Music Popularity Prediction Using Advanced Machine Learning Ensemble Methods

Sahan L. Gunasekara
dept. Computer Science & Engineering
University of Moratuwa
Sri Lanka
sahang.22@cse.mrt.ac.lk

Chandupa H. Ambepitiya
dept. Computer Science & Engineering
University of Moratuwa
Sri Lanka
chandupa.22@cse.mrt.ac.lk

Dewmike L. Amarasinghe
dept. Computer Science & Engineering
University of Moratuwa
Sri Lanka
dewmike.22@cse.mrt.ac.lk

Sugith G. Munasinghe
dept. Computer Science & Engineering
University of Moratuwa
Sri Lanka
geesan.22@cse.mrt.ac.lk

Pasindu P. Manujaya
dept. Computer Science & Engineering
University of Moratuwa
Sri Lanka
manujaya.22@cse.mrt.ac.lk

Mohommad A. Jusail
dept. Computer Science & Engineering
University of Moratuwa
Sri Lanka
jusail.18@cse.mrt.ac.lk

***Abstract*—This paper presents a comprehensive study on predicting music track popularity scores using machine learning ensemble methods. We analyzed a dataset of 61,609 Billboard-tracked music releases to predict continuous popularity scores (0–100) based on audio features, metadata, and contextual information. Our research compared twelve distinct modeling approaches across six team members, ranging from basic linear regression to advanced ensemble meta-models. The top-performing approach achieved an RMSE of 9.0035 using a sophisticated ensemble meta-model combining six diverse algorithms (XGBoost, LightGBM, Extra Trees, Histogram Gradient Boosting, Random Forest, and a second XGBoost variant) with advanced preprocessing techniques including KNN imputation and temporal feature engineering. Analysis revealed that metadata features significantly outweigh audio characteristics in predictive power, with track identification (SHAP value 4.71) and artist information (3.82) contributing more than audio features like instrumental density (0.88). Tree-based ensemble methods consistently outperformed linear models, with ensemble approaches occupying five of the top seven rankings. Our findings demonstrate that combining metadata-rich feature engineering with robust ensemble techniques significantly enhances music popularity prediction accuracy, achieving superior performance compared to individual models.**

***Index Terms*—music popularity prediction, ensemble learning, feature engineering, gradient boosting, machine learning**

I. Introduction

The prediction of music popularity remains a challenging problem with significant commercial implications for the music industry. Artists, producers, and record labels would benefit from understanding which musical characteristics contribute to commercial success before investing in production and marketing efforts. This study tackles the problem of predicting continuous popularity scores for music tracks using machine learning techniques applied to real-world Billboard data. Six team members independently developed and implemented twelve different modeling approaches to address this prediction challenge. These approaches varied significantly in their preprocessing strategies, feature engineering techniques, and algorithmic choices. Methods ranged from simple linear regression to sophisticated ensemble meta-models combining multiple algorithms. Each approach was evaluated using the same dataset to enable direct performance comparison. The diversity of approaches allows for comprehensive analysis of what works best for music popularity prediction. Some methods focused on advanced feature engineering and temporal analysis, while others emphasized ensemble techniques and hyperparameter optimization. Several approaches explored different strategies for handling missing data and categorical variables. This paper presents the comparative results of all twelve approaches, identifies the most effective techniques, and analyzes which features contribute most significantly to prediction accuracy. The findings provide practical insights for both the technical development of prediction models and understanding of the factors that drive music popularity.

II. Related work

Music popularity prediction, often known as Hit Song Science (HSS), is a significant research area within Music Information Retrieval (MIR) that has attracted attention across multiple disciplines, combining techniques from signal processing and machine learning [1] [2]. The core goal is to predict the commercial success or popularity of a musical track prior to its distribution, serving as a valuable multimedia application for various stakeholders, including artists, producers and recommendation systems [1]. The early methods in this field primarily focused on extracting technical features from the audio signal [3]. These studies often framed the problem as a binary classification task, with the aim of distinguishing between hit and non-hit songs [1].

The feasibility of predicting popularity based on such technical features has historically been debated [3]. Some early research using standard acoustic or human-generated features found them not sufficiently informative to predict popularity, suggesting that this subjective judgment was not

well captured by these techniques [3]. In contrast, other work utilizing alternative methods and feature sets, such as a time-shifting perceptron applied to UK chart data, reported better-than-random accuracy, indicating that prediction might be possible with relevant data and models [4]. More recent efforts leverage advancements in deep learning and explore the use of multimodal information, including audio, lyrics, and metadata [1], to enhance prediction accuracy. The creation of datasets like the SpotGenTrack Popularity Dataset (SPD) [5] addresses limitations of previous resources such as the Million Song Dataset (MSD), which often lacked comprehensive raw audio and lyrics, by providing a unified, multimodal source. Modern models, such as the proposed HitMusicNet, explore both classification and regression approaches to popularity prediction, with regression offering the potential for a more nuanced, continuous popularity score rather than a hard binary decision [1]. This highlights the complexity of music popularity prediction and the evolving methods used to address this multifaceted problem [2].

