Tree-Based Models Outperform Transformers for Music Engagement Prediction

João Victor de Souza Albuquerque
Department of Computer Science
Federal Institute of Ceará (IFCE)
Fortaleza, Brazil
joao.victor08@aluno.ifce.edu.br

Amauri Holanda de Souza Junior
Department of Computer Science
Federal Institute of Ceará (IFCE)
Fortaleza, Brazil
amauriholanda@ifce.edu.br

***Abstract*—Youtube and Spotify have transformed media consumption by enabling creators to reach global audiences without traditional intermediaries. Music content, in particular, dominates user engagement, motivating the development of predictive tools to forecast content popularity. Despite the emergence of Transformer-based models as the state-of-the-art across various domains, their effectiveness in music engagement prediction remains unexplored. In this paper, we investigate whether Transformers for tabular data (FT-Transformer) can outperform traditional tree-based models such as XGBoost and Random Forest on a real dataset combining Spotify and YouTube metadata. Our findings reinforce prior results that highlight the dominance of tree-based models on tabular data, with the FT-Transformer failing to surpass them across all considered settings. However, we show that the FT-Transformer remains competitive against XGBoost, particularly when GPU resources are available. These results provide further empirical support for the limitations of current Transformer-based approaches in low-dimensional, heterogeneous tabular data regimes, and suggests future research combining tabular data with multimodal inputs like lyrics or video content.**

***Index Terms*—FT-Transformer, Tabular Data, Music Engagement Prediction, Random Forest, XGBoost**

I. Introduction

Platforms like YouTube have revolutionized music, video, and media distribution by democratizing content creation and enabling global access, allowing artists and creators to reach vast audiences without traditional gatekeepers. According to Pedroso *et al.* [1], in the last quarter of 2016 alone, approximately 4.95 million videos were watched on the platform, with 300 hours of video content uploaded every minute. Additionally, YouTube receives around 900 million monthly visits, accounting for a total of 3.25 billion hours of video watched. These numbers highlight the immense visibility that content can achieve once it becomes popular on the platform. The same study also found that music is the most frequently searched type of content by users, underlining the importance of intelligent tools for predicting the performance (popularity) of musical content on YouTube.

Recent data [2] show that the Brazilian music market alone generated R$3.486 billion in revenue in 2024, ranking ninth globally. Worldwide, the music industry amassed a total of 29.6 billion USD in revenue during the same year. These figures underscore not only the economic significance of the music industry but also the value of reliable predictive and analytical tools to support decision-making and market strategies.

The Transformer architecture, introduced by A. Vaswani *et al.* [14] in the groundbreaking paper "Attention Is All You Need," has quickly became the *model of choice* for a plethora of learning tasks. Its versatility and power have led to widespread adoption across a variety of domains, including serving as the foundation for large-scale language models such as OpenAI's ChatGPT. Motivated by the success of Transformers across different data types, interest has grown in applying these models to tabular data — still prevalent in many real-world applications. In this context, Gorishniy *et al.* [15] introduced the FT-Transformer, which adapts the Transformer architecture to the tabular domain via a specialized feature embedding mechanism. Nevertheless, tree-based models such as XGBoost continue to dominate practical applications on tabular datasets [20]. Understanding the conditions under which deep learning (DL) methods outperform tree-based models in this setting remains an active area of research.

In this regard, Grinsztajn *et al.* [20] investigate the performance of deep learning models on tabular data. In an extensive empirical study, they found that tree-based models still beat DL methods, and claim that "*neural networks struggle to learn irregular patterns of the target function, and their rotation invariance hurt their performance, in particular when handling the numerous uninformative features present in tabular data*".

For music engagement prediction, Pareek *et al.* [21] and Yee *et al.* [23] explore the use of machine learning to predict popularity. However, their approaches relied on traditional architectures, overlooking modern DL ones. As a result, the question of whether Transformer-based models can outperform tree-based methods in this specific domain remains open.

In this paper, we aim to answer this question. In particular we evaluate whether a modern architecture — namely, the FT-Transformer [15] — can outperform traditional models for tabular data such as XGBoost and Random Forest in predicting music engagement. To this end, we considered the "Spotify and YouTube" dataset [3]. In addition, to account for temporal information, a new feature was engineered to encode temporal patterns into the data. The baseline models evaluated in this study include Linear Regression, Random Forest, XGBoost, and a Multi-Layer Perceptron (MLP). Regarding model selection, we considered both default hyperparameters — as implemented in broadly used toolboxes and according to the guidelines in [15] — and *hold-out* validation using grid-search and Optuna [7], which allows exploiting Bayesian optimization for hyperparameter selection.

