TEAM-35

CONTENT

Overview

Music has excellent qualities of healing a person emotionally and mentally. It is a form of meditation. While composing or listening to music, one tends to forget all his worries, sorrows, and pains. It has the power to cure diseases such as anxiety, depression, insomnia, etc. It is composed and performed for many purposes, ranging from aesthetic pleasure, religious or ceremonial purposes, or entertainment to the marketplace.

The ability to predict a music track's popularity is one of the most crucial tasks of record labels who purchase a music track's rights. These predictions help the record labels to place their bids for these music tracks. A higher bid can result in the purchase of music tracks at a higher price, whereas a lower bid results in losing a good music track altogether.

Here we analyzed the data of music tracks and used the data to predict these music tracks' popularity. According to the problem statement, we have to maximize the revenue by correctly predicting these music tracks' popularity. A major record label will then use this expected popularity to place their bid to purchase these music tracks and produce maximum revenue for their organization. Here a wrong prediction can also be considered a valid bid as long as we bid a less popular music track at the cost of a more popular music track, but vice versa is not possible.

Exploratory Data Analysis

Data Summary

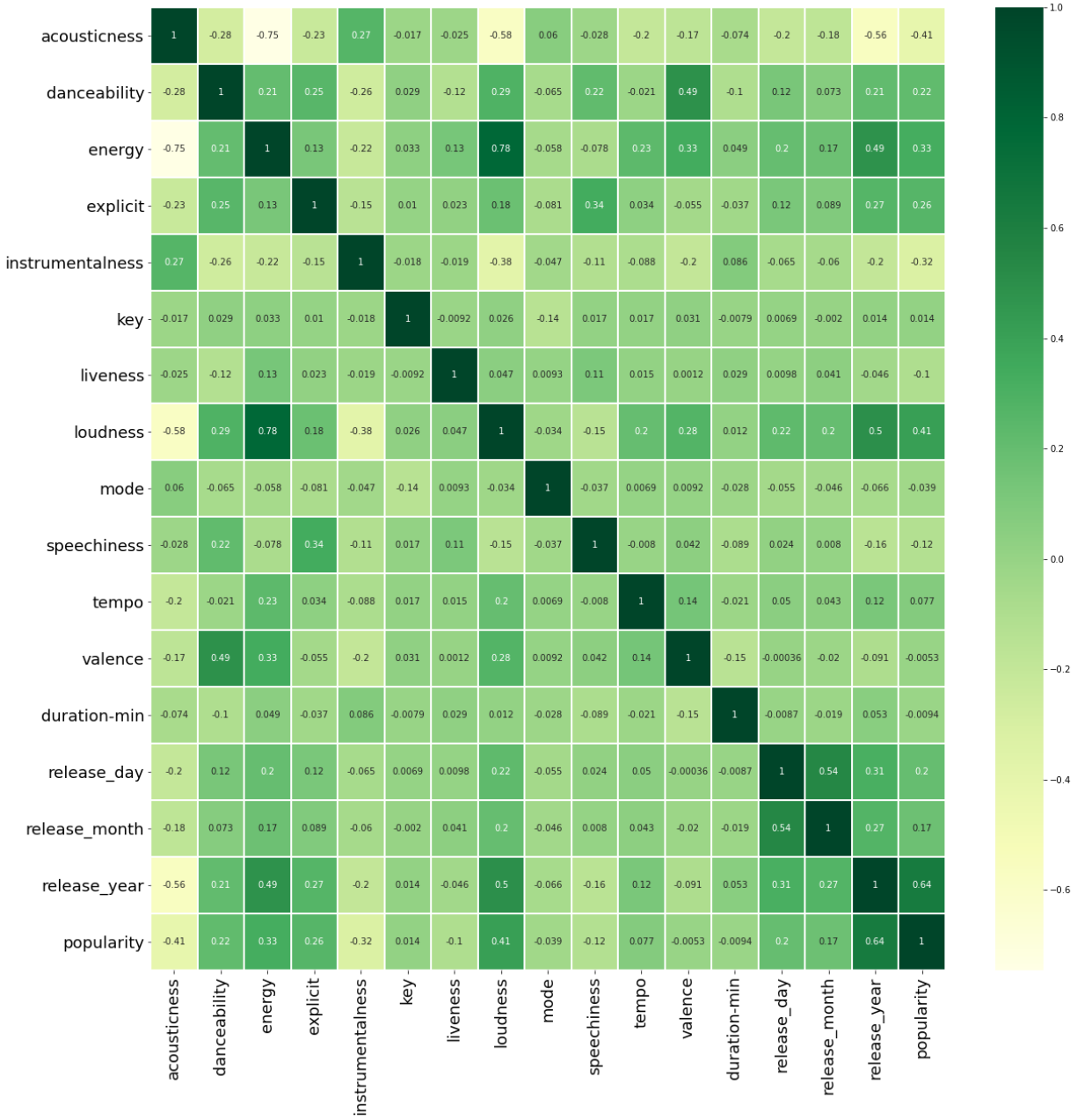
Summary Statistics

The dataset consists of **12227 rows and 17 columns**. There are **three categorical features, one feature of datetime and 13 numerical features.** From the mean, 75% and max columns, we can see that there are outliers in several columns.

Feature/Stat	Mean	Std	Min	25%	50%	75%	Max
Acousticness	0.430578	0.366893	0.000001	0.05895	0.354000	0.80500	0.996
Danceability	0.556353	0.175373	0.000000	0.43800	0.569000	0.68500	0.980
Energy	0.522129	0.262482	0.000020	0.30300	0.534000	0.73900	1.000
Instrumentalness	0.149321	0.297954	0.000000	0.00000	0.000115	0.05565	1.000
Key	5.205202	3.526954	0.000000	2.00000	5.000000	8.00000	11.000
Liveness	0.201365	0.173987	0.014700	0.09620	0.132000	0.25200	0.997
Loudness	-10.6686	5.506888	-43.738	-13.6560	-9.5840	-6.57150	1.006
Speechiness	0.097680	0.155895	0.000000	0.03470	0.045600	0.07890	0.968
Tempo	118.1674	30.200	0.000000	95.05050	116.9150	136.1085	216.843
Valence	0.525300	0.258205	0.000000	0.32100	0.532000	0.73700	1.000
Year	1984.517	25.9119	1920.00	1966.00	1987.00	2008.00	2021.000
DurationMin	3.888133	2.383133	0.200000	2.90000	3.600000	4.40000	2021.000

Correlation Matrix

Loudness refers to how loud or soft a sound seems to a listener. The loudness of sound is determined, in turn, by the intensity of the sound waves. Intensity



We made the correlation matrix between various features and found that the **loudness** and **energy** are **highly positively correlated**. nsity is a measure of the amount of energy in sound waves.
In loud music, musical energy is easy to identify. We notice the energy more as the drums get busier and play louder and as the singer sings higher.

