Hit Song Prediction Through Machine Learning and Spotify Data

Andrej Natev
89221050@student.upr.si
Faculty of Mathematics, Natural Sciences
and Information Technologies,
University of Primorska
Glagoljaška ulica 8
SI-6000 Koper, Slovenia

ABSTRACT

This study[2] aims to predict hit songs using metadata gathered from the Spotify API[8]. The extracted dataset comprises over 20 genres, each with 40 songs equally divided between hits and flops, sourced directly from the Spotify web API[8] using spotipy[7]. The prediction was based on the popularity feature, also from Spotify's API[8], that presents current popularity with an integer governing from 0-100. Models were trained on various features including danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, and tempo. The dataset was partitioned using three techniques: train_test_split (test set of 10%, 20%, and 33%), and standard and stratified kfold cross-validation with k values of 2, 5, and 10.

Models were trained, evaluated, and tested, with performance analyzed and saved from all three techniques. The kfold cross-validation techniques presented us the best accuracy, and also they have the least chance of overfitting. It utilized all of scikit-learn's classification models, ensemble models and MLPClassifier as a neural network. PassiveAgressiveClassifier showed a 60% accuracy as did AdaBoost, with Tree ensemble methods as a runner up together with the MLPClassifier using the logistic activation function. Noteworthy performances were observed from extra trees and random forest as ensemble models, and Gaussian Process/NaiveBayes and ridge classifiers stood out as more standard models.

These findings hint us that with further enhancing of models, not just using the default models from scikit-learn[3], we could predict hit songs using Spotify API[8] audio features. Particularly neural networks and decision tree ensembles, could be used to enhance predictive efficacy. Prospective research avenues, including frequency and lyric analysis, hold potential for uncovering the hit song formula.

KEYWORDS

music, genre, song, Spotify, machine learning, classification, ensemble model, support vector, neural networ, artificial intelligence

1 INTRODUCTION

This research delves into the intersection of music and data science, leveraging the Spotify Web API[8] in conjunction with the Spotipy library[7], and machine learning models. By harnessing these tools, the study[2] aims to extract and analyze track data across various genres. The primary objective is to find machine learning models capable of categorizing songs into two distinct groups: "hit" and "flop", based on a range of audio features. Popularity is a feature in Spotify's Web API[8], that represents a song's popularity the past three days from the day of extraction. It is an integer that ranges from 0-100, such that a flop is any song below 60 popularty and everything above is a hit song.

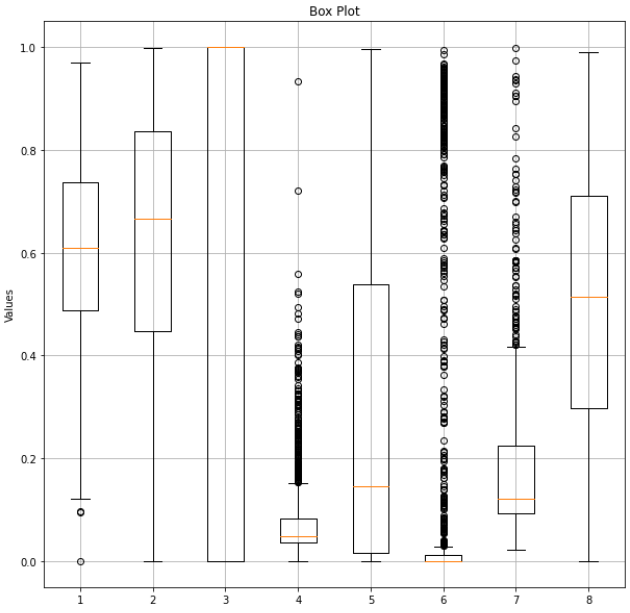

Figure 1: Box Plot for Feature Outliers

2 MATERIALS, MODELS, METHODS

2.1 Materials

In this study[2], firslt two datasets from Kaggle were utilized: "Most Streamed Songs 2023"[6] and "30000 Spotify Songs"[5]. These datasets provided a rich source of music metadata for analysis, with one of them being a training dataset and the other a evaluating dataset. Later, both of them were discarded because of the chance of overfitting. So then Spotipy[7], a Python library, was used for data extraction from the Spotify Web API[8]. Pandas and NumPy were employed for data manipulation, while the Scikit-learn[3] library facilitated feature engineering, preprocessing, data splitting, model implementation, and evaluation. And also MatPlotLib was used to be able to visually analyze the features and to present the results.

