TEAM 41

CONTENTS:

OVERVIEW:

The Music Industry:

The music industry is not one, but a number of different industries that are all closely related but are based on different logics and structures. The overall music industry is based on the creation and exploitation of music-based intellectual properties - composers and songwriters create songs, lyrics, and arrangements that are performed live on stage; recorded and distributed to consumers; or licensed for some other kind of use.

Music Royalties and Record Labels:

Music royalties are payments that go to recording artists, songwriters, composers, publishers, and other copyright holders for the right to use their intellectual property. Record labels market and distribute an artist's original work. They often have the master rights to a recorded song.Record labels generate income from mechanical and public performance royalties. They issue contracts that allow them to exploit the recordings in exchange for royalty payments over a set length of time. The artist then receives a flat rate or percentage of these record label royalties.

Task :

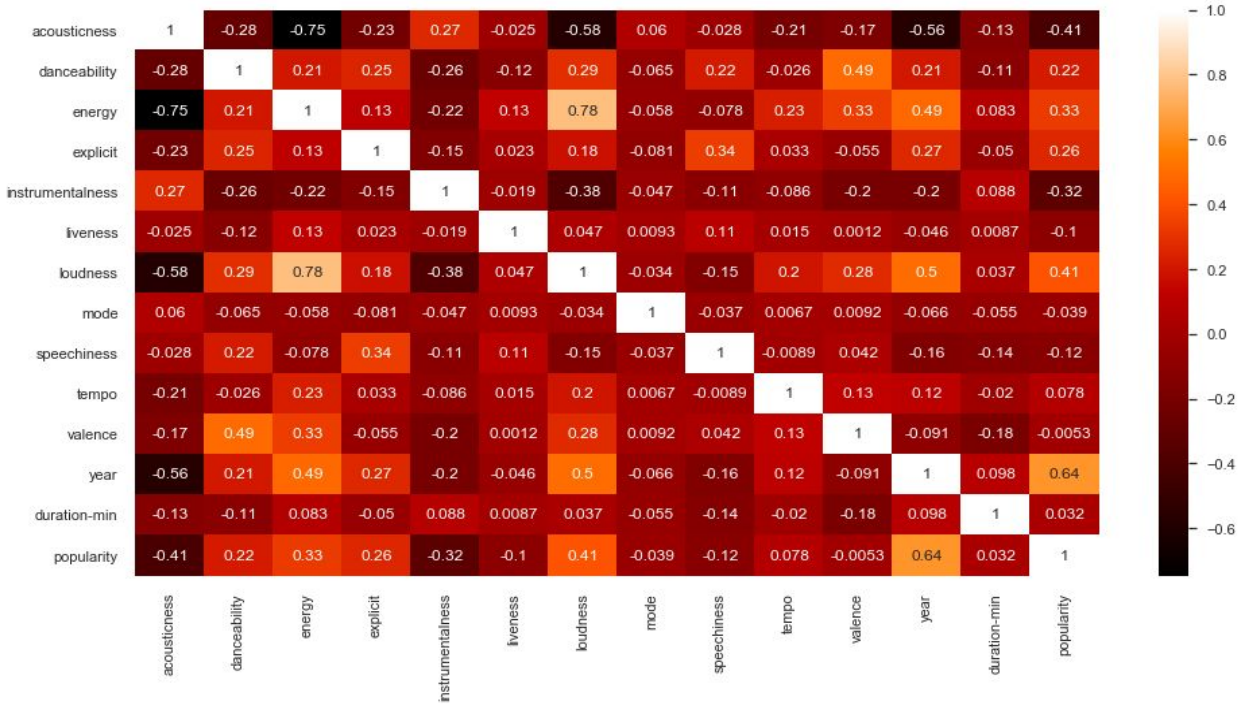
Buying rights to all songs are not equally beneficial - the maximum revenue is generated from the most popular songs. Hence, it becomes important to predict beforehand the expected popularity of the song to maximise the profit. A major record label is planning to invest 10,000 (10k $ units) on 4000 songs. We are expected to help them in ensuring that they do not encounter losses with promotion and distribution of the track. Using the features given, we need to predict the songs as belonging to 5 classes of popularity categories: 'Very high', 'high', 'average', 'low' and 'very low' in such a way that the revenue is maximised.

Approach :

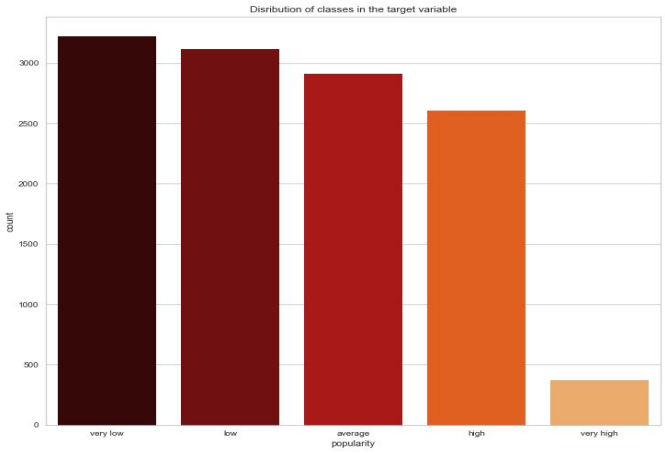
We have used data visualization tools to generate insights and developed intelligent features to support our forecasting models. Identification of anomalies was done and treated suitably. The data was highly imbalanced which we took care of by upsampling the minority classes to ensure proportional distribution. We have developed various classification models and evaluated them on the validation set for model selection. There was no one superior model, so we used an ensemble approach, stacking results from the best performing models to produce robust predictions and prevent overfitting.

EXPLORATORY DATA ANALYSIS :

Data Summary:

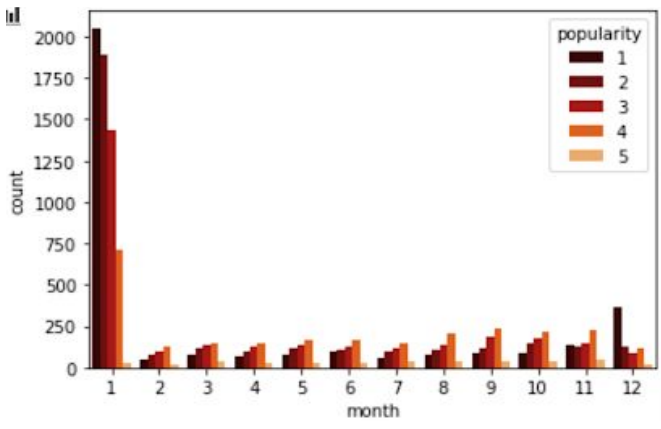


As can be seen from the heatmap, **year shows the greatest correlation** with the target variable (popularity), followed by loudness and acousticness.



