Spotify Track Popularity Prediction

Prerna Chander & *Nidhi Mankala*

Abstract

There has been a rise in popularity of music streaming platforms in recent years with Spotify being the most popular, holding 31% of the total music streaming market in 2021. In today's increasingly data-driven world, if Spotify can quantify music using this data, they can generate more revenue for themselves and provide a better experience to their users. This project aims to compare ensemble learning algorithms (XGBoost, Categorical Boosting, Light Gradient Boosting, AdaBoost, Random Forest) and observe if improved predictions are being made on the data in order to see which one most effectively predicts the popularity of a song and what auditory features contribute to this popularity. The highest accuracies are given by the AdaBoost and Random Forest classifiers, with 0.9111 and 0.9113 respectively.

I. Introduction

Music analytics is a highly up-and-coming field - decisions made in the music industry are increasingly becoming about innovative insights and discovering new trends and artists. It allows streaming services to better understand their users and their listening habits. By analyzing data about what users are listening to, how often they listen, and how they interact with the service, they can identify trends and patterns that can help them to improve their algorithms, make more personalized recommendations, and develop new features. In specific, popular track prediction also has interesting applications in music analytics. Identifying acoustic properties that popular music is composed of can be used towards parameterizing audio for artists to create "popular" sounds/music.

It presents multiple benefits to Spotify as well. Firstly, it can help the service to make personalized recommendations to its users. By predicting which songs are likely to be popular, the service can suggest music that a user is more likely to enjoy, which can improve their experience and keep them engaged with the service. Additionally, accurate popular music prediction can also help Spotify to identify and target potential new subscribers, by suggesting songs that are similar to ones that they already enjoy. This can help to increase the number of users on the platform, which can in turn help to increase the service's revenue. Finally, popular music prediction can also help Spotify to identify trends in user listening habits, which can be useful for a variety of purposes, such as developing new features or improving the service's algorithms.

To achieve these goals, the project aims to make use of ensemble machine learning classifiers. These are considered to be better than individual classifiers because they can provide more accurate predictions. This is because ensemble classifiers combine the predictions of multiple individual classifiers, which can help to reduce errors and improve overall performance. For example, if several individual classifiers each make slightly different predictions for a particular data point, the ensemble classifier can combine these predictions to make a more accurate overall prediction. Additionally, ensemble classifiers can also help to reduce overfitting, which is when a classifier performs well on the training data but poorly on new,

unseen data. By combining the predictions of multiple individual classifiers, an ensemble classifier can smooth out the predictions and improve generalizability. This project focuses on using and comparing the performance of five ensemble classifiers, XGBoost, Categorical Boosting, Light Gradient Boosting, AdaBoost, and Random Forest.

II. Preliminaries

A. Dataset Description

The Spotify Tracks DB was used for this project. It consists of 232726 rows x 18 columns, with data relating to music tracks and their audio properties [1]. The dataset head is shown below [Figure 1]. The summary statistics of the dataset is also shown below, it is helpful in understanding the distribution of the features of the dataset [Figure 2].

	genre	artist_name	track_name	track_id	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	time_signature	valence
0	Movie	Henri Salvador	C'est beau de faire un Show	0BRjO6ga9RKCKjfDqeFgWV	0	0.611	0.389	99373	0.910	0.000	C#	0.3460	-1.828	Major	0.0525	166.969	4/4	0.814
1	Movie	Martin & les fées	Perdu d'avance (par Gad Elmaleh)	0BjC1NfoEOOusryehmNudP	1	0.246	0.590	137373	0.737	0.000	F#	0.1510	-5.559	Minor	0.0868	174.003	4/4	0.816
2	Movie	Joseph Williams	Don't Let Me Be Lonely Tonight	0CoSDzoNIKCRs124s9uTVy	3	0.952	0.663	170267	0.131	0.000	C	0.1030	-13.879	Minor	0.0362	99.488	5/4	0.368
3	Movie	Henri Salvador	Dis-moi Monsieur Gordon Cooper	0Gc6TVm52BwZD07Ki6tIvf	0	0.703	0.240	152427	0.326	0.000	C#	0.0985	-12.178	Major	0.0395	171.758	4/4	0.227
4	Movie	Fabien Nataf	Ouverture	0IusIXpMROHdEPvSl1fTQK	4	0.950	0.331	82625	0.225	0.123	F	0.2020	-21.150	Major	0.0456	140.576	4/4	0.390

