Song Popularity Prediction using Ordinal Classification

Michael Vötter, Maximilian Mayerl, Eva Zangerle, Günther Specht
Department of Computer Science
Universität Innsbruck
Innsbruck, Austria
{firstname.lastname}@uibk.ac.at

ABSTRACT

Predicting a song's success based on audio descriptors before its release is an important task in the music industry, which has been tackled in many ways. Most approaches utilize audio descriptors to predict a song's success, typically captured by either chart positions or listening counts. The popularity prediction task is then either modeled as a regression task, where the popularity metric is precisely predicted, or as a classification task by, e.g., transforming the popularity task to distinct classes such as hits and non-hits. However, this way of modeling the task neglects that most popularity measures form an ordinal scale. While classification ignores the order, regression assumes that the data is in interval (or ratio) scale. Therefore, we propose to model the task of popularity prediction as an ordinal classification task. Further, we propose an approach that utilizes the relative order of classes in an ordinal classification setup to predict the popularity (class) of songs. Our presented approach requires a machine learning model able to predict the relative order of two pieces of music, and hence can flexibly be applied using many types of predictors. Furthermore, we investigate how different ways of mapping the underlying popularity metrics to ordinal classes influence our model. We compare the proposed approach with regression as well as classification models and show its robustness w.r.t. different numbers of ordinal classes and the distribution of the number of songs assigned to them. Additionally, we show that, for some prediction settings, our approach results in a better predictive performance than classical regression and classification approaches, while it achieves similar predictive performance on other settings.

1. INTRODUCTION

Song popularity prediction is an important task in the music industry, where sales, charts, or listening data are used to determine the popularity of music. The goal is to predict the success of a song before or shortly after its release. Such systems could be utilized by musicians to tweak their songs towards success. For instance, they could use such predictors during the creation process of a song to determine if a given song will be successful. Record labels could use such a system to determine which songs they should support and promote.

To this end, the song popularity prediction task is either modeled as a regression [1–3] or classification task [4–6]. Existing approaches make use of audio descriptors such as Essentia audio features [7] for prediction. To determine the popularity of songs, chart-based measures such as peak position and number of weeks in charts and listener- and play counts on streaming platforms are used. Typically, they are gathered from the Billboard Hot 100 charts [1] among other country-specific charts or from streaming platforms such as last.fm [2] and Spotify [3]. A wide range of machine learning approaches such as linear models [8, 9], SVM models [6, 8, 10], tree models [5], and neural network models [1–3] have been used for prediction. In a recent work, we present two datasets (HSP-S and HSP-L) [11] and compared such models on both types of prediction tasks in our follow-up work [12].

Despite these previous efforts, modeling the song popularity prediction task as either a regression or classification task is insufficient. Modeling it as a regression task assumes equidistant and continuous popularity values, while modeling it as a classification task ignores the natural order encoded in the utilized measures of popularity. Hence, modeling it as a regression task requires that the resulting popularity measure is continuous. This is obviously not the case for popularity measures such as chart position, as they are always given as positive integers. Further, not all popularity measures allow assuming equidistant gaps between successive popularity values. E.g., chart metrics such as the weeks in charts contradict this assumption, as all songs that did not made it to the charts exhibit a value of zero. Obviously, the songs that did not make it into the charts are not all equally popular. Similar arguments are valid for listening event-based measures of popularity. Again, listener counts and play counts are always positive integers. Further, songs that have no listening events on a particular platform are not necessarily equally popular. Additionally, metrics based on listening events have a high resolution in terms of distinct popularity values and form a power scale. We argue that predictions do not have to be that fine-grained to be useful for the previously mentioned applications. Considering the fact that these counts

[1] https://www.billboard.com/charts/hot-100
[2] https://www.last.fm/
[3] https://www.spotify.com/

are in power scale, it is quite obvious that different "levels" of popularity show a large difference in the number of listening events, especially when considering truly popular songs. Hence, we argue that binning successive values to reduce the number of distinct values is legit and still keeps the majority of relevant information on the popularity of a song. Binning is also done by modeling the task as a classification task. Modeling it as a classical classification task ignores the natural order of popularity values, which in contrast means that valuable information is lost. As a result, commonly used classification models cannot use ordering information for their predictions.

To resolve these issues, we propose to model music popularity prediction as an ordinal classification task that encodes the order information. This allows exploiting the natural ordering of popularity classes. We already laid out in [11] that popularity measures are in ordinal scale. This means that successive popularity values are not necessarily equidistant, but they have an order. Hence, modeling the task as an ordinal classification task addresses the fact that popularity measures are in ordinal scale. Furthermore, this task formulation allows adjusting the number of distinct popularity classes by binning multiple successive popularity values into a single popularity class, similar to a classical classification task. Combining popularity values is enabled by the ordinal nature of those classes, without the requirement that all classes have an equal size or distances between each other. The extreme case would be to reduce the number of classes to two, resulting in the already known hit/non-hit classification. Note that in addition to classical classification, ordinal classification preserves the order of classes. Here, we propose a novel pairwise approach for learning the ordering of pairs of songs, thereby allowing us to rank songs. In addition, we derive representatives for the individual ordinal classes. They are used to compare a given song with the pairwise model to determine its rank. Doing so, allows inferring the popularity based ordinal classes (cf. Section 3.5) resulting in the final popularity prediction.