Our results align well with the findings of [20]. More specifically, tree-based models are the best-performing methods in all scenarios. The results are analyzed and contextualized with respect to existing literature, particularly the findings of L. Grinsztajn *et al.* [20], illustrating how this study contributes to emerging research directions in applying Transformer-based models to tabular prediction tasks.

The rest of the paper is organized as follow. In Section II, we discuss related works. Section III provides a detailed description of the task (and dataset) of interest. Section IV overviews the baselines and FT-Transformer. Section V presents the evaluation setup and methodology. In Section VI, we report the obtained results. Finally, Section VII concludes the paper.

II. Related Works

Most studies involving the prediction of music performance on streaming platforms primarily focus on Spotify. Many of these works employ different machine learning models and propose diverse preprocessing techniques, as observed in the following studies.

Pareek *et al.* [21] explore the prediction of song popularity by leveraging audio features provided by Spotify, such as loudness, energy, and acousticness. The authors applied three machine learning classifiers — Random Forest, K-Nearest Neighbors (KNN), and Linear Support Vector Classifier (LSVC) — to classify songs as popular or not. Their findings indicate that the Random Forest classifier outperformed the others, achieving the highest accuracy, precision, recall, and F1-score, thereby demonstrating its effectiveness in predicting music popularity based on audio features alone.

Yee *et al.* [23] investigate whether combining audio features from Spotify with social media metrics from YouTube can enhance the prediction of a song's popularity. The study constructed a dataset of newly released tracks, collecting audio features from Spotify and social media data (such as views, likes, and comments) from YouTube. Popularity was quantified using five metrics derived from Spotify's Top 200 daily chart. Machine learning models were trained using two feature sets: one with only audio features and another combining audio and social media features. The results demonstrated that incorporating social media data significantly improved model performance, with accuracy gains ranging from 10% to 60%, highlighting the value of multimodal data in predicting music popularity.

Other studies have explored more advanced deep learning techniques to design specialized models for the task of music popularity prediction on the Spotify platform. These works often go beyond traditional approaches by also proposing customized data preprocessing methods tailored to the characteristics of the problem. For instance, Martín-Gutiérrez *et al.* [22] introduce HitMusicNet, an innovative end-to-end deep learning architecture designed to predict the popularity of music recordings. The model integrates multiple data modalities, including audio features, metadata, and potentially other relevant information, to capture the complex factors influencing a song's success. The authors also present the SpotGenTrack Popularity Dataset (SPD) to facilitate benchmarking and comparison of predictive models in this domain. HitMusicNet demonstrated strong performance in predicting music popularity, underscoring the effectiveness of multimodal deep learning approaches in this field.

The work presented in this study differs from previous research by focusing on the predictive capabilities of various models specifically for music data on the YouTube platform. Also, it compares the performance of well-established classical models, commonly used in tabular data prediction tasks, with that of a modern architecture (FT-Transformer) designed specifically to operate in the tabular domain.

III. Dataset Description

This section describes the data used for model training. The data were obtained from Spotify and YouTube dataset. This dataset was created by collecting various statistical and descriptive information from ten different songs by various artists. It contains approximately twenty thousand samples and twenty-six features that describe each song. These features are divided into musical features and descriptive features.

The musical features describe technical aspects of the songs using statistical measurements. These features include:

- **Danceability**: Indicates how suitable a track is for dancing. The value ranges from 0.0 (least danceable) to 1.0 (most danceable).
- **Energy**: Represents a perceptual measure of intensity and activity, ranging from 0.0 to 1.0.
- **Key**: An integer value that represents the key of the track, based on standard Pitch Class notation.
- **Loudness**: The overall loudness of the track in decibels (dB).
- **Speechiness**: Measures the presence of spoken words in a track. Values close to 1.0 indicate tracks that are primarily speech-like (e.g., audiobooks or podcasts), while values near 0.0 indicate music with little or no speech content.
- **Acousticness**: A confidence measure from 0.0 to 1.0 indicating the likelihood that the track is acoustic.
- **Instrumentalness**: A confidence measure indicating whether a track contains no vocals.
- **Liveness**: Detects the presence of an audience in the recording. Higher values suggest a greater probability that the track was recorded live.
- **Valence**: Describes the musical positiveness conveyed by a track, ranging from 0.0 to 1.0.
- **Tempo**: The estimated tempo of a track in beats per minute (BPM), representing its speed based on the average beat duration.
- **Duration_ms**: The duration of the track in milliseconds.