2.2 Most Accurate Models

2.2.1 MLPClassifier with ReLU and Logistic Activation Functions. The MLPClassifier is a neural network model with multiple layers

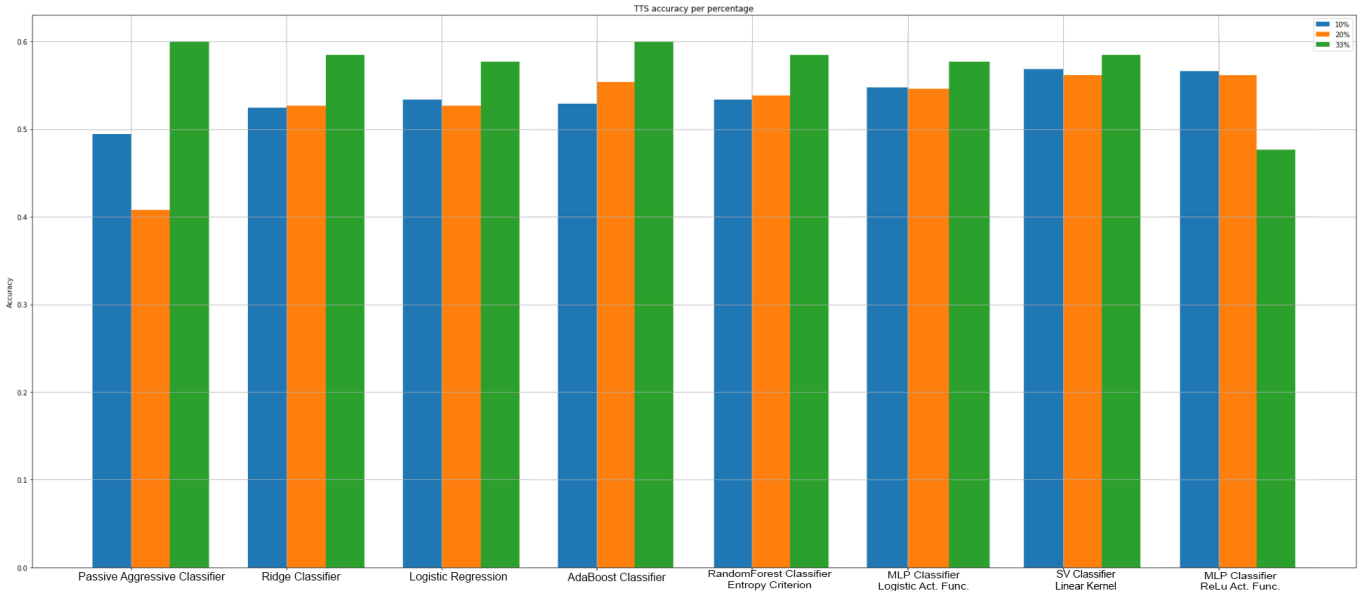

Figure 2: Bar Graph for TTS Accuracy Percentages

of interconnected nodes. It uses the ReLU (Rectified Linear Unit) activation function, which outputs the input directly if positive or zero otherwise, helping to avoid the vanishing gradient problem. The logistic (sigmoid) activation function, which maps the input into a range between 0 and 1, is particularly useful for binary classification tasks. These activation functions enable the MLPClassifier to capture complex data patterns, improving prediction accuracy.[3]

2.2.2 ExtraTreesClassifier with Gini and Entropy Criteria. The ExtraTreesClassifier is an ensemble method that builds many decision trees using randomized splits of the training data. It uses Gini impurity or entropy as criteria to evaluate the quality of splits within the trees. Gini measures the likelihood of misclassification, while entropy measures uncertainty. By averaging predictions from multiple trees, ExtraTreesClassifier reduces overfitting and improves generalization, making it a robust choice for classification tasks.[3]