To summarize, the contributions of the presented work are: (i) We model the song popularity prediction task as an ordinal classification task; (ii) We propose a novel approach for music popularity prediction based on a learned pairwise comparator; (iii) We present an extensive evaluation comparing different types of models (regression, classification, and pairwise) applied to multiple variations of the ordinal classification task and (iv) share the source code of the approach and all conducted experiments [4].

2. RELATED WORK

In the following, we give an overview of different song popularity prediction approaches and their evaluation.

Frieler et al. [5] utilize melodic features to distinguish successful from non-successful pop songs, resulting in a binary classification task. In [10], Dhanaraj and Logan apply classifier models that consume acoustic and lyrics features to predict whether a song is a hit or not, resulting in a comparable binary classification task. Further, Singhi and Brown [13] use classifier models consuming lyrics features to predict whether a song is in the hit or non-hit class. They consider a song a hit if it made it to the Billboard Year-End Hot 100 singles charts. In contrast, flops (non-hit songs) are considered songs of the same artists that produced hits that did not occur in the charts. In total, their dataset contains 492 hits and 6,323 flops, showing the natural imbalance of the two classes. A different definition of hits and non-hits is used by Ni et al. [14]. They distinguish songs that made it to the top five of the UK charts (hits) from songs that resided in the range 30-40 (non-hits). Pachet and Roy [15] introduce the HiFind Database containing popularity measures as three popularity classes (low, medium, and high).

In contrast, both Yang et al. [2] and Yu et al. [3] use a neural network model trained on a dataset that contains streaming-based popularity measures from KKBOX Inc., a Taiwanese music streaming platform. They model the task as a regression task and predict the popularity value based on listener- and play counts. Similarly, we applied a neural network model to Essentia audio features [7] to predict the highest chart position of a song in a regression setup in our work [1]. In that work, we also compute accuracy scores based on the regression value by transforming the regression value to a hit (range 1-100) and non-hit (any other value) class. Moreover, in a recent work, we created the two datasets HSP-S and HSP-L and used them to compare various models on song popularity prediction tasks [11]. In that work, we tackle regression tasks predicting the top position and weeks in charts. Further, we predict the listening event-based popularity measures listener count, play count, and Yang's hit score [2]. We use these measures to transform the regression tasks into binary classification tasks by splitting the value range using the median. Note that we use two distinct classes rather than a single class for classification. To evaluate the predictive performance of the regression experiments, we use Spearman's ρ and Kendall's τ. Additionally, we report accuracy and F_1 for the classification experiments. In a follow-up work, we provide additional results for further models in [12].

In contrary to the previously presented approaches, ordinal classification approaches utilize the relative order of songs to learn how to rank songs based on popularity. Related previous work on ordinal classification, also called ordinal regression, was done by Ren and Kauffman [16]. They investigate the development of the popularity of a song over time using audio features, and external features such as the artist's voice, or if it was produced by a major label. Based on these features, they aim to predict the ranking of a song in an ordinal classification setup, where they predict popularity for one week in advance. In contrast, we solely use audio descriptors which do not change over time to predict popularity measures. This results in a different prediction setup as we predict an overall popularity measures that is not bound to a specific date (derived from the full observations period covered in a dataset) while they try to predict a popularity measure at a certain date (e.g., they predict the popularity one week in the future).

[4] https://github.com/dbis-uibk/hit-prediction-code/tree/smc2023

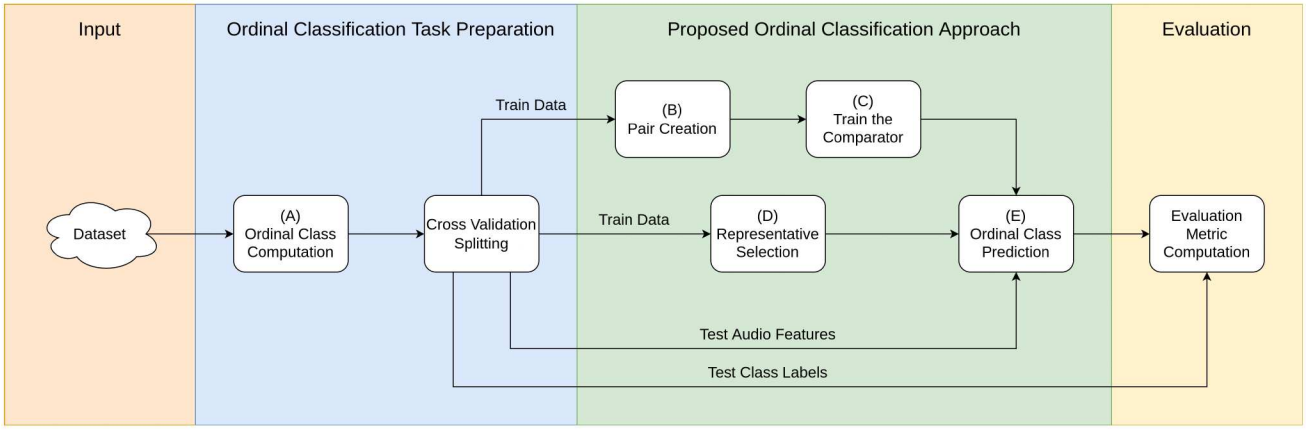


Figure 1: The overall setup of our approach, showing the main components used to compute the ordinal class predictions.