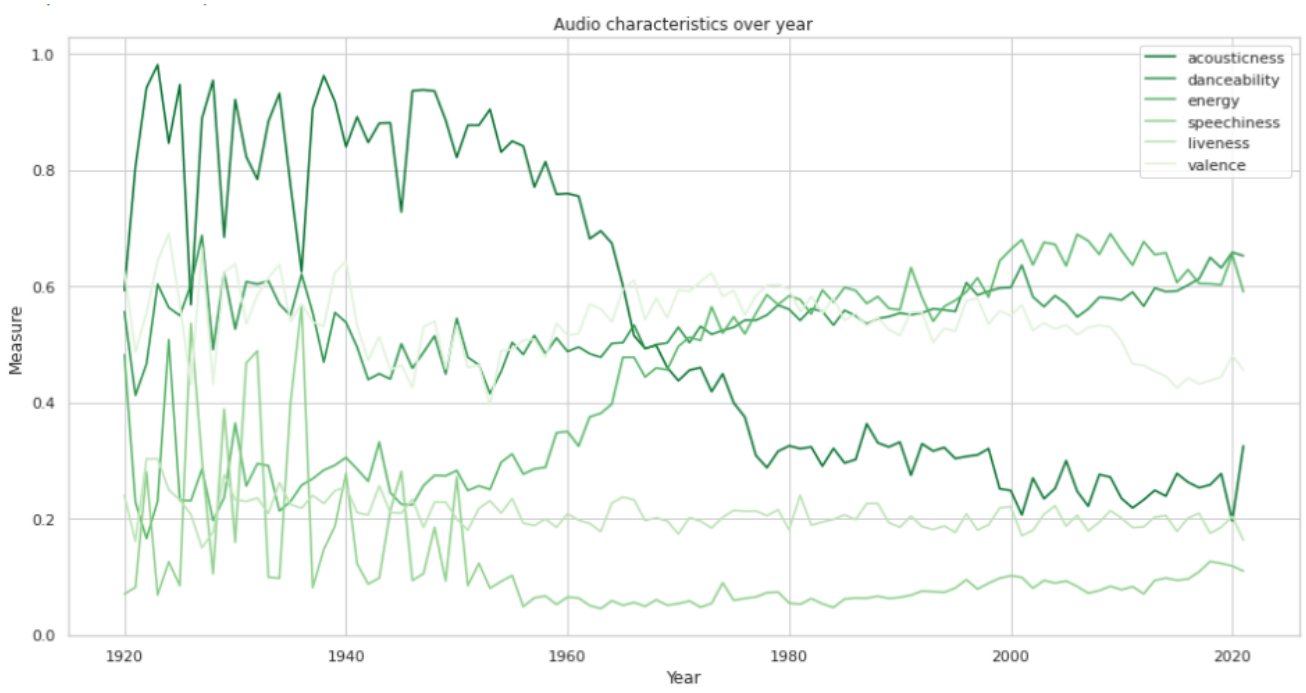
We see that the **acousticness** is **highly negatively correlated** with the **song's energy and loudness**, which makes sense since acousticness represents the degree of purity of a song. A

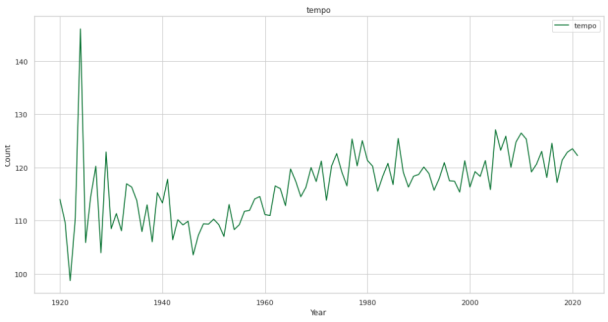
highly acoustic sound is a sound that is not electrically enhanced and modified, and hence they have less energy and are less loud. We can also infer that **the year column and release date column** are **highly correlated**, so we decided to remove the year column and divided the release date column into day, month and year.

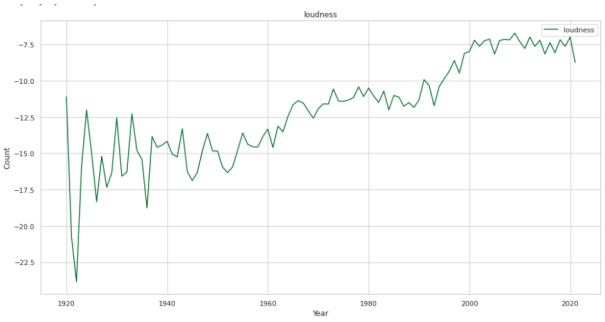
Data Visualisation and Insights

Trend Over The Years

We can see some of the observations that the **acousticness in songs decreased** over the years while the **energy increased**. Similarly, the **loudness** and the **tempo** of the songs **increased** over the years.

