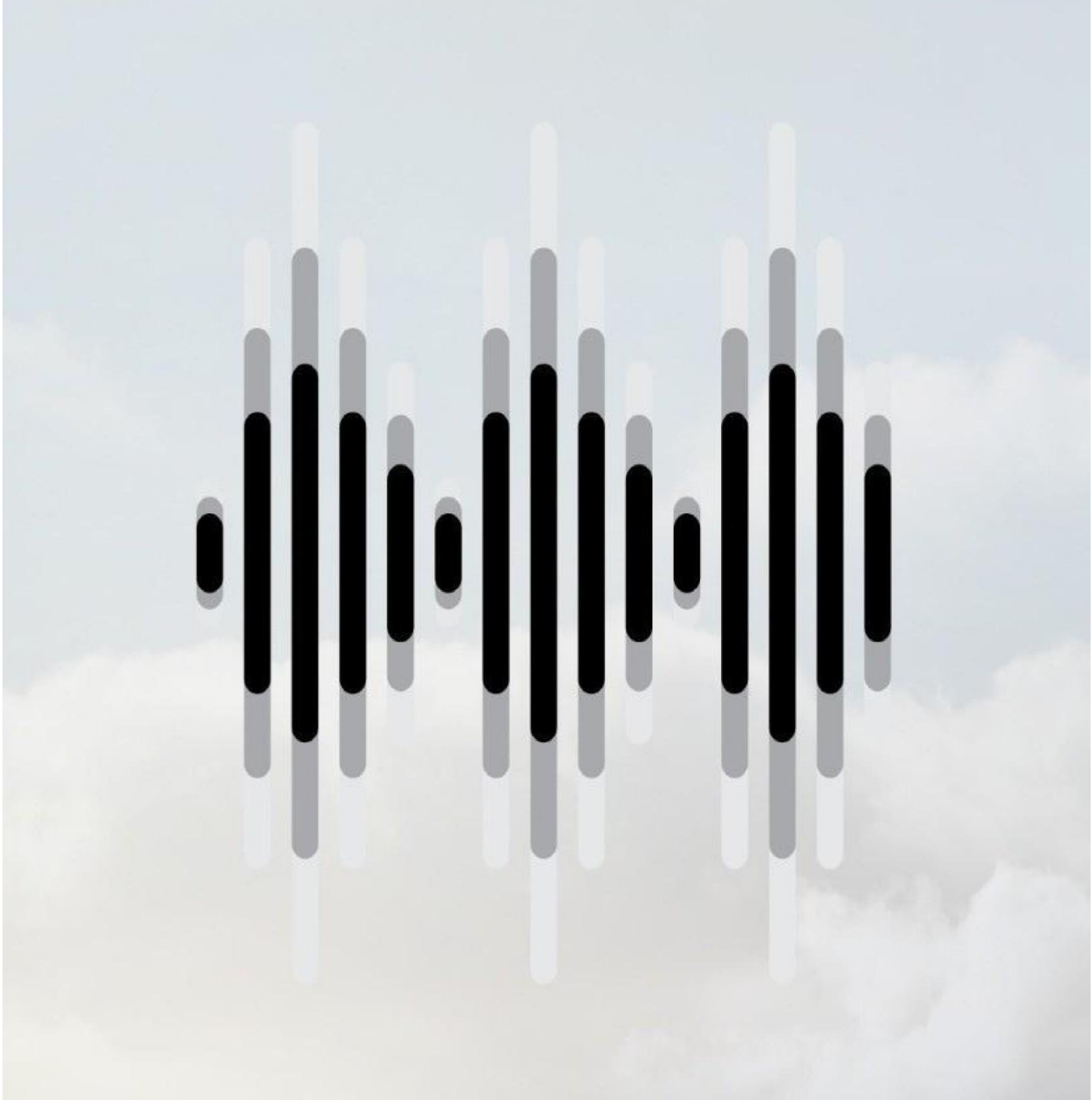


DATA ANALYTICS

CONTENTS

1. PROBLEM STATEMENT:

A major record label wants to purchase the rights to a music track. It does not want to encounter any losses with promotion and distribution of the track. It needs to decide on the royalties to be paid to the artists and composers

Objective: You need to predict the popularity of the music tracks based on the features provided in the dataset.