3. METHODS

Opposed to previous regression and classification models that aim to predict the popularity of songs based on the features of a given song, our pairwise approach learns to predict the relative order of *pairs of songs* at its core. The overall setup forming our proposed approach is depicted in Figure 1. In the following, we describe the depicted components: (A) the creation of ordinal classes based on a dataset containing popularity measures, (B) the creation of pairs, (C) the training of our pairwise comparator model, (D) the selection of representatives used for prediction, and (E) the subsequent popularity prediction.

3.1 Ordinal Class Computation

Contrasting previous approaches that model the popularity prediction task as either a regression or classification task [1, 2, 4, 6, 8, 10], we suggest modeling it as an ordinal classification task. Regression approaches assume that the predicted values are in interval scale, while classification approaches ignore the order. We argue that, in general, it cannot be assumed that popularity values form an interval scale as laid out in [12]. One example is the number of weeks in charts where all songs that did not make it to the charts share a value of zero. Assuming an interval scale would mean that these songs are all equally popular. This is obviously not the case when comparing their respective listener or play counts. Using such popularity measures to train a regression model will result in a model that learned that these songs are all equally popular. Hence, this encoding hinders such model to correctly derive the popularity of a song. Using ordinal classes eliminates this problem, as they only define an order between classes without the requirement that all songs within a class are equally popular. Ordinal classes can hence be seen as equivalence classes for a range of popularity measures. If the number of popularity classes is equal to the number of popularity measure values, the transformation results in a more accurate encoding of information without losing information, as it basically just changes the interpretation of the given popularity values. Further, it is possible to bin successive values without violating properties of an ordinal scale, in contrast to regression where equidistant values are a requirement.

To create ordinal classes, we propose to map popularity measures to ordinal classes. For this, we again propose two methods. The first method is quantile binning (i.e., we split the data based on k percentiles), resulting in k balanced classes in terms of number of songs contained. The second method is uniform binning (i.e., we split the data based on k uniformly spaced values), possibly resulting in imbalanced classes. We argue that the latter is the natural choice when considering it from a popularity standpoint. E.g., for chart positions with 100 discrete positions, the most obvious choice would be to assign them to 100 ordinal classes. To reduce the number of classes, it seems natural to combine neighboring classes by splitting the 100 positions into classes that have the same size in terms of positions contained, rather than number of songs contained in each class. Note that this type of binning is independent of the distribution of popularity in a dataset, but it possibly results in an uneven distribution of the numbers of songs per class. Further note that these different kinds of binning are enabled by the ordinal definition of the task.

3.2 Pair Creation

The input to our approach is a set of songs (represented as N feature vectors). To create a predetermined number of pairs, we randomly select pairs of songs from the input. For the feature representation of song pairs, we have two options to combine the songs' features. The first option, *concat*, concatenates the feature vectors of the two songs in the pair. Note that this doubles the vector size. We expect this pair representation to allow learning how to compare the songs' features. The second option, *delta*, computes a delta encoding of the song features; in other words, we compute the difference of both feature vectors, of the two songs in the pair, component-wise. Lastly, we encode the relative ordering of the two songs (a, b) of a pair based on their individual popularity values $p(a)$ and $p(b)$. The function p returns the index of the ordinal class representing the popularity value. The resulting relative order is defined by the following comparator function:

$$C(a,b) = \begin{cases} -1 & \text{for } p(a) < p(b) \\ 0 & \text{for } p(a) = p(b) \\ 1 & \text{for } p(a) > p(b) \end{cases} \quad (1)$$

3.3 Train the Comparator

Our approach aims to train a model that compares two songs to derive their relative order, relying on the feature format (*concat* or *delta*) and a target value (cf. Section 3.2). We train a pairwise comparator model that learns to predict the three classes $\{-1, 0, 1\}$ (cf. Equation (1)), representing the relative order of the pair's two songs. Since we use this prediction to determine the order of two songs, it is sufficient to predict the correct sign or 0 as this prediction is then used to determine the prediction of the ordinal class (cf. Section 3.5). Relying on the sign allows utilizing both classification and regression models as the pairwise comparator without modifying their output.

3.4 Representative Selection

To predict the popularity of a song using the pairwise comparator model, we convert the learned ranking prediction to an ordinal class prediction. We first derive a representative sample for each ordinal class during training to compare them with songs for which we are seeking popularity predictions (cf. Section 3.5). Note that the binning of popularity values in ordinal classes (cf. Section 3.1) does not allow distinguishing the popularity of songs within an ordinal class. Nevertheless, it would be possible to add further labeling to songs within a class that enables that. To keep our results comparable with traditional classification models that cannot use such additional labeling, we decided to only utilize the popularity encoded in the ordinal classes (cf. Section 3.1). Hence, to determine a representative sample, we consider two distinct approaches. For the first approach, we select a random song per class from the training set to act as the representative sample of each class. This random selection leads to a high variance in the predictive performance when the experiment is repeated multiple times. We attribute this to the fact that a purely random selection can either (a) happen to choose a good representative for a class, i.e., a song that is very characteristic of the class, or (b) choose a bad representative, i.e., a song that is effectively an outlier which is dissimilar to the other songs in the same class. Hence, this approach is not suitable to find songs that represent the ordinal class well. These representatives are used to derive predictions; hence, it is crucial to select a good representative. Therefore, we propose a second approach, where we compute the average of the feature vectors of all songs in a given class in the training set. The deterministic nature of this approach leads to stable results in the predictive performance of the overall approach across repeated runs of the same experiment. Therefore, we will utilize this selection method of representatives for our further experiments. Note that this average represents the centroid of each class in terms of features, resulting in an artificial feature vector.

