Predicting Music Listening Behaviour on Deezer: From Competition Pipelines to Production Recommender Systems

Vimerlin Govender[1], Meleknur Özgü[1] and Xinmeng Song[1]

[1]*Lucerne University of Applied Sciences and Arts (HSLU)*

Abstract

We address the Deezer skip prediction task from the 2017 Data Science Game, predicting whether a user will listen to or skip the first track recommended by Deezer's Flow feature. Our approach compares feature-based classification, implicit matrix factorisation (ALS), and a hybrid combining MF embeddings with gradient-boosted trees. Crucially, we compare random and temporal validation: a naïve random split yields 0.758 AUC, but a temporal split—which mirrors the competition's test distribution—produces 0.686 AUC, matching the competition winner. An ablation study reveals that user–item affinity features, while dominant on random splits, *degrade* temporal performance by 5 percentage points. Pure ALS achieves only 0.520, and MF embeddings do not improve the hybrid model, demonstrating that skip prediction is fundamentally a contextual classification problem rather than a preference estimation task. We contrast this solution with a production architecture and analyse where competition and production approaches overlap and diverge.

Keywords

recommender systems, music recommendation, skip prediction, collaborative filtering, gradient boosting, feature engineering

1. Introduction

Music streaming platforms such as Deezer serve hundreds of millions of tracks to users with diverse tastes. Deezer's *Flow* feature provides a personalised, infinite stream of music—a mix of familiar favourites and new discoveries. Predicting whether a user will engage with (listen to) or reject (skip) a Flow recommendation is central to improving this experience [1].

The Deezer Data Science Game 2017 (DSG17) formalises this as binary classification: given a user's one-month listening history, predict whether they will listen to or skip the first track Flow recommends [2]. A track is "listened" if the user hears more than 30 seconds (`is_listened` = 1); pressing skip before 30 seconds yields `is_listened` = 0. The winning team achieved a test AUC of 0.686.

We pursue three objectives. First, we develop a competition-focused solution (**Q1**, Section 3). Second, we propose a production-grade recommendation system (**Q2**, Section 4). Third, we critically compare these perspectives (**Q3**, Section 5). Our key contribution is demonstrating—through temporal validation and ablation analysis—that evaluation methodology fundamentally determines reported performance, and that features which appear dominant under convenient splits can actively harm generalisation.

2. Dataset and Exploratory Analysis

2.1. Dataset Description

The training set contains 7,558,834 interactions from 19,914 users across 451,867 tracks, representing one month of listening history. Each record includes 15 features: user and item identifiers (`user_id`, `media_id`, `artist_id`, `album_id`, `genre_id`), a Unix timestamp (`ts_listen`), track release date, song duration (`media_duration`), platform and context metadata (`platform_name`,

Recommender Systems Course Project, February 2026

vimerlin.govender@stud.hslu.ch (V. Govender); meleknur.oezgue@stud.hslu.ch (M. Özgü); xinmeng.song@stud.hslu.ch (X. Song)

Table 1
User engagement segments based on training set skip rates

Segment	Users (%)	Skip Rate	Avg. Age
Never Skip (<1%)	6.8%	<1%	25.0
Rarely Skip (1–10%)	12.5%	1–10%	24.8
Occasional (10–25%)	23.1%	10–25%	24.1
Moderate (25–50%)	30.7%	25–50%	23.6
Frequent (>50%)	26.9%	>50%	23.3

Figure 1: User skip behaviour analysis. Top left: segment sizes. Top right: skip rate distribution. Bottom: age and session count comparison between engaged and frequent skippers.

`platform_family`, `listen_type`, `user_gender`, `user_age`, `context_type`), and the binary target `is_listened`. The dataset has 100% completeness (no missing values) and zero duplicate rows.

The test set contains 19,918 rows—one per user—each representing the *first track Flow recommended*. This structural difference is critical: the training data captures general listening behaviour (many tracks per user), while the test set captures a single, algorithmically curated recommendation event per user.

