Musical Alchemy

Navigating Music Recommendations through Machine Learning

Kaily Mejia
Lally School of Management
Rensselaer Polytechnic Institute
Troy, NY United States
mejiak@rpi.edu

Executive Summary

For as long as there have been humans there has been music. Music is an indiscriminate part of human identity and culture. As human technologies have advanced so has the way humans interact and enjoy music. Instruments have become more sophisticated, music recording has become higher quality, and music has become more accessible than ever with the advent of the internet. More recently, the music industry has seen major changes to its business model due to digital sale and music streaming. Now for a monthly fee, someone can listen to as many songs as they want without ever having to own the song. Streaming is now the music industry's biggest revenue source. Streaming brings in so much money that streaming services are competing to create a better listening experience for users. The biggest attempt in improving user experience has been recommendations. At the moment, algorithms struggle to recommend songs without any historical data. The KKBox Music Recommendation Challenge Competition on Kaggle uses historical user data to determine if a user will listen to a song more than once within a month of listening to the song the first time. By using both user and historical data to build models that can accurately predict a user's listening activity these models can later be applied to recommending songs that the user is more likely to enjoy.

The data given by the competition consists of 6 different datasets: train, test, sample submission, songs, members, and extra song info. Below are the data dictionaries for each dataset:

Figure 1: sample_submission.csv Data Dictionary

Feature Name	Feature Description	Feature Type
id	Row ID	Integer
target	target variable	Integer (Binary)

Figure 2: test.csv Data Dictionary

Feature Name	Feature Description	Feature Type
msno	User ID	String
song_id	Song ID	String
source_system_tab	Tab where the event was triggered (Search, Library, etc)	String
source_screen_name	Layout a user sees	String
source_type	Entry point a user first plays music on mobile apps (Album, Playlist, etc)	String
target	target variable	Integer (Binary)

Figure 3: song_extra_info.csv Data Dictionary

Feature Name	Feature Description	Feature Type
song_id	Song ID	string
Song name	Song's Name	string
isrc	International Standard Recording Code	string

Figure 4: test.csv Data Dictionary

Feature Name	Feature Description	Feature Type
msno	User ID	String
song_id	Song ID	String
source_system_tab	Tab where the event was triggered (Search, Library, etc)	String
source_screen_name	Layout a user sees	String
source_type	Entry point a user first plays music on mobile apps (Album, Playlist, etc)	String
id	Row ID	Integer

Figure 5: songs.csv Data Dictionary

Feature Name	Feature Description	Feature Type
song_id	Song ID	string
song_length	Song length in milliseconds	integer
genre_ids	Song Genre (could be multiple)	string
artist_name	Name of Artist	string
composer	Name of Composer	string
lyricist	Name of Lyricist	string
language	Language identified by number	integer

Figure 6: members.csv Data Dictionary

Feature Name	Feature Description	Feature Type
msno	User ID	String
city	User's City	integer
bd	User's age	integer
gender	User's gender	string
registered_via	Registration Method	integer
registration_init_time	Registration date (%y%m%d)	integer
expiration_date	Expiration date (%y%m%d)	integer

Benchmarking of Other Solutions

There were many solutions to choose from as this was a public competition. The following are the three solutions for the competition chosen for an in-depth analysis:

Figure 7: Solution One Table

	Solution 1
Score	0.74787 (1st Place)
Feature Summarization	• Merged member table with training set on msno • Merged song table with the training table on song ID • Created a new feature column to have first genre ID and second genre ID • Created a feature for every option of the features in the member table and song table • There were then song components and member components that affected the model
Modeling Approach	This solution used embedding layers, dense layers, batch normalization layers, and dropout layers.

Figure 8: Solution Two Table

	Solution 2
Score	0.73015 (6th Place)
Feature Summarization	• Merged member table with training set on msno • Merged song table with the training table on song ID • Changed the expiration date and registration date to be stored as an actual date instead of an integer • Created a matrix with user listening history and how often they listen to a song • Focused on user listening data over more than the other features
Modeling Approach	This solution used multiple LightGBM models, matrix factorization, regression, and feature engineering.

Figure 9: Solution Three Table

	Solution 3
Score	N/A (Not a competition submission)
Feature Summarization	• Merged member table with training set on msno • Merged song table with the training table on song ID • Used expiration date and registration date to create a column for a day month and year for both dates • Dropped the original registration and an expiration date given • Used ISRC to create a feature for song year
Modeling Approach	This solution used the LightGBM classifier and XGBoost classifier.