2.2.3 GradientBoostingClassifier. GradientBoostingClassifier incrementally builds a strong classifier by sequentially adding weak learners, typically decision trees. Each new model is trained on the residual errors of the previous models, allowing the ensemble to focus on earlier mistakes. This process iteratively reduces error, enhancing accuracy and robustness. GradientBoostingClassifier is particularly effective in complex prediction tasks requiring high precision.[3]

2.2.4 PassiveAggressiveClassifier. PassiveAggressiveClassifier is a linear model suited for large-scale learning tasks, especially in online learning. It updates parameters only when misclassification occurs (passive) and aggressively corrects errors when they do. This combination allows the model to adapt quickly, making it efficient for real-time classification tasks where speed is crucial.[3]

2.2.5 AdaBoostClassifier. AdaBoostClassifier is an ensemble method that builds a strong classifier by combining multiple weak classifiers. It adjusts the weights of misclassified instances in each iteration, focusing subsequent classifiers on difficult cases. By concentrating on previous errors, AdaBoostClassifier progressively improves accuracy, making it a powerful tool for various classification tasks.[3]

2.2.6 RidgeClassifier. RidgeClassifier is a linear model that uses L2 regularization to prevent overfitting by penalizing the magnitude of coefficients. This regularization is particularly useful in high-dimensional spaces where features outnumber observations. RidgeClassifier balances bias and variance, making it effective for classification tasks requiring generalization to unseen data.[3]

2.2.7 LogisticRegression. LogisticRegression is a linear model for binary classification tasks. It models the probability of class membership by applying the logistic function to a linear combination of input features. Trained by maximizing the likelihood of observed data, LogisticRegression effectively handles cases where the feature-target relationship is approximately linear, making it widely used for various classification scenarios.[3]

2.2.8 SVC (Linear Kernel). SVC with a linear kernel is a supervised learning model that constructs a hyperplane in a high-dimensional space to separate classes. The linear kernel computes the dot product of feature vectors, making it effective for linearly separable data. By maximizing the margin between classes, SVC with a linear kernel provides reliable and interpretable classification results.[3]

2.3 Methods

2.3.1 Data Understanding. The first step in our methodology involved understanding the datasets and the task at hand. Initially, songs were obtained from the Spotify Web API[8] based on their popularity ratings, ensuring a balanced representation of hits and flops. The popularity feature served as a crucial criterion for categorizing songs as hits or flops, with hits defined as songs with