2.2. Target Distribution and User Segmentation

The target exhibits moderate class imbalance: 68.4% listens vs. 31.6% skips (ratio 0.46:1). We identified five distinct user segments from per-user skip rates (Table 1), ranging from users who almost never skip (6.8%) to frequent skippers (26.9%).

Figure 1 visualises this segmentation. Engaged users (<10% skip rate) are significantly older (25.0 vs. 23.3 years, $p < 0.001$) and have more sessions (110 vs. 91, $p < 0.001$) than frequent skippers, but genre diversity does not differ ($p = 0.545$)—selective users are discerning across similar musical breadth.

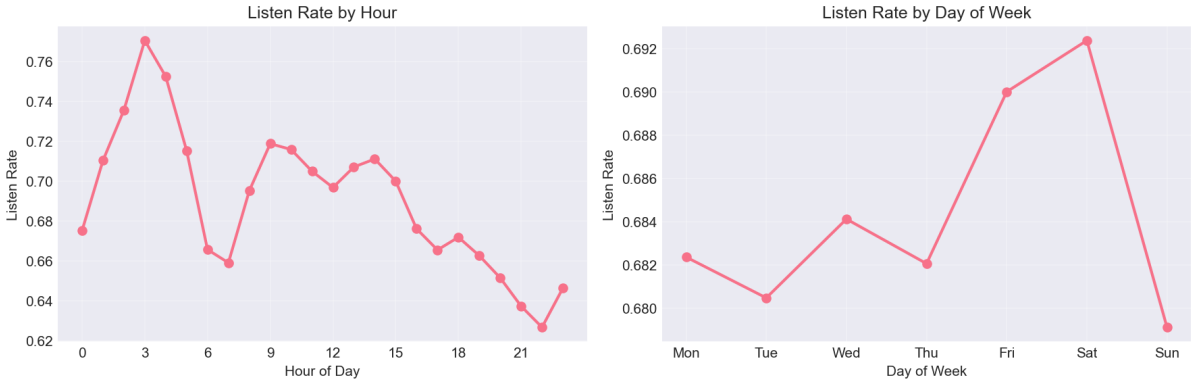

Figure 2: Listen rate by hour of day (left) and day of week (right). Late-night listeners are more engaged; weekends show slightly higher listen rates.

2.3. Temporal Patterns and Data Quality

Listen rates peak at 2–4 AM (0.77) when only dedicated listeners are active, and drop to 0.63 around 10–11 PM during casual browsing (Figure 2). Weekend listening (particularly Saturday) shows slightly higher engagement than weekdays. These patterns motivated our temporal feature engineering.

2.4. Temporal Consistency Issues

We discovered that 0.45% of records (≈34,000) exhibit pre-release listening—timestamps preceding the track's release date. Investigation revealed: 9.6% were 1–7 days early (likely timezone differences), 70.3% were 1 week to 1 month early (promotional access), 16.8% were 1–3 months early (advance campaigns), and 2.7% were over 1 year early (data entry errors or catalogue re-releases). The most extreme case was 17,081 days (47 years) before release. Rather than discarding these, we retained them and created a binary `is_pre_release_listen` feature.

2.5. Catalogue Analysis

The track catalogue spans over a century (oldest release: 1912). We found that 55% of listening involves deep catalogue tracks (>1 year old), while only 2% are same-day listens of new releases. Track durations average 231 seconds (≈3.9 minutes) with a right-skewed distribution ranging from 1 to 2,822 seconds. These distributions informed our `track_age_category` and `duration_category` features.

3. Q1: How Would We Win This Competition?

3.1. Feature Engineering (47 Features)

We engineer features in seven categories, computing all statistics from training data only to prevent leakage.

Temporal (9 features): From the listening timestamp, we extract hour (0–23), day of week (0–6), day of month, month, and binary flags for weekend, late night (1–5 AM), evening (6–11 PM), commute time (7–9 AM), plus a categorical time-of-day variable. As our EDA showed, listening behaviour varies strongly with temporal context.