The foundational step common to all three solutions involved merging the training set data with member and song data. However, their subsequent approaches diverged significantly.

In the first solution, which emerged as the competition's best, a neural network was defined with embedded layers for categorical features, interactions between user and song components, and various dense layers for prediction. The expansion of features in this solution focused on member-song interactions, introducing new features to provide contextual information for the neural network's learning process.

The second solution primarily employed multiple LightGBM models through ensemble averaging. These models added context to given features through a latent space representation of the member-user interactions matrix and collaborative filtering based on the top hundred neighborhood of member-song interactions, akin to KKbox's existing algorithm. The source feature was transformed into a categorical variable rate, and the solution attempted to replicate KKbox's algorithm further by using word embedding for song features such as artist, lyricist, composer, and genre. While the competition ranking difference between the first and second solutions was notable, the score difference was marginal.

The third solution, although not a competition entry, shared similarities with the models developed in this paper. It utilized both the LightGBM classifier and XGBoost classifier after expanding the feature set. Additional features included a year feature for the song release year, and the expiration date and sign-up date for the streaming service were transformed into date, month, and year features. This approach, focusing on classifiers, may result in lower accuracy compared to the first two solutions.

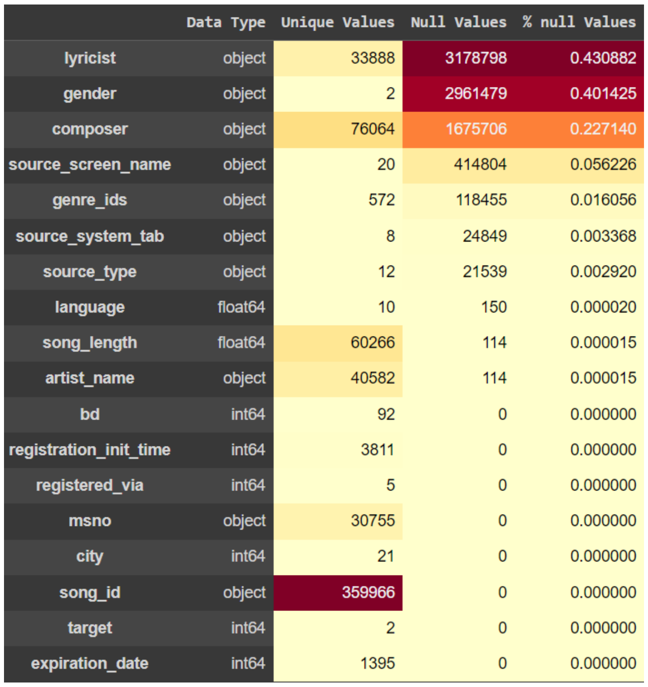
Data Description & Initial Processing

The initial observation highlights a significant absence of timestamps for instances of user interactions with the target song, as well as typical times for the user's listening activities. This absence may pose a potential drawback to model development. Furthermore, the song data file contains numerous songs not included in the user's listening history, necessitating a merger with the training data to focus solely on songs listened to by users in the training set. Similar considerations apply to the file with additional song data, which appears to provide limited utility in model development.

To construct the model, various files containing essential features were utilized. The member data was merged with the training data based on member ID, followed by the integration of relevant song data using song ID. With the training data now encompassing all necessary information, the data cleaning process commenced.

All feature columns underwent scrutiny for the presence and count of null values. A visualization using color gradients identified features with the highest null values and unique values. Lyricist, gender, and composers emerged as features with the most null values. These null values were addressed by filling string features with "unknown" and numeric features with their respective means. A matrix visualizing null values confirmed the successful correction.