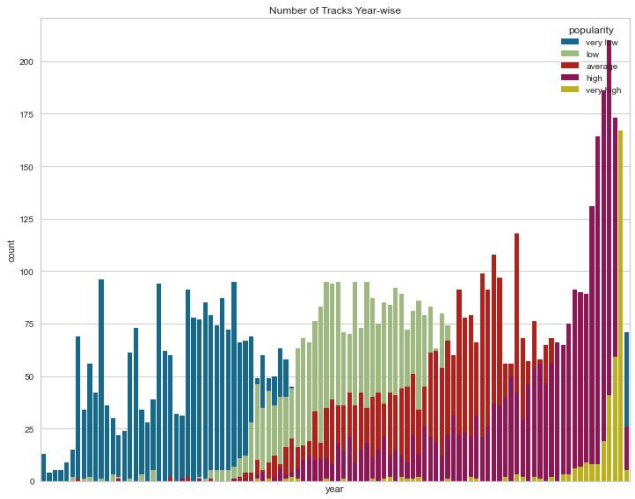
Highly imbalanced dataset, with songs belonging to “very high” popularity constituting less than 4% of all songs.

Rest of the classes are more or less proportional to each other.



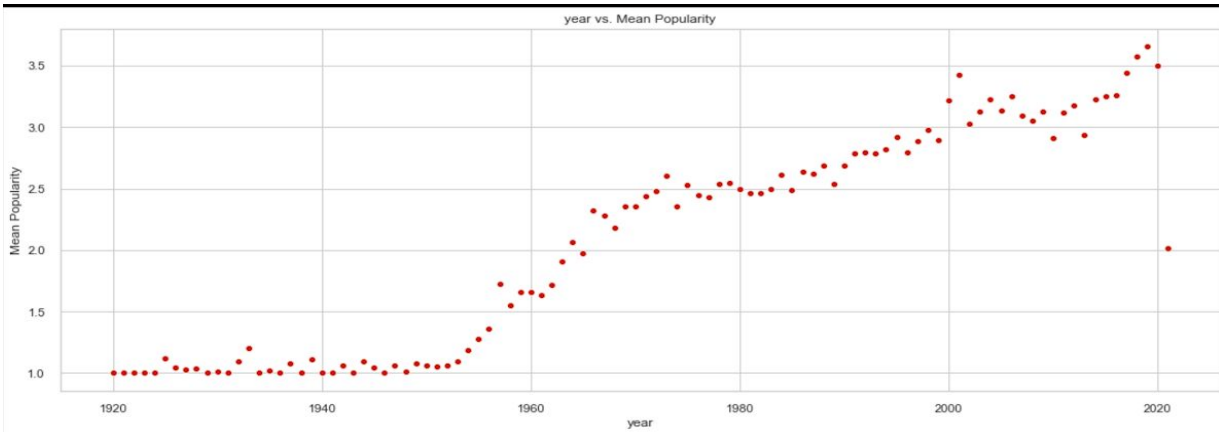
A far **greater number of songs released in January** compared to the rest of the year, most of which are among the less popular tracks.

Time Series Analysis:

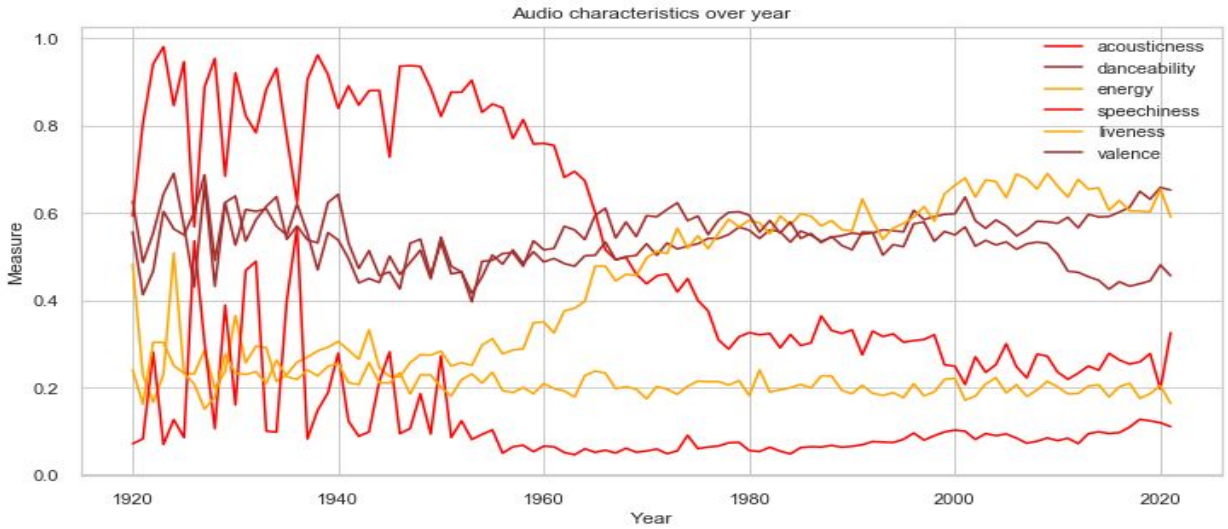


This graph shows the count of songs released yearwise based on their popularity. The **initial years are dominated by songs belonging to "very low" popularity** with the trend shifting towards higher popularity with time

The graph below shows the mean popularity of the songs released yearwise. There is a **clear upward trend, post 1960.**



Variance of a few audio characteristics with time. Some, like acousticness, show clearly visible trends while some others remain more or less static.

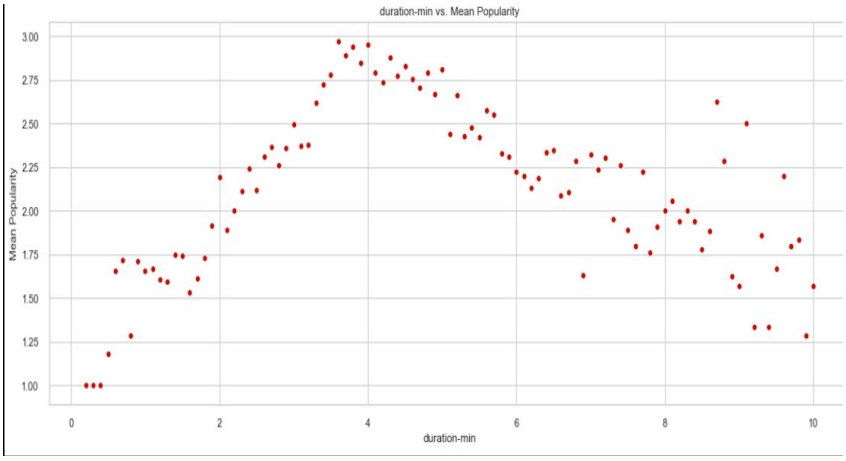


NOTE: Further graphs and visualization have been included in the ANNEXURE section.