III. Methodology

A. Dataset Description

The dataset consists of 61,609 Billboard-tracked music releases, with each entry represented by 62 features. Each row corresponds to a unique music release and includes detailed information for up to the first three tracks. The target variable is the popularity score ranging from 0 to 100. Out of the 62 features, 53 are numerical and 9 are categorical.

For each of the first three tracks, the dataset includes core audio attributes `duration_ms`, `rhythmic_cohesion`, `intensity_index`, `harmonic_scale`, `tonal_mode`, `organic_texture`, `beat_frequency`, `time_signature`, and `composition_label`. These features describe aspects of duration, danceability, energy, musical key, modality, acousticness, tempo in BPM, meter, and track titles respectively.

The dataset also includes several derived metrics: emotional charge (energy $\times$ valence), groove efficiency (energy/danceability), and organic immersion (acousticness $\times$ duration). At the release level, additional aggregated features are provided, including duration consistency (standard deviation of durations), tempo volatility (range of BPM values), and key variety (number of distinct keys across tracks).

Contextual and categorical features include the number of credited artists, total track count in the release, album title length (in characters), day of the week and seasonal quarter of release, and the lunar phase on the release date.

B. Model Training Approaches

1) ***Ensemble with Six Models (RMSE: 9.0035)****:*

a) ***Data Preprocessing****:* Missing numerical values were handled using KNN Imputer with 5 neighbors due to high missing percentages exceeding 10%, where traditional mean or median imputation would introduce bias. The KNN approach estimates missing entries based on similarity to other data points, providing more reliable imputation for datasets with substantial missing data patterns.

All categorical values were converted to lowercase to ensure consistency and prevent identical values with different cases from being treated as separate categories. In columns `composite_label_[0-2]`, `track_identifier`, and `creator_collective`, values appearing only in the training set were labeled `'unseen_train'` while those only in the test set received `'unseen_test'` labels. Common values were retained if they appeared above specified frequency thresholds, otherwise they were labeled as `'common_rare'`. Following this preprocessing, all categorical columns underwent label encoding. Missing values in encoded categorical columns were filled using KNN Imputer with 5 neighbors, which proved more effective than standard mode imputation.

b) ***Feature Engineering****:* Publication year was calculated from the `publication_timestamp` column and analyzed against average target values. This analysis revealed noticeable differences across time periods(eras), leading to the creation of five distinct musical eras through statistical grouping. Box plots were used to visualize target distribution patterns for each period, while Mann-Whitney U tests were conducted for each pair of periods to confirm statistical significance of the observed differences.

The following two new features were created:

- `publication_year` represents the published year
- `data_period` represents the era song belongs to

The `data_period` feature was label encoded to make it suitable for machine learning algorithms while preserving the ordinal relationship between different musical periods. After that the missing values in `publication_year` and `data_period` were filled with KNN imputer. This method allowed to estimate more reliable values for missing entries rather than filling with mean or median.

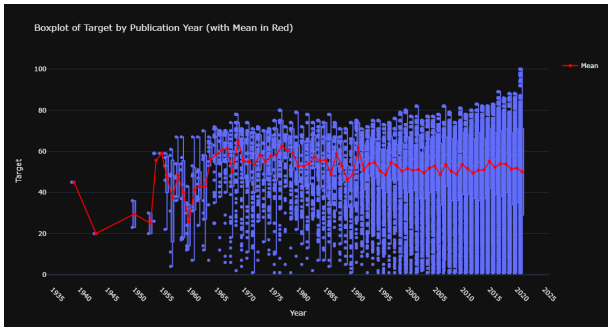

Fig. 1. Avg. Song Popularity Score Distribution vs Publication Year

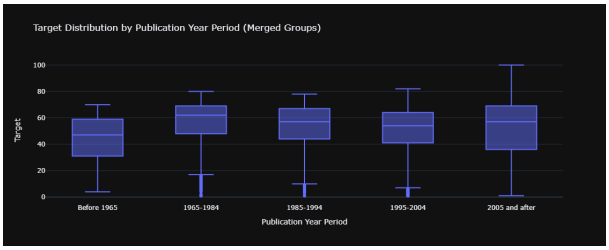

Fig. 2. Song Popularity Score Distribution vs Binned Years

*c) **Feature Selection**:* Feature importance evaluation was conducted using XGBRegressor to identify the most relevant predictors. Based on importance scores, the two least contributing features, `lunar_phase` and `id` were removed from the dataset. Additionally, the original `publication_timestamp` was dropped due to the high correlation with the newly derived `publication_year` and `data_period` features. This dimensionality reduction helped minimize redundancy while preserving essential temporal information through the engineered features.