3.5 Ordinal Class Prediction

To predict the ordinal class of a song, we propose a procedure inspired by the widely known insertion sort algorithm. We utilize the ordinal class definition and compare the song with the representatives of all classes (cf. Section 3.4). Beginning with the representative of the least popular class (lowest class index; depending on the popularity measure this can also be the most popular class), we successively compare the song with all further representatives in the order determined by their ordinal popularity class. We continue this as long as the pairwise model, acting as a comparator, predicts a value $>= 0$, meaning that the currently compared representative is less popular (in terms of class index) than the given song. Hence, we predict the first class for which the pairwise model predicts a value of < 0 as its popularity class. This results from our interpretation of the representatives. We account the song that resides "between" two representatives to the latter, as otherwise, it would not be possible to ever predict a song for the first popularity class following the same procedure.

4. EXPERIMENTS

This section describes the experiments conducted to evaluate the pairwise approach for popularity prediction modeled as an ordinal classification task. We compare our approach to baseline models performing traditional classification or regression tasks to show the usefulness of the proposed ordinal regression approach. We train the baseline regression models on the index of a one-hot encoded representation of the ordinal classes, and then transform their prediction to the closest class (index in the one-hot encoded vector). To investigate the impact of the number of classes, we run experiments with different numbers of classes: two classes, five classes (inspired by [17]), and from there on in steps of five up to 100 classes. Note that using many classes can be considered an approximation of a regression task, while small numbers are comparable with previous binary classification tasks.

4.1 Evaluated Models

We evaluate three types of models: (1) classification models, (2) regression models, and (3) our pairwise ordinal classification approach and chose baseline models based on their results in [12].

For classification, we use a variety of models to compare our approach against: The logistic regression classifier (Logit) of scikit-learn [18] as a representative for a linear classification model, and a dense feed-forward neural network using the multi-layer perceptron classifier (MLPC) of scikit-learn. Similar to the logistic regression model, we relied on the default parameters except for the layer structure, as this would result in a single hidden layer with 100 neurons as this would result in a rather small network compared to previous network-based approaches (cf. [1–3]). In contrast to this default, we use five hidden layers. The first hidden layer following the input layer has a size of 256, followed by three layers with 128 neurons and a further hidden layer containing 64 neurons. This results in a neural network with seven layers in total, including the input and output layer. We enabled early stopping provided by the scikit-learn with a maximum of 200 epochs.

In addition, we evaluate regression models for prediction: a linear regression model (Linear) and a MLP regression

model (MLPR). In both cases, we utilize the scikit-learn implementations and keep the default parameters. Again, we change the layer configuration of the MLP to be the same as the classifier model. To arrive at predictions in the form of ordinal classes, we map the prediction of the regression model to the closest ordinal class, effectively converting a single value to a one-hot encoded vector.

For our pairwise approach, we use the four models introduced above to learn the comparator at its core. As stated in Section 3.5 it is possible to use the output of a regression model as a comparator without further modification.

4.2 Experimental Setup

The main goals of the experiments are to tune our pairwise approach and to compare our proposed pairwise approach with classical regression and classification models.

Our experiments are based on the HSP-S and HSP-L datasets published in our previous work [11]. These datasets mainly differ in their size, as the HSP-S dataset is balanced in terms of hits and non-hits (hits are defined by a song's occurrence in the Billboard Hot 100). The HSP-S dataset contains 7,736 songs while HSP-L contains 73,482 songs. Further, they differ in the distribution of popularity measures. The different properties of both datasets (HSP-S and HSP-L) allow showing the generalizability of the results.

As input features, we use the set of Essentia audio features [7] already utilized by [1,11,12] to predict the ordinal class derived from Yang's hit score [2] (cf. Section 3.1). These features include the low-level descriptors extracted by Essentia describing bark bands, erb-bands, mel-bands, and average loudness to name some of the features that describe the spectral and dynamics of a song. Further, the input includes high-level features capturing the mood, vocal, genre, and danceability derived using the classifiers included in Essentia [5] . Yang's hit score is computed by multiplying the logarithm of the listener count with the logarithm of the play count, and has been shown to be well suited for popularity prediction. We run experiments using both binning strategies presented in Section 3.1 and different numbers of classes in a 5-fold cross validation setup. In particular, we aim to determine how quantile binning and uniform binning impacts the performance of the different models. This allows analyzing the effects of the distribution of the number of songs contained in each class. Further, we aim to gather insights on how models are affected by different numbers of ordinal classes. Note that in case of uniform binning, our pairwise approach cannot make predictions for large numbers of classes. This is due to the dataset size, as splitting it to a larger number of classes leads to empty classes, preventing the selection of a representative for this class that is needed for the prediction (cf. Section 3.5). This effect can be observed beginning at 45 (HSP-S) or 55 classes (HSP-L).