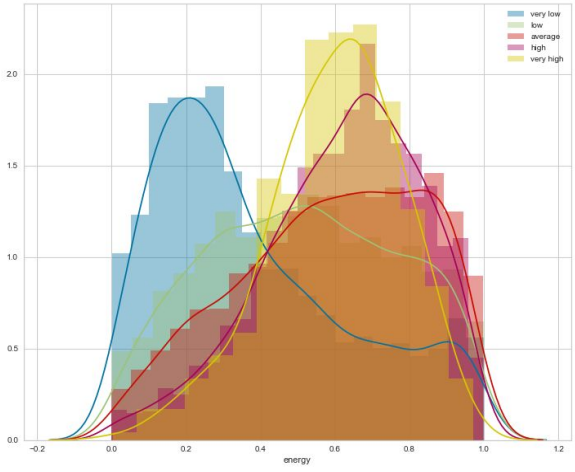
Other Visualizations:

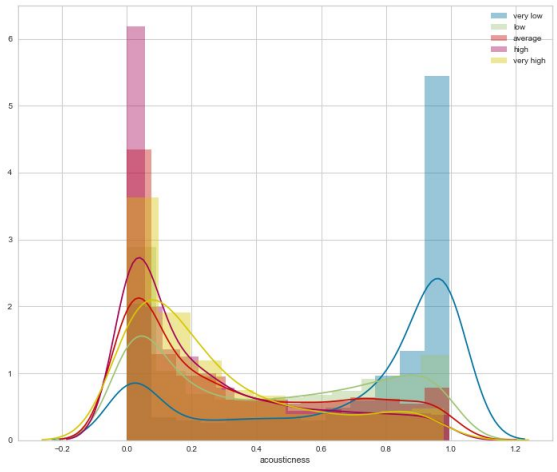
Mode is almost invariant - hence was dropped from being used as a feature in models, both classes of mode have remarkably similar statistical properties with respect to the target variable.

Mode	Count	Mean popularity	Mode popularity	std
Major	3740	2.56	3.0	1.22
Minor	8487	2.460	2.0	1.15



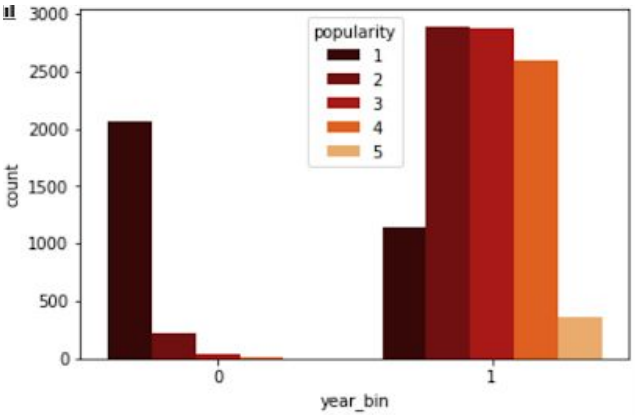
The following is a plot of **duration vs popularity.** The trend shows that there exists an optimal song duration for which mean popularity is the highest. This matches our intuition as neither too long nor too small songs would be appreciated.
Hence, this is an important feature for the dependent variable, but does not have a high linear correlation value due to inverse trend in certain parts.





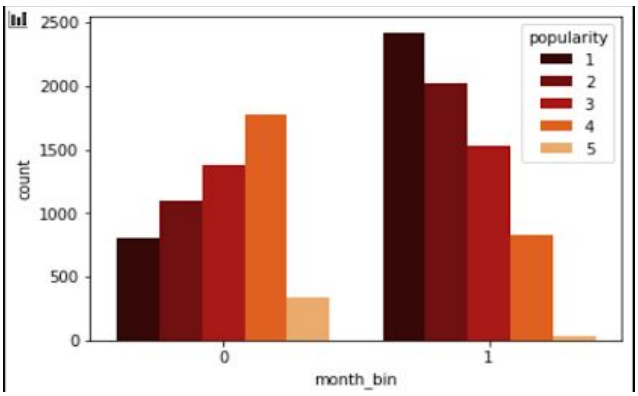
Density distribution plot of features separately for each class , to observe different trends in different class values. Trends for energy and acousticness are very contrasting.

FEATURE ENGINEERING:



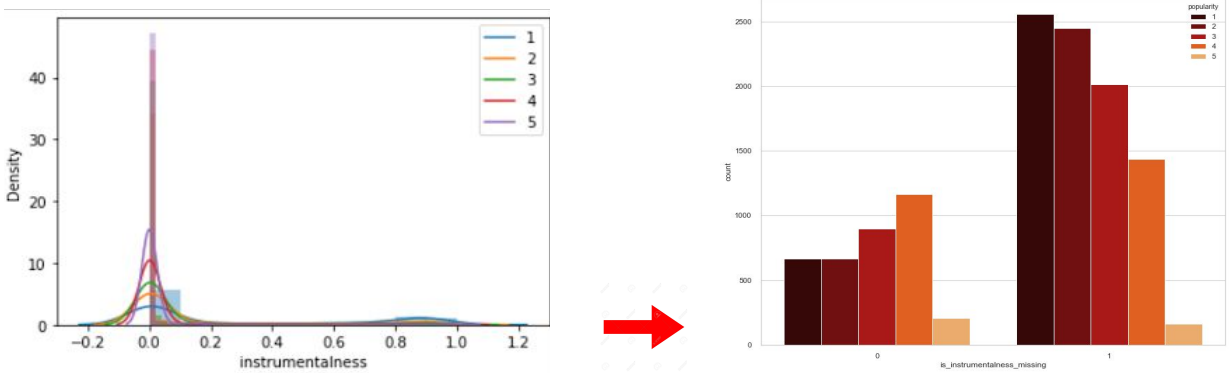
We created this feature **" year_bin"** based on previous results of EDA.

1 represents songs released post 1960 and 0 represents those before 1960
Both categories have a very different popularity distribution and led to increased accuracy of models as this captures the information in the year of release well.



"Month_bin" has been created based on our findings in EDA that a greater number of songs have been released in January.

1 represents songs released in January while 0 represents those released in other months.
The 2 bins differ in distribution and have positively affected the model performance.



Around 30% of data samples have **instrumentalness** value as 0. This abnormality can be seen clearly in the left graph. We capture this information by engineering a feature having **binary values - 1 for non-zero values and 0 for others**. The two bins of the transformed features have clearly different class-wise distributions of popularity thus adding more discriminative power which led to increase in the model performance.

NOTE: We had tried building some other features as well, but they did not add significant value to our model and so we have not discussed them here.

CLASSIFICATION MODELS:

At a glance, our problem looks like one pertaining to classification (since there are 5 labelled classes of popularity). But the values for popularity are ordered from high to low and hence, we tried a regression model as well. However, the performance of the regression model was pretty poor and so we decided to stick to classification models only.

Classification is a subset of supervised learning and is the process of grouping objects into a fixed number of preset classes or labels. There are a wide variety of classification models available and are used in a variety of fields such as biometric identification, document classification, text and image recognition, etc. We will be using these models to build an efficient and robust song popularity prediction model.

We have used some of the commonly used classification models which are reliable for giving high quality output and have their own unique mechanism.
For our required 5-class classification problem, we have used: Bagging models like Random Forest Classifier; Boosting models like XGBoost and LightGBM as well as simple classifiers like Support Vector Classifier and Decision Trees.

The dataset was severely imbalanced with songs belonging to "Very High" popularity constituting less than 4% of the entire data. We have used SMOTE to upsample the minority class and ensure a comparable proportion of all 5 classes in hope of better prediction results.

