Modeling Temporal Dynamics: The Impact of User Rating Sequences on Future Song Rating Predictions

Zhuoyang Hao 1255309 Keith Howen 1214132 Zilin Su 1155122

1 Introduction

In the digital era, music streaming platforms have access to large amounts of user data, including detailed records of user interactions and preferences. Understanding user ratings and preferences over time can provide valuable insights for personalized recommendations. This project explores how temporal dynamics in user ratings impact their preferences using the Yahoo Music dataset.

The Yahoo Music dataset [1] contains user interactions with various musical items, including songs, albums, artists, and genres. Each record in the dataset includes fields such as `user_id`, `item_id`, `score` and `time`, representing the time sequence of ratings for each user.

The key research question addressed in this study is: *Do the temporal dynamics of user ratings affect user preferences?* To explore this, we design a classification problem where the goal is to predict the user's rating category (Low, Medium, High) for each song in the test set. We employ three models: Long Short-term Memory network (LSTM) to capture the sequential patterns in user ratings, Neural Additive Models (NAM) to model feature importance under the assumption that ratings within each user are independent (that is, NAM solely accounts for temporal dynamics through an ordinal timestep variable signifying the order of rating predictions for each user), and Transformer-based models for leveraging complex temporal dependencies.

Through this study, we aim to uncover patterns in user preferences over time and evaluate the efficacy of different machine learning models in predicting future user ratings based on historical data. The insights drawn from this research can contribute to improved recommendation systems on music streaming platforms.

2 Literature review

Despite the vast amount of relationships that Deep Neural Networks/DNNs could capture, the interpretability of such models remains as an obstacle. The NAM aims to partially mitigate this obstacle by extending the Generalized Additive Model/GAM, which aim to estimate $f(x_1, .., x_p)$ in the extended generalized linear model $g(E(y)) = f(x_1, .., x_p)$ as $f_1(x_1) + \cdots f_p(x_p)$ (where f_i is a nonlinear function), with the possibility of the f_is themselves being DNNs, allowing for complex, nonlinear relationships. The additive nature of this model makes it interpretable; the contributions of each feature are independently additive, allowing the possibility to visualise how the NAM arrives at its predictions [2].

Sequential data, characterized by temporal dependencies, is common in various domain, including natural language processing, speech recognition, and music ratings. Recurrent Neural Networks (RNNs) are designed to process sequential data by using a recurrent structure where the output from the previous step is fed as input to the next. However traditional RNNs suffer from the vanishing gradient problem, which limits their ability to learn long-term dependencies in sequences [3]. To address this limitation, Long Short-term Memory (LSTM) were introduced. LSTMs use memory cells and gating mechanisms to selectively retrain relevant information over long sequences, thus improving performance on sequential tasks [4].

The Transformer architecture [5] revolutionized sequence modelling by using self-attention mechanisms to capture global dependencies in the data. Unlike RNNs and LSTMs, which process sequences sequentially, Transformers can attend to different parts of a sequence simultaneously, enabling faster training and better handling of long-range dependencies. This property makes Transformers highly effective for modeling complex sequential patterns in large datasets.

3 Methods

3.1 Feature construction and preprocessing

In this project, the first step involved concatenating the training and validation sets based on the `user_id` field to ensure consistency across users. The `item_id` in the dataset can correspond to a track, artist, album, or genre. To

distinguish these categories, we created four additional binary columns: `isTrack`, `isArtist`, `isAlbum`, and `isGenre`, which indicate whether the `item_id` belongs to one of these respective categories.

The user ratings, originally numerical, were transformed into three discrete categories: ratings greater than 70 were labeled as "High" (label 2), ratings between 30 and 70 were labeled as "Medium" (label 1), and ratings less than 30 were labeled as "Low" (label 0). This transformation simplifies the rating prediction task into a multi-class classification problem.

To introduce the temporal dynamics into Transformers and NAM, we engineered an additional feature `timestep` representing the order in which each use provided their ratings. For each user, the method goes through their ratings and assigns a sequential number starting from 1, which serves as the timestep. This number reflects the order in which the ratings appear, ensuring that the chronological order is maintained for each user individually.

During preprocessing, we observed an imbalance in the sequence lengths of user interactions, which could impact model performance. To address this, we chose an entropy threshold of 1.3825 based on exploratory analysis, as users with entropy values above this threshold exhibited a more balanced and diverse rating pattern across different item categories. This filtering step ensured that we focused on users with more variability in their ratings (i.e. rated a diverse set of items), thereby improving the quality of the training data for our models.