*d) **Model Architecture**:* The ensemble consisted of six diverse algorithms selected for complementary strengths:

- XGBoost Regressor for handling nonlinear relationships and mitigating overfitting through gradient boosting,
- LightGBM Regressor for computational efficiency on high-dimensional data while maintaining competitive performance,
- Extra Trees Regressor for variance reduction through randomized tree construction,
- Histogram Gradient Boosting Regressor for efficient training on large, dense datasets with its histogram-based approach,
- Random Forest Regressor for robust modeling of complex feature interactions through bagging,
- A second XGBoost Regressor with different parameter configuration with increased tree depth to capture more complex data patterns.

Robust Scaler was applied before training to reduce outlier influence and ensure stable feature distributions across all algorithms. All models underwent hyperparameter optimization using Optuna, an automated optimization framework that efficiently searches hyperparameter spaces.

*e) **Meta Learning**:* The ensemble used stacking methodology rather than simple averaging or voting schemes. Individual model predictions served as input features for a meta-model, which learned optimal combination weights through supervised learning. This stacking approach allows the meta-model to identify which base models perform best under different data conditions and weight their contributions accordingly, resulting in superior generalization compared to fixed combination strategies.

*2) **XGBoost 1 (Deep XGBoost) (RMSE: 9.3012)**:*

*a) **Data Preprocessing**:* Missing values in numerical columns were handled using Iterative Imputer, which models each feature with missing values based on other features. This method is better for datasets with complex feature relationships compared to KNN Imputer which only uses nearby values. Iterative Imputer captures broader patterns for more consistent results.

For categorical columns, the same steps from Approach 1 were used: lowercase conversion, labeling unseen categories as `'unseen_train'`, `'unseen_test'`, or `'common_rare'`, and label encoding. Missing values in categorical columns were filled using Iterative Imputer instead of KNN to maintain consistency and better capture inter-column dependencies.

Log transformation was applied to highly right-skewed numerical columns to reduce skewness, minimize outlier influence, and create more balanced distributions. This helps improve model learning and performance.

*b) **Feature Engineering**:* The same temporal features from Approach 1 were used: `publication_year` and `data_period` extracted from `publication_timestamp`. These features capture temporal patterns in the data. The original timestamp column was dropped since its information was already represented in the new features.

*c) **Model Training**:* Robust scalar was applied on numerical columns. A single XGBoost model was used with increased tree depth and increased search space to capture more complex patterns in the data. The deeper tree configuration allows the model to learn more complex decision boundaries and feature interactions.

*3) **Ensemble with XGB, LGBM 1 (RMSE: 9.0283)**:*

*a) **Data Preprocessing**:* Duplicate rows were removed, reducing the training set from 61,609 to 61,515 samples. The `publication_timestamp` was decomposed into year, month, day, day of week, and quarter. Sinusoidal transformations were applied to monthly and daily features, creating `month_sin`, `month_cos`, `day_sin`, and `day_cos` variables because music popularity follows seasonal patterns and weekly release schedules.

Missing values in categorical columns were filled with the placeholder `'Unknown'`, uncommon labels between training and test sets were standardized to `'Other'` and rare categories appearing fewer than 3–5 times were grouped into a `'Rare'` category. This ensures model generalization while preserving meaningful categorical distinctions.

Missing values in numerical features were filled with `-1` instead of zero because zero values could be meaningful in audio features, so `-1` distinguishes actual missing data from zero measurements.

*b) **Feature Engineering**:* Statistical aggregations were computed across track collections for each audio attribute prefix (`duration`, `dance`, `energy`, `acoustic`, `tempo`, `key`, `mode`). These included mean, standard deviation, maximum, minimum, range, median, and skewness because albums or collections with consistent audio characteristics might have different popularity patterns than diverse collections.

Cross-feature interactions were created to capture musical relationships that individual features miss:

- Energy-to-dance ratio: identifies high-energy but non-danceable tracks.
- Tempo-energy ratio: captures rhythmic intensity.
- Acoustic-energy: contrast distinguishes between organic and electronic music styles.

Metadata features were transformed into categorical indicators: tracks were classified as singles, EPs, or albums based on component count, and collaboration indicators were created from artist counts. These features capture the commercial and artistic context that influences music popularity beyond raw audio characteristics.

*c) **Feature Selection**:* `SelectKBest` with `f_regression` scoring identified the most predictive features for regression. From 134 engineered features, the top 100 were selected based on their statistical relationship with the target variable. This reduces computational complexity, minimizes overfitting risk, and eliminates redundant features that could degrade performance.