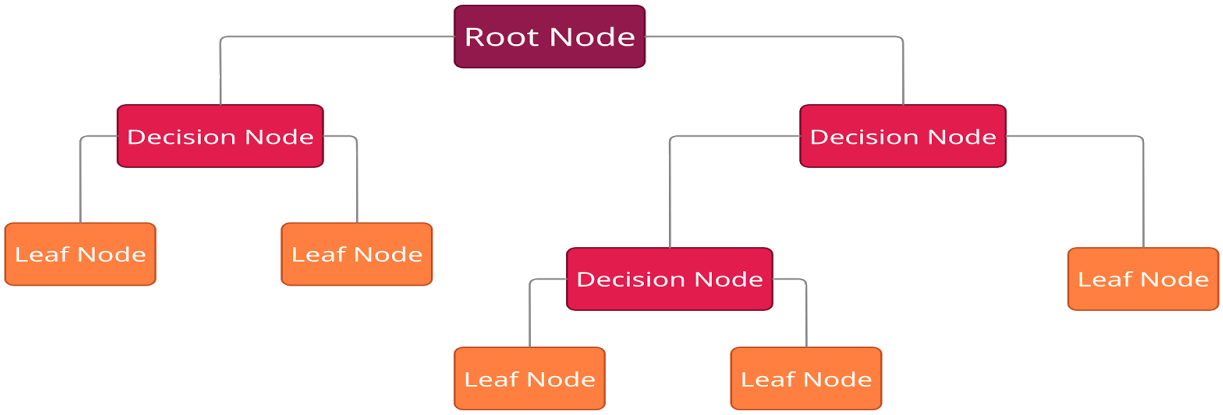
We have tried to compare the performances of each of these models on the validation set and then stacked the best performers (using Voting Classifier) to make the model robust and less prone to overfitting on unseen data.

NOTE: We tried Neural Networks but couldn't observe any significant improvement in the performance. So, we have decided to avoid them as there is often a risk of overfitting involved with neural network based models.

Decision Trees:

Theory:

Decision tree is a predictive model which uses a tree based algorithm..The decision trees are built by splitting the source set, constituting the root node of the tree, into subsets—which constitute the successor children. The splitting is based on a set of splitting rules following a greedy approach.This process is repeated on each derived subset in a recursive manner until the subset at a node has all the same values of the target variable, or when splitting no longer adds value to the predictions



Why Decision Trees?

Decision trees have a powerful representation and are one of the best performers on a large, disjunctive hypothesis space. They also work well when the data is noisy, the target function is discrete-valued and the instances are describable by a fixed set of attributes and their values.These reasons make Decision Trees a sensible choice as a starting model

Results:

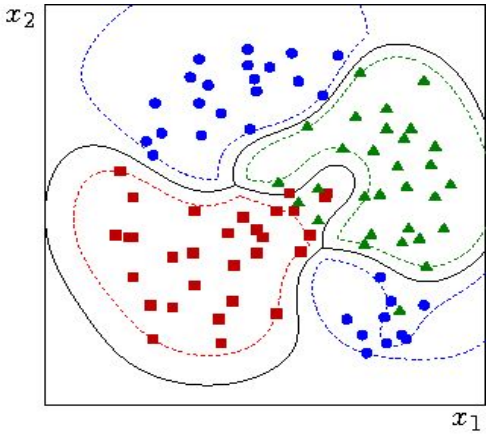
The revenue obtained in this model was: $ 8516

The amount left was: $ 188

SVC (Support Vector Classifiers):

Theory:

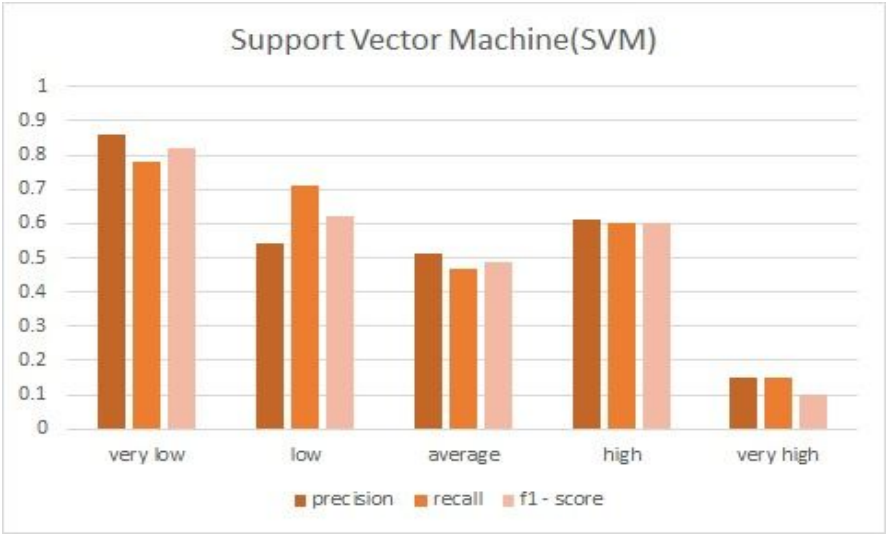
Support Vector Classifier is a supervised classification algorithm in which each data item is plotted as a point in n-dimensional space (where n is the number of features) with the value of each feature being the value of a particular coordinate. For non-linear data, it uses kernel trick to map the data in a higher dimension and then searches for optimal decision boundaries to separate the transformed data accurately.



Why SVC?

Non-linear SVM provides the benefit of capturing much more complex relationships between the data points without having to perform difficult transformations on our own. We have used our kernel as the Radial Basis Function for this purpose. This algorithm is accurate yet less prone to overfitting.

Results:

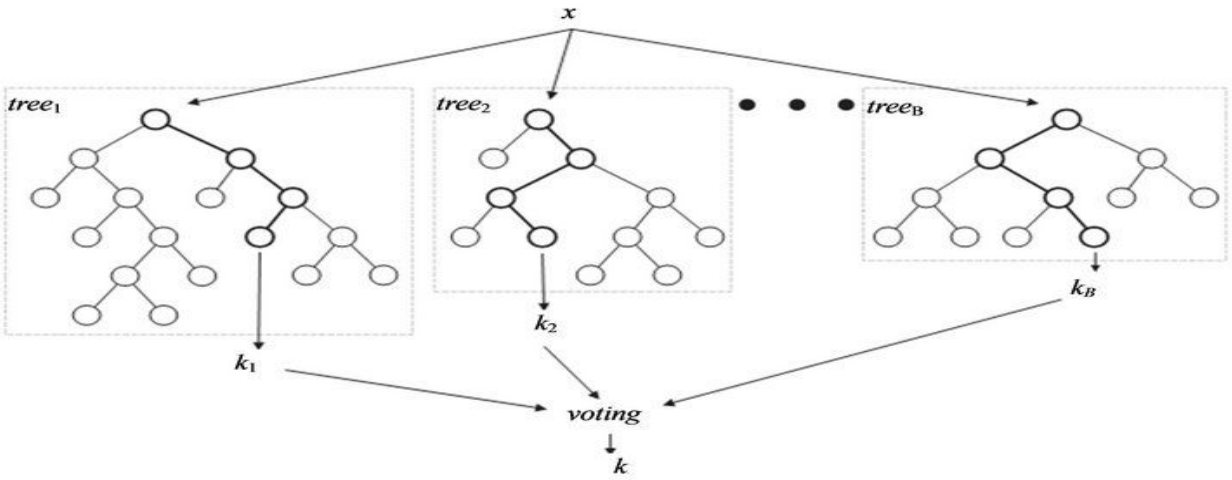


The revenue in this case was: $8325
The amount left was: $233

Random Forest Classifier:

Theory:

Random forest is a bagging model which consists of a large number of individual decision trees generated parallelly which are the base learners and operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction.