After preprocessing, the dataset contains 36,627 ratings from 441 users.

user_id	item_id	ratings	istrack	isalbum	isartist	isgenre	ratings_discretized	timestep
716610	0	80	0	0	0	1	2	1

Table 1: Example of a user rating with timestep

3.2 Algorithms

Long Short-term Memory network (LSTM)

This model integrates sequences of user interactions with items, additional features (such as whether the interaction involves a track, album, artist, or genre), and uses nested cross-validation to tune hyperparameters and ensure robust and accurate predictions. To improve efficiency, we sort the input sequences by length and pad the shorter sequences, ensuring that all sequences in a batch can be processed effectively. The padded parts of the sequences are masked in the training prcess to ensure they do not affect the cross-entropy loss calculation.

The core model consists of an embedding layer, an LSTM layer, and a fully connected layer. First, the `itemid` is converted into a 64-dimensional vector by the embedding layer, which helps the model capture item similarities. Additional features are then combined with the item embeddings and processed by the LSTM layer, providing the model with more context for making predictions. Finally, the LSTM output is passed through a fully connected layer to produce the predicted ratings for each interaction.

Neural Additive Models (NAM)

This model is essentially a multicategorical logistic model, where the log-odds of being in class i is $g_i(x_1, \cdots, x_K) = \mathbf{w}_i^T (f_1(x_1), \cdots f_K(x_K))$ for $f_i,\ i = 1, \cdots, K$ being a feature-specific neural network and $\mathbf{w}_i$ being class-specific weight vectors. Class membership is then determined by converting log-odds into softmax probabilities and selecting the class with highest probability. This model assumes that interactions within each user are somewhat independent, so the resulting rating prediction only depends on the additive contribution of the 4 binary variables indicating the item category being rated, the timestep value (the order in which a user rates its items) and the `itemid`. The architecture consisted of multiple feature-specific neural networks, where each network (with fully connected layers) models the contributions of the individual features. In addition, the model also includes a neural network for embedding `itemid`s, which includes an embedding layer and fully connected layers, producing a scalar contribution. Finally, the outputs of these feature-specific networks are then passed through a fully connected layer to compute the logits.

Transformer

This model utilizes both item-specific information and auxiliary features to predict the rating category (Low, Medium, High) in a multi-class classification task. The model processes the following inputs: `item_id`, track, album, artist, and genre information. The model includes an embedding layer with an output size of 128 to transform the `item_id`. Instead of using a timestep feature, the model applies *positional encoding* in [5] to capture the sequential nature of the input data, allowing the model to understand the order of interactions. The embedded `item_id` is concatenated with the auxiliary features (track, album, artist, and genre), and this combined feature vector is passed through a Transformer encoder with multiple layers and attention heads. After passing through the Transformer encoder, the output is processed by a fully connected layer, which maps the encoded features into 3 classification logits corresponding to the rating categories (Low, Medium, High).

3.3 Cross Validation

The dataset was partitioned based on user IDs to prevent data leakage, ensuring that interactions from the same user did not appear in both training and validation sets. This entire process ensures that the model is evaluated on unseen data, providing a reliable estimate of its generalization ability.

We implemented nested cross-validation, where the outer loop is used for evaluating model performance, and the inner loop is used for hyperparameter tuning. The outer loop divides the dataset into 20 folds, with each fold serving sequentially as the outer testing set and the remaining folds as the outer training set. For each outer training set, we further split it into 5 folds for hyperparameter tuning. The best hyperparameters were then used to train the model on the outer training set, with early stopping implemented (patience of 3 epochs) to prevent overfitting. We incorporate early stopping in all three models training process to reduce overfitting and thereby improving generalization.

To ensure a fair comparison among different algorithms, we used the same data partitioning scheme for all algorithms. By setting the same random seed, we ensured consistency in the data splits each time. This means that each algorithm was trained and evaluated on identical training and validation sets, making the comparison of model performance equitable.

4 Results and Discussion

	Accuracy	Precision	Recall	F1-score	Precision Error bars (in percentage)
LSTM	0.5083	0.4160	0.4255	0.3870	41.6±2.20
NAM	0.4924	0.4301	0.4184	0.3860	43.01±1.70
Transformer	0.4935	0.2787	0.3736	0.2980	27.87±4.15

Table 2: Metrics of 3 Algorithms

Table 2 compares the different metrics of 3 algoritms. Since user experience is crucial in the recommendation system, we choose to further analyze the precision (i.e. we don't want users to be recommended music that they are not interested in). From Figure 1, we can see LSTM performs better than Transformer, which preliminarily shows that users' rating behavior would change over time. For such locally dependent rating data, Transformer's complexity and global dependency capture capabilities are not fully utilized.