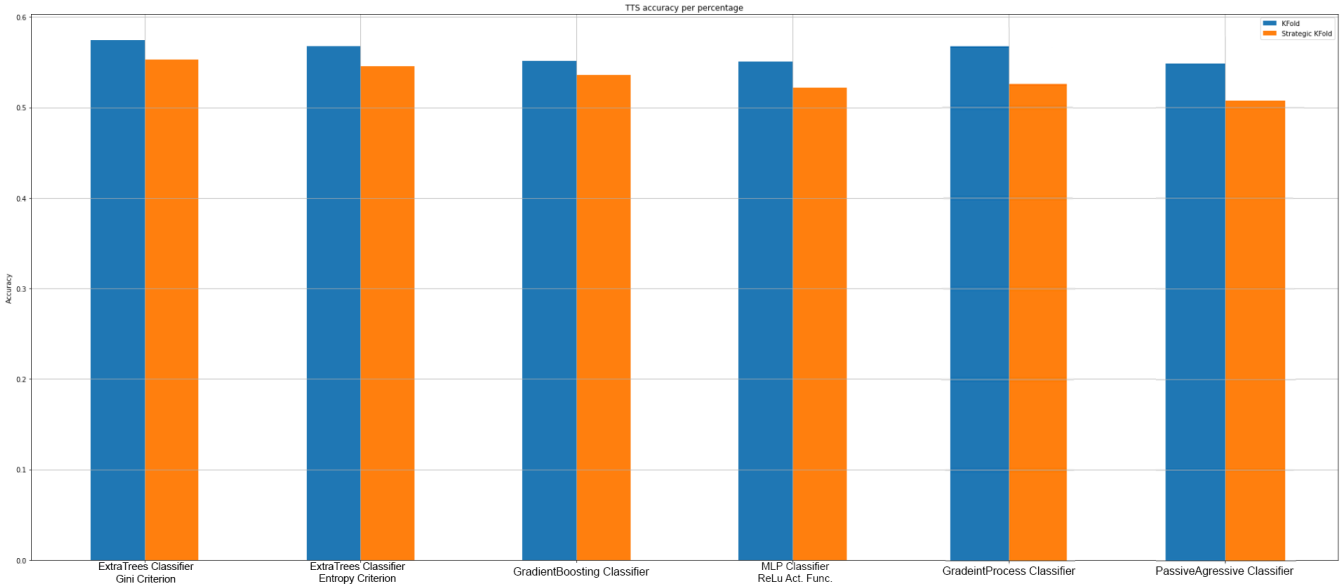

Figure 3: Bar Graph for CV Accuracy Percentages

a popularity score of 60 or above, and non-hits as songs with a popularity score below 60.

2.3.2 Data Extraction. Data extraction was conducted using the Spotipy library along with the Spotipy-random add-on. The extraction process involved sourcing track data directly from the Spotify Web API[8]. In addition to leveraging Spotipy-random, genres were carefully selected to ensure diversity and fairness in the broad range of the features' values. These genres ranged from unpopular to popular and encompassed different audio features to ensure distributivity across the dataset.
During the extraction process, it was observed that songs with popularity ratings above 85 and below 60 were particularly challenging to find, even when considering genres that were both unpopular and popular. This highlights the inherent difficulty in obtaining a balanced dataset, especially when targeting specific popularity ranges.

2.3.3 Data Preparation. Following data acquisition, the dataset was prepared for analysis by incorporating audio features obtained from the Spotify API[8]. These features included danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, and tempo. Additionally, data types were adjusted to ensure categorical representation for key, mode, time signature, and hit categories. After which another dataset was made that contained the same features but just scaled using the default StandardScaler from scikit-learn[3].

2.3.4 Model Training and Evaluation. During the model training and evaluation phase, it was observed that KFold and StratifiedKFold cross-validation techniques[3] encountered difficulties when handling larger splits due to the relatively small size of the dataset. With only 1299 songs available and an almost 50/50 balance between hits and non-hits, the dataset size posed challenges for these cross-validation methods to effectively cover all instances Despite these challenges, various machine learning models, including ensemble models and standard classifiers, were trained on the dataset. The performance of each model was evaluated using Train-Test-Split[3] (TTS) with varying test sizes (10%, 20%, and 33%), as well as KFold and StratifiedKFold cross-validation techniques with different fold splits (2, 3, 5, and 10).

3 RESULTS

The study[2] aimed to predict hit songs using machine learning algorithms trained on Spotify API metadata. Results revealed varying accuracies across different models and evaluation techniques. Notably, the AdaBoost Classifier and Passive Agressive Classifier achieved the highest accuracy of 60% on a test size of 33%, followed by the RandomForest (entropy criterion), and Ridge Classifiers, and Logistic Regression that demonstrated stable performance on the same test size. The MLP (logistic activation function) and the SupportVector Classifier demonstrated the highest nad the most constant through out all test sizes, 10%, 20% and 30%.
Regarding the cross-validation techniques, the stratified kfold had a constant lower accuracy throughout all models and throughout all kfolds, 2, 3, 5, and 10. ExtraTrees Classifier had the highest accuracy with both techniques and using both the gini criterion and the entropy criterion, with an an almost 60%. It was followed by the GradientProcess Classifier that had a percent higher accuracy than similarly named the GradientBoosting Classifier. Other notable mentions using the cross-validation techniques are the Passive Agressive Classifier, and the MLP Classifier with the ReLu activation function with a very similar accuracy.