Why Random Forests?

Being a bagging model, random forest allows each individual tree to randomly sample from the dataset with replacement, resulting in different trees trained on different sets of data as well as different subset of features. Thus, independent trees are grown, which help to simultaneously improve accuracy and reduce variance (and thus overfitting)

Results:

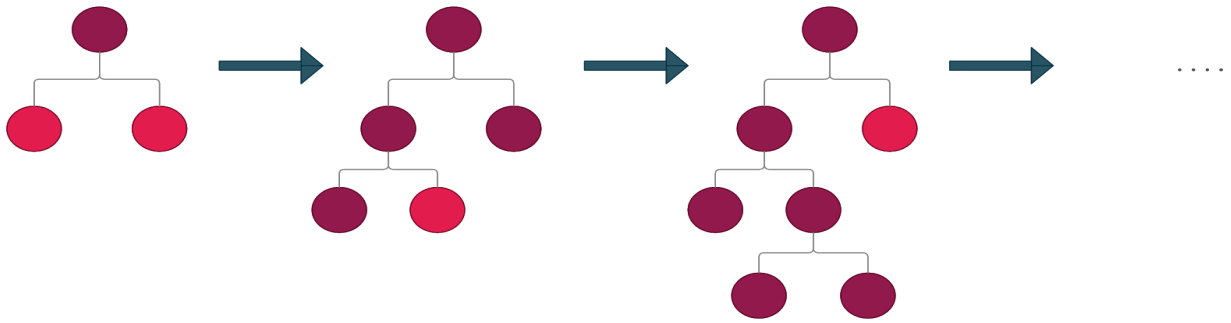
The revenue obtained in this case was: $ 8644

The amount left was: $ 72

LightGBM:

Theory:

LightGBM is a fast, distributed, high-performance gradient boosting framework that uses tree based learning algorithms that splits the tree leaf wise with the best fit. When growing on the same leaf in Light GBM, the leaf-wise algorithm can reduce more loss than the level-wise algorithm and hence results in much better accuracy.



Leaf-Wise Tree Growth

Why LightGBM?

LightGBM does not use the widely used sorting-based decision tree learning algorithm, which searches the best split point on sorted feature value. Instead, it implements a highly optimized histogram-based decision tree learning algorithm, which yields great advantages on both efficiency and memory consumption.

Results:

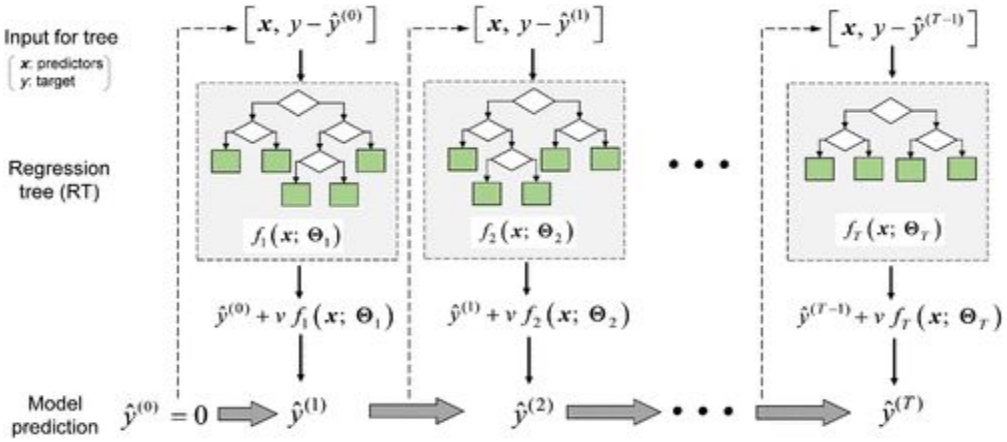
The revenue obtained in this case was: $ 8471

The amount left was: $65

eXtreme Gradient Boosting (XGB):

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 the residuals or errors of prior models 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?

● Execution Speed: It is really fast when compared to other implementations of gradient boosting.

● Model Performance: It dominates structured or tabular datasets on classification problems. It is the go-to algorithm for Kaggle competition winners.

Results:

In this model, the revenue obtained was: $8232

The amount left was: $156

THE REVENUE CONSTRAINT:

At this point of time, it becomes important to discuss a bit about the revenue being generated, because our main goal is maximising revenue, and not accuracy.

As is the case usually, we formed our validation set by splitting our given training set into 2 parts of 80% and 20%. For a particular validation set (random_state=123), a perfect prediction on each and every observation would require us to bid a total of $7,341 units suitably.
Now, we have been asked to bid a total of $10,000 units on 4,000 songs. Taking into account this 2.5 multiplier factor and our validation set having 2446 observations, we get a threshold of $6115 (2446*2.5), which is around 83% of the total required money

Thus, inherently, we can't get a very high accuracy. Because of the same reason, it's impossible to get the max revenue with that amount which is 6115*2 = 12230 (since we get double the revenue on each song as is the bid value).

So far, our revenue values have been around $8,500 units which is around 70% of the maximum amount (which, as we discussed, is impossible to achieve). Therefore, we can say that our model is actually performing better than the suggested accuracy, as far as the goal of maximising revenue goes.

ENSEMBLE METHODS:

As we have seen above, the models perform more or less similarly if we are judging based on the F1 score or accuracy but do give different values of revenue. Since we are primarily concerned with that, it makes sense to try ensemble learning techniques.
Also, we chose to exclude XGBoost since it was performing the worst.

We have used the inbuilt Voting Classifier model, giving more weightage to Random Forest, as it was the best performing individual model. Also, we have made use of Random Forest's class weight parameter , intentionally biasing the model towards the top 2 classes because of two reasons:

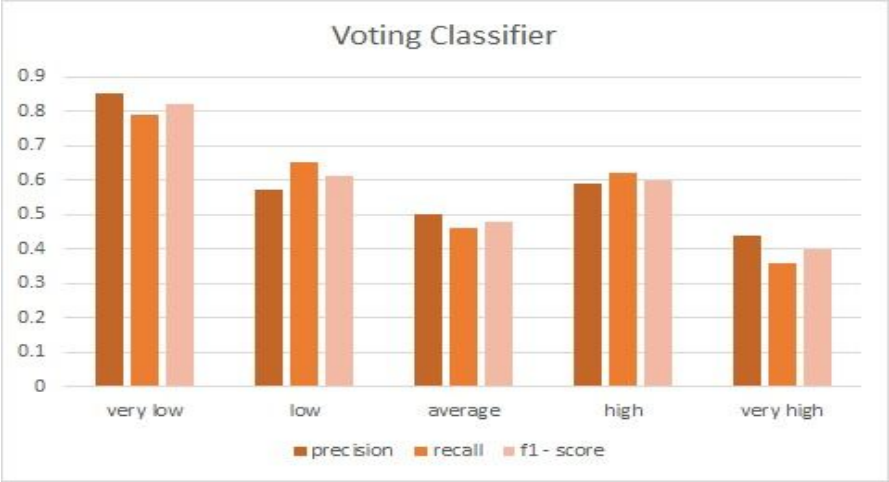
1. To take care of the imbalance in the dataset.
2. Keeping in mind the fact that a higher bid can still fetch us the song (though at a higher price)