`f_regression` scoring was used because it measures the linear relationship between each feature and the continuous target variable, which aligns with the regression objective. Selection was applied after all preprocessing steps to ensure engineered features competed fairly with original features.

*d) **Model Training**:* LightGBM and XGBoost were used because they are effective with structured data and mixed feature types. LightGBM was configured with 2000 estimators, a learning rate of 0.03, and regularization (`alpha=0.1`, `lambda=0.1`) to prevent overfitting. Feature and bagging fractions were set to 0.8 to introduce randomness and improve generalization. `max_depth` was set to 8 with 64 leaves to balance model complexity and interpretability.

XGBoost used similar hyperparameters for fair comparison: same learning rate, estimator count, and regularization parameters. Both models used early stopping with 100-round patience to prevent overfitting. `subsample` and `colsample_bytree` parameters were set to 0.8 to match LightGBM's approach.

5-fold cross-validation with shuffling was used to ensure robust performance estimation. Each fold used early stopping based on validation performance. The ensemble combined predictions using inverse RMSE weighting – XGBoost achieved RMSE 9.0283 and LightGBM achieved RMSE 9.2717, resulting in 50.7% weight for XGBoost and 49.3% for LightGBM.

*4) **Ensemble with XGB, LGBM 2 (RMSE: 9.4690)**:*

*a) **Data Preprocessing**:* Same duplicate removal and categorical feature handling as the previous approach. Missing values across all features were filled with `-1` as a sentinel value instead of using other imputation methods. The `publication_timestamp` column was dropped as temporal features were not utilized in this implementation.

*b) **Feature Engineering**:* Same statistical aggregations for audio features as the Multi-Stage Feature Engineering approach. Categorical features underwent frequency encoding for high-cardinality variables like composition labels and track identifiers, where each category was replaced with its relative frequency in the training set. Low-cardinality categorical features such as `season_of_release` and `lunar_phase` received label encoding to convert them into numerical representations while preserving ordinal relationships.

*c) **Feature Selection**:* A preprocessing-based approach was used rather than explicit selection algorithms. All engineered features were retained, expanding the feature space from 61 to 79 dimensions. High-cardinality categorical features were consolidated through frequency encoding, effectively reducing dimensionality while preserving information content. This approach prioritized information retention over dimensionality reduction, allowing tree-based models to perform implicit feature selection through their splitting criteria.

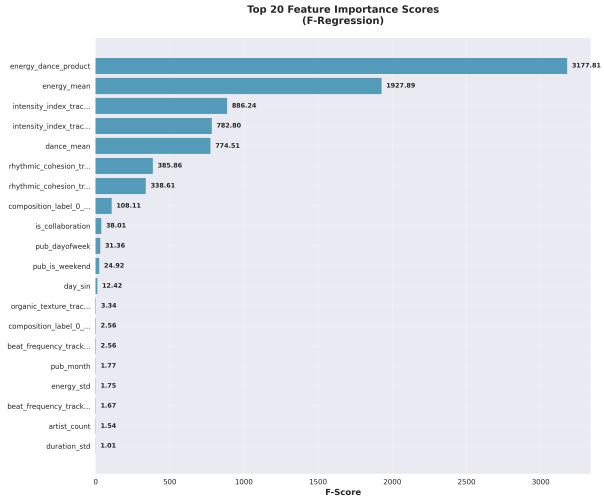

Fig. 3. Top 20 features based on F-Regression

*d) **Model Training**:* Two gradient boosting models were implemented with complementary strengths. LightGBM used 1000 estimators with a learning rate of 0.05, maximum depth of 6, and regularization through `feature_fraction = 0.8` and `bagging_fraction = 0.8` to prevent overfitting. XGBoost employed similar hyperparameters: 1000 estimators, 0.05 learning rate, maximum depth of 6, and `subsample`/`colsample_bytree` ratios of 0.8.

Both models utilized early stopping mechanisms to prevent overfitting. 5-fold cross-validation with shuffle was used for robust performance estimation. The ensemble used weighted averaging based on inverse RMSE scores – XGBoost received weight of 0.508 (RMSE: 9.4690) while LightGBM received 0.492 (RMSE: 9.7561).

*5) **Voting Ensemble with XGB, LGBM, CatBoost (RMSE: 9.7538)**:*

*a) **Data Preprocessing**:* Missing values in numerical features were handled using different strategies based on their missing data percentage. Features with $\leq$10% missing values used `KNNImputer` because the relatively low missing percentage allows KNN to find sufficient similar data points for reliable imputation. For features with $>$10% missing values, mean imputation was applied since high missing percentages

would make KNN imputation unreliable due to insufficient similar cases.

Categorical features were imputed using the mode since it preserves the original distribution of categories and provides the statistically most likely value for missing entries.