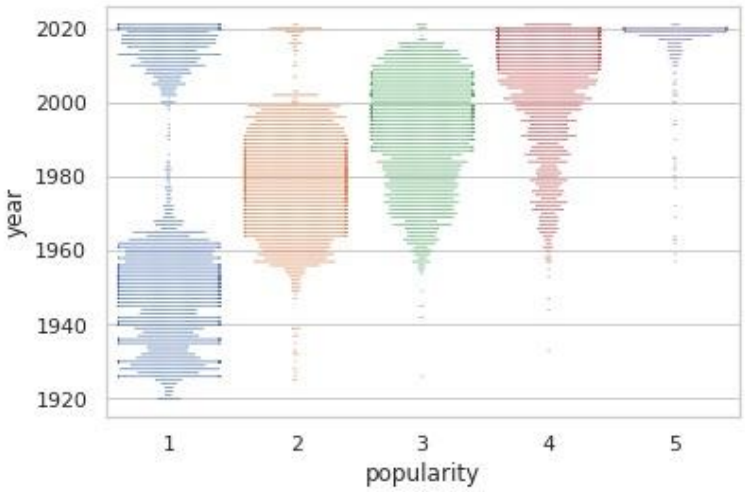






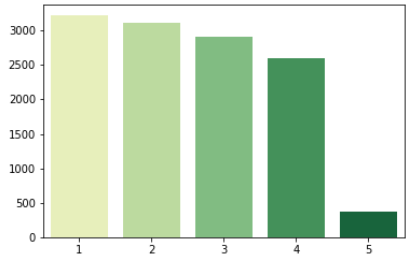
Year Column

From the following graph between Year vs Popularity, most of the songs with **very high popularity are from 2015 onwards**. The reason could be because of the advancement of technology. Because of apps in smartphones, songs have been easily accessible to a large group of the audience because of which each song now gets more listened to and thus becomes more popular.