Figure 2 and Figure 3 illustrate that LSTM and Transformer both performs well in predicting High ratings, while struggling with the Medium category.

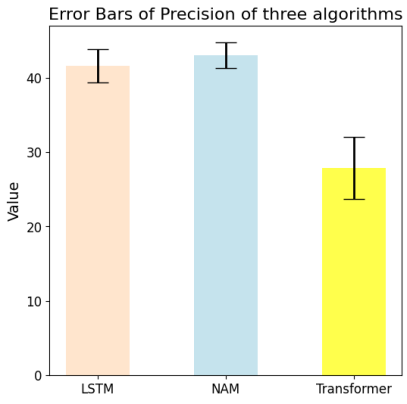


Figure 1: Precision Comparisons

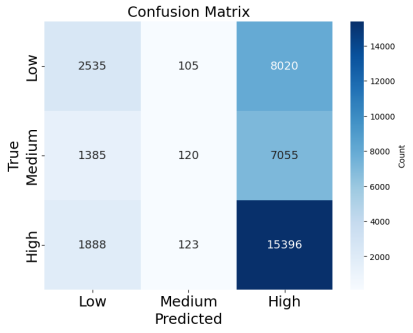


Figure 2: Transformer Confusion Matrix

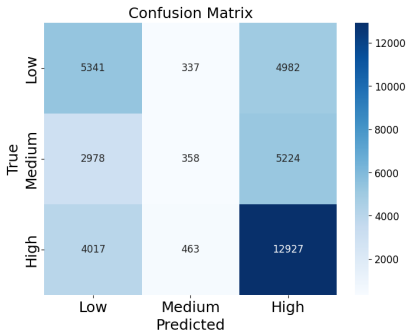


Figure 3: LSTM Confusion Matrix

The NAM was fit into the entire dataset to get an idea of how it makes its predictions during cross-validation. Right hand side of Figure 4 shows the feature contributions $w_{ik}f_k(x_k)$ to the logits (hence probabilities) of classes $0 \leq i \leq 2$ and features $1 \leq k \leq 5$ (excluding `itemid`). From the right hand side of Figure 4, the timestep feature seems to yield a linear contribution to the log-odds of being in a particular class. While the slopes of the contributions for class 0/1 are slightly steeper (though not clearly visible due to font size), this difference in slope compared to class 2 exhibits a multiplicative increase in odds of being assigned to class 0/1 compared to 2. This suggests that overtime, on average users are less likely to give extremely high rating (class 2). Related to this, further empirical analysis shows that (as per the left plot in Figure 4) overtime users are, on average, more likely to give moderate ratings. This could be deemed as one attempt to model temporal dynamics, however to an extent this may be a simplification since, in addition to rating class 2 dominating the distribution, the model struggles in distinguishing class 2 from the other classes (Figure 5a, green curve). This simplification, as a consequence of modelling features in isolation, is further highlighted by the 2nd row in the right plot of Figure 4. Namely, user interactions with items other than tracks tend to influence extreme ratings, shown by the extreme negative weights assigned to such features for class 1.