4 DISCUSSION

The findings of this study[2] shed light on the possibility of machine learning algorithms to be used to predict hit songs based on Spotify metadata[8]. While some models exhibited promising

Table 1: Model Accuracy Comparison

Model	Accuracy (%)	
	Train-Test-Split	k-fold CV
RandomForestClassifier (Entropy)	57.69 / 54.62 / 52.68	56.12
AdaBoostClassifier	60.00 / 55.38 / 52.91	53.58
ExtraTreesClassifier	48.46 / 51.54 / 55.24	56.81
ExtraTreesClassifier (Entropy)	54.62 / 55.38 / 54.31	57.81
MLPClassifier (Identity)	50.00 / 58.46 / 53.85	55.66
MLPClassifier (Logistic)	58.46 / 58.46 / 54.78	55.50

accuracy rates for a really general approach to the problem, deviations from expected outcomes were observed, prompting deeper analysis. The AdaBoost Classifier achieved the highest accuracy, but only in the train-test-split. Additionally, the MLPClassifier with identity and logistic activation functions showed accuracy, suggesting the potential of neural network architectures in capturing nonlinear relationships within the data. Theoretical implications suggest the need for further investigation into ensemble models and neural networks, and hyperparameter tuning to optimize model performance.

4.1 Previous Research

Compared to prior research endeavors, which often grappled with issues of data imbalance and feature scaling, this study's results represent a significant improvement. The utilization of a more balanced dataset, coupled with standardized feature scales, has led to more reliable and interpretable models. The transition from overfitted models, which yielded inflated accuracy rates, to robust and generalizable models underscores the importance of methodological rigor in data science research.[1]

5 CONCLUSION

In summary, this study[2] investigated the application of machine learning algorithms for hit song prediction using Spotify metadata[8]. Through rigorous experimentation and evaluation, we have demonstrated the potential of various classifiers and ensemble methods in categorizing songs into hits and non-hits with reasonable accuracy for a general approach. The findings contribute to the existing body of research by providing insights into the performance characteristics of different models and the impact of algorithmic parameters on predictive outcomes.

Despite achieving competitive accuracy rates, the study[2] also revealed nuances and deviations from expected results, making an even bigger need for further investigation.

Moving forward, it is imperative to address open questions surrounding the generalizability of models across diverse music genres, the robustness of predictions over time, and the incorporation of additional features such as lyrics and user-specific preferences. Moreover, future research should focus on refining model architectures, exploring ensemble models and neural networks, and optimizing hyperparameters to enhance predictive efficacy.

ACKNOWLEDGMENTS

Special recognition is also extended to the Google Developer Student Club of the University of Primorska for organizing multiple events focused on ML&AI. It was during these events that the seeds of the initial research, which had overfitting issues, were sown.[4]

REFERENCES

[1] Andrej Natev. 2023. Initial Notebook. https://www.kaggle.com/code/andrejnatev/hit-song-prediction
[2] Andrej Natev. 2024. Main Notebook. https://www.kaggle.com/code/andrejnatev/spotify-api-spotipy-hit-song-prediction.
[3] David Cournapeau. 2007. Scikit-learn Documentation. https://scikit-learn.org/stable/index.html
[4] GoogleDSC University of Primorska. 2023. ML&AI Summit. https://gdsc.community.dev/university-of-primorska-koper-slovenia/
[5] Joakim Arvidsson. 2023. 30000 Spotify Songs dataset. https://www.kaggle.com/datasets/joebeachcapital/30000-spotify-songs
[6] Nidula Elgiriyrwithana. 2023. Most Streamed Songs 2023 dataset. https://www.kaggle.com/datasets/nelgiriyewithana/top-spotify-songs-2023
[7] Paul Lamere. 2014. Spotipy Documentation. https://spotipy.readthedocs.io/en/2.24.0/
[8] Spotify. 2013. Spotify Web API Documentation. https://developer.spotify.com/documentation/web-api