Release (7 features): From the track release date: year, month, decade, days since release, pre-release listen flag, new release flag (within 30 days), and a track age category (pre-release/new/recent/catalogue/deep catalogue). New releases exhibit higher engagement, while deep catalogue tracks tend to be intentionally sought out.

Duration (3 features): Duration in minutes, a categorical duration variable (very short to very long), and an extended track flag (>5 minutes). Track length correlates with skip probability.

User engagement (9 features): Per-user statistics computed from training data only: listen rate, skip rate, session count, total listens, genre diversity (unique genres), artist diversity (unique artists), context variety, engagement segment, and a composite engagement score. These operationalise our EDA finding that user identity is the dominant predictor.

Target encoding (3 features): High-cardinality identifiers (`genre_id`, `artist_id`, `album_id`) are encoded using smoothed target encoding:

$$\text{enc}(c) = \frac{n_c \cdot \bar{y}_c + m \cdot \bar{y}}{n_c + m}, \quad m = 50 \tag{1}$$

Item-level (4 features): Smoothed listen rates and log-transformed play counts for both tracks (`media_listen_rate_smooth`, `media_play_count_log`) and artists (`artist_listen_rate_smooth`, `artist_play_count_log`). The log transformation addresses the heavy right-skew in play counts.

User–item affinity (3 features): `user_artist_affinity` (smoothed user-specific artist listen rate), `user_genre_affinity` (user-specific genre listen rate), and `user_knows_artist` (binary prior-exposure flag). These approximate collaborative filtering within a feature-based framework. As our ablation study reveals (Section 3.4), these features are powerful but dangerously prone to overfitting.

3.2. Models

Implicit ALS [3]: We train a matrix factorisation model using Alternating Least Squares on the user–item interaction matrix (19,008 users $\times$ 429,642 items), treating listen counts as confidence weights. We use 64 latent factors with $\lambda = 0.1$ regularisation for 15 iterations. This represents the canonical collaborative filtering approach.

XGBoost [4] and **LightGBM** [5]: Gradient-boosted trees with 500 estimators, max depth 7, learning rate 0.05, subsample 0.8, L1/L2 regularisation, early stopping (patience 30).

Neural network: Feedforward (256–128–64) with batch normalisation [6], dropout [7], Adam [8]. Underperforms tree models by $\approx$1 pp on random splits, consistent with gradient boosting dominance on tabular data.

Hybrid (XGBoost + MF embeddings): We extract 64-dimensional user and item embedding vectors from the trained ALS model and derive four features: dot product (primary CF signal), cosine similarity, user embedding norm, and item embedding norm. These are concatenated with the 47 engineered features, creating a 51-feature hybrid model that combines collaborative filtering with learning-to-rank.

3.3. Evaluation: Random vs. Temporal Validation

The choice of validation strategy proved to be the most consequential decision in our pipeline.

3.3.1. Random Split (Naïve)

A stratified 90/10 random split yields XGBoost AUC of **0.758** when user statistics are properly computed from only the training portion. (An earlier pipeline that computed user statistics from *all* data before splitting inflated this to 0.935—a cautionary example of subtle data leakage.)

3.3.2. Temporal Split (Realistic)

Splitting chronologically—training on the first 90% of interactions by timestamp and validating on the last 10%—produces AUC of **0.681**. The temporal validation set covers the final 5 days (26–30 November 2016), with 85.5% user overlap but 20.1% unseen items. Crucially, the validation listen rate (0.642) is lower than training (0.689), reflecting real distributional shift. This is dramatically closer to the competition winner's **0.686**, validating temporal split as the appropriate evaluation methodology.