The target variable, "popularity", has 5 categories: 'Very high', 'high', 'average', 'low', 'very low'. The order is in decreasing popularity. For each category, there is initial bid price (for royalties to be paid) and expected revenue collections(in 10k $) as follows:

.

POPULARITY	BID PRICE	EXPECTED REVENUE (in 10k $)
Very low	5	10
Low	4	8
Average	3	6
High	2	4
Very High	1	2

A bid will be valid only when, either the predicted popularity matches the actual popularity or predicted popularity is higher than actual popularity. However, in the second case, the revenue generated will be equal to the expected revenue corresponding to the actual popularity class.

2. LITERATURE REVIEW:

When the characteristics of an entity are converted into some measurable form, they are called features. The performance of a predictive model is heavily dependent on the quality of the features in the dataset used to train that model. Our literature survey led us to the following conclusions - key is a subset of instrumentalness. Explicit is subset of speechiness since, speechiness focuses on lyrics and explicit specifically on abusive lyrics. Valence is a function of loudness and mode.

3. DATASET:

In training dataset:
Total no. of rows : 12227
Total no. of columns i: 17
Number of categorical variables : 4
Number of null values : 0 in all the columns
Train-test split ratio : 80:20
Outliers: no outliers were removed
Duplicate values : no duplicate values

4. UNDERSTANDING OF FEATURES:

Feature Description:

- **Acousticness.** Value representing the probability that a track was created using acoustic instruments, including voice. Float; range, 0–1.
- **Danceability.** A track's "foot-tapping" quality, based on tempo, rhythm stability, beat strength, and isochrony. Float; range, 0–1.
- **Energy.** A perceptual estimation of frenetic activity throughout a track. High-Energy tracks have increased entropy, and tend to feel fast, loud, and noisy (e.g., Death Metal). Float; range, 0–1.
- **Explicit.** The explicit filter tells whether a music contains common explicit words and phrases or no
- **Instrumentalness.** Value representing the probability that a track was created using only instrumental sounds, as opposed to speech and/or singing. Float; range, 0–1.

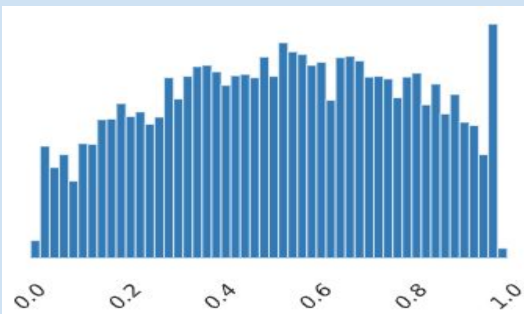
- **Key.** Key, in music, a system of functionally related chords deriving from the major and minor scales, with a central note, called the tonic (or keynote). **Liveness.** Value representing the probability that a track was recorded in the presence of an audience rather than in a studio. Float; range, 0–1.
- **Loudness.** The average loudness of a track in decibels. Loudness is the psychological correlate of signal amplitude.
- **Mode.** The term 'mode' has always been used to designate classes of melodies. The seven main categories of mode have been part of musical notation since the middle ages.So, the list goes: Ionian, Dorian, Phrygian, Lydian, Mixolydian, Aeolian and Locrian. Some of them are major modes, some are minor, and some are ambiguous. Some modes are sadder or holier than others.
- **Speechiness.** Value representing the presence of spoken words in a track, e.g., talk show, audio book, poetry, rap. Float; range, 0–1.
- **Tempo.** The estimated tempo of a track in beats per minute. Float; range, 0–294.
- **Valence.** A perceptual estimation of a track's positive/negative affect, e.g., happy and cheerful, or sad and depressed. Float; range, 0–1.
- **Duration.** The duration of a track in seconds as calculated by the Spotify analyzer. Float; maximum value, 6,060 s.