Figure 10: Null Value Heatmap



	Data Type	Unique Values	Null Values	% null Values
lyricist	object	33888	3178798	0.430882
gender	object	2	2961479	0.401425
composer	object	76064	1675706	0.227140
source_screen_name	object	20	414804	0.056226
genre_ids	object	572	118455	0.016056
source_system_tab	object	8	24849	0.003368
source_type	object	12	21539	0.002920
language	float64	10	150	0.000020
song_length	float64	60266	114	0.000015
artist_name	object	40582	114	0.000015
bd	int64	92	0	0.000000
registration_init_time	int64	3811	0	0.000000
registered_via	int64	5	0	0.000000
msno	object	30755	0	0.000000
city	int64	21	0	0.000000
song_id	object	359966	0	0.000000
target	int64	2	0	0.000000
expiration_date	int64	1395	0	0.000000

Figure 11: Null Value Matrix (Before Filling)

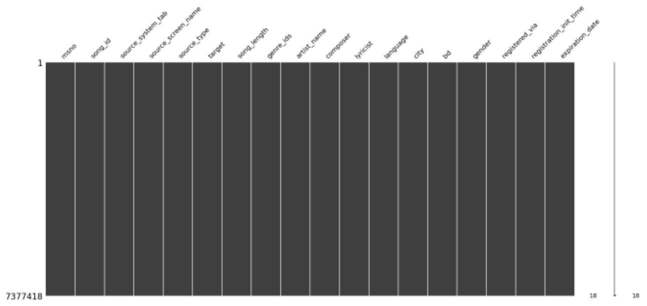
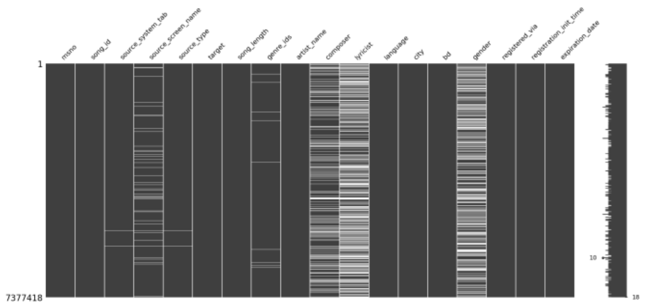


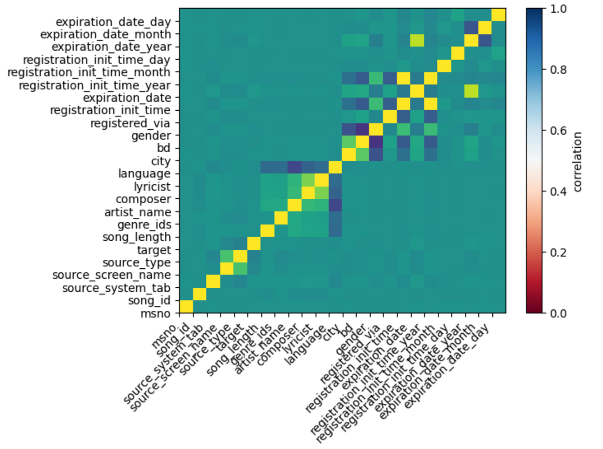
Figure 12: Null Value Matrix (After Filling)



In two of the three explored solutions, the format of expiration date and registration date features was reformatted to include separate columns for date, month, and year. This reformatting aimed to enhance the model's consideration of features and increase its predictive capabilities. Categorical features underwent encoding, employing label encoding for ordinal data and one-hot encoding for nominal data.

A correlation matrix revealed positive correlations between specific feature sections, such as consecutive song features and song-related features. Similarly, positive correlations were observed in the section containing registration and expiration information. Notably, correlated features originated from the same dataset, and intriguingly, no features exhibited negative correlations.

Figure 13: Feature Correlation Matrix



The final training dataset, with over seven million rows and 24 columns, included additional features containing song and member information. Key statistics indicated an average song length of 4.6 minutes, with male users representing 40.14%, female users at 31.14%, and 28.71% labeled as unknown. Due to the dataset's size, a 25% randomly sampled subset was used for the test-train split in model development on Google Colab.

Modeling

Three distinct models, namely Decision Tree, LightGBM, and Random Forest, were constructed using the training dataset. The training data was partitioned into test and train subsets to facilitate the development of these models. Subsequently, each model underwent evaluation utilizing key metrics, including accuracy, precision, recall, and F1-score. These metrics collectively provide a comprehensive assessment of the performance and predictive capabilities of each model.

1 Decision Tree Model

The initial model employed was a Decision Tree classifier, a supervised learning algorithm. This model utilizes a tree-like structure to represent decisions or actions, with each node corresponding to a decision based on input feature values. The tree undergoes splitting based on the most significant features at each node, aiming to create homogeneous subsets for enhanced predictability. While Decision Trees are proficient at capturing intricate relationships within datasets, there is a risk of overfitting.