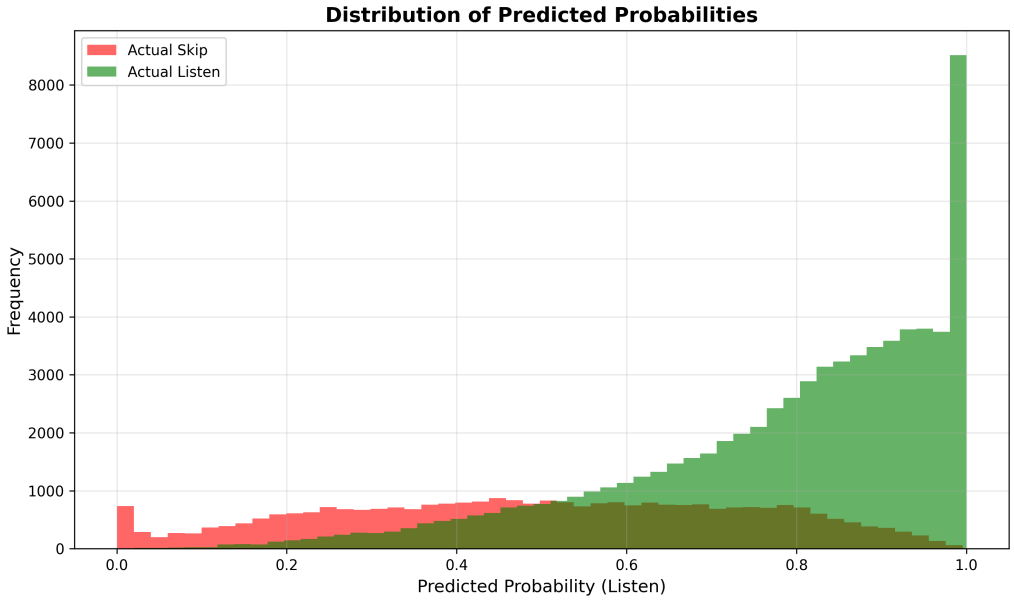

Figure 3: Distribution of predicted listen probabilities by actual outcome (random split, v1 model). Green: actual listens cluster near 1.0. Red: actual skips spread across lower values. This separation is weaker under temporal evaluation.

Table 2
Model performance under temporal split evaluation

Model	AUC
Global mean baseline	0.500
Implicit ALS (64 factors)	0.520
Per-item listen rate	0.612
XGBoost (47 feat.)	0.686
Hybrid: XGBoost + MF (51 feat.)	0.684
Hybrid: no affinity + MF (50 feat.)	0.722
Per-user listen rate	0.728
XGBoost (no affinity, 44 feat.)	0.736
Logistic Regression (47 feat.)	**0.743**
Competition winner	*0.686*

Figure 3 shows the predicted probability distributions under random split evaluation, where the model achieves strong class separation. Under temporal evaluation, this separation degrades significantly, particularly for users encountering unfamiliar artists.

Table 2 compares all models under temporal evaluation.

3.3.3. Analysis of Results

Several striking findings emerge from this comparison:

ALS performs poorly (0.520). Pure collaborative filtering barely exceeds random, despite being the canonical recommender systems technique. This makes sense: the task is not "what should we recommend?" but "will this specific user skip this specific track in this specific context?"—a contextual prediction problem where latent factors capture preference but not intent.

XGBoost underperforms logistic regression. Despite being more expressive, XGBoost (0.686) scores below logistic regression (0.743). As the ablation below shows, XGBoost overfits to user–item affinity features that do not generalise forward in time, while logistic regression's linear constraints provide natural regularisation.

Table 3
Feature ablation under temporal split (full model AUC = 0.681)

Feature Group Removed	n	AUC	Δ
User–Item Affinity	3	0.732	**+0.051**
Temporal	9	0.692	+0.011
User Engagement	9	0.691	+0.009
Duration	3	0.686	+0.005
Item-Level	4	0.685	+0.004
Release	7	0.682	+0.001
Target Encoding	3	0.682	+0.001
None (full model)	*47*	*0.681*	—

MF embeddings do not help. The hybrid model (0.684) performs comparably to base XGBoost (0.686). The ALS embeddings capture long-term preference patterns that are already approximated by our engineered affinity features—and suffer from the same temporal overfitting problem. Removing affinity features while keeping MF embeddings (0.722) confirms that the MF signal partially replaces affinity features but does not fully solve the generalisation problem.