`OneHotEncoder` was used for categorical variables. This encoding method was chosen because it creates binary features for each category, preventing the model from assuming ordinal relationships between categories.

`StandardScaler` was applied to all numeric features. This standardization ensures all features contribute equally to the learning process regardless of their original magnitude.

*b) **Model Training**:* The ensemble used `VotingRegressor` to combine three gradient boosting algorithms: XGBoost, LightGBM, and CatBoost. This combines strengths of each algorithm. XGBoost excels at handling complex patterns, LightGBM is memory efficient and fast, while CatBoost has built-in categorical handling capabilities. By combining these different approaches, the ensemble captures a broader range of data patterns than any single model could.

Each model was individually tuned using `Optuna` with 30 trials per model. The 30-trial limit balanced optimization quality with computational cost.

*6) **XGBoost 2 (RMSE: 9.8695)**:*

*a) **Model Training**:* The same preprocessing pipeline from the previous ensemble approach was maintained to ensure fair comparison between methods. A single XGBoost model was used with `Optuna` hyperparameter tuning over 50 trials, which is more than the 30 trials used per model in the ensemble approach. This increased optimization effort was chosen because focusing computational resources on one model allows for more thorough hyperparameter exploration, potentially finding better configurations than the ensemble's distributed optimization approach.

XGBoost was selected as the single model as it consistently performs well on structured data problems and handles both numerical and categorical features effectively after preprocessing.

*7) **Ensemble with XGB, RandomForest, ExtraTrees (RMSE: 10.2795)**:*

*a) **Data Preprocessing**:* Object-type columns related to dates were converted to proper `datetime` type using pandas for easier manipulation and feature extraction.

Missing values were imputed using different strategies based on the data distribution:

- Mean imputation for continuous numerical features that were symmetrically distributed without notable outliers
- Median imputation for skewed continuous variables and discrete numerical features where outliers could distort the mean
- Mode imputation for categorical features to preserve the natural class distribution

The `creator_collective` column required cleanup due to inconsistencies in label formats. The `unicodedata` library normalized characters to a consistent alphabet, while the `difflib` library identified and grouped similar but inconsistent labels for standardization. Only the primary (first) creator was extracted since over 95% of records included at least one creator.

*b) **Feature Engineering**:* Feature engineering addressed high correlations between emotional attributes:

- `emotional_charge_0` ↔ `emotional_resonance_0`: 0.82
- `emotional_charge_1` ↔ `emotional_resonance_1`: 0.83
- `emotional_charge_2` ↔ `emotional_resonance_2`: 0.82

wriey that list part without piintforms remvoin list

New features were created: `emotional_gap_i = emotional_resonance_i - emotional_charge_i`, which reduced multi-collinearity and better represented emotional dynamics.

The `publication_date` field was decomposed into year, month and day components to capture temporal trends in music popularity.

*c) **Feature Selection**:* Feature selection was applied based on mutual information scores with the target variable. Variables exhibiting MI scores below 0.1 were excluded from the training process, including `time_signature_i`, `artist_count`, `tonal_mode_i`, `key_variety`, and `lunar_phase`. This filtering reduced noise and improved model performance by eliminating low-predictive features. insensitive to the scale or distribution of input features.

*d) **Model Training**:* No feature scaling was applied to the tree-based models due to their inherent insensitivity to variable scales and distributions. An ensemble regressor was trained on the processed dataset using an 80-20 train-test split for evaluation. Random Forest, Extra Trees, and XGBoost algorithms were implemented with default hyperparameters to leverage their capacity for capturing non-linear relationships and feature interactions.

*8) **Linear Models (RMSE: 19.6991)**:*

*a) **Model Training**:* The same preprocessed dataset from the tree-based approach was used for linear model training. Three linear regression models were trained on the processed dataset: Linear Regression, Ridge Regression and Lasso Regression.

Ridge regression incorporated L2 regularization while Lasso used L1 regularization to address potential overfitting and feature selection within the linear framework. The same 80-20 train-test split methodology was employed for evaluation consistency across approaches. Best performing model Ridge Regression was selected as the final model.

*9) **RandomForests (RMSE: 11.9678)**:*

*a) **Data Preprocessing**:* New features `pub_year`, `pub_month`, and `pub_day` were derived from `publication_timestamp`. Missing values in these derived columns were filled with a designated invalid placeholder.

For remaining features, missing value imputation was performed based on the nature of each variable. The skewness of

each numerical feature was calculated. For features exhibiting significant skewness, the median was used to impute missing values, while the mean was applied to those with approximately normal distributions. Categorical features were imputed using the mode of the respective columns.

After completing missing value imputation, all features excluding the target variable were standardized to ensure consistency in scale.