Throughout the analysis, each split of the data consistently yielded similar precision and recall scores, as evident in the recall-precision graph. Notably, this classifier incorporated feature importance, leading to a bar graph that highlighted song length and song ID as the most influential features in this iteration of the Decision Tree.

Figure 14: Decision Tree Precision and Recall Graph

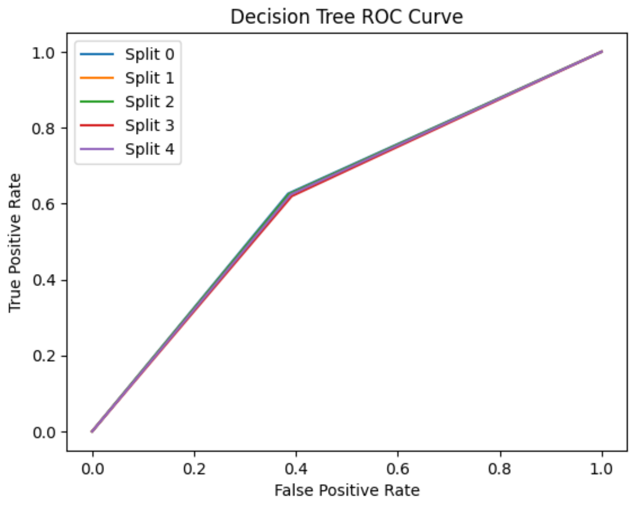


Figure 15: Decision Tree Feature Importance Bar Graph

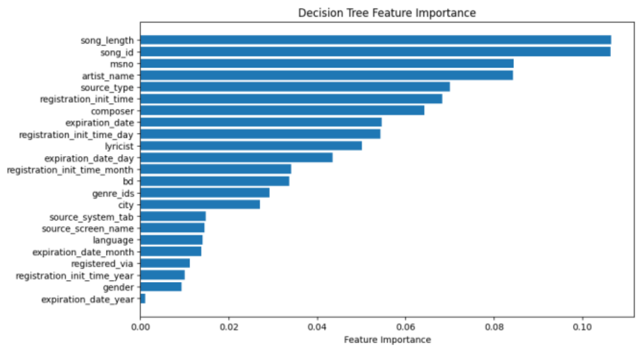


Figure 16: Decision Tree Evaluation Metrics Summary Table

Split	Accuracy	Precision	Recall	F1 Score
0	0.6188	0.6194	0.6202	0. 6188
1	0.6203	0.6207	0.6264	0.6241
2	0.6232	0.6235	0.6284	0.6264
3	0.6227	0.6230	0.6285	0.6261
4	0.6225	0.6246	0.6244	0.6238
Avg.	0.6215	0.6222	0.6256	0.6246

The Decision Tree classifier, on average, demonstrates moderate performance in predicting whether a user will listen to a song. The average accuracy of approximately 61.8% suggests that, on average, the model correctly predicts the target variable (listened to or not) for about 61.8% of instances in the test sets across the five splits. Additionally, models exhibit a balanced trade-off between precision and recall. The average precision of around 61.9% indicates that, on average, when the model predicts a song will be listened to, it is correct about 61.9% of the time. The average recall of approximately 62.4% means that, on average, the model captures about 62.4% of instances where the target variable is positive (i.e., songs that are listened to). The average F1 score of around 62.2% combines precision and recall into a single metric. Overall, the model does moderately, the moderate accuracy could be attributed to a few factors including the shortcomings of the decision tree classifier itself.

2 LightGBM Model

The LightGBM classifier represents a gradient boosting framework known for its efficiency and speed. Like the Decision Tree, it operates in a supervised learning paradigm, constructing an ensemble of weak learners to form a robust predictive model. LightGBM excels in handling large datasets and complex relationships, making it particularly suitable for this analysis. The model's unique approach to tree building prioritizes nodes that contribute the most to the overall loss, ensuring a strategic split based on feature importance.

In this analysis, as multiple precision and recall scores were calculated across diverse data splits, LightGBM exhibited consistency with minimal variations in the metrics. LightGBM highlighted key contributors that significantly shaped its decision-making process, expiration date, registration time, and song length emerged as the most influential features.