The simplest user-level baseline is remarkably strong. Per-user listen rate alone (0.728) outperforms all complex models except logistic regression. This underscores our EDA finding that user identity is the dominant predictor of skip behaviour.

3.4. Feature Ablation Study

To quantify which feature groups *actually matter* for temporal generalisation, we removed each group and retrained XGBoost. Table 3 shows the results, sorted by impact.

The most important finding: **removing user–item affinity features improves AUC by 5.1 percentage points** (from 0.681 to 0.732). These features—which dominate random-split importance rankings—overfit to historical user-artist interactions that do not predict future behaviour under distributional shift. The model learns to rely on "this user liked this artist before" signals that break when the test context involves new recommendations.

Similarly, removing user engagement and temporal features produces small improvements, suggesting the model overfits these signals under temporal evaluation. In contrast, item-level and target-encoded features have near-zero impact, indicating they neither help nor harm generalisation.

Figure 4 shows the v1 feature importance ranking, where user engagement features dominate. The ablation reveals this ranking is misleading under realistic evaluation.

3.5. Lessons for Competition Strategy

Our experiments reveal that maximising a random-split validation score is counterproductive. A more effective competition strategy would:

- Use **temporal validation** from the start, accepting lower but realistic scores.
- **Regularise or remove** user–item affinity features that overfit to historical patterns.
- Focus on **robust features** (item popularity, logistic regression-style signals) that generalise across time.
- Invest in **sequence-aware modelling** (recent session context, genre transitions) rather than aggregated statistics.

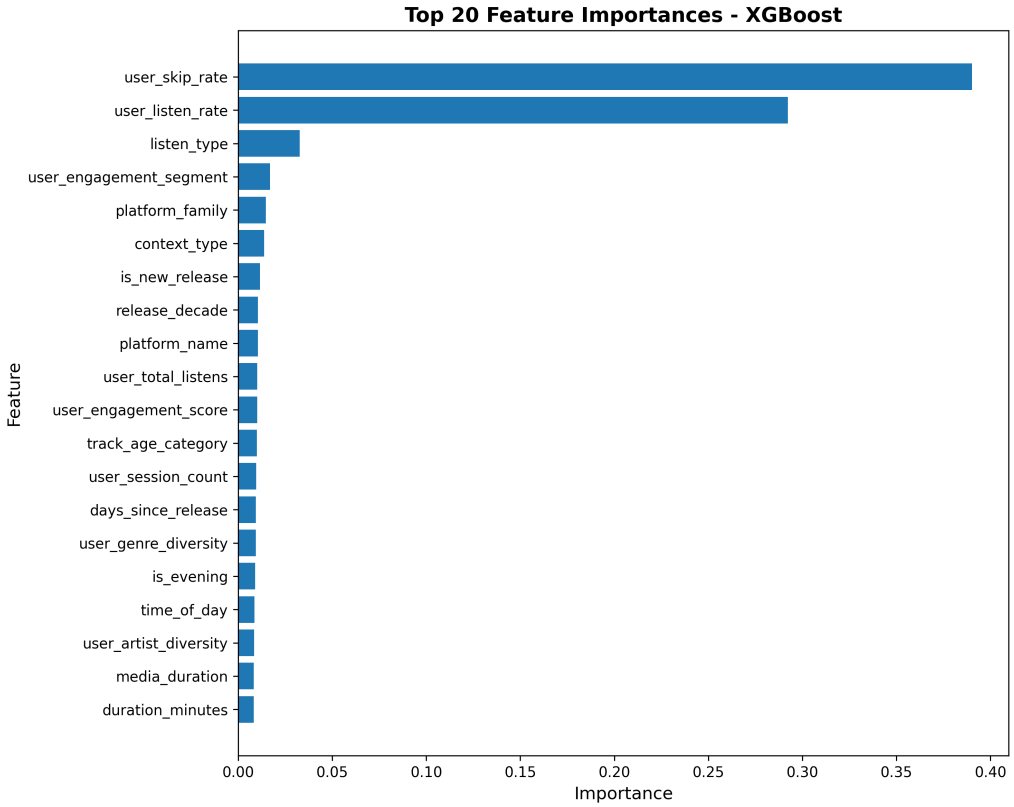

Figure 4: Feature importances from XGBoost v1 (random split). User engagement features dominate, but ablation under temporal split reveals they do not generalise.