*b) **Model Training**:* Random Forest was selected as it works well for regression problems and can handle both linear and non-linear relationships. The model deals better with missing values and reduces overfitting by combining results from multiple decision trees. Given the dataset's many features, Random Forest was chosen to capture complex patterns and improve prediction accuracy.

*10) **XGBoost 3 (RMSE: 12.2388)**:*

*a) **Model Training**:* The same preprocessed dataset from the Random Forest approach was used for XGBoost training. XGBoost Regressor was trained as a gradient boosting family tree-based model. The model includes advanced features like regularization, weighted quantile sketching, and native handling of missing values.

*11) **XGBoost 4 (RMSE: 12.5847)**:*

*a) **Data Preprocessing**:* The dataset included a `publication_timestamp` column containing datetime information. This column was converted to datetime format, and three new columns were created: `year`, `month` and `day`. These new features were used in place of the original timestamp, which was then dropped. Missing values in the `year`, `month`, and `day` columns were filled using the most frequent value from each respective column.

Next, the features were divided into numerical and categorical types. Numerical columns were scaled using `RobustScaler`, which reduces the influence of outliers by using the median and interquartile range. Categorical columns were encoded using `OrdinalEncoder` with settings that allowed it to handle unknown categories during prediction. These preprocessing steps were combined into a single transformation pipeline using `ColumnTransformer`, which was later integrated with the model.

*12) **Extra Trees Regressor (RMSE: 13.1245)**:*

*a) **Model Training**:* The same preprocessed dataset was used to train an Extra Trees Regressor. Numerical features were scaled using `RobustScaler` and categorical features were encoded with `OrdinalEncoder`. The preprocessing steps remained consistent across all phases of the experiment.

The model was trained using the Extra Trees Regressor, with preprocessing applied manually prior to model fitting. To evaluate performance and reduce the risk of overfitting, 5-fold cross-validation was used.

C. Selection of Best Methodology

After evaluating all 12 modeling approaches implemented by the team, the Ensemble Meta-Model with Six Models was selected as the optimal solution. This approach indicated superior performance on multiple evaluation metrics, achieving the lowest RMSE in both validation (8.916) and test data sets (9.0035). In addition, it recorded the best MAE and R^2 scores compared to all other methods, indicating consistent predictive accuracy between different evaluation criteria.

The ensemble architecture combines six different models to use individual strengths while reducing the inherent drawbacks and algorithm-specific biases of each model. Due to the diversity of the model, it can handle a wide range of unseen data patterns and scenarios. The meta-learning model assigns weights to each base model's contribution based on performance, unlike simple averaging or voting methods that treat all models equally.

This approach outperforms the second-best method "Ensemble with XGB, LGBM 1" using only 60 features compared to 100, reducing model complexity and overfitting risk. Feature engineering used hypothesis testing for validation when creating new features and retained only statistically significant variables, avoiding unnecessary feature creation, as seen in Methods 3 and 4.

The missing value handling used KNN Imputer, which was more accurate than mean, median, or mode imputation methods used in other approaches. This preprocessing technique preserves data relationships better, contributing to the ensemble model's performance.

IV. Results

A. Model Performance

Figure 4 shows the RMSE values for all 12 approaches implemented. The Ensemble with Six Models outperformed all other methods across all evaluation metrics, achieving the lowest RMSE of 9.0035, MAE of 4.878, and highest R^2 of 0.829. This was followed by Ensemble with XGB, LGBM 1 at 9.0283 RMSE and XGBoost 1 (Deep XGBoost) at 9.3012 RMSE in third position.

Out of the top 7 methods, 5 are ensemble approaches. The remaining methods showed progressively higher RMSE values, with Ensemble using XGB, RF, and ExtraTrees achieving 10.2796 RMSE, RandomForests reaching 11.9678 RMSE, and various XGBoost implementations ranging from 12.2388 to 12.5847 RMSE.

The deeper XGBoost method (XGBoost 1) achieved significantly higher accuracy compared to other XGBoost versions, with RMSE of 9.3012 versus 9.8695, 12.2388, and 12.5847 for XGBoost 2, 3, and 4 respectively. Linear models performed poorly with RMSE of 19.6991, indicating they are not suitable for predicting music popularity due to the non-linear nature of the problem.