Figure 1: Dataset Head

	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence
count	232725.000000	232725.000000	232725.000000	2.327250e+05	232725.000000	232725.000000	232725.000000	232725.000000	232725.000000	232725.000000	232725.000000
mean	41.127502	0.368560	0.554364	2.351223e+05	0.570958	0.148301	0.215009	-9.569885	0.120765	117.666585	0.454917
std	18.189948	0.354768	0.185608	1.189359e+05	0.263456	0.302768	0.198273	5.998204	0.185518	30.898907	0.260065
min	0.000000	0.000000	0.056900	1.538700e+04	0.000020	0.000000	0.009670	-52.457000	0.022200	30.379000	0.000000
25%	29.000000	0.037600	0.435000	1.828570e+05	0.385000	0.000000	0.097400	-11.771000	0.036700	92.959000	0.237000
50%	43.000000	0.232000	0.571000	2.204270e+05	0.605000	0.000044	0.128000	-7.762000	0.050100	115.778000	0.444000
75%	55.000000	0.722000	0.692000	2.657680e+05	0.787000	0.035800	0.264000	-5.501000	0.105000	139.054000	0.660000
max	100.000000	0.996000	0.989000	5.552917e+06	0.999000	0.999000	1.000000	3.744000	0.967000	242.903000	1.000000

Figure 2: Dataset Statistics

The different features of the dataset and their descriptions are as follows [2]:

1. ***Acousticness***: A measure from 0.0 to 1.0 of whether the track is acoustic.
2. ***Danceability***: Describes from a range of 0.0 to 1.0 if a track is danceable based on musical elements.
3. ***Duration_ms***: Track duration in milliseconds.
4. ***Energy***: Describes how energetic a track is from 0.0 to 1.0.
5. ***Track ID***: Spotify track ID.
6. ***Instrumentalness***: Describes how instrumental (song with no vocals) a track is from 0.0 to 1.0.
7. ***Key***: Key of the track (*eg:* C, D#)
8. ***Liveness***: A measure from 0.0 to 1.0 of if the track was performed live.
9. ***Loudness***: Loudness of a song in decibels.

10. ***Mode***: Modality of a song (*eg:* major, minor)
11. ***Speechiness***: A measure in the range 0.0 to 1.0 of if the track has spoken words.
12. ***Tempo***: The tempo of a song in beats per minute.
13. ***Time Signature***: Measure of how many beats are in a bar.
14. ***Genre***: Genre of song (*eg:* R&B, Pop)
15. ***Valence***: A measure from 0.0 to 1.0 conveying the "positivity" of a song.
16. ***Artist Name***: Name of the artist of the track.
17. ***Track Name***: Name of the track.
18. ***Popularity***: A measure from 0 to 100 of the popularity of the song, with 100 being the most popular. Popularity is calculated by the Spotify algorithm as the total number of plays the track has had and how recent those plays are. This is our target variable.

B. Exploratory Data Analysis

Exploratory data analysis was performed to summarize the main characteristics of the data, with visual methods in order to discover new insights and identify trends that can be further investigated. It is a key step in understanding the Spotify data to identify any unusual or unexpected patterns in the data.
First, our target variable, *Popularity*, was visualized [Figure 3]. This helps us understand the distribution of the feature in the dataset - how many samples of each we have.