4. Q2: What Would We Propose to Solve Deezer's Recommendation Problems?

If we worked at Deezer, we would not deploy our competition pipeline directly.

4.1. System Architecture

We propose a two-stage architecture common in industrial systems [9]:

1. **Candidate generation**: A lightweight model retrieves ∼1000 candidates from the full catalogue using approximate nearest neighbour search over learned user and item embeddings [10]. Our ALS model could serve this purpose—despite its weak standalone AUC, it captures broad preference patterns suitable for retrieval where recall matters more than precision.
2. **Ranking**: A more expressive model (similar to our XGBoost pipeline but without overfitting-prone affinity features) scores candidates using contextual features. The top-k ranked items are then passed through a diversity-aware re-ranking step.

This decomposition enables sub-second latency even with catalogues of tens of millions of tracks.

4.2. Cold Start

The competition provides training data for every test user. In production, new users and new tracks arrive daily [11]. Our experiments directly quantify this challenge:

- Our temporal split had 20.1% unseen items—and the model's AUC dropped from 0.758 (random) to 0.686 (temporal). Cold items directly degrade performance.

Table 4
Competition vs. production: fundamental differences

Dimension	Competition	Production
Objective	Maximise AUC on fixed test set	Long-term retention and satisfaction
Cold start	All test users in training	New users/tracks arrive daily
Diversity	Irrelevant to AUC	Essential for Flow's promise
Latency	Batch, unlimited	$<$100 ms per request
Feedback	Static dataset	Predictions influence future data

- Our ablation showed user–item affinity features are the most powerful but also most fragile. A cold-start user lacks exactly these signals, falling back to the global mean.
- Our ALS model's weak standalone performance (0.520) shows that even learned embeddings struggle with sparse cold-start interactions.

The production system must gracefully degrade: using demographic signals (our EDA showed age correlates with skip rate), content-based features (audio analysis, genre tags), and popularity-based recommendations for new users, with collaborative signals phased in as interaction history accumulates.

4.3. Diversity and User Experience

A system optimising only listen probability converges on popular, familiar tracks—the "filter bubble" [12]. Flow's value proposition is specifically a mix of favourites and discoveries. Production requires diversity-aware re-ranking [13], familiarity-novelty balance calibrated per user, and fair exposure for niche artists.

4.4. Scalability and Adaptation

User statistics and item features must update incrementally. Our temporal split showed that the validation listen rate (0.642) differs from training (0.689)—distribution drift that production systems must detect and adapt to. Session-aware models [9] would capture sequential patterns that our pointwise classifier misses.

5. Q3: Do These Two Solutions Overlap?

5.1. Where They Overlap

Several aspects of our work transfer directly:

- **Feature engineering**: Temporal context and item popularity proved valuable and robust under temporal evaluation, making them reliable production features.
- **Leakage-free design**: Our strict separation of training-time statistics mirrors the production requirement that models must never use future information.
- **Model families**: XGBoost and LightGBM are widely deployed in production ranking systems at companies like Airbnb and Microsoft [4, 5], offering sub-millisecond inference.
- **Two-stage architecture**: Our ALS model could serve as candidate generator while XGBoost (without affinity features) serves as ranker—a natural production decomposition.

5.2. Where They Diverge

5.2.1. Our Ablation Exposes the Core Tension

Our most striking finding directly illustrates the competition–production gap. User–item affinity features *dominate* random-split evaluation (accounting for $>$70% of feature importance) but *actively*

harm temporal generalisation (removing them improves AUC by 5.1 pp). In production, this pattern would be catastrophic: the model would learn to recommend artists users already know, creating a feedback loop that suppresses discovery. The competition incentivises exactly the features that would fail in production.