B. Feature Importance

The SHAP analysis (Figure 5) reveals distinct patterns in feature contribution to music popularity prediction. Track identification features dominate the model with `track_identifier` contributing the highest impact at 4.71, while artist information through `creator_collective` ranks second at 3.82. These

Method No	Approach Number	Test RMSE	Validation RMSE	MSE	MAE	R2	Rank
1	**Ensemble with Six Models**	**9.0035**	**8.916**	**79.488**	**4.878**	**0.829**	**1**
2	XGBoost 1 (Deep XGBoost)	9.3012	9.265	85.838	5.459	0.815	3
3	Ensemble with XGB, LGBM 1	9.0283	9.167	84.028	5.4148	0.8194	2
4	Ensemble with XGB, LGBM 2	9.4691	9.653	93.182	6.2066	0.7997	4
5	Voting Ensemble with XGB, LGBM, CatBoost	9.7538	9.845	95.124	6.415	0.7956	5
6	XGBoost 2	9.8695	9.923	97.401	6.578	0.7907	6
7	Ensemble with XGB, RF, ExtraTrees	10.2796	10.117	102.354	6.543	0.7801	7
8	Linear Models	19.6991	19.474	379.237	16.1513	0.5842	12
9	RandomForests	11.9678	11.5827	134.159	5.6062	0.6928	8
10	XGBoost 3	12.2388	12.0674	145.622	6.7256	0.6724	9
11	XGBoost 4	12.5847	12.384	7.012	158.374	0.6435	10
12	Extra Trees	13.1245	12.897	7.458	172.251	0.6124	11

Fig. 4. Model Performance Comparison

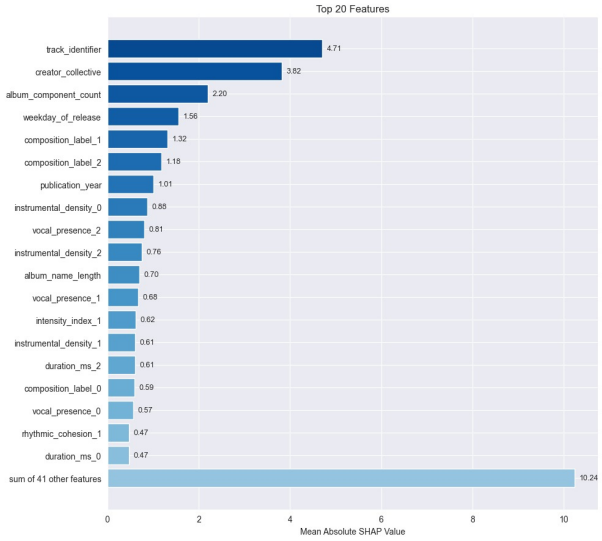

Fig. 5. Top 25 features according to the best method

metadata features substantially outweigh audio characteristics in predictive power.

Release structure emerges as a critical factor, with `album_component_count` achieving 2.20 SHAP value, indicating that the number of tracks in a release significantly influences popularity scores. Temporal factors demonstrate notable influence through `weekday_of_release` (1.56) and `publication_year` (1.01), establishing release timing as a measurable predictor.

Audio feature analysis shows distributed importance across track positions. Instrumental density features exhibit position-dependent predictive power, with the primary track (`instrumental_density_0`: 0.88) demonstrating higher importance than secondary tracks (`instrumental_density_2`: 0.76, `instrumental_density_1`: 0.61). Popularity scores respond to instrumental density variations, indicating that specific orchestration patterns, whether sparse or dense, influence audience reception. Vocal presence follows a similar position-based importance hierarchy, ranging from 0.81 to 0.57.

Track labeling contributes consistently to predictions, with composition labels across different track positions showing values between 1.32 and 0.59. Duration features demonstrate position-specific influence, with `duration_ms_2` at 0.61 and `duration_ms_0` at 0.47. Album naming characteristics, captured through `album_name_length` (0.70), provide additional predictive signal.

The feature distribution shows the top 19 features accounting for approximately 70% of the total influence. The remaining 41 features collectively contribute 10.24 SHAP value (30% of the total influence), representing distributed predictive power across numerous lower-impact variables. This distribution indicates that while primary metadata features drive predictions, comprehensive accuracy requires integration of multiple audio and contextual characteristics.

V. Discussion

A. Feature Impact Importance

Feature importance analysis shows that metadata features have higher predictive power than audio characteristics. Track identification (SHAP value 4.71), artist information (3.82) and album component count (SHAP value 2.20) have the highest impact on predictions, while audio features like `instrumental_density` (0.88) contribute slightly less. This finding shows that audio content alone does not determine popularity and that artist reputation, official title of the album and track titles, album size, track placement (order) within albums/playlists.

Among the core numerical audio features, `intensity_index`, `instrumental_density` and `duration` displayed higher importance values. The `intensity_index`, which defines the energy level of the music, has higher impact (SHAP 0.62) on popularity predictions, indicating that energy levels significantly influence track success and that musicians can optimize these acoustic properties when producing tracks to increase commercial appeal and market success.

The temporal dimension proves significant, with release timing features contributing measurably to predictions. The `weekday_of_release` (SHAP 1.56) and `publication_year` (1.01) demonstrate that strategic timing decisions impact commercial success. This temporal sensitivity suggests that market dynamics and audience behavior patterns follow predictable cycles that can be leveraged for popularity forecasting.