Figure 17: LightGBM Precision and Recall Graph

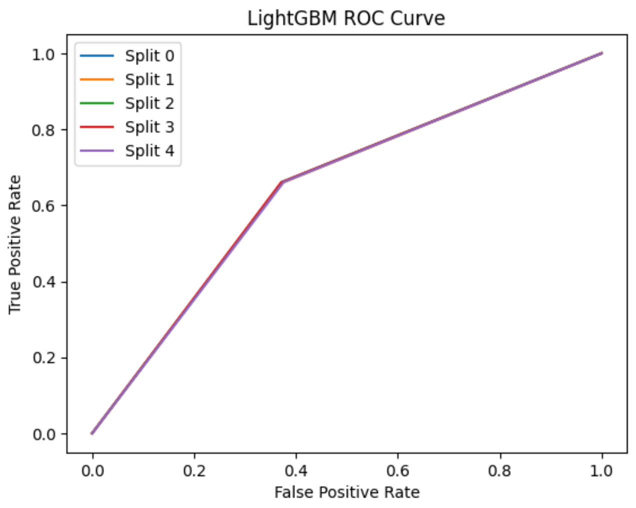


Figure 18: LightGBM Feature Importance Bar Graph

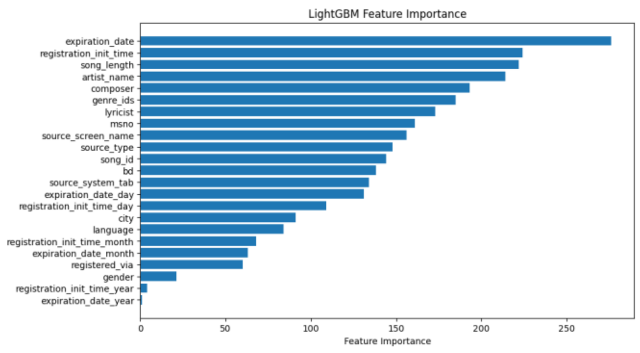


Figure 19: LightGBM Evaluation Metrics Summary Table

Split	Accuracy	Precision	Recall	F1 Score
0	0.6444	0.6414	0.6612	0.6412
1	0.6436	0.6427	0.6581	0.6503
2	0.6438	0.6418	0.6612	0.6514
3	0.6426	0.6430	0.6604	0.6516
4	0.6448	0.6403	0.6611	0.6505
Avg.	0.6438	0.6418	0.6604	0.6512

3 Random Forest Model

Similar to its counterparts, the Random Forest classifier, a great example in ensemble learning, operating on bagging principles, Random Forest combines multiple decision trees for enhanced accuracy and stability. As with the Decision Tree and LightGBM, this classifier underwent multiple evaluations through precision and recall assessments. Similarly, to the previous models, the metrics remained consistent across the various test-train splits. Delving into feature importance, akin to the Decision Tree, the most notable features were song length and song ID.