Inspired by ordinal classification, we use a confusion matrix to derive multiple classification metrics. We compute correlation scores based on this confusion matrix. Sakai et al. [19] evaluated different evaluation measures for ordinal prediction and suggest to use Cohen's linear weighted kappa [20] for ordinal classification. We follow this suggestion and use Cohen's linear weighted kappa as the primary evaluation metric for ordinal classification tasks. Note that the ordinal information encoded in the confusion matrix by the order of classes allows computing error metrics such as mean absolute error (MAE).

[5] These features are further described in Essentia's documentation: `https://essentia.upf.edu/streaming_extractor_music.html`.

5. RESULTS AND DISCUSSION

In the following, we present the results of our experiments. First, we share insights into how different configurations of the pairwise approach perform in terms of predictive power. Second, we present a comparison of the pairwise approach with baseline models. It is important to note that for Cohen's linear weighted kappa, higher values are better and for the MAE, lower values are better. Further, the $@n$ (e.g., $@5$ and $@10$) notation in Table 1 refers to the number of ordinal classes used for the particular experiment.

5.1 Different Pairwise Approach Setups

In a first step, we investigate the effects of the feature encoding strategies for our pairwise approach on both datasets. The results can be seen in Table 1. The experiments on the HSP-S dataset reveal that *delta* encoding overall leads to significantly (paired Student's t-test; $p < .05$) better results than *concat* encoding when comparing the individual results of the folds of all selected comparator models for our pairwise approach. For the HSP-L dataset, this also holds for linear comparator models (linear and logistic regression) but it does not hold true for the multi-layer perceptron classifier (MLPC) and regressor models (MLPR) comparators. For these, *concat* performs significantly better than *delta*. We suspect that this is due to the large number of songs. This might enable the MLP comparator models to learn to compare the concatenated features of two songs to determine their relative order. Nevertheless, linear models overall outperformed these MLP models and our experiments show that *delta* encoding is the preferred type of encoding for linear models. Consequently, we present detailed results utilizing this encoding (cf. Section 5.2).

Second, we investigate the impact of different comparator models on our pairwise approach. Comparing all four comparator models on the HSP-S dataset using uniform binning shows that the linear comparator models (linear and logistic regression) significantly outperform the multi-layer perceptron models. Table 1 shows that linear models achieve approximately twice the kappa score of the multi-layer perceptron models. This also holds true for MAE. The finding that neural network-based comparators achieve a lower kappa score compared to their linear counterpart is also evident on the HSP-L dataset for uniform binning. Similar behavior can be seen on the HSP-S dataset with quantile binning. This indicates that for the current pairwise model setup, linear models outperform more complex models such as the utilized neural network model. We hypothesize that only encoding the order of songs in a pair (cf. Section 3.2) might not be descriptive enough to tune

