimators, a learning rate of 1, and a maximum depth of 1. In addition to this, we added a histogram

(i) Previous popularity of artist	0.135
(ii) Genre average	0.069
(iii) Time to reach 200 listenings	−0.201
(iv) Time between repeated listenings	−0.193
(v) Number of repeated listeners	0.039
(vi) Area under the trend curve	−0.223
(vii) Area under the trend curve (normalized)	−0.044
(viii) Area under the trend curve (subtracted from diagonal)	−0.085
(ix) Influence 6 h	0.159
(x) Influence 12 h	0.169
(xi) Influence 24 h	0.174
(xii) Homophily	0.081
(xiii) Degree (friendship NW)	0.090
(xiv) Pagerank (friendship NW)	−0.054
(xv) Nearest neighbor degree (friendship NW)	−0.081
(xvi) Clustering coefficient (friendship NW)	0.069
(xvii) Degree (influencer NW)	0.132
(xviii) Pagerank (influencer NW)	0.110
(xix) Nearest neighbor degree (influencer NW)	0.090
(xx) Clustering (influencer NW)	0.062

Table 4. Pearson correlation coefficients between prediction parameters and song popularity. The prediction parameters were computed on the first 200 listenings for each song. Song popularity was defined as the maximum number of times a song was listened to *last.fm* in its best month. All p-values are below 10^{-50}.

gradient boost classifier, which uses binned input variables. We use the Python sklearn implementation[51] with a maximum of 100 iterations, a learning rate of 0.1 and a maximum depth of 3.

K-nearest neighbor classification is based on a multidimensional feature space that is populated by the feature vectors of the training set and their labels[52]. During classification, one looks at the nearest k neighbors of an element and attaches the label to it that is most common among its neighbors. In this study, we use the Python sklearn implementation[53] with the number of nearest neighbors k equal to 5.

A multilayer perceptron is a fully connected, feed-forward artificial neural network that consists of at least three layers: an input, a hidden layer, and an output layer[54]. All nodes in each layer are connected to all other nodes of the following layer. It uses non-linear activation functions and learns through back-propagation. In this study, we use the Python sklearn implementation[55] with 2 layers, 100 neurons per hidden layer, RELU activation function, ADAM solver, and a maximum number of iterations of 1000.

These machine learning models are each trained on (the same) 60% of the data and then used to classify the other 40%. The data is classified according to the majority vote of the different models.

Data availability

The data that support the findings of this study are available from *last.fm* but restrictions apply to the availability of these data, which were used under license for the current study, and so are not publicly available. Data are however available from the authors upon reasonable request and with permission of *last.fm*. *Last.fm* offers limited public access to their data via their public API https://www.last.fm/api. The authors have created an open-source software tool to download the data used in this study through the public API, which is available at https://github.com/NiklasRz/lastfm-data-downloader.

Received: 9 February 2023; Accepted: 5 April 2024
Published online: 18 April 2024

References

1. Ingham, T. Over 60,000 tracks are now uploaded to Spotify every day. That's nearly one per second. https://www.musicbusinessworldwide.com/over-60000-tracks-are-now-uploaded-to-Spotify-daily-thats-nearly-one-per-second/. Accessed 25 Aug 2022 (2021).
2. Houghton, B. 60,000 tracks are uploaded to Spotify every day. https://www.hypebot.com/hypebot/2021/02/60000-tracks-are-uploaded-to-Spotify-every-day.html. Accessed 25 Aug 2022 (2021).
3. Hu, H.-B. & Han, D.-Y. Empirical analysis of individual popularity and activity on an online music service system. *Phys. A Stat. Mech. Appl.* **387**, 5916–5921. https://doi.org/10.1016/j.physa.2008.06.018 (2008).
4. Rosen, S. The economics of superstars. *Am. Econ. Rev.* **71**, 845–858 (1981).
5. Pham, J., Kyauk, E. & Park, E. Predicting song popularity. In Technical Report. Vol. 26. Department Computer Science, Stanford University (2016).
6. Dhanaraj, R. & Logan, B. Automatic prediction of hit songs. In *Proceedings of the International Conference on Music Information Retrieval* (*ISMIR*). 488–491 (2005).
7. Pachet, F. & Roy, P. Hit song science is not yet a science. In *ISMIR*. 355–360 (2008).