CONCLUSION:

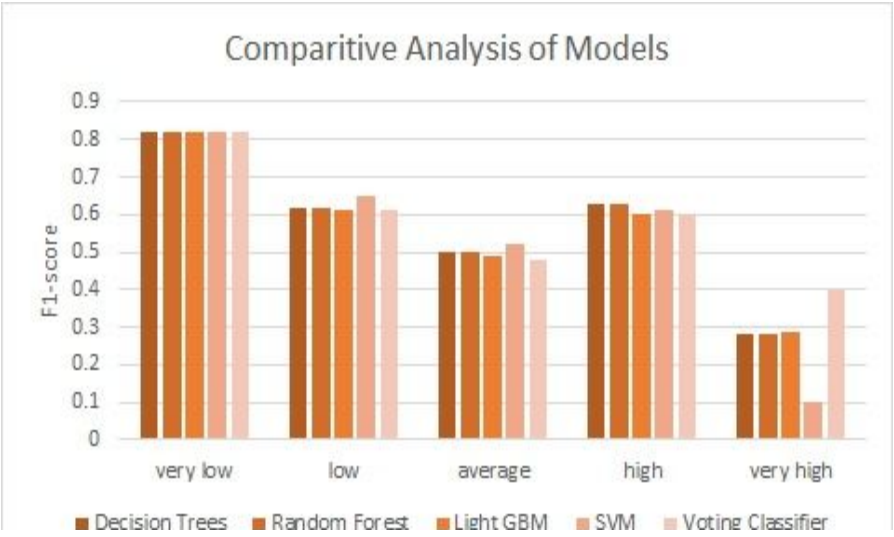
Our ensemble model returned desirable results, with the revenue reaching a higher figure than those obtained in any of the individual models.
Revenue obtained was $8,760 while $126 was being left over (both in 10k Units).

Since there is no use of remaining money, we used that to increase the last 126 values predicted as 1,2 or 3 by 1 each. We tried other combinations too but based on our confusion matrix of the validation set and trial and error, this would help in generating maximum additional revenue. (it went up to $9,136) While this possibly decreases the accuracy, our total revenue will definitely not go down, even on the unseen data!

Here are the results:



What's particularly encouraging is the fact that the F1 score as well as other metrics of the "very high" class, which was the cause of the imbalance, has improved significantly



At the same time, the model results aren't extremely accurate which ensures that it isn't prone to overfitting on unseen data. Overall, the model result was satisfactory and we have used the same technique to make predictions on the test set.

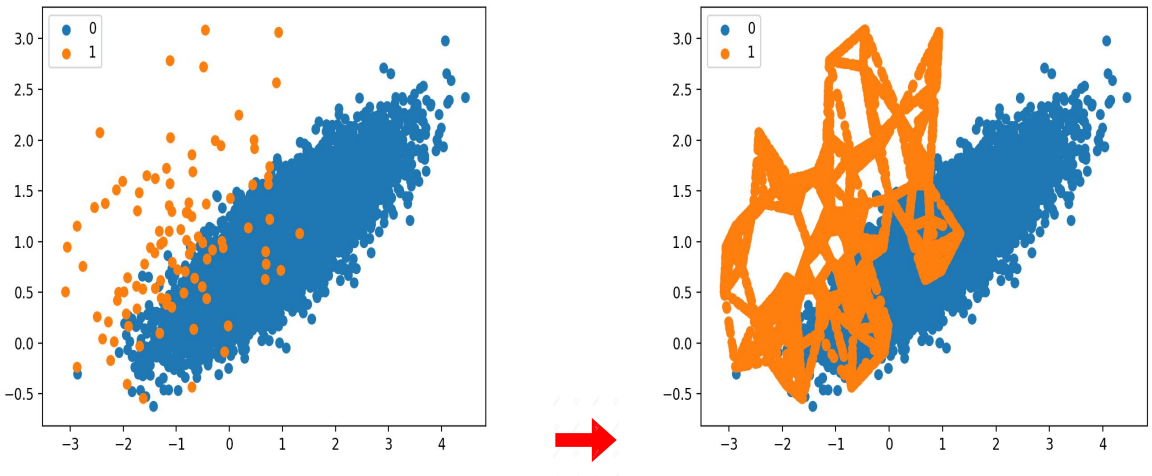
ANNEXURE

Annexure A.1 (Balancing Dataset using SMOTE):

Class imbalance is a common problem associated with many classification problems. It is a scenario that arises when we have an unequal distribution of class in a dataset i.e. the no. of data points in a particular class (or classes) is very large compared to that of some other class (or classes).
In such cases where class distribution is skewed, the accuracy metric is biased and not preferable. F1 score is a commonly chosen metric instead.

Resampling data is the most commonly preferred approach to deal with imbalanced dataset. There are two types of methods for this i) Undersampling ii) Oversampling.
In most cases, oversampling is preferred over undersampling techniques since undersampling removes instances from data that may be carrying some important information. Also, oversampling is an apt choice in our case since only one class is an overwhelming minority.

SMOTE (Synthetic Minority Oversampling Technique) is an oversampling technique where the synthetic samples are generated for the minority class. This algorithm helps to overcome the overfitting problem posed by random oversampling. It focuses on the feature space to generate new instances with the help of interpolation between the positive instances that lie together.



Imbalanced Binary Data

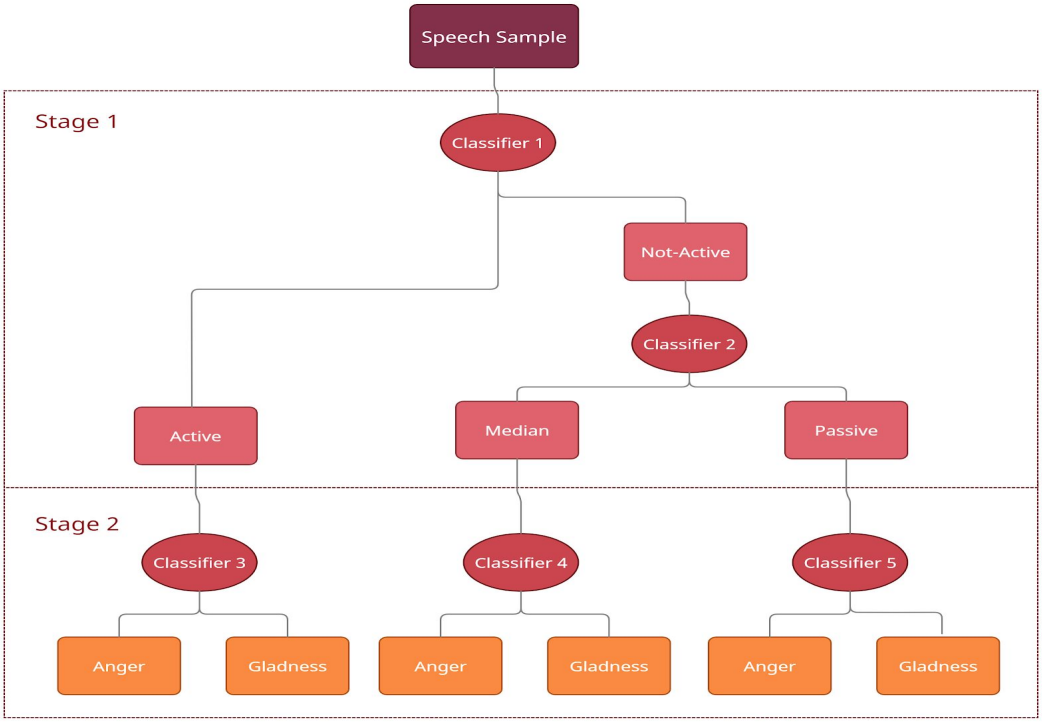
After transformation by SMOTE

Annexure A.2 (Using 2 Stage Classifiers to deal with Imbalance):

Another way of approaching the problem of imbalanced dataset is to use two-stage classifiers. As we had observed for the individual classifiers, the class “very high” was consistently being misclassified to a higher degree.