The descriptive features provide non-statistical information and details related to the song's performance on Spotify and YouTube. These features include:

- **Track**: The name of the song on Spotify.
- **Artist**: The name(s) of the artist(s) featured on the track.
- **Album**: The name of the album that contains the song on Spotify.
- **Album_type**: Indicates whether the song was released as a single or as part of an album.
- **Uri**: A Spotify link used by the platform's API to locate the track.
- **Url_youtube**: The URL of the song's video on YouTube.
- **Title**: The title of the video on YouTube.
- **Channel**: The name of the YouTube channel that published the video.
- **Description**: The video's description on YouTube.
- **Licensed**: Indicates whether the video is licensed content.
- **Official_video**: A boolean value indicating whether the video is the official music video.
- **Stream**: The number of times the song was streamed on Spotify.
- **Views**: The number of views of the video on YouTube.
- **Likes**: The number of likes the video received on YouTube.
- **Comments**: The number of comments on the YouTube video.

After analyzing the data dependency over time, it was determined that some attributes vary significantly with the passage of time. Therefore, two additional columns were added: one indicating the release date of the video on YouTube, and another indicating the release date of the song on Spotify.

IV. Models

This section briefly overviews the models used in this work.

A. Random Forest

Random Forest is an ensemble learning method that constructs a multitude of decision trees during training and outputs the mean prediction (in regression) of the individual trees. It introduces randomness by selecting subsets of the data and features for each tree, which helps reduce overfitting and increases generalization performance. The algorithm is capable of capturing complex nonlinear relationships and interactions between features without requiring extensive preprocessing. Random Forest has been successfully applied in various scientific and industrial domains due to its robustness and high predictive accuracy [17], [18].

B. Extreme Gradient Boosting (XGBoost) Model

According to Jin et al. [12], Extreme Gradient Boosting (XGBoost) is an implementation of the boosting method that seeks to build a strong classifier by combining multiple weak classifiers. As Friedman [13], cited in Zeraatgari *et al.*. [8], states, XGBoost is an advanced version of the gradient boosting algorithm, offering high efficiency, flexibility, and powerful applicability. Jin *et al.* [12] highlight recent applications of XGBoost in astronomy, such as pulsar signal separation and classification of unidentified sources in the Fermi-LAT catalog.

C. Linear Regression

Linear regression is one of the most fundamental and widely used algorithms for regression tasks. It assumes a linear relationship between the input variables and the target variable, aiming to estimate the coefficients that minimize the error between the predicted and actual values, typically using the Mean Squared Error (MSE) as the loss function. Despite its simplicity, linear regression is useful as a baseline model and often performs well when the relationship between features and the target is approximately linear. Its interpretability and low computational cost make it a valuable tool in both research and practical applications [16].

D. Multi-Layer Perceptron (MLP) Model

Multilayer Perceptrons (MLPs) are a class of artificial neural networks composed of an input layer, one or more hidden layers, and an output layer. Each hidden layer contains neurons that apply a linear transformation followed by a non-linear activation function to the input received from the previous layer. The network learns by adjusting its weights through the backpropagation algorithm to minimize a predefined loss function [8]. These models can include multiple hidden layers with a large number of neurons, allowing them to capture complex patterns in the data.

E. ft-Transformer Model

The Transformer architecture, proposed by A. Vaswani *et al.* [14], marked a significant shift in the design of deep learning models for sequence modeling. It dispensed with recurrent and convolutional components in favor of a fully attention-based mechanism. The core innovation is the multi-head self-attention mechanism, which allows each element in a sequence to attend to all others, capturing long-range dependencies efficiently. The architecture is composed of stacked encoder and decoder blocks, each containing layers of self-attention, feedforward neural networks, layer normalization, and residual connections. Furthermore, positional encodings are added to the input embeddings to retain information about the order of the sequence, which is essential due to the absence of recurrence. This highly parallelizable structure enables faster training and has established the Transformer as the backbone of many state-of-the-art models in natural language processing.