8. Ni, Y., Santos-Rodriguez, R., Mcvicar, M. & De Bie, T. Hit song science once again a science. In *4th International Workshop on Machine Learning and Music* (Citeseer, 2011).
9. Interiano, M. *et al.* Musical trends and predictability of success in contemporary songs in and out of the top charts. *R. Soc. Open Sci.* **5**, 171274 https://doi.org/10.1098/rsos.171274. eprint https://royalsocietypublishing.org/doi/pdf/10.1098/rsos.171274 (2018).
10. Lassche, A., Karsdorp, F. & Stronks, E. Repetition and popularity in early modern songs. In *DH 2019: Proceedings of the 2019 Digital Humanities Conference* (2019 Digital Humanities Conference, 2019).
11. Yang, L.-C., Chou, S.-Y., Liu, J.-Y., Yang, Y.-H. & Chen, Y.-A. Revisiting the problem of audio-based hit song prediction using convolutional neural networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing* (*ICASSP*). 621–625 https://doi.org/10.1109/ICASSP.2017.7952230 (2017).
12. Middlebrook, K. & Sheik, K. Song hit prediction: Predicting billboard hits using Spotify data. *CoRR eprint*arXiv:1908.08609 *(2019).*
13. Martin-Gutierrez, D., Penaloza, G. H., Belmonte-Hernandez, A. & Garcia, F. A. A multimodal end-to-end deep learning architecture for music popularity prediction. *IEEE Access* **8**, 39361–39374. https://doi.org/10.1109/access.2020.2976033 (2020).
14. Askin, N. & Mauskapf, M. What makes popular culture popular? Product features and optimal differentiation in music. *Am. Sociol. Rev.* **82**, 910–944. https://doi.org/10.1177/0003122417728662 (2017).
15. Shin, S. & Park, J. On-chart success dynamics of popular songs. *Adv. Complex Syst.* **21**, 1850008. https://doi.org/10.1142/S021952591850008X (2018).
16. Im, H., Song, H. & Jung, J. A survival analysis of songs on digital music platform. *Telem. Inform.* **35**, 1675–1686. https://doi.org/10.1016/j.tele.2018.04.013 (2018).
17. Kim, S. T. & Oh, J. H. Music intelligence: Granular data and prediction of top ten hit songs. *Decis. Supp. Syst.* **145**, 113535. https://doi.org/10.1016/j.dss.2021.113535 (2021).
18. Yu, H., Li, Y., Zhang, S. & Liang, C. Popularity prediction for artists based on user songs dataset. In *Proceedings of the 2019 5th International Conference on Computing and Artificial Intelligence, ICCAI '19*. 17–24. https://doi.org/10.1145/3330482.3330493 (Association for Computing Machinery, 2019).
19. Kim, Y., Suh, B. & Lee, K. #nowplaying the future billboard: Mining music listening behaviors of Twitter users for hit song prediction. In *Proceedings of the First International Workshop on Social Media Retrieval and Analysis, SoMeRA '14*. 51–56. https://doi.org/10.1145/2632188.2632206 (Association for Computing Machinery, 2014).
20. Tsiara, E. & Tjortjis, C. Using Twitter to predict chart position for songs. In *Artificial Intelligence Applications and Innovations* (Maglogiannis, I., Iliadis, L. & Pimenidis, E. eds.). 62–72 (Springer, 2020).
21. Rosati, D. P., Woolhouse, M. H., Bolker, B. M. & Earn, D. J. D. Modelling song popularity as a contagious process. *Proc. R. Soc. A Math. Phys. Eng. Sci.* **477**, 20210457. https://doi.org/10.1098/rspa.2021.0457 (2021).
22. Christakis, N. A. & Fowler, J. H. The spread of obesity in a large social network over 32 years. *N. Engl. J. Med.* **357**, 370–379. https://doi.org/10.1056/nejmsa066082 (2007).
23. Smirnov, I. & Thurner, S. Formation of homophily in academic performance: Students change their friends rather than performance. *PLOS ONE* **12**, e0183473. https://doi.org/10.1371/journal.pone.0183473 (2017).
24. Miller McPherson, L.S.-L. & Cook, J. M. Birds of a feather: Homophily in social networks. *Annu. Rev. Sociol.* **27**, 415–444 (2001).
25. Franken, A., Keijsers, L., Dijkstra, J. K. & ter Bogt, T. Music preferences, friendship, and externalizing behavior in early adolescence: A SIENA examination of the music marker theory using the SNARE study. *J. Youth Adolesc.* **46**, 1839–1850. https://doi.org/10.1007/s10964-017-0633-4 (2017).
26. Mechant, P. & Evens, T. Interaction in web-based communities: A case study of last.fm. *Int. J. Web Based Commun.* **7**, 234–249 https://doi.org/10.1504/IJWBC.2011.039513. eprint https://www.inderscienceonline.com/doi/pdf/10.1504/IJWBC.2011.039513 (2011).
27. Bischoff, K. We love rock 'n' roll: Analyzing and predicting friendship links in last.fm. In *Proceedings of the 4th Annual ACM Web Science Conference, WebSci '12*. 47–56. https://doi.org/10.1145/2380718.2380725 (Association for Computing Machinery, 2012).
28. Granovetter, M. S. The strength of weak ties. *Am. J. Sociol.* **78**, 1360–1380 (1973).
29. Sadilek, M., Klimek, P. & Thurner, S. Asocial balance—How your friends determine your enemies: understanding the co-evolution of friendship and enmity interactions in a virtual world. *J. Comput. Soc. Sci.* **1**, 227–239. https://doi.org/10.1007/s42001-017-0010-9 (2017).
30. Guidotti, R. & Rossetti, G. "Know thyself" how personal music tastes shape the last.fm online social network. In *Formal Methods. FM 2019 International Workshops* (Sekerinski, E. *et al.* eds.) 146–161 (Springer, 2020).
31. Duricic, T., Kowald, D., Schedl, M. & Lex, E. My friends also prefer diverse music: Homophily and link prediction with user preferences for mainstream, novelty, and diversity in music. In *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM '21*. 447–454. https://doi.org/10.1145/3487351.3492706 (Association for Computing Machinery, 2022).
32. Di Bona, G. *et al. Social Interactions Affect Discovery Processes*. https://doi.org/10.48550/ARXIV.2202.05099 (2022).
33. Pálovics, R. & Benczúr, A. A. Temporal influence over the last.fm social network. *Soc. Netw. Anal. Min.*https://doi.org/10.1007/s13278-014-0244-y *(2015).*
34. Pálovics, R. & Benczúr, A. A. Temporal influence over the last.fm social network. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM '13*. 486–493 https://doi.org/10.1145/2492517.2492532 (Association for Computing Machinery, 2013).
35. Wasserman, S. & Faust, K. *Social Network Analysis: Methods and Applications. Structural Analysis in the Social Sciences* (Cambridge University Press, 1994).
36. Zhou, Z., Xu, K. & Zhao, J. Homophily of music listening in online social networks of China. *Soc. Netw.* **55**, 160–169. https://doi.org/10.1016/j.socnet.2018.07.001 (2018).
37. Bisgin, H., Agarwal, N. & Xu, X. A study of homophily on social media. *World Wide Web* **15**, 213–232. https://doi.org/10.1007/s11280-011-0143-3 (2011).
38. Salganik, M. J., Dodds, P. S. & Watts, D. J. Experimental study of inequality and unpredictability in an artificial cultural market. *Science* **311**, 854–856. https://doi.org/10.1126/science.1121066 (2006).
39. Salganik, M. J. & Watts, D. J. Leading the herd astray: An experimental study of self-fulfilling prophecies in an artificial cultural market. *Soc. Psychol. Q.* **71**, 338–355. https://doi.org/10.1177/019027250807100404 (2008).
40. Lynn, F. B., Walker, M. H. & Peterson, C. Is popular more likeable? Choice status by intrinsic appeal in an experimental music market. *Soc. Psychol. Q.* **79**, 168–180. https://doi.org/10.1177/0190272516645603 (2016).
41. Monechi, B., Gravino, P., Servedio, V. D. P., Tria, F. & Loreto, V. Significance and popularity in music production. *R. Soc. Open Sci.* **4**, 170433. https://doi.org/10.1098/rsos.170433 (2017).
42. van Kemenade, H. *Pylast*. https://github.com/pylast/pylast (2024). Accessed 13 Jan 2024.
43. Hsu, C.-W., Chang, C.-C. & Lin, C.-J. A practical guide to support vector classification. *J. Mach. Learn. Res.* **101**(1), 1396–1400 (2003).
44. scikit-learn developers. *Svc in Python Sklearn*. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. Accessed 22 Nov 2022 (2022).
45. Biau, G. & Scornet, E. A random forest guided tour. *TEST* **25**, 197–227. https://doi.org/10.1007/s11749-016-0481-7 (2016).
46. scikit-learn developers. *Random Forest in Python Sklearn*. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. Accessed 22 Nov 2022 (2022).