One way to tackle it is to cluster all the other classes as a superclass named A and leave the minority class as it is. The first stage of this classifier will basically group the observations as belonging to the class “very high” or not belonging to it. This is a proven technique, since the characteristics of a class which is in extreme minority (or majority) are also expected to differ by a great degree.

In the second stage, the task is to disintegrate the superclass into its component original classes and thus get the final predictions.

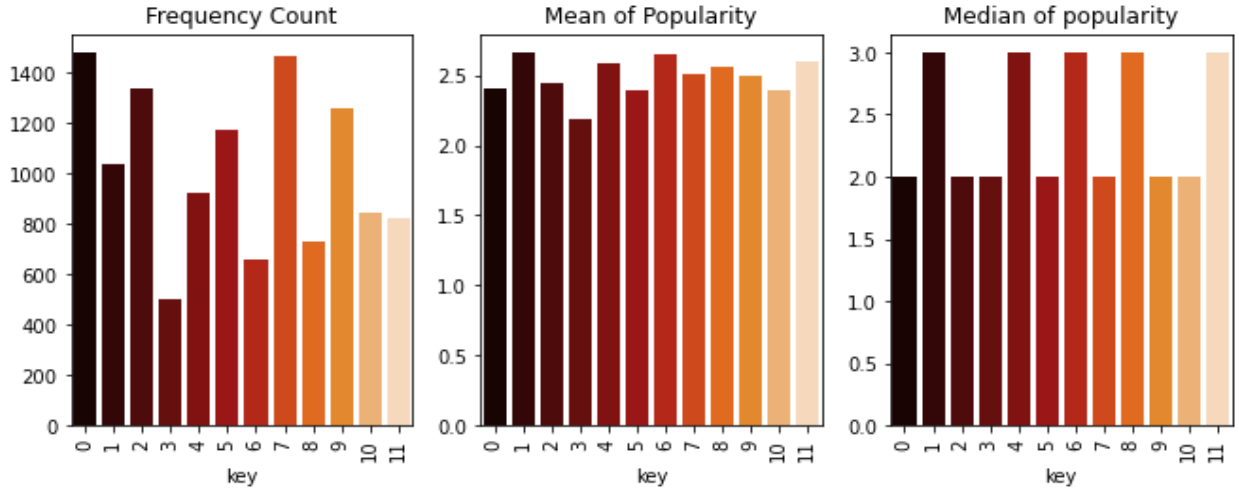


An Example of a Multi-Stage Classifier

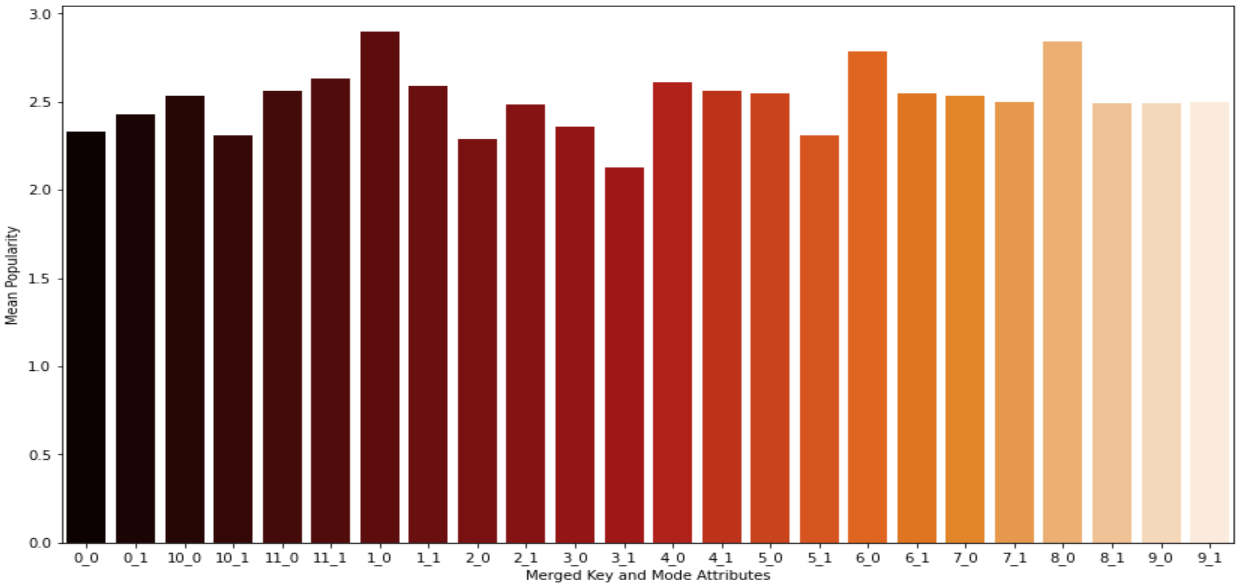
We had tried this method as well during our training phase.
However, the results obtained using SMOTE technique were more satisfactory compared to this. Hence, we chose to use SMOTE only for our purpose.

Annexure B (Further Graphical Analysis):

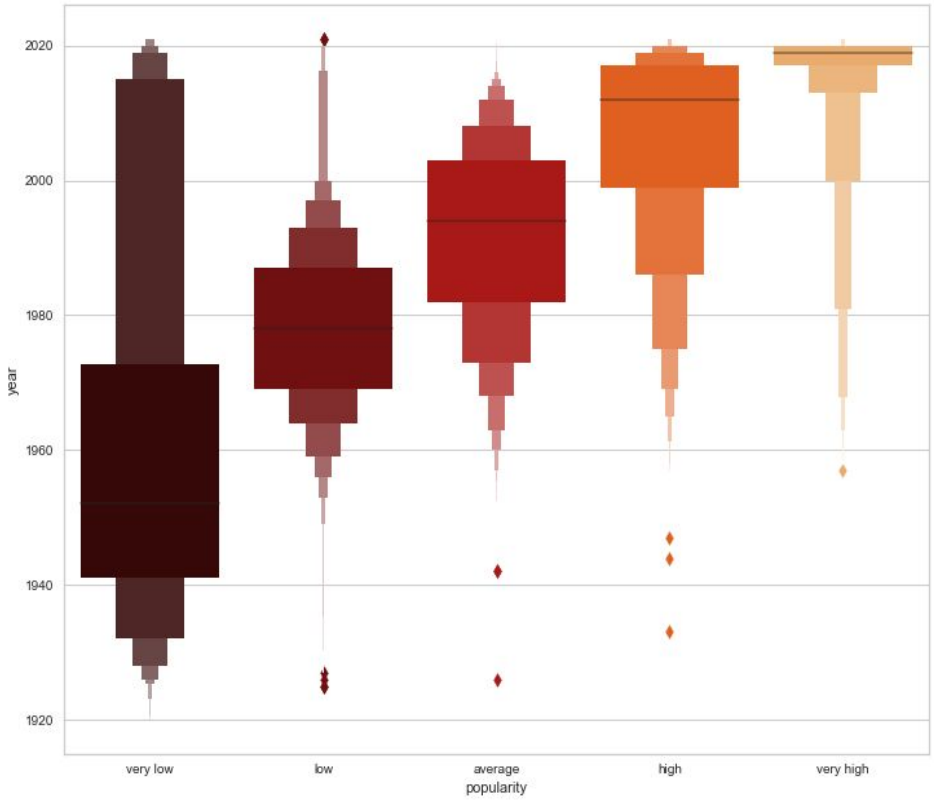
Key and Mode:



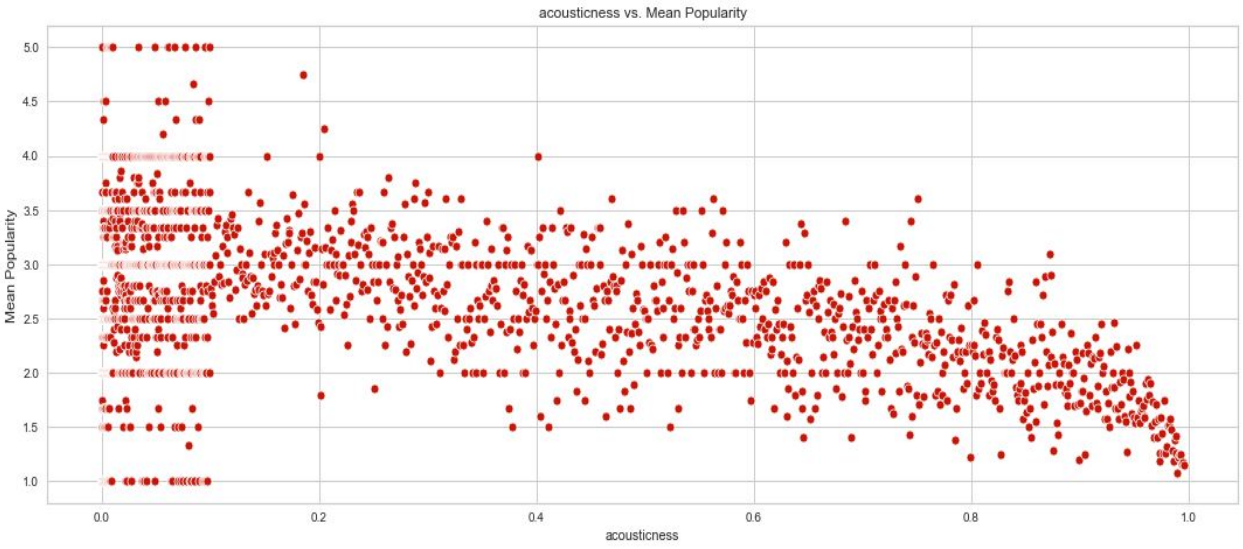
There is not much variation in the popularity of tracks with respect to each note. We tried to encode the key note by replacing it with their median value, thereby creating a binary feature, but this did not improve the performance of the model and was rejected.



Since only the key feature did not add any significant value to our data, we tried merging the Mode and Key attributes to see if they carry any meaningful patterns. Looking at the granular level, there wasn't much information that could be extracted from their combinations and the normalized distribution did not vary much either.



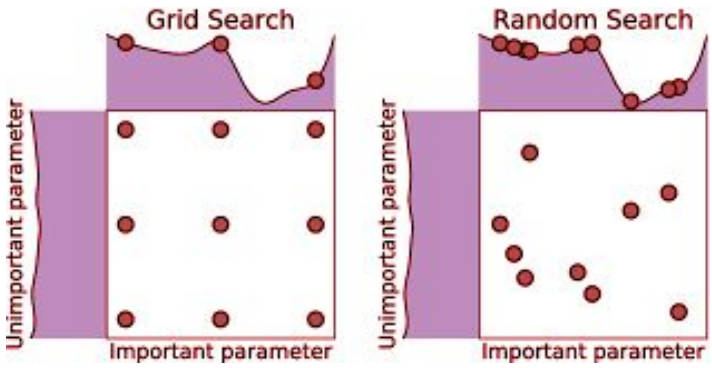
The Box plot of **popularity vs year** clearly shows that mean value of popularity tends to be higher in the recent years.



Acousticness vs mean popularity - shows a soft trend of decreasing acousticness with higher popularity, but the trend is not very steep. We already know that acousticness is one of the features with negative correlation to target variable.

Annexure C.1 (GridSearch Parameters):

A machine learning model has multiple parameters that are not trained by the training set. These parameters control the accuracy of the model. These parameters, referred to as the hyperparameters are particularly important in data science projects as they directly influence the model performance.



Grid search is a tuning technique that attempts to compute the optimum values of hyperparameters. It is an exhaustive search that is performed on the specific parameter values of a model. The model is also known as an estimator.

Here, we present the parameters returned by grid search on the models we have trained:

Estimator	Parameters
Random Forest Classifier	n_estimators=500,min_samples_leaf=5, max_features=sqrt,c lass_weight = {0:3, 1:3, 2:4, 3:7, 4:38}
Support Vector Classifier	C=100,kernel="rbf",gamma =0.1, probability = True , class_weight={0:4, 1:4, 2:4, 3:5, 4:25}
Light GBM	lambda_l1=1.5, lambda_l2= 0, min_data_in_leaf = 30, num_leaves= 31, reg_alpha=0.1
Decision Trees	ccp_alpha=0.001, criterion = 'entropy', max_depth = 10, max_features = 0.9, min_samples_leaf = 6, min_samples_split = 15

Annexure C.2 (Intuition behind class_weights for RMF):

A simple technique for modifying a decision tree for imbalanced classification is to change the weight that each class has when calculating the "impurity" score of a chosen split point.
Impurity measures how mixed the groups of samples are for a given split in the training dataset and is typically measured with Gini or entropy. The calculation can be biased so that a mixture in favor of the minority class is favored, allowing some false positives for the majority class.

This modification of random forest is referred to as Weighted Random Forest and can be achieved by setting the class_weight argument on the RandomForestClassifier class. This argument takes a dictionary with a mapping of each class value (e.g. 0 and 1) to the weighting. The argument value of 'balanced' can be provided to automatically use the inverse weighting from the training dataset, giving focus to the minority class.

From the confusion matrices, we observed that even after using the 'balanced' function, very few observations were being labelled as 'very high'. This is a concern for us, as these are the songs which will generate highest revenue. So, we intentionally put more weight on the classes of higher popularity. Keeping in mind the goal of maximum revenue, we even used higher weight for class "high" than lower ones even though their distribution was somewhat similar

The final chosen weights were: class_weight = {0:4, 1:4, 2:4, 3:7, 4:35}