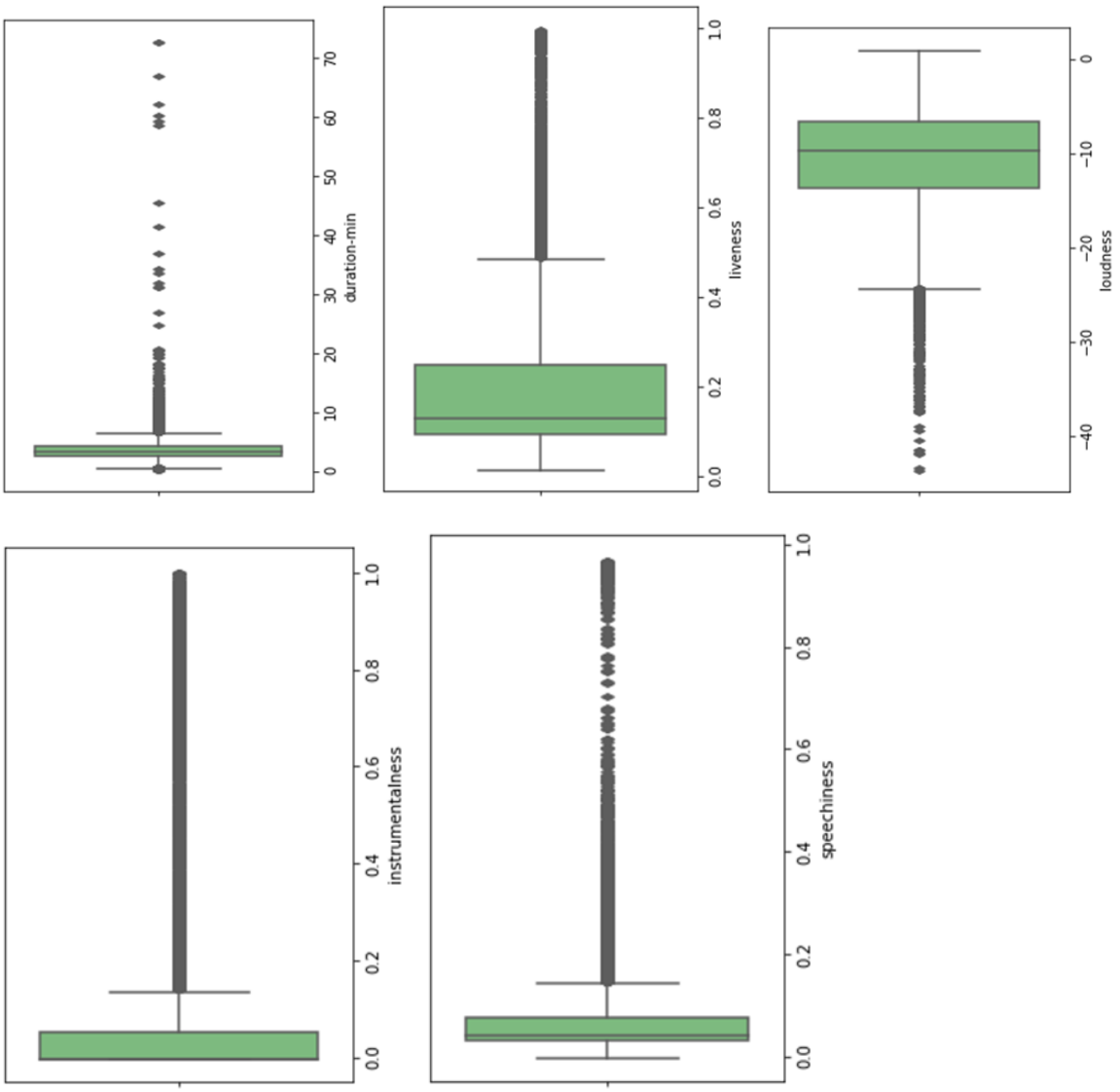
Imbalance in the Dataset

Following is the chart of the frequency of different popularity classes. We can see that songs with very high popularity are the smallest, which shows an **imbalance in the dataset.** The **very high** popularity constituted **3%** of the total dataset. To tackle this problem, **we used oversampling.**



Anomaly Detection

After plotting boxplots for the features, we found significant outliers in the following features, which are -> **Duration-min**, **Liveness**, **Loudness**, **Instrumentalness**, and **Speechiness**.



The outliers constituted **50%** of the dataset, so removing them wasn't an option. We tried replacing them with their median and also eliminating the columns, but the score decreased after both of them. So we decided **not to treat** the outliers. The **reason** was that we were going to

train decision tree models, and outliers wouldn't affect the model much, which was visible in the results.

Feature Engineering

Variance Threshold

This is the simplest form of selecting features. A threshold value is used, and any feature having variance less than a threshold is removed from the dataset.

SelectFromModel

Scikit-learn offers a SelectFromModel class that helps to choose features directly from a given model. It selects a given number of features **based on the importance weights**.

SelectKBest

Select features according to the **k highest scores** where the scoring function can be the ANOVA, F-Test, Chi-Squared Test, etc.

Greedy Feature Selection

Followed the following steps ->

- The first step is to choose a model.
- The second step is to select a loss/scoring function.
- And the third and final step is to iteratively evaluate each feature and **add it to the list of "good" features if it improves loss/score.**

Common features selected are ->

Year	**Valence**
Danceability	**Acousticness**
Instrumentalness	**Liveness**
Duration-min	

Classification Models

In machine learning, **classification** refers to a predictive modeling problem where a class label is predicted for a given example of input data.

Examples of classification problems include:

- Given an example, classify if it is spam or not.
- Given a handwritten character, classify it as one of the known characters.
- Given recent user behavior, classify as churn or not.

There are perhaps **four main types of classification** tasks that you may encounter; they are:

- Binary Classification
- Multi-Class Classification
- Multi-Label Classification
- Imbalanced Classification

Our problem is of an imbalanced classification. To predict the classes we use the following models:

- Cost-sensitive Logistic Regression
- Cost-sensitive Decision Trees
- Cost-sensitive Support Vector Machines

The scores of Logistic Regression and Support Vector Machines were a lot less as compared to Decision Trees. The **models having the better score** are the following:

- XGBoost
- CatBoost
- Light GBM
- Random Forest

XGBoost

Theory

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable. It implements machine learning algorithms under the Gradient Boosting framework. It is an approach where new models are created that predict prior models' residuals or errors and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

Why XGBoost?

The **two reasons to use XGBoost** are also the two goals of the project:

- **Execution Speed**: It is really fast when compared to other implementations of gradient boosting.
- **Model Performance**: It dominates structured or tabular datasets on classification and regression predictive modeling problems.

Finally, the evidence is that it is the go-to algorithm for competition winners on the Kaggle competitive data science platform.

Results

Class	Precision	Recall	F1-score
Very Low	0.85	0.78	0.81
Low	0.56	0.67	0.61
Average	0.45	0.44	0.44
High	0.68	0.56	0.61
Very High	0.95	1.00	0.97

Test Accuracy -> 71.2	**Training Accuracy -> 99.95**
Bidding Value-> 7540	**Revenue Collected -> 12488**

LightGBM

Theory

It is a gradient boosting framework that makes use of tree-based learning algorithms that is considered to be a very powerful algorithm when it comes to computation. LightGBM is called "Light" because of its **computation power** and giving **results faster**. It takes **less memory** to run and can deal with large amounts of data.