In this study, we adopt an adapted Transformer model tailored for tabular data, known as the FT-Transformer, introduced by Gorishniy *et al.* [15]. Unlike the original Transformer, which was designed for sequential inputs, the FT-Transformer operates on sets of tabular features, combining both numerical and categorical variables. It encodes each feature as a token using learnable embeddings, enabling the attention mechanism to capture inter-feature dependencies in a data-driven manner. Additionally, it employs a simplified structure with a [CLS] token for aggregation, similar to BERT-based models, and leverages a Feature Tokenizer to handle heterogeneous inputs. This adaptation has shown competitive performance in various tabular data benchmarks, offering a powerful alternative to traditional models like gradient boosting or fully connected neural networks in structured data tasks.

V. Methodology

A. Data Preprocessing

1) Data Engineering: In addition to the existing columns, new features were engineered to enrich the dataset. The first feature, *days_on_platform*, represents the number of days since a video was uploaded to the platform. The second feature, *artist_number*, captures the total number of distinct artists involved in a given track. The categorical column *album_type* was processed using one-hot encoding for use in what we refer to as *classical models*, which include Random Forest, XGBoost, Linear Regressor, and Multi-Layer Perceptron (MLP). For the FT-Transformer, this column was left unprocessed, as it is designed to handle categorical variables internally without requiring traditional encoding methods.

Finally, the *engagement_rate* feature was constructed to serve as the target variable for the regression task. This metric combines key indicators of content performance—including likes, shares, and views—into a single value, offering a robust and comprehensive measure of audience engagement for predictive modeling. The engagement formula used was:

$$\text{engagement} = \frac{\text{likes} + \text{comments}}{\text{views}} \tag{1}$$

2) Feature Selection: After the feature engineering step, the dataset underwent a feature selection process to determine which variables would be included in the final dataset. All musical features were retained, as the goal was to enable the model to predict the target variable based on the intrinsic characteristics of the music, in addition to a few descriptive attributes.

The selected descriptive features were *artist_number*, *album_type*, *stream*, and *engagement_rate* — the latter being the target variable. Other features were excluded either due to redundancy or because they were too difficult to interpret in a way that could contribute positively to the experiment. The *stream* feature was specifically retained because empirical testing showed that its inclusion improved model performance across different algorithms.

3) Data Cleaning: The next preprocessing step involved cleaning the selected features. The first cleaning operation consisted of removing all rows containing missing values in any of the selected features. This strategy was adopted due to the large number of samples in the dataset, which made the data loss from this operation negligible.

The second cleaning procedure addressed outliers in the target feature. To accomplish this, the data were first segmented into time-based ranges: 0 to 30 days, 31 to 90 days, 91 to 365 days, and 366 days to the maximum value observed. This segmentation strategy was motivated by the observation that engagement tends to decline over time for videos on the platform. Without this segmentation, applying a global outlier threshold could result in recent videos with naturally higher engagement being misclassified as outliers.

By segmenting the data, the analysis focused on identifying anomalous engagement values relative to each specific time range. The Interquartile Range (IQR) method, a widely used technique for detecting statistical outliers [19], was applied to each segment. Data points exceeding the upper bound were removed from the dataset.

4) Data Splitting Strategy: Following the segmentation described in the previous step, the subset of songs that had been on the platform for 0 to 30 days was selected as the basis for the main data split. This subset was first divided into two separate sets: one for training and validation, and the other for testing. A 75%-25% split was used, allocating 75% of the data to the training/validation set and 25% to the test set. Subsequently, the training/validation set was further split into training and validation sets, following an 80%-20% ratio. Random seeds were applied to ensure the reproducibility of the experiment.

To enrich the training data, the training set composed of songs with up to 30 days on the platform was combined with the remaining data from songs older than 30 days. This data splitting and training set formation strategy was motivated by two main factors. First, the objective was to evaluate the model's ability to predict engagement for newly released songs, particularly during their first few weeks on the platform. This justifies the 30-day window as a meaningful definition of "new" music. Second, experimental results showed that including older songs in the training set significantly improved model performance when compared to training with only recent songs. Thus, all older data were incorporated exclusively into the training process.

5) Normalization Process: The final step in data preparation was normalization, which was handled differently for classical models and the modern model.

For the classical models, Min-Max normalization was applied using the `MinMaxScaler()` function from the `scikit-learn` library. This transformation was performed on the feature columns (i.e., the predictors X) of the datasets to scale the values to a standard range.