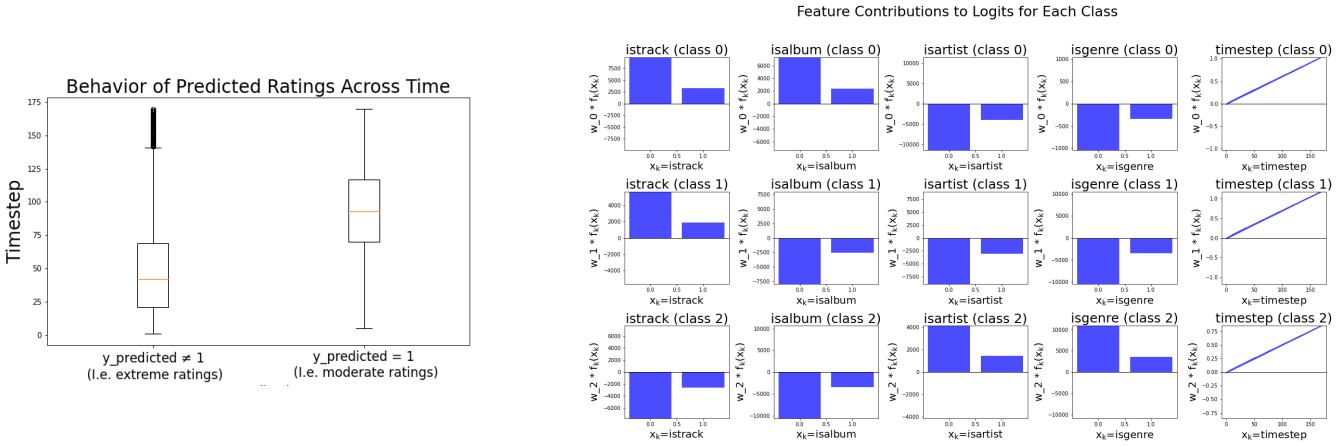
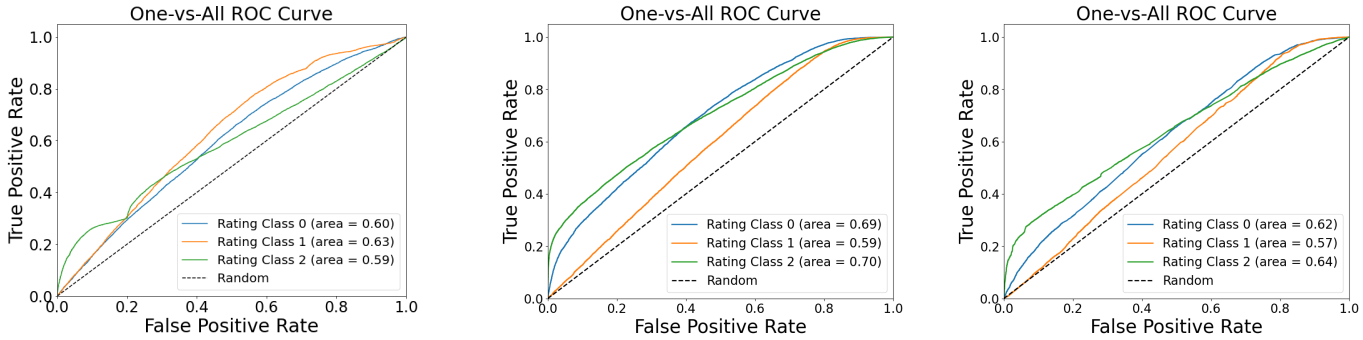


Figure 4: NAM Prediction Overtime and Feature Contributions

From Figure 5, the LSTM model demonstrates the best overall performance in capturing the temporal dynamics of user ratings, particularly for Class 2 (the majority class), where it achieves an AUC of 0.69, suggesting that temporal patterns significantly influence user preferences in this class. The Transformer model performs moderately, with a lower AUC of 0.64 for Class 2 and struggling more with Class 1 (AUC of 0.57), indicating that while it captures some long-range dependencies, it doesn't leverage temporal information as effectively as the LSTM. The NAM model shows balanced, yet lower, AUC scores (around 0.60 for all classes), suggesting that its non-temporal nature limits its ability to capture sequence information, although it slightly improves Class 1 prediction compared to the Transformer. In conclusion, the LSTM provides the strongest evidence that temporal dynamics affect user preferences, while the Transformer and NAM models show limited success, particularly for minority classes.



(a) NAM ROC (b) LSTM ROC (c) Transformer ROC

Figure 5: ROC Curves for all three models

5 Conclusion

In conclusion, our analysis shows that temporal dynamics play a significant role in shaping user preferences, with the LSTM model performing best due to its ability to capture temporal dependencies. The Transformer and NAM models showed moderate success, with the Transformer slightly better at capturing long-range dependencies but still underperforming compared to the LSTM, and NAM struggling without a strong temporal component. The LSTM's superior performance, particularly for the majority class, supports the hypothesis that sequential information is crucial for predicting user ratings, emphasizing the importance of modeling time in user behavior analysis.

References

[1] Yahoo! Inc. *Yahoo! Music user ratings of musical tracks, albums, artists and genres.* `https://webscope.sandbox.yahoo.com/catalog.php?datatype=c&did=48`. 2011.

[2] Rishabh Agarwal et al. "Neural Additive Models: Interpretable Machine Learning with Neural Nets". In: *arXiv preprint arXiv:2004.13912* (2020).

[3] S Hochreiter. "Long Short-term Memory". In: *Neural Computation MIT-Press* (1997).

[4] Alex Graves. "Generating sequences with recurrent neural networks". In: *arXiv preprint arXiv:1308.0850* (2013).

[5] A Vaswani. "Attention is all you need". In: *Advances in Neural Information Processing Systems* (2017).