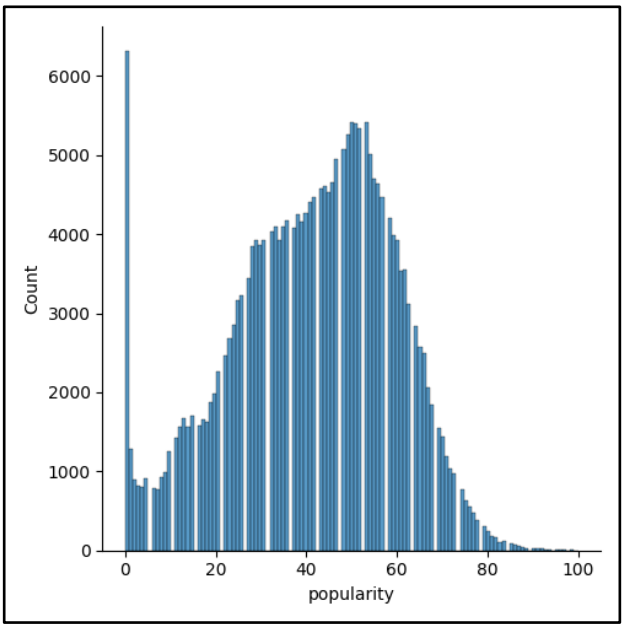

Figure 3: Popularity Distribution

From the figure, we can see that the distribution of *Popularity* is not even. The dataset consists of too many '0' popularity values, and the rest of the values are not dispersed. There are very few samples of actual "popular" songs above a score of '60'. This can cause an imbalance issue while training the model.

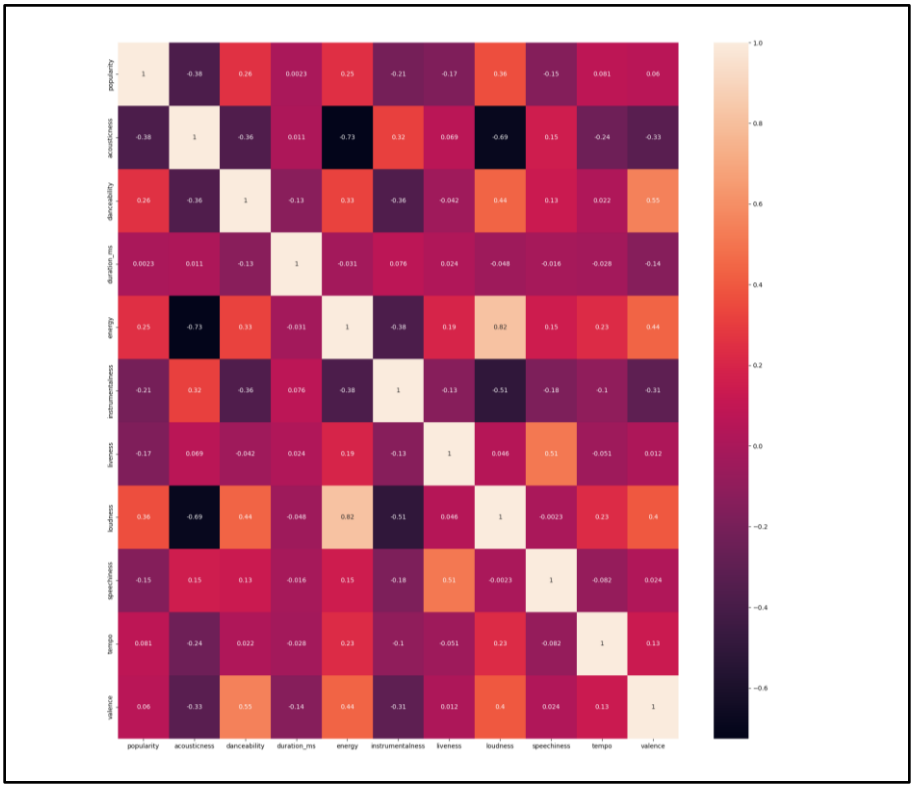

Figure 4: Correlation Heatmap

Next, a correlation matrix was created to show the correlation coefficients of the different features of the dataset in order to visualize strong and weak ones. As seen in [Figure 4], we can identify some relationships between the features. For example, *Loudness* has the highest positive correlation with *Popularity*, and *Acousticness* has the highest negative correlation with *Popularity*. *Energy* and *Acousticness* have a highly-correlated inverse relationship. The more acoustic a song is, the less energy it tends to be.

Making note of some of the important features in the correlation matrix, a few more plots were made to visualize the relationships between features.