Figure 20: Random Forest Precision and Recall Graph

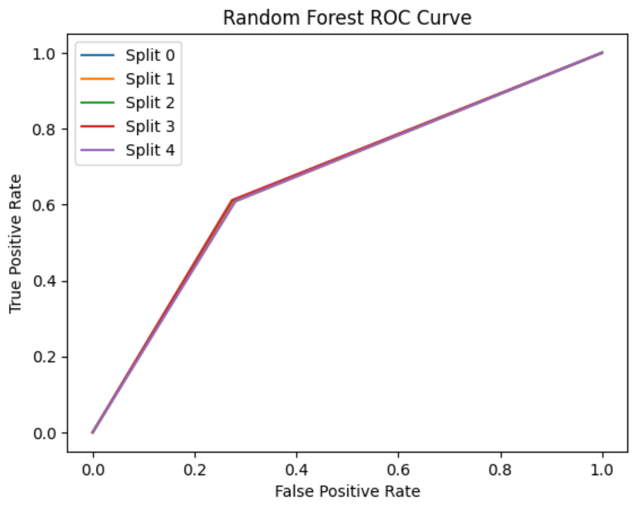


Figure 21: Random Forest Feature Importance Bar Graph

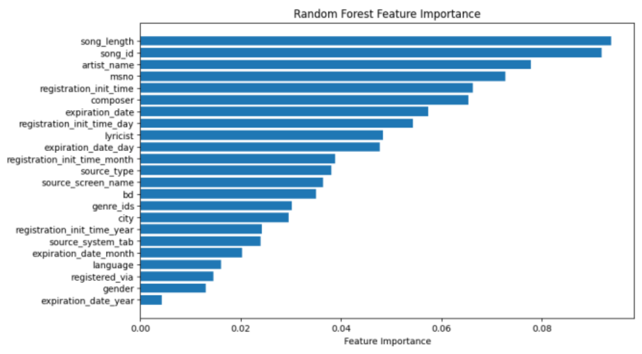


Figure 22: Random Forest Evaluation Metrics Summary Table

Split	Accuracy	Precision	Recall	F1 Score
0	0.6650	0.6891	0.6064	0.6451
1	0.6646	0.6897	0.6074	0.6459
2	0.6680	0.6933	0.6103	0.6492
3	0.6687	0.6934	0.6117	0.6500
4	0.6642	0.6873	0.6102	0.6465
Avg.	0.6661	0.6906	0.6092	0.6473

The Random Forest classifier, on average, performs similarly to the LightGBM classifier. The average accuracy of approximately 66.6% indicates that, on average, the Random Forest model correctly predicts the target variable for about 66.6% of instances in the test sets across the five splits. The average precision of around 69.1% means that, on average, when the Random Forest model predicts a song will be listened to, it is correct about 69.1% of the time. The average recall of approximately 60.96% suggests that, on average, the Random Forest model captures about 60.96% of instances where the target variable is positive. The average F1 score of around 64.73% combines precision and recall into a single metric. It balances the trade-off between precision and recall, providing a comprehensive measure of the Random Forest model's performance. While this model has the highest accuracy and precision when compared to the other model where it doesn't perform as well is with its recall and F1 score. The accuracy and precision were improved in this model at the expense of the recall.

Appendix

For the proposed solution, the chosen model depends on what metric is most important, for the highest accuracy and precision Random Forest is the best solution but if a balance between precision and recall is desired, the LightGBM model could be a good choice. The Decision Tree model shows slightly lower performance compared to the other two in this context. Depending on the goals of the company, for instance, if minimizing false positives is critical, the Random Forest model should be used, however, if capturing as many positive instances as possible is crucial, the LightGBM model could be favored.

The solution proposed by the Decision Tree model is effective in capturing the nonlinear relationship within data. It produces an easy to interpret tree that allows for a clear understanding of the decision-making process. However, the drawbacks of the using decision tree classifier come from the model being prone to overfitting and can be sensitive to outliers in the data. Additionally, the single decision tree produced by this classifier might not capture all the patterns as effectively as ensemble methods. The drawbacks for this classifier can probably be attributed to why this classifier performed the worst out of the three models developed.

The solution proposed by the LightGBM model is effective at capturing complex patterns and relationships in the data. However, gradient Boosting models, including LightGBM, can be complex and act as black boxes, making it challenging to interpret the underlying decision rules and performance may heavily depend on tuning hyperparameters, and improper tuning might lead to suboptimal results. Though generally, LightGBM tends to provide high predictive accuracy and efficiency, especially on large datasets.

The solution proposed by Random Forest model achieves high accuracy and robustness by aggregating predictions from multiple decision trees. Similar to LightGBM, Random Forests can be challenging to interpret due to its black-box nature. However, the ensemble nature of Random Forests helps reduce overfitting compared to a single Decision Tree.

The decision to use a particular model depends on the specific goals of your music recommendation system. Decision Trees offer interpretability but might lack predictive power compared to ensemble methods. LightGBM and Random Forest are powerful in capturing complex patterns but are more challenging to interpret. The trade-off between interpretability and predictive accuracy should be considered when it comes to deciding which models to apply.

REFERENCES

[1] KKBOX (no date) WSDM - KKBOX's Music Recommendation Challenge, Kaggle. Available at: https://www.kaggle.com/competitions/kkbox-music-recommendation-challenge/overview (Accessed: 07 December 2023).

[2] 1.10. decision trees (no date) scikit. Available at: https://scikit-learn.org/stable/modules/tree.html (Accessed: 07 December 2023).

[3] Lightgbm (Light Gradient Boosting Machine) (2023) GeeksforGeeks. Available at: https://www.geeksforgeeks.org/lightgbm-light-gradient-boosting-machine/ (Accessed: 07 December 2023).

[4] Sklearn.ensemble.randomforestclassifier (no date) scikit. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (Accessed: 07 December 2023)...