In contrast, the normalization approach for the modern model, FT-Transformer, followed the official implementation provided by the original authors on their GitHub repository [15]. According to their experimental setup and recommendations, the X features were preprocessed using the `QuantileTransformer()` from `scikit-learn`, combined with Gaussian noise injection. The target variable y was normalized using its standard deviation, following the custom normalization function provided in the same repository. This preprocessing pipeline aligns with the configuration used by Gorishniy *et al.* [15] in their experiments.

Figure 1 illustrates the preprocessing pipeline previously described.

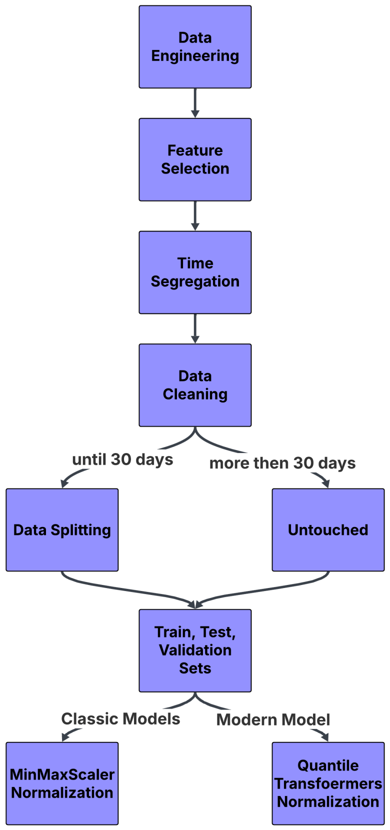

Fig. 1. The Preprocessing Pipeline.

B. Model Selection

We considered two cases: (1) no model selection, (2) model selection either via grid search or Optuna.

In the first case, the experiment was designed to evaluate how each model, when configured with default hyperparameters from the `scikit-learn` toolbox, performs on the dataset — as unexperienced practitioners would apply ML. For this purpose, each model underwent ten independent trials. In each round, the data were partitioned according to the procedure described in the *Data Splitting Strategy* section. However, in this case, the training and validation sets were not separated, as no hyperparameter tuning was conducted.

In the second case, we aim to assess the benefits of hyperparameter optimization. Each model underwent ten independent trials, allowing the collection of both the average performance across the best configurations and the performance of the best single configuration in terms of the lowest Mean Squared Error (MSE) on the test set.

This experiment employed two different hyperparameter optimization strategies: GridSearch and Optuna. Both optimizers followed the same experimental procedure, which is detailed below. It is important to note that GridSearch was applied only to the classical models due to the computational constraints associated with tuning the FT-Transformer.

The training and validation sets, along with the model, were provided to the hyperparameter optimizer. Once the best hyperparameters for the round were identified, the final model for that round was retrained using the combined training and validation sets. The performance of this final model was then evaluated on the test set. Among the ten rounds, the model yielding the lowest test MSE was identified and retained for comparison purposes.

Additionally, the MSE from each round's final model was recorded to compute the overall mean and standard deviation, providing a more comprehensive view of the tuning process. The specific hyperparameters considered for each model during tuning are described in the Appendix. Figure 1 provides a visual overview of the model selection procedure.

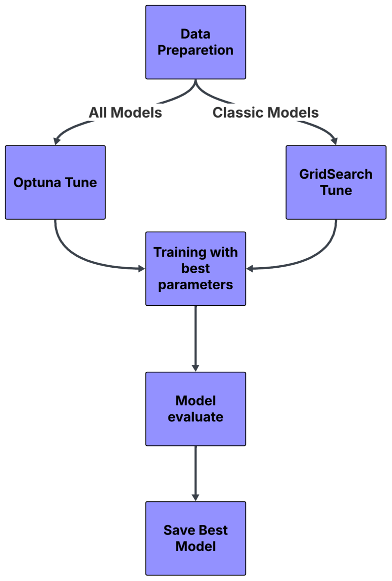

Fig. 2. Model tune pipeline.

C. Evaluation

As previously discussed, we use MSE as performance metric. The values obtained in each round were recorded to compute the mean and standard deviation of the MSE across all ten runs. Moreover, the performance of the models was evaluated through a multi-step comparison. First, the results of the default versions of all models were compared with each other. Next, the tuned versions of each model were compared against their respective default versions to assess the impact of hyperparameter optimization. The improvement achieved by the best tuned configuration for each model was then quantified in relation to its default version. Finally, the best tuned models were compared to determine which achieved the highest overall performance.