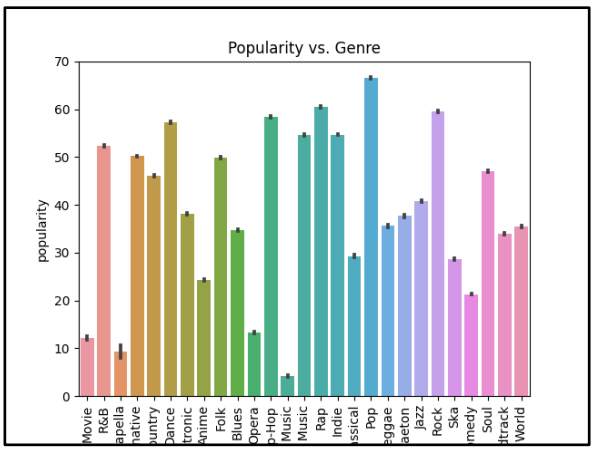

Figure 5: Popularity vs. Genre

From [Figure 5] we can infer that the *Genre* that contributes most to *Popularity* is Pop, followed by Rap, Rock and Hip-Hop. As seen in [Figure 6], C# is the most popular key, while G# Minor is the most popular key mode combination [Figure 7].

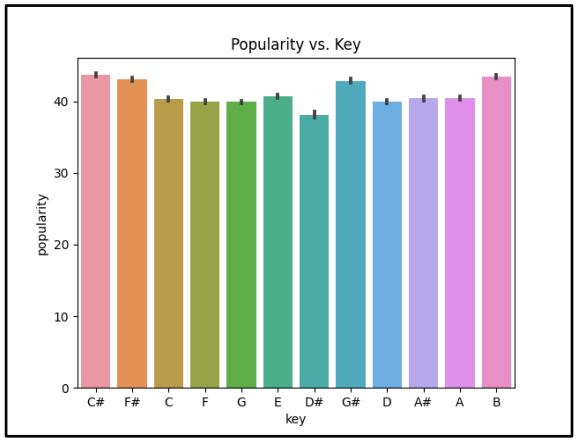

Figure 6: Popularity vs. Key

As seen in [Figure 8], we understand that the most popular mode is 'minor' and so is 4/4 time signature [Figure 9].