47. Schapire, R. E. Explaining AdaBoost. In *Empirical Inference*. 37–52. https://doi.org/10.1007/978-3-642-41136-6_5 (Springer, 2013).
48. scikit-learn developers. *Ada Boost in Python Sklearn*. https://scikit-learn.org/stable/modules/ensemble.html#adaboost. Accessed 22 Nov 2022 (2022).
49. Natekin, A. & Knoll, A. Gradient boosting machines, a tutorial. *Front. Neurorobot.* **7**, 21 (2013).
50. scikit-learn developers. *Gradient Tree Boosting in Python Sklearn*. https://scikit-learn.org/stable/modules/ensemble.html#gradient-tree-boosting. Accessed 22 Nov 2022 (2022).
51. scikit-learn developers. *Histogram Gradient Tree Boosting in Python Sklearn*. https://scikit-learn.org/stable/modules/ensemble.html#histogram-based-gradient-boosting. Accessed 16 Sep 2023 (2023).
52. Cover, T. & Hart, P. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory* **13**, 21–27. https://doi.org/10.1109/TIT.1967.1053964 (1967).
53. scikit-learn developers. *K Neighbors Classifier in Python Sklearn*. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html. Accessed 22 Nov 2022 (2022).
54. Hastie, T., Tibshirani, R. & Friedman, J. *The Elements of Statistical Learning* (Springer, 2009).
55. scikit-learn developers. *Multi-Layer Perceptron Classifier*. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html. Accessed 22 Nov 2022 (2022).

Author contributions

NR, ST, VDPS conceptualized the work and designed the model. NR collected and prepared the data and performed the analysis. NR and ST wrote the paper.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-024-58969-w.

Correspondence and requests for materials should be addressed to S.T.

Reprints and permissions information is available at www.nature.com/reprints.