VI. Results

A. Default Models Performance

By analyzing the results of the default models in Table I, it becomes evident that the Linear Regressor significantly underperformed compared to all other models. This likely stems from its inability to construct a regression function capable of adequately fitting the data samples, which severely compromised its predictive power. This limitation is particularly critical for the Linear Regressor, as it lacks hyperparameters that could be adjusted to improve performance.

In contrast, the XGBoost, FT-Transformer, and Random Forest models delivered comparable results, making it challenging to identify the most suitable option for this regression task at this stage. Although the MLP model did not achieve competitive performance in the default configuration, its results were not substantially behind those of the top-performing models. As such, it remains a potentially viable approach, particularly after hyperparameter optimization.

Focusing specifically on the performance of the FT-Transformer, it is noteworthy that even in its default configuration, the model outperforms the MLP, which is a commonly used architecture for tasks in the tabular domain. Additionally, its performance is not far behind that of XGBoost—a widely adopted model in various tabular data applications due to its strong predictive capabilities and efficiency. Random Forest is the best performing method in this case.

TABLE I
PERFORMANCE OF DEFAULT MODELS OVER TEN INDEPENDENT TRAINING ROUNDS

Model	***Mean MSE***	***STD MSE***
Linear Regressor	7.2316	0.7596
Random Forest	3.1358	0.3833
MLP	4.1114	0.4997
XGBoost	3.3707	0.4148
FT-Transformer	3.5601	0.6433

B. *Tuned models Performance*

By analyzing the tuned models presented in Table II, it becomes evident that the performance of the MLP deteriorated after hyperparameter optimization using both Optuna and GridSearch, with average MSE values higher than those obtained with the default configuration. However, this behavior was present to some extent across all models, which may be explained by the distribution of engagement over days in the training and test sets: the data split was not stratified either by day or by engagement level. Consequently, the training set may have contained less representative patterns and engagement levels for accurate prediction. Furthermore, when comparing the best MLP model (identified through Optuna) to the best-performing versions of the other models, the MLP remained the worst overall, with an MSE approximately 0.4 points higher.

Regarding the FT-Transformer's performance in contrast to XGBoost and Random Forest, the average MSE of the modern model was approximately 0.47 points higher than that of Random Forest and 0.48 points higher than XGBoost. These results are aligned with the conclusions presented by L. Grinsztajn *et al.* [20], who argue that tree-based models often outperform deep learning models on tabular data. Nevertheless, when considering only the best-performing configuration of the FT-Transformer, the model achieved competitive results compared to XGBoost, with a negligible difference of just 0.0011 in MSE. Consequently, the choice between these two models may largely depend on the available computational resources. It is important to highlight that the entire tuning process for XGBoost was conducted on a local machine equipped with an 11th-generation Intel Core i5 processor and 8 GB of RAM, whereas the tuning of the FT-Transformer required the use of Google Colab with a T4 due to the need for a GPU-enabled environment.

TABLE II
PERFORMANCE OF TUNED MODELS USING TWO OPTIMIZATION METHODS

Model	***Optimizer***	***Mean MSE***	***STD MSE***	***Best MSE***
Random Forest	GridSearch	3.3750	0.4742	2.5401
	Optuna	3.3223	0.4055	2.5819
MLP	GridSearch	4.9773	0.9203	3.5314
	Optuna	4.2563	0.5586	3.1479
XGBoost	GridSearch	3.3049	0.4324	2.7117
	Optuna	3.3086	0.4402	2.7636
FT-Transformer	Optuna	3.7982	0.9625	2.7647

TABLE III
TIME SPENT ON HYPERPARAMETER TUNING (IN MINUTES) FOR EACH MODEL AND OPTIMIZER

Model	***Optimizer***	***Time (min)***
Random Forest	GridSearch	82.2833
	Optuna	118.6333
MLP	GridSearch	220.05
	Optuna	828.8333
XGBoost	GridSearch	612.95
	Optuna	32.9667
FT-Transformer	Optuna	212.87

Comparing the best FT-Transformer configuration with the best-performing Random Forest model, we observe a more significant difference of 0.22 in MSE. This further reinforces Random Forest as the most effective model for this task, not only in terms of raw performance but also due to its algorithmic simplicity and lower computational demands.