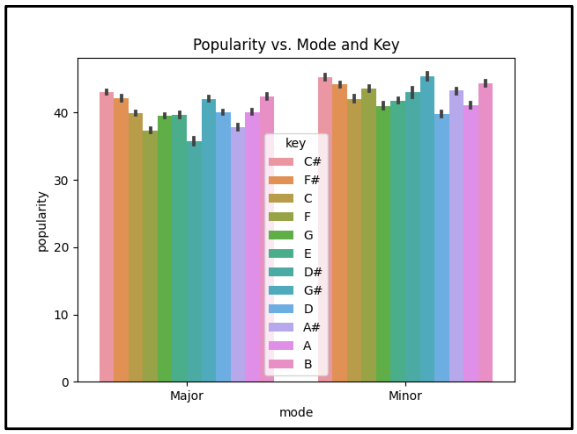

Figure 7: Popularity vs. Mode and Key

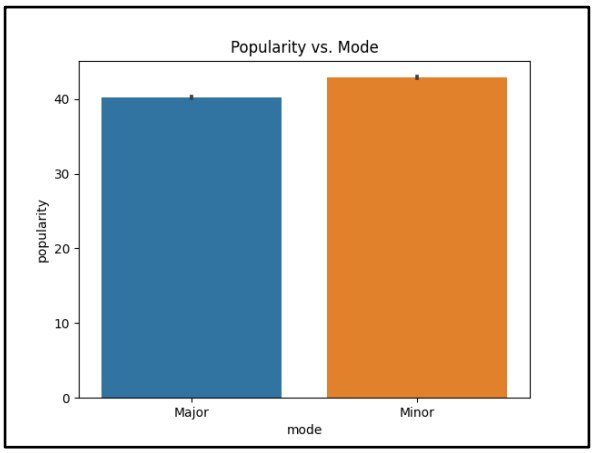

Figure 8: Popularity vs. Mode

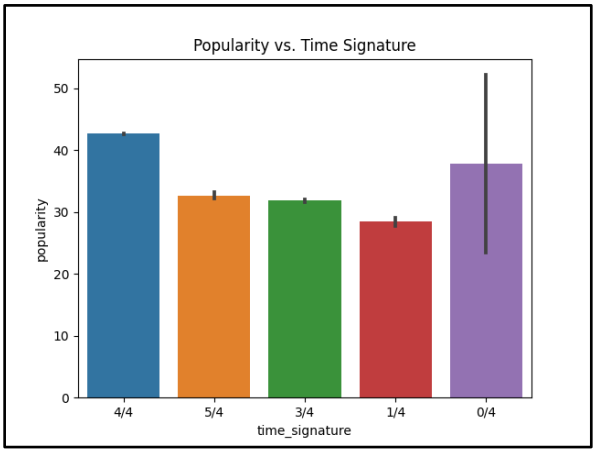

Figure 9: Popularity vs. Time Signature

C. Data Tidying and Transformation

For the data cleaning, we drop NAs and the unneeded columns of *Track Name*, *Track ID*, *Artist Name*. We also drop all duplicate data as there exists multiple copies of the same track by the same artist. From [Figure 5] we observe that the 'Children's Music' genre appears twice, this is because of a data quality issue where the apostrophe appears as ' and '. This was cleaned as well.

Class Label	Count of Samples
1	107246
2	76646
0	45877
3	2956

Table 1: Count of Samples in Each Class

As observed in the Exploratory Data Analysis [Figure 3], the distribution of *Popularity* is uneven. In order to combat this imbalance of data, we first group *Popularity* into four classes for the ranges of 0-25, 25-50, 50-75, 75-100, turning this into a classification problem. We then observe the counts of each of the four classes [Table 1]. This also clearly shows that data is unevenly spread out. Next, we oversample the data to have an equal number of data in all four classes. This changes the counts of all classes to be the same, however this introduces the possibility of model overfitting.

One hot encoding is also performed on the categorical features - *Genre*, *Time Signature*, *Key*, and *Mode*. One hot encoding is a method used to encode categorical variables as binary vectors. A categorical value can be represented as a binary vector with each element in the vector as 0 except for the index corresponding to the categorical value, which is 1.

After performing these steps, the correlation between the features of the dataset and *Popularity* was visualized once again [Figure 10]. As can be observed *Loudness* and Hip Hop *Genre* have the highest positive correlation with Popularity, while *Acousticness* and *Opera* Genre have the most negative correlation.