Statistics and Correlation

Valence

Distinct	1256
Distinct (%)	10.3%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	0.5253000728

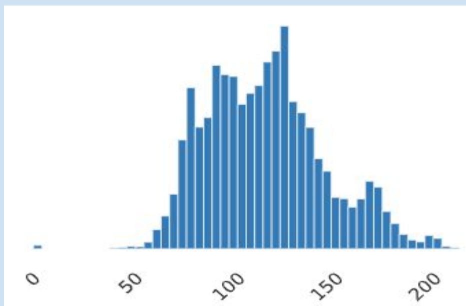
Minimum	0
Maximum	1
Zeros	17
Zeros (%)	0.1%
Negative	0
Negative (%)	0.0%
Memory size	95.6 KiB



Tempo

Distinct	11264
Distinct (%)	92.1%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	118.1674949

Minimum	0
Maximum	216.843
Zeros	13
Zeros (%)	0.1%
Negative	0
Negative (%)	0.0%
Memory size	95.6 KiB

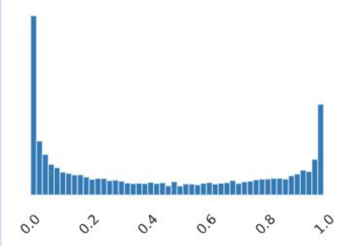


Statistics and Correlation:

Acousticness

Distinct	2714
Distinct (%)	22.2%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	0.4305783602

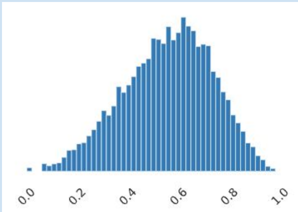
Minimum	1.04×10^{6}
Maximum	0.996
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	95.6 KiB



Danceability

Distinct	898
Distinct (%)	7.3%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	0.556352654

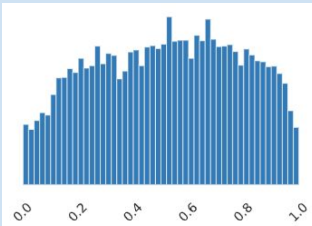
Minimum	0
Maximum	0.98
Zeros	13
Zeros (%)	0.1%
Negative	0
Negative (%)	0.0%
Memory size	95.6 KiB



Energy

Distinct	1396
Distinct (%)	11.4%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	0.5221287124

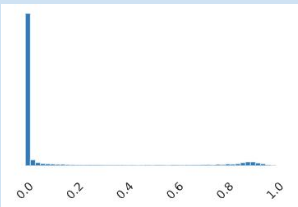
Minimum	2.03×10^{5}
Maximum	1
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	95.6 KiB



Instrumentalness

Distinct	3658
Distinct (%)	29.9%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	0.1493205587

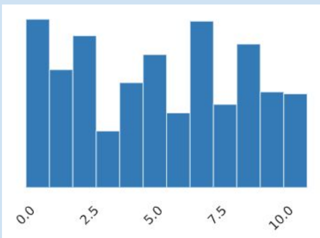
Minimum	0
Maximum	1
Zeros	3602
Zeros (%)	29.5%
Negative	0
Negative (%)	0.0%
Memory size	95.6 KiB



Key

Distinct	12
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	5.205201603

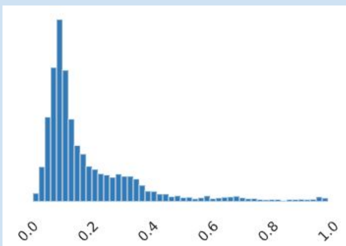
Minimum	0
Maximum	11
Zeros	1481
Zeros (%)	12.1%
Negative	0
Negative (%)	0.0%
Memory size	95.6 KiB



Liveness

Distinct	1477
Distinct (%)	12.1%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	0.201364562

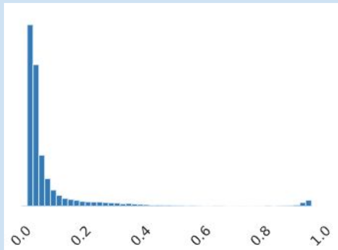
Minimum	0.0147
Maximum	0.997
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	95.6 KiB