			HSP-S				HSP-L			
			Kappa		MAE		Kappa		MAE	
Binning	**Model**	**Encoding**	@5	@40	@5	@40	@5	@40	@5	@40
Uniform	Linear	-	**0.212 (0.009)**	**0.212 (0.008)**	**1.027 (0.022)**	**8.001 (0.298)***	**0.194 (0.005)***	0.158 (0.003)	**1.098 (0.008)***	8.935 (0.179)
Uniform	Logit	-	0.207 (0.014)	0.173 (0.010)	1.052 (0.013)	8.465 (0.189)	0.156 (0.005)	0.134 (0.007)	1.173 (0.013)	9.493 (0.269)
Uniform	Linear Pairwise	concat	0.199 (0.007)	0.184 (0.007)	1.091 (0.021)	10.151 (0.410)	0.169 (0.004)	**0.164 (0.004)**	1.123 (0.017)	10.267 (0.129)
Uniform	Linear Pairwise	delta	0.210 (0.009)	0.202 (0.004)	1.065 (0.033)	9.688 (0.376)	0.171 (0.003)	**0.164 (0.003)**	1.121 (0.012)	10.291 (0.173)
Uniform	Logit Pairwise	concat	0.190 (0.007)	0.177 (0.008)	1.147 (0.024)	10.502 (0.394)	0.150 (0.005)	0.140 (0.002)	1.221 (0.027)	10.895 (0.346)
Uniform	Logit Pairwise	delta	0.203 (0.008)	0.198 (0.008)	1.087 (0.029)	9.853 (0.423)	0.169 (0.003)	0.153 (0.002)	1.163 (0.016)	10.603 (0.306)
Uniform	MLPR	-	0.157 (0.015)	0.155 (0.019)	1.117 (0.026)	8.546 (0.203)	0.140 (0.006)	0.160 (0.007)	1.163 (0.010)	**8.720 (0.181)***
Uniform	MLPC	-	0.161 (0.022)	0.161 (0.015)	1.111 (0.024)	8.638 (0.125)	0.137 (0.005)	0.148 (0.002)	1.186 (0.013)	9.241 (0.279)
Uniform	MLPR Pairwise	concat	0.103 (0.008)	0.076 (0.014)	1.435 (0.062)	14.329 (0.968)	0.113 (0.008)	0.095 (0.020)	1.445 (0.035)	13.246 (1.173)
Uniform	MLPR Pairwise	delta	0.117 (0.011)	0.104 (0.011)	1.395 (0.038)	13.103 (0.420)	0.081 (0.005)	0.071 (0.002)	1.552 (0.049)	14.071 (0.510)
Uniform	MLPC Pairwise	concat	0.075 (0.009)	0.062 (0.006)	1.581 (0.053)	15.147 (0.351)	0.111 (0.008)	0.099 (0.011)	1.467 (0.060)	12.940 (0.818)
Uniform	MLPC Pairwise	delta	0.109 (0.013)	0.095 (0.008)	1.462 (0.027)	13.221 (0.558)	0.071 (0.005)	0.065 (0.004)	1.613 (0.042)	14.848 (0.652)
Quantile	Linear	-	0.210 (0.008)	0.211 (0.010)	**1.053 (0.011)***	**8.708 (0.100)***	0.153 (0.003)	0.155 (0.003)	**1.088 (0.003)***	**8.944 (0.032)***
Quantile	Logit	-	0.255 (0.009)	0.225 (0.010)	1.247 (0.016)	10.903 (0.193)	0.223 (0.006)	0.203 (0.004)	1.362 (0.015)	12.113 (0.064)
Quantile	Linear Pairwise	concat	0.266 (0.016)	0.247 (0.012)	1.280 (0.025)	11.726 (0.198)	0.253 (0.005)	0.230 (0.014)	1.342 (0.010)	12.471 (0.147)
Quantile	Linear Pairwise	delta	**0.276 (0.009)**	**0.264 (0.013)**	1.261 (0.014)	11.362 (0.270)	**0.255 (0.004)***	**0.236 (0.006)**	1.331 (0.005)	12.397 (0.068)
Quantile	Logit Pairwise	concat	0.259 (0.012)	0.247 (0.016)	1.312 (0.028)	11.963 (0.344)	0.224 (0.006)	0.206 (0.003)	1.394 (0.009)	13.056 (0.052)
Quantile	Logit Pairwise	delta	**0.276 (0.014)**	0.260 (0.009)	1.272 (0.027)	11.586 (0.187)	0.241 (0.005)	0.222 (0.005)	1.360 (0.008)	12.712 (0.044)
Quantile	MLPR	-	0.182 (0.021)	0.180 (0.012)	1.240 (0.029)	10.397 (0.177)	0.157 (0.003)	0.166 (0.007)	1.269 (0.011)	10.116 (0.191)
Quantile	MLPC	-	0.190 (0.021)	0.180 (0.027)	1.292 (0.026)	11.269 (0.493)	0.155 (0.003)	0.173 (0.007)	1.361 (0.005)	11.582 (0.130)
Quantile	MLPC Pairwise	concat	0.126 (0.016)	0.102 (0.014)	1.624 (0.043)	15.799 (0.466)	0.190 (0.010)	0.130 (0.035)	1.515 (0.029)	15.708 (0.862)
Quantile	MLPC Pairwise	delta	0.151 (0.029)	0.144 (0.021)	1.572 (0.056)	14.675 (0.306)	0.104 (0.007)	0.097 (0.008)	1.681 (0.014)	16.079 (0.170)

Table 1: Summary of our results containing linear-weighted kappa scores and MAEs on both datasets and binning for baseline models and multiple setups of our pairwise approach with standard deviation in brackets. The best results per dataset, binning, and metric are in bold. $\star$ means significantly better (tested with a paired Student's t-test; $p < .05$)

the large number of parameters for the utilized multi-layer perceptron approach.

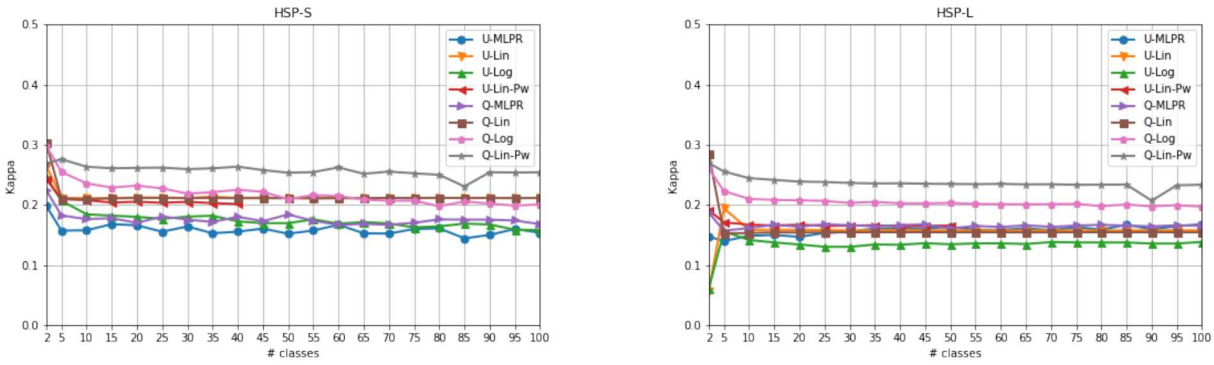
As a further observation, the linear regression model as a comparator obtains significantly higher kappa scores compared to the logistic regression model, across both datasets and class mappings. Similar effects can be seen when comparing the respective MAEs. The same effects can be observed by comparing MLPR and MLPC comparator models for our pairwise model based on kappa score. We lead this back to the comparator (cf. Section 3.3): regression models are not strictly bound to the three values $\{-1, 0, 1\}$ for prediction, while classification models are bound to these three classes. As a consequence, we only report detailed results for different numbers of classes using a linear regression comparator model in the following section.