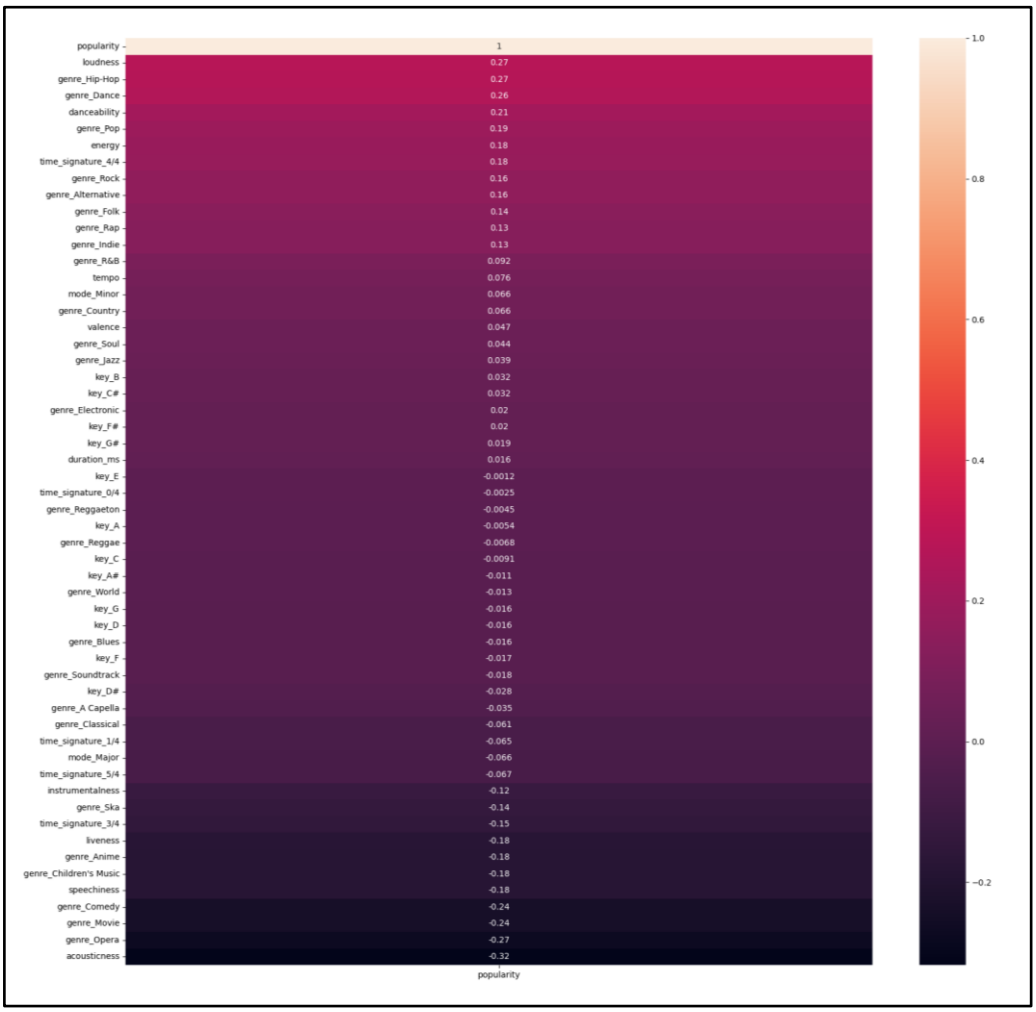

Figure 10: Correlation Heatmap with Popularity

D. Feature Engineering

As the features increased greatly with the introduction of one hot encoding, there needed to be a way to choose only the most important features useful for modeling. For this Principal Component Analysis (PCA) was performed. This helps reduce the dimensionality of data by identifying patterns in the data and projecting them onto a lower-dimensional space. Total of twenty features were selected at the end for modeling.

E. Data Modeling

For the purposes of data modeling, ensemble classifiers were used to fit the data and predict the output. Light gradient boosting, XGBoost, CatBoost, AdaBoost, and

Random Forests are all ensemble machine learning algorithms that can be used for classification and regression tasks. Light gradient boosting, also known as LightGBM, is a gradient boosting algorithm that uses decision trees as the base learners. It is designed to be fast and efficient, and has been shown to outperform many other popular boosting algorithms on a variety of machine learning tasks [3]. LightGBM uses a technique where data is bucketed into bins using the histogram distribution and these are used to split the data. It also reduces dimensionality, making it more efficient [4].

XGBoost, which stands for "Extreme Gradient Boosting," is another popular gradient boosting algorithm that uses decision trees as the base learners. It is known for its speed and performance [3]. XGBoost uses residuals based on the predicted value and the observed values. A decision tree is created with the residuals, and the output of the tree becomes the new residual which is used to make another tree. This process is repeated until the residuals stop decreasing, with each tree learning from the previous [5].

CatBoost is a gradient boosting algorithm developed by the Russian online retailer Yandex. Like LightGBM and XGBoost, it uses decision trees as the base learners. CatBoost implements a regular Gradient Boosting algorithm with the addition of two things, namely, ordered boosting and an algorithm for processing categorical features. It is specifically designed to handle categorical data, which is common in many real-world datasets [3].

AdaBoost, short for "Adaptive Boosting," is a gradient boosting algorithm that uses weak learners as the base models. Unlike decision trees, which are often used as the base learners in other gradient boosting algorithms, weak learners are simple models that are only slightly better than random guessing. AdaBoost works by iteratively training the weak learners and combining them to produce a strong learner that can make accurate predictions. At each iteration, the weights of the misclassified examples are adjusted such that the weights of the misclassified examples are increased and the weights of the correctly classified examples are decreased [3].

Random Forests are a type of ensemble learning algorithm that uses multiple decision trees to make predictions. Unlike gradient boosting algorithms, which build the trees in a sequential manner, the trees in a Random Forest are trained independently of each other, using a random subset of the training data. The final predictions are made by aggregating the predictions of all the individual trees. This can help to reduce overfitting and improve the overall performance of the model [3].

III. Results

Precision, recall, F1 score, and accuracy are all metrics that can be used to evaluate the performance of a machine learning model. We judged all of the classifiers used in our project based on these metrics.

Precision is a measure of the accuracy of a model when it predicts positive outcomes. It is defined as the number of true positive predictions divided by the total number of positive predictions made by the model [6]. The precision values for our classifiers can be found in [Table 3].