Speechiness

Distinct	1275
Distinct (%)	10.4%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	0.09767980698

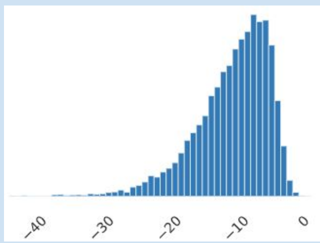
Minimum	0
Maximum	0.968
Zeros	13
Zeros (%)	0.1%
Negative	0
Negative (%)	0.0%
Memory size	95.6 KiB

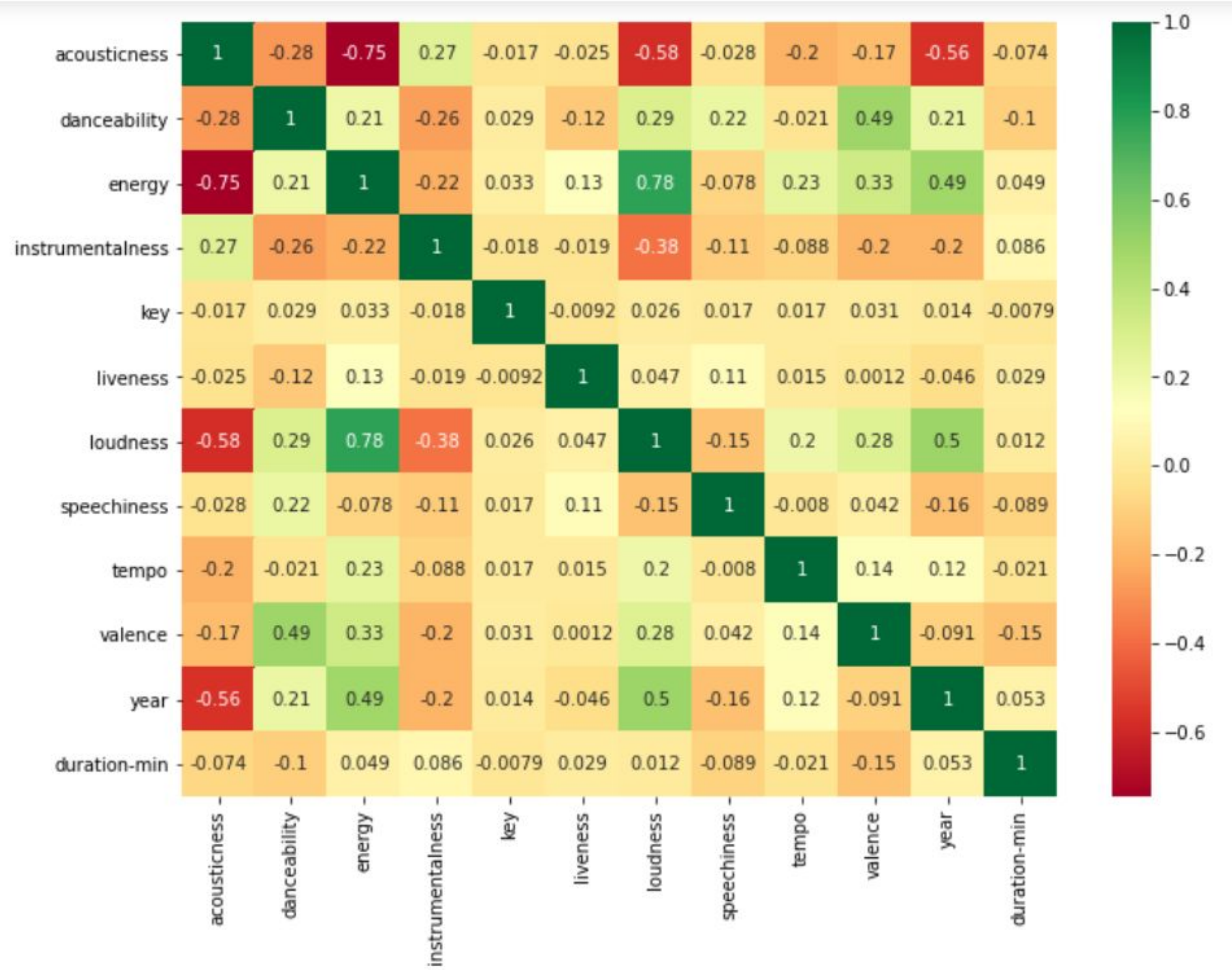


Loudness