5.2 Different Numbers of Ordinal Popularity Classes

Figure 2a shows the results for different models applied to the HSP-S dataset using different numbers of ordinal classes. We provide results for a multi-layer perceptron regression model (MLPR), a linear regression (Lin), a logistic regression model (Log) and our pairwise approach utilizing *delta* encoding and a linear regression model as its comparator (Lin-Pw). Further, we include results for both uniform (L) and quantile binning (Q). E.g., L-Lin-Pw stands for linear binning with a linear regression comparator used in the pairwise model. The pairwise model applied to uniform binning could not be evaluated for 45 classes and above. The reason for this is that we need to select representatives (cf. Section 4.2) that are not available for all ordinal classes beyond this number of classes.

For both datasets, the pairwise approach with linear regression as its comparator resides among the best performing models in terms of kappa score for uniform binning. It significantly outperforms all models on both datasets, except for the pure linear regression model on the HSP-S dataset. Note that for uniform binning we only included results up to 40 classes for HSP-S and up to 50 classes for HSP-L in our significance tests as we do not have results for higher numbers of classes for the pairwise approach (cf. Section 3.5). Further, we see in both Figure 2a and Figure 2b that the linear pairwise approach significantly outperforms all other approaches on quantile binning. An explanation for this effect could be that quantile binning leads to the same number of songs for each class. Hence, it is equally likely for each class's songs to be selected for each pair—in contrast to the uniform binning experiments where this is dependent on the distribution of songs among the classes. This effect is caused by the random selection of songs during the pair creation (cf. Section 3.2), which leads to a lower probability of being selected for samples from a class with lower numbers of songs. Additionally, the logistic regression model as a representative of classifier models benefits from the balanced numbers of songs per class resulting from quantile binning. We observe that on both datasets it performs significantly better than the same model run on uniform binning. Further, the logistic regression classifier model performs significantly better than the linear regression model on the HSP-S as well as the HSP-L dataset with quantile binning. Further, we observe that the opposite is true for uniform binning. For this setup, the linear regression model significantly outperforms the logistic regression classifier model.

The MLP regression model reveals comparable results as the linear regression models on the HSP-L dataset (cf. Figure 2b), independent of the used binning. While the MLP regression model performs worse than the linear regression model on the smaller HSP-S dataset (cf. Figure 2a). These findings are similar to the results we reported in our previous work [11], where we showed that simple linear models achieve comparable results to those achieved by neural network models. This is similar to the observations



(a) Cohen's linear-weighted kappa on the HSP-S dataset.

(b) Cohen's linear-weighted kappa the HSP-L dataset.

Figure 2: Comparison of baseline models (*MLP R*egression, *Lin*ear Regression, *Log*istic Regression) with our pairwise approach (*Lin-Pw*) for different numbers of classes. (*U* and *Q* in the model name indicates the binning *U*niform or *Q*uantile.)

that we made in [12]. A reason for that could be that every MLP should be able to converge to a linear regression model, given that it has seen enough training samples compared to the number of tuneable parameters. This could also explain why both types of models perform similarly well on the larger HSP-L dataset. Further, it might explain why the neural network-based model performs worse on the smaller HSP-S dataset, as HSP-L is approximately ten times the size of HSP-S. Nevertheless, additional experiments are necessary to test if the hypothesis that the MLP model converges to a linear regression model holds true.

The tested models show constant performance for larger numbers of classes. Performance differences between regression and classification models looking at both datasets can likely be explained by the fact that the HSP-S dataset is balanced in terms of hits and non-hits. This means that half of this dataset contains very popular songs, while there is no such restriction for the HSP-L dataset. Further, we see that the binary case (two classes; popular/unpopular) is easier to predict than larger numbers of classes.

In summary, we find that the pairwise approach outperforms other models if the ordinal classes are created using quantile binning, and it achieves comparable results to other approaches using uniform binning. Hence, the proposed approach improves the predictive performance.

6. LIMITATIONS

We acknowledge that the used datasets are biased toward commercial western music. Further, the used measure of popularity has a platform bias as it is solely based on data stemming from last.fm. Hence, it remains to be shown that our pairwise approach can perform similarly well on other types of music and other definitions of success. Nevertheless, we believe that it is generalizable. Further, the current findings are solely based on experiments using audio features. Note that audio features alone cannot grasp external factors that might impact the popularity of a song. Such factors are, for instance, money invested in advertising the song, the general "visibility" of a song such as its usage in a movie, or concerts given by the artists. Most of these factors, especially those related to the money invested for advertising a song, are not publicly available. Therefore, it remains unclear whether our approach performs well utilizing external factors to predict the popularity of a song. Further, the overall approach of predicting the future success of a song has natural limitations. There will be unforeseeable events in the future that might have effects on the future popularity of a given song. E.g., there is no way to predict, which other songs will be released at the same time and compete with the song. This obviously affects popularity measures such as the peak position in charts. Such limitations are also present in other prediction tasks. E.g., future stock prices being influenced by geopolitical factors or the COVID-19 pandemic being influenced by future mutations and unknown properties thereof.