Why LightGBM?

- **Faster training speed and higher efficiency**: Light GBM uses histogram based algorithm i.e it buckets continuous feature values into discrete bins which fasten the training procedure.
- **Lower memory usage**: Replaces continuous values to discrete bins which result in lower memory usage.
- **Better accuracy than any other boosting algorithm**: It produces much more complex trees by following leaf wise split approach rather than a level-wise approach which is the main factor in achieving higher accuracy.

Results

Class	Precision	Recall	F1-score
Very Low	0.56	0.68	0.62
Low	0.84	0.81	0.82
Average	0.47	0.46	0.47
High	0.67	0.47	0.55
Very High	0.90	0.98	0.97

Test Accuracy -> 70.05	**Training Accuracy -> 89.95**
Bidding Value-> 7538	**Revenue Collected -> 14046**

CatBoost

Theory

CatBoost is a machine learning algorithm that uses gradient boosting on decision trees.CatBoost does not follow similar gradient boosting models. Instead, CatBoost grows oblivious trees, which means that the trees are grown by imposing the rule that all nodes at the same level test the same predictor with the same condition. Hence, an index of a leaf can be calculated with bitwise operations. The oblivious tree procedure allows for a simple fitting scheme and efficiency on CPUs, while the tree structure operates as a regularization to find an optimal solution and avoid overfitting.

Why CatBoost?

The **advantages of CatBoost** compared to other models are

- It provides a symmetrical tree structure
- It takes less prediction time as compared to other algorithms
- It provides support to both numerical and categorical features

Results

Class	Precision	Recall	F1-score
Very Low	0.84	0.81	0.82
Low	0.56	0.66	0.60
Average	0.46	0.47	0.46
High	0.63	0.40	0.49
Very High	0.85	0.98	0.91

Test Accuracy -> 68.00	**Training Accuracy -> 77.05**
Bidding Value-> 7539	**Revenue Collected -> 13972**

Random Forest

Theory

Random Forest operates as an ensemble of a large number of decision trees. In this algorithm, all the trees spit out the prediction, and the class with the most number of votes becomes the model's prediction. It can be used for classification and regression problems.
One of the primary reasons for the random forest's success is that there are a large number of uncorrelated trees combined that will outperform any of the individual constituents models. The Random Forest algorithm makes it very easy to measure the relative importance of each feature. The Random Forest algorithm is very handy because the hyperparameters it uses by default often produce good results.

Why Random Forest?

The following points explain the **reason for using Random Forest Algorithm:**

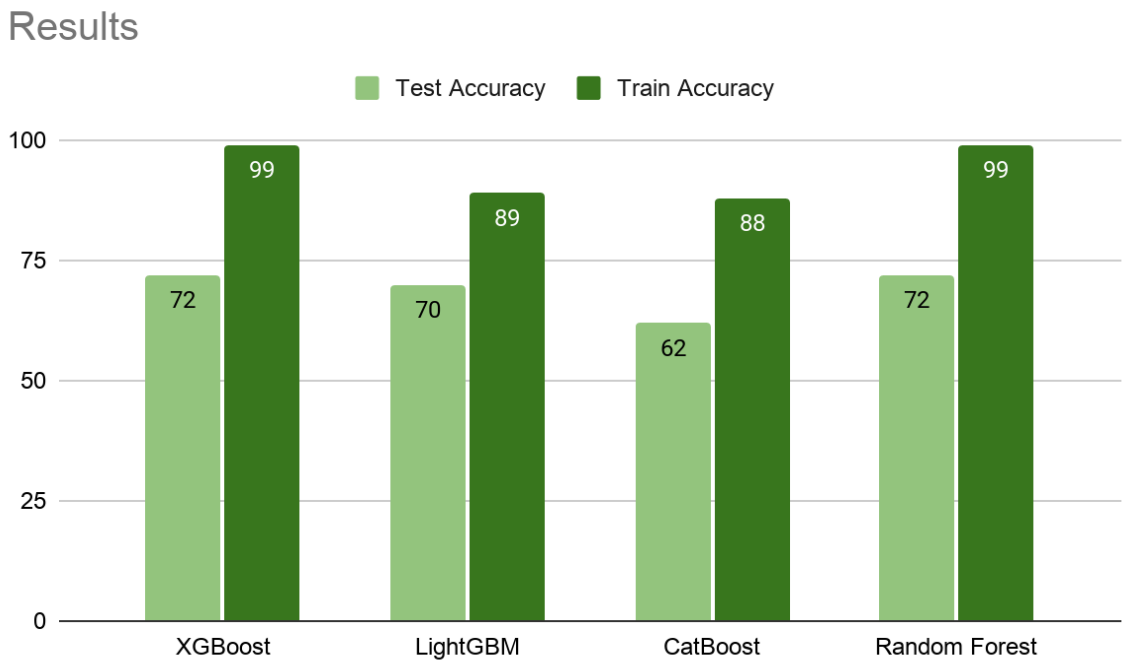
- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.
- The Random Forest algorithm adds randomness to your model.