B. Ensemble Model Performance

The experimental results show that ensemble methods outperform individual algorithms for music popularity prediction. The top-performing ensemble meta-model achieved an RMSE of 9.0035, compared to the best individual model (XGBoost 1 with RMSE 9.3012). Five of the seven best-performing

approaches used ensemble techniques, showing that model diversity and combination strategies are important for music popularity prediction.

The superior performance of the six-model ensemble results from the different algorithmic approaches of its constituent models. The ensemble consists of six regression models, each contributing uniquely. The first XGBoost regressor captures nonlinear relationships and feature interactions using gradient boosting. The second XGBoost regressor employs deeper trees and a larger search space to model more complex patterns that the first may miss. LightGBM introduces diversity through its leaf-wise tree growth and histogram-based learning, resulting in different prediction patterns compared to XGBoost. Extra Trees increases ensemble diversity by using fully randomized splits, which reduces correlation among trees and improves generalization. Histogram Gradient Boosting contributes by efficiently building trees with histogram binning, which improves training speed and captures subtle data distribution differences not easily detected by other models. Random Forest stabilizes predictions through bootstrap aggregating and random feature selection, capturing patterns that boosting methods might overlook. Together, these models combine distinct algorithmic approaches to enhance the ensemble's accuracy and robustness.

C. Feature Engineering Effectiveness

Feature engineering approaches produced different success rates. The most effective strategies used domain-specific transformations (method 1: Ensemble with 6 models), including temporal pattern extraction (`publication_year`) and deriving song eras (`data_period`). Excessive feature creation without statistical validation reduced performance in method 3 and 4 (Ensemble with XGB, LGBM 1 and Ensemble with XGB, LGBM 2). The best-performing method used meticulous feature engineering with hypothesis testing and significance validation, showing that targeted feature development works better than creating many features without validation.

D. Potential Improvements

a) Market Context Parameters Integration: The dataset lacks market-related parameters like streaming metrics, radio airplay data, and social media engagement. Integrating streaming platform data (playlist inclusion, skip rates, user engagement) and traditional media exposure metrics would capture commercial dynamics beyond audio features, potentially improving prediction accuracy.

b) External Dataset Validation: Current evaluation uses only the Billboard dataset, limiting generalization assessment. Testing on independent datasets from different time periods, regions, or platforms would provide robust real-world performance evaluation and identify potential overfitting.

c) Ensemble Optimization for Deployment: The six-model ensemble achieves optimal performance but requires substantial computational resources. Investigating smaller 2–3 model configurations could maintain accuracy while reducing overhead, improving deployment feasibility in resource-constrained environments.

d) Deep Learning Architecture Implementation: CNNs applied to audio spectrograms could capture complex patterns that tree-based models miss. Raw audio processing through CNN architectures would automatically extract features rather than using pre-computed attributes. Multi-task learning frameworks predicting genre, mood, and popularity simultaneously could enhance generalization through shared representations across related music analysis tasks.

VI. Conclusion

This study demonstrates that machine learning models can effectively predict music popularity using audio features and metadata. The ensemble approach achieved superior performance over individual models, and XGBoost provided the strongest individual contributions. Although current results show promise for practical applications in music recommendation and A&R decision-making, the identified improvements represent clear pathways for improved prediction accuracy. Future work incorporating these enhancements could significantly advance automated music popularity prediction capabilities for industry applications.

Acknowledgment

The authors thank the University of Moratuwa Department of Computer Science for providing computational resources and the Kaggle platform for hosting the music popularity prediction competition that motivated this research.

References

[1] D. Martín-Gutiérrez, G. Hernández Peñaloza, A. Belmonte-Hernández, and F. Álvarez, "A Multimodal End-To-End Deep Learning Architecture for Music Popularity Prediction," IEEE Access, 2019. [Online]. Available: https://oa.upm.es/79105/1/IEEE_Open_access_music_hit_prediction.pdf

[2] A. Boughanmi and M. Ansari, "Feature Learning and Deep Architectures: New Directions for Music Informatics," 2021. [Online]. Available: https://www.semanticscholar.org/paper/Dynamics-of-Musical-Success:-A-Machine-Learning-for-Boughanmi-Ansari/e84c61d8317faba3e86eb22fc9b7fb39d645abe5

[3] P. Knees et al., "Hit Song Science Is Not Yet a Science," SIGIR, 2011. [Online]. Available: http://sigir.org/wp-content/uploads/2021/09/p21.pdf

[4] P. Knees, M. Schedl, and G. Widmer, "Hit Song Science Once Again a Science," in *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, 2013, pp. 465–470.

[5] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere, "The Million Song Dataset," in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, Miami, FL, USA, 2011, pp. 591-596.