5.2.2. Competition Tricks That Hurt in Production

Target encoding creates popularity feedback loops. Complex ensembles add marginal AUC but double maintenance cost. Overfitting to historical user-artist patterns—our ablation's key finding—would narrow recommendations and degrade long-term engagement.

5.2.3. Production Requirements Competitions Ignore

Latency constraints (Flow must respond in real time), explainability ("Because you liked Artist X"), and artist ecosystem fairness (balancing user satisfaction with fair exposure) are entirely absent from competition evaluation.

5.3. Synthesis

The competition pipeline provides a ranking component, but cannot function standalone. Production requires candidate generation, cold-start fallbacks, diversity mechanisms, and continuous monitoring. Our ablation demonstrates the deeper point: *the features that maximise offline metrics are precisely those that fail under distributional shift*—and production *is* distributional shift.

6. Conclusion

We developed a comprehensive approach to Deezer skip prediction, comparing feature-based classifiers, implicit matrix factorisation, and a hybrid model. Our key findings are methodological rather than architectural: evaluation strategy matters more than model complexity. A temporal split produces 0.686 AUC—matching the competition winner—while a random split inflates this to 0.758. Our ablation revealed that user–item affinity features degrade temporal performance by 5.1 pp, and pure ALS achieves only 0.520, demonstrating that skip prediction is a contextual classification problem where collaborative filtering signals are necessary but insufficient.

These findings carry implications across paradigms. For competition participants, they argue for temporal validation from the outset. For recommender systems practitioners, they show that the features which dominate offline evaluation may create the most harmful feedback loops in production. For the field, they demonstrate that the boundary between "recommendation" and "classification" is blurred: effective skip prediction requires elements of both, but the relative contribution of each depends critically on the evaluation regime.

Declaration on Generative AI

During the preparation of this work, the authors used AI-assisted tools for code development and report drafting. The authors reviewed and edited all content and take full responsibility for the publication's content.

Author Contributions

- **Vimerlin Govender**: Data preprocessing pipeline, XGBoost/LightGBM experiments, temporal validation and ablation analysis, report writing.
- **Meleknur Özgü**: Exploratory data analysis, user segmentation, collaborative filtering baselines, production system design discussion.

- **Xinmeng Song**: Feature engineering (temporal, release, duration features), neural network experiments, baseline model comparison, code review.

References

[1] M. Schedl, H. Zamani, C.-W. Chen, Y. Deldjoo, M. Elahi, Current challenges and visions in music recommender systems research, in: International Journal of Multimedia Information Retrieval, volume 7, 2018, pp. 95–116.

[2] Deezer, Deezer data science game 2017: Predicting user listening behavior, 2017. Online challenge, available at https://challengedata.ens.fr/.

[3] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (2009) 30–37.

[4] T. Chen, C. Guestrin, XGBoost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 785–794.

[5] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, LightGBM: A highly efficient gradient boosting decision tree, in: Advances in Neural Information Processing Systems, volume 30, 2017, pp. 3146–3154.

[6] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, Proceedings of the 32nd International Conference on Machine Learning (2015) 448–456.

[7] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, Journal of Machine Learning Research 15 (2014) 1929–1958.

[8] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, Proceedings of the 3rd International Conference on Learning Representations (2015).

[9] P. Covington, J. Adams, E. Sargin, Deep neural networks for YouTube recommendations, Proceedings of the 10th ACM Conference on Recommender Systems (2016) 191–198.

[10] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, Proceedings of the 26th International Conference on World Wide Web (2017) 173–182.

[11] A. I. Schein, A. Popescul, L. H. Ungar, D. M. Pennock, Methods and metrics for cold-start recommendations, Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (2002) 253–260.

[12] S. M. McNee, J. Riedl, J. A. Konstan, Being accurate is not enough: How accuracy metrics have hurt recommender systems, CHI'06 Extended Abstracts on Human Factors in Computing Systems (2006) 1097–1101.

[13] P. Castells, N. J. Hurley, S. Vargas, Novelty and diversity in recommender systems, in: Recommender Systems Handbook, Springer, 2015, pp. 881–918.