Based on the findings of L. Grinsztajn *et al.* [20] and the results reported by A. Charchyan [24], the FT-Transformer did not achieve the best performance for this task. One key reason lies in the nature of the input and output data, which still exhibit relatively abrupt variations—even after the normalization process—as highlighted by Grinsztajn *et al.* [20]. This limitation became even more evident during the preliminary experimentation phase, where the best approach to using the FT-Transformer was investigated. It was consistently observed that removing the normalization step from either the input features (X) or the target variable (y) caused a significant performance drop. In contrast, as shown by L. Grinsztajn *et al.* [20], tree-based models are more robust to such irregularities in the data.

Additionally, early data analysis raised the hypothesis that a song's engagement level is not uniformly influenced by all its musical characteristics. This suggested that certain audio features might be more important than others, as these features may exhibit recurring patterns within specific genres—and some genres are more popular than others. This hypothesis was later supported by the work of Charchyan [24], which demonstrated that the features *danceability* and *loudness* are significantly more relevant than others.

While this imbalance in feature relevance might initially appear to impact all models, it aligns with the third hypothesis presented by L. Grinsztajn *et al.* [20], which states that the presence of low-signal features can severely impair the performance of MLP-like neural networks. In their study, this was one of the main reasons why tree-based models continue to outperform deep learning approaches in tabular domains. Nevertheless, these insights point to new directions for approaching this problem through alternative data representations.

When analyzing the computational cost of the models in terms of execution time (Table III), it can be observed that the FT-Transformer does not exhibit a disproportionately high computational cost compared to the other models. Among all approaches, XGBoost achieved the fastest tuning process, demonstrating its efficiency and ability to quickly adapt to the characteristics of the problem..

VII. Conclusion

In conclusion, considering the task of predicting song engagement on the YouTube platform, the assertion by L. Grinsztajn *et al.* [20]—that tree-based models generally perform better on tabular data—remains valid for the problem addressed in this work. Among the tree-based models evaluated, Random Forest delivered the best overall performance in solving the task. Furthermore, the FT-Transformer model demonstrated competitive results when compared to XGBoost, suggesting that although tree-based models outperformed in this specific scenario, there may be other tabular prediction tasks where FT-Transformers can match or even surpass their performance. This supports the idea that the conclusion of L. Grinsztajn *et al.* [20] may represent a strong trend rather than a strict rule, particularly when considering an effective use of the embedding mechanism of the FT-Transformer. Such a mechanism would make it feasible to include features like the ”Artist” column, which carries a strong correlation with the level of engagement and audience of a song. This attribute was not used in the classical models because encoding it through one-hot encoding would generate more than two thousand additional columns in the dataset. However, the FT-Transformer is capable of learning an appropriate tokenization strategy for such a large number of classes, thus allowing this type of information to be exploited more efficiently.

Our results, aligned with the findings of A. Charchyan [24], also open up promising directions for future research. For both YouTube engagement prediction and Spotify popularity forecasting, one potential avenue involves combining tabular features with textual information extracted from song lyrics using natural language processing (NLP) techniques. For researchers focusing on Spotify, this fusion could enhance the predictive models. Conversely, for YouTube-related tasks, integrating tabular features with video content through image processing techniques represents a compelling approach for improving performance prediction models.

References

[1] E. Pedroso, L. T. Borges, and P. Oliveira, ”YOUTUBE,” B.S. thesis, Instituto Superior Técnico, Lisboa, Portugal, 2016.

[2] UBC, ”Música gravada no Brasil supera, pela primeira vez, os R$ 3 bi em receitas em 2024,” *UBC*, Mar. 19, 2025. [Online]. Available: https://www.ubc.org.br/publicacoes/noticia/23178/musica-gravada-no-brasil-supera-pela-primeira-vez-os-r-3-bi-em-receitas-em-2024

[3] S. Rastelli, M. Sallustio, and M. Guarisco, “Spotify and YouTube,” *Kaggle*, 2022. [Online]. Available: https://www.kaggle.com/datasets/salvatorerastelli/spotify-and-youtube

[4] C. A. Fernandes, *Inteligência Artificial: Fundamentos, Desenvolvimento e Aplicações*, São Paulo: Érica, 2005.

[5] J. H. Boose, ”A Survey of Knowledge Acquisition Techniques and Tools,” in *Knowledge Acquisition for Knowledge-Based Systems*, San Diego: Academic Press, 1994.

[6] M. V. Mohammed, B. Javed, and S. Mushtaq, “Machine Learning Algorithms for Disease Prediction,” in *2018 7th Int. Conf. on Reliability, Infocom Technologies and Optimization (ICRITO)*, 2018.

[7] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2019.