Recall [Table 4] is a measure of the ability of a model to correctly identify all relevant instances. It is defined as the number of true positive predictions divided by the total number of actual positive cases in the dataset [6].

F1 score, as seen in [Table 2], is a metric that combines precision and recall. It is defined as the harmonic mean of precision and recall, with a value between 0 and 1, with a higher score indicating a better performing model [6].

Accuracy is a measure of the overall correct classification rate of a model. It is defined as the number of correct predictions made by the model divided by the total number of predictions made [6]. The accuracies of all models are observed in [Table 5].

Model	**F1 Score**
Random Forest	0.9096
AdaBoost	0.9097
XGBoost	0.8260
LightGBM	0.7770
CatBoost	0.8024

Table 2: F1 Scores for Models

Model	**Precision**
Random Forest	0.9109
AdaBoost	0.9101
XGBoost	0.8267
LightGBM	0.7827
CatBoost	0.8033

Table 3: Precision for Models

Model	Recall
Random Forest	0.9113
AdaBoost	0.9109
XGBoost	0.8259
LightGBM	0.7791
CatBoost	0.8027

Table 4: Recall for Models

Model	Accuracy
Random Forest	0.9113
AdaBoost	0.9111
XGBoost	0.82386
LightGBM	$0.7801 \rightarrow 0.8364$
CatBoost	0.8034

Table 5: Accuracy for Models

As the accuracy for Light Gradient Boosting was lower than others, we performed hyperparameter tuning using cross validation and GridSearchCV and obtained an increase in accuracy from 0.7801 to 0.8364.

IV. Discussion

From the results above, we can infer that both AdaBoost and Random Forest performed best on this data based on accuracy alone. AdaBoost and Random Forest both showed accuracies of 0.9111 and 0.9113 respectively. However, accuracy alone is not enough to assess the operation of the classifiers. Comparing their precision scores, we can see that Random Forest once again edges out AdaBoost slightly with a 0.9109 compared to 0.9101. High precision means a low false positive rate. Similarly with recall, Random Forest performs slightly better than AdaBoost with a 0.9113 in comparison to 0.9109. The F1 Score is indistinguishable, with Random Forest having 0.9096 and AdaBoost having 0.9097. Both models can be considered as high performing for this data.

AdaBoost works well because it is simple to implement and achieves good performance with little need for hyperparameter tuning. It is also resistant to overfitting and is less sensitive to the scale of the features compared to other algorithms [3]. Random Forests can handle a large number of features and can perform well even without feature selection or hyperparameter

tuning. They are also resistant to overfitting and are robust to the presence of noisy or missing data [3].

Some limitations of the project which we would like to address in future work are that the popularity of a song also depends on the popularity of the artist, not necessarily only on audio features of a song. This is not being accounted for in the dataset. We would like to combine this data in the future with artist popularity as well to make it more exhaustive and gain deeper insights. Further feature engineering and parameter tuning can also be performed to gauge if the models' performances improve.

V. References

[1] Hamidani, Z. (2019, July 23). Spotify tracks DB. Kaggle. Retrieved from https://www.kaggle.com/datasets/zaheenhamidani/ultimate-spotify-tracks-db

[2] Web API reference: Spotify for developers. Home. (n.d.). Retrieved from https://developer.spotify.com/documentation/web-api/reference/#/operations/get-audio-features

[3] Gupta, S. (2022, October 13). XGBoost vs. CatBoost vs. Lightgbm: How do they compare? Springboard Blog. Retrieved from https://www.springboard.com/blog/data-science/xgboost-random-forest-catboost-lightgbm/

[4] How LightGBM algorithm works—ArcGIS Pro | Documentation. (n.d.). https://pro.arcgis.com/en/pro-app/latest/tool-reference/geoai/how-lightgbm-works.htm

[5]How XGBoost algorithm works—ArcGIS Pro | Documentation. (n.d.). https://pro.arcgis.com/en/pro-app/latest/tool-reference/geoai/how-xgboost-works.htm

[6] B, H. N. (2021, December 12). Confusion Matrix, Accuracy, Precision, Recall, F1 Score. Medium. https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd

VI. Appendix

Link to GitHub code repository: https://github.com/nidhimankala/DS-5220-project.git