7. CONCLUSION

In this work, we presented a task formulation for song popularity prediction in the form of an ordinal classification task. This can be considered a combination of the best fitting properties of previous task formulations that either modeled song popularity prediction as a regression task or a classification task. Modeling the task as an ordinal classification task allows training classification as well as regression models on the particular task. Note that using two classes results in a binary classification task and using many classes is comparable to a regression task with the same number of discrete values. This highlights the flexibility of the proposed ordinal classification task formulation in comparison to a plain regression of classification task formulation. Further, we propose a novel pairwise approach. At its core, this approach learns a model that predicts the relative order of pairs of songs. Additionally, representatives that separate the ordinal classes are computed during the train phase of the model. Using this method, it is possible to predict the popularity of a given song by comparing it to its representatives. We show that (1) our pairwise approach outperforms previous regression and classification approaches, (2) that this holds true for different datasets, and (3) that our approach can be applied using

many models and setups, which demonstrates the generalizability of the proposed approach. For the presented work, we decided to keep the input as ordinal classes to maintain comparability to the plain classification task. In future work, we plan to investigate whether the approach could further be improved by utilizing the exact popularity value to train the model instead of the ordinal class representation. This would also allow computing better fitting representatives by selecting them close to the boundary between two ordinal classes. Additionally, we plan to investigate the effects of more complex pair creation strategies that consider the distribution of songs among popularity classes. We hypothesize that this might contribute to improving the performance of the uniform binning setup by compensating for the class imbalance. Hence, it will be interesting to see whether more advanced pair creation techniques will improve the overall performance of the model.

8. REFERENCES

[1] E. Zangerle, M. Vötter, R. Huber, and Y.-H. Yang, "Hit Song Prediction: Leveraging Low- and High-Level Audio Features," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 319–326.

[2] L.-C. Yang, S.-Y. Chou, J.-Y. Liu, Y.-H. Yang, and Y.-A. Chen, "Revisiting the problem of audio-based hit song prediction using convolutional neural networks," in *Proceedings of the IEEE International Conference Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 621–625.

[3] L.-C. Yu, Y.-H. Yang, Y.-N. Hung, and Y.-A. Chen, "Hit Song Prediction for Pop Music by Siamese CNN with Ranking Loss," *arXiv preprint arXiv:1710.10814*, 2017.

[4] J. Lee and J. Lee, "Music Popularity: Metrics, Characteristics, and Audio-Based Prediction," *IEEE Transactions on Multimedia*, vol. 20, no. 11, pp. 3173–3182, 2018.

[5] K. Frieler, K. Jakubowski, and D. Müllensiefen, "Is it the song and not the singer? Hit song prediction using structural features of melodies," *Yearbook of Music Psychology*, pp. 41–54, 2015.

[6] F. Pachet, "Hit Song Science," in *Music Data Mining*, 1st ed. Chapman & Hall/CRC Press Boca Raton, 2012, pp. 305–326.

[7] D. Bogdanov, N. Wack, E. Gómez Gutiérrez, S. Gulati, H. Boyer, O. Mayor, G. Roma Trepat, J. Salamon, J. R. Zapata González, X. Serra *et al.*, "Essentia: An audio analysis library for music information retrieval," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2013, pp. 493–498.

[8] D. Herremans, D. Martens, and K. Sörensen, "Dance hit song prediction," *Journal of New Music Research*, vol. 43, no. 3, pp. 291–302, 2014.

[9] J. Fan and M. Casey, "Study of Chinese and UK hit songs prediction," in *Proceedings of the International Symposium on Computer Music Multidisciplinary Research (CMMR)*, 2013, pp. 640–652.

[10] R. Dhanaraj and B. Logan, "Automatic Prediction Of Hit Songs," in *In Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, 2005, pp. 488–491.

[11] M. Vötter, M. Mayerl, G. Specht, and E. Zangerle, "Novel Datasets for Evaluating Song Popularity Prediction Tasks," in *IEEE International Symposium on Multimedia (ISM)*, 2021, pp. 166–173.

[12] M. Vötter, M. Mayerl, G. Specht, and E. Zangerle, "HSP Datasets: Insights on Song Popularity Prediction," *International Journal of Semantic Computing (IJSC)*, pp. 1–23, May 2022.

[13] A. Singhi and D. G. Brown, "Hit song detection using lyric features alone," *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[14] Y. Ni, R. Santos-Rodriguez, M. Mcvicar, and T. De Bie, "Hit song science once again a science?" in *Proceedings of the International Workshop on Machine Learning and Music (MML)*, 2011, pp. 2–3.

[15] F. Pachet and P. Roy, "Hit Song Science Is Not Yet a Science." in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2008, pp. 355–360.

[16] J. Ren and R. J. Kauffman, "Understanding music track popularity in a social network," *Proceedings of the European Conference on Information Systems (ECIS)*, pp. 374–388, 2017.

[17] Y. Kim, B. Suh, and K. Lee, "#nowplaying the future billboard: mining music listening behaviors of twitter users for hit song prediction," in *Proceedings of the International Workshop on Social Media Retrieval and Analysis (SoMeRA)*, 2014, pp. 51–56.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research (JMIR)*, vol. 12, pp. 2825–2830, 2011.

[19] T. Sakai, "Evaluating Evaluation Measures for Ordinal Classification and Ordinal Quantification," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2021, pp. 2759–2769.

[20] J. Cohen, "Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit," *Psychological Bulletin*, vol. 70, no. 4, pp. 213–220, 1968.