[8] F. Z. Zeraatgari et al., “Machine learning-based photometric classification of galaxies, quasars, emission-line galaxies and stars,” *Monthly Notices of the Royal Astronomical Society*, vol. 525, no. 1, pp. 199–211, 2023.

[9] P. O. Baqui et al., “Photometric redshift estimation using machine learning models with the miniJPAS dataset,” *Astronomy & Astrophysics*, vol. 644, A55, 2020.

[10] E. Charles, “Árvores de decisão aplicadas ao problema da separação estrela/galáxia,” Universidade Federal do Rio Grande do Sul, 2011.

[11] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, 1995.

[12] X. Jin et al., “A machine learning method to separate pulsar signals from noise,” *Monthly Notices of the Royal Astronomical Society*, vol. 485, no. 3, pp. 3561–3574, 2019.

[13] J. H. Friedman, “Greedy Function Approximation: A Gradient Boosting Machine,” *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[14] A. Vaswani et al., “Attention Is All You Need” in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[15] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, ”Revisiting Deep Learning Models for Tabular Data,” in Advances in Neural Information Processing Systems, vol. 34, pp. 18932–18943, 2021.

[16] J. James, D. Witten, T. Hastie, and R. Tibshirani, An Introduction to Statistical Learning: with Applications in R, Springer, 2013.

[17] L. Breiman, “Random forests,” Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[18] G. Louppe, “Understanding Random Forests: From Theory to Practice,” arXiv preprint, arXiv:1407.7502, 2014.

[19] J. W. Tukey, *Exploratory Data Analysis*. Reading, MA: Addison-Wesley, 1977.

[20] L. Grinsztajn, E. Oyallon, and G. Varoquaux, “Why do tree-based models still outperform deep learning on tabular data?,” Adv. Neural Inf. Process. Syst., vol. 35, pp. 22977–22990, 2022.

[21] P. Pareek, P. Shankar, P. Pathak, and N. Sakariya, ”Predicting Music Popularity Using Machine Learning Algorithm and Music Metrics Available in Spotify,” Journal of Development Economics and Management Research Studies, vol. 9, pp. 10–19, Jan.–Mar. 2022.

[22] 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, vol. 8, pp. 39361–39374, 2020, doi: 10.1109/ACCESS.2020.2976033.

[23] Y. K. Yee, M. Raheem, "Predicting Music Popularity Using Spotify and YouTube Features," Indian Journal of Science and Technology, vol. 14, no. 28, pp. 2347–2354, Jul. 2021.

[24] A. Charchyan, "Exploring Trends in Music Platforms: A Comparative Analysis of Key Factors for Trending Songs on Spotify and YouTube," B.S. thesis, Dept. of Data Science, American Univ. of Armenia, Yerevan, Armenia, 2024.

Appendix

The hyperparameters considered for each model are:

1) Random Forest:

- **n_estimators**: 300, 400, 500.
- **max_samples**: 0.5, 0.75.
- **max_features**: 'sqrt', 'log2'.
- **max_depth**: 30, 40, 50.
- **min_samples_split**: 5, 10.
- **min_samples_leaf**: 2, 4.

2) MLP:

- **hidden_layerz_sizes**: (100, 100), (50, 50, 50), (100, 50, 50), (100, 100, 50), (100, 100, 100).
- **activation**: 'relu', 'tanh'.
- **solver**: 'adam', 'sgd'.
- **learning_rate**: 'constant', 'adaptive'.
- **alpha**: 0.0001, 0.001.

3) XGBoost:

- **n_estimators**: 500, 600, 700, 800, 900, 1000.
- **learning_rate**: 0.001, 0.01.
- **gamma**: 0.1, 0.2, 0.3.
- **min_child_weight**: 1, 2, 3.
- **subsample**: 0.8, 1.0.
- **max_depth**: 3, 5, 7, 9.

4) FT-Transformer:

- **n_blocks**: [1, 4].
- **d_token**: [64, 512] (step=64).
- **attention_n_heads**: 8, 16, 32, 64.
- **d_ffn_factor**: [1.0, 4.0].
- **attention_dropout**: [0.0, 0.35] (step=0.05).
- **ffn_dropout**: [0.0, 0.5] (step=0.05).
- **residual_dropout**: [0.0, 0.2].
- **learning_rate**: [1e-5, 1e-3].
- **weight_decay**: [1e-6, 1e-3].
- **batch_size**: 128, 256, 512.
- **n_epochs**: , [1000, 10000] (step=200).