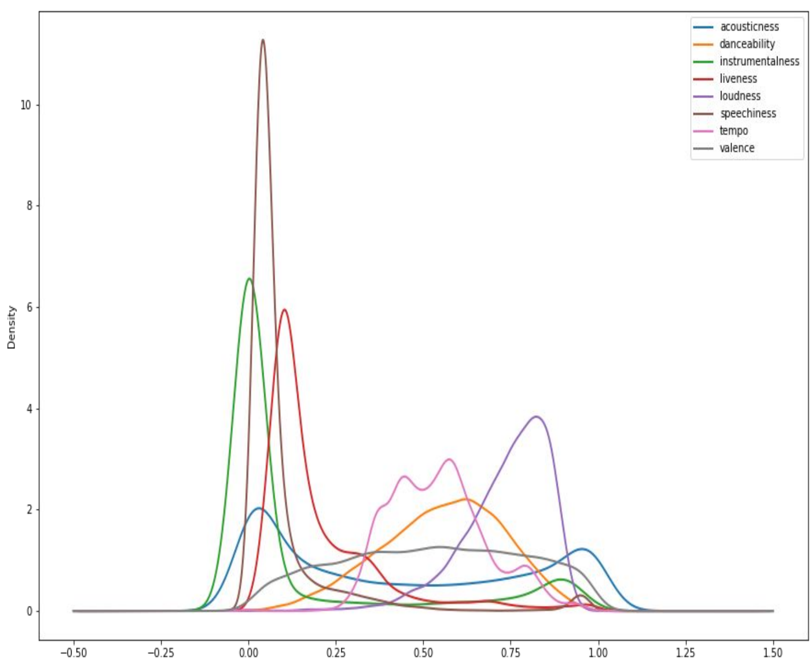
Distinct	8718
Distinct (%)	71.3%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	-10.66868651

Minimum	-43.738
Maximum	1.006
Zeros	0
Zeros (%)	0.0%
Negative	12226
Negative (%)	> 99.9%
Memory size	95.6 KiB





5. EXPLORATORY DATA ANALYSIS

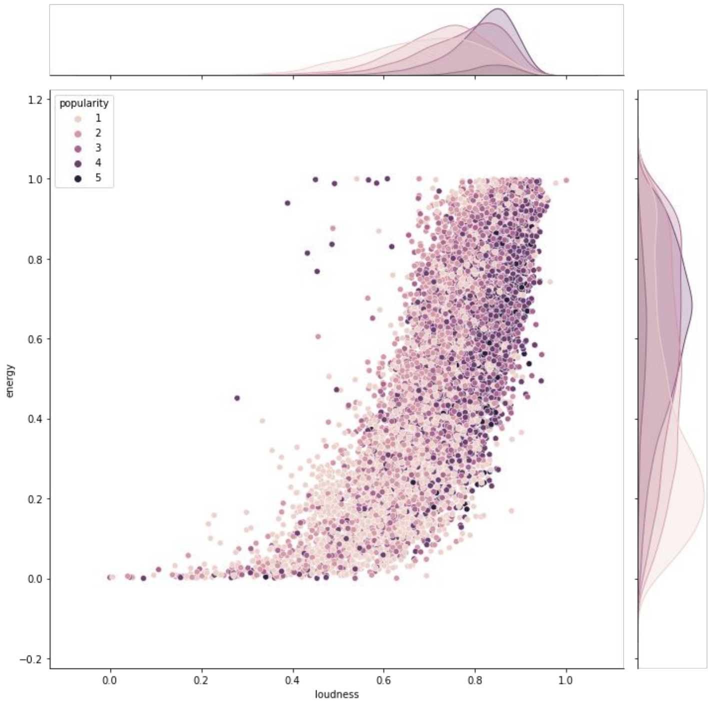