Results:

Class	Precision	Recall	F1-score
Very Low	0.86	0.79	0.83
Low	0.57	0.70	0.63
Average	0.47	0.48	0.48
High	0.70	0.54	0.61
Very High	0.95	1.00	0.97

Test Accuracy -> 71.35	**Training Accuracy -> 99.95**
Bidding Value-> 7538	**Revenue Collected -> 14086**

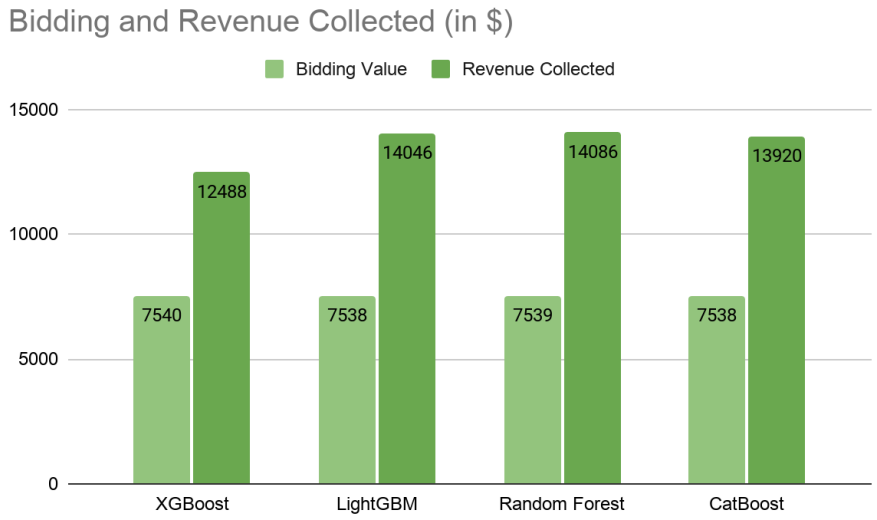
Comparative Analysis



Since both XGBoost and Random Forest models are overfitting, and the score of CatBoost is less, we are going to choose **LightGBM as our final model.**

Evaluation Metric

We split the dataset into **80:20 ratio** and calculated the bidding and the revenue value collected. The chart shows the revenue and bidding value for the validation dataset of each model.



We used two methods to calculate the revenue. In the first method we used **brute force bidding**, i.e we bid for all the songs till the bidding limit reached. In the second method, we calculated the probability of each class's prediction and **sorted** each row in decreasing order based on the highest probability achieved for a class and bid till the bidding limit was reached. We finally chose the max revenue based on these methods.

Bidding Limit = (Length of Validation Set)*(10,000/4,000)

Conclusion

Based on the **revenue collected** and **train** and **test accuracy** as well as **F1 scores** of XGBoost, LightGBM, Random Forest and CatBoost models, we chose **LightGBM** to be our **final model**. The bidding total of the model on our validation set of size **3016** rows is **$7538**, and the revenue collected is **$14046**. Our model has **good predictions** for **Very High** and **Low** popularity, which is visible in our **classification report** of the Light GBM model. So for most of our bidding, we are having a profit of at least 1. Because of this, we have a high revenue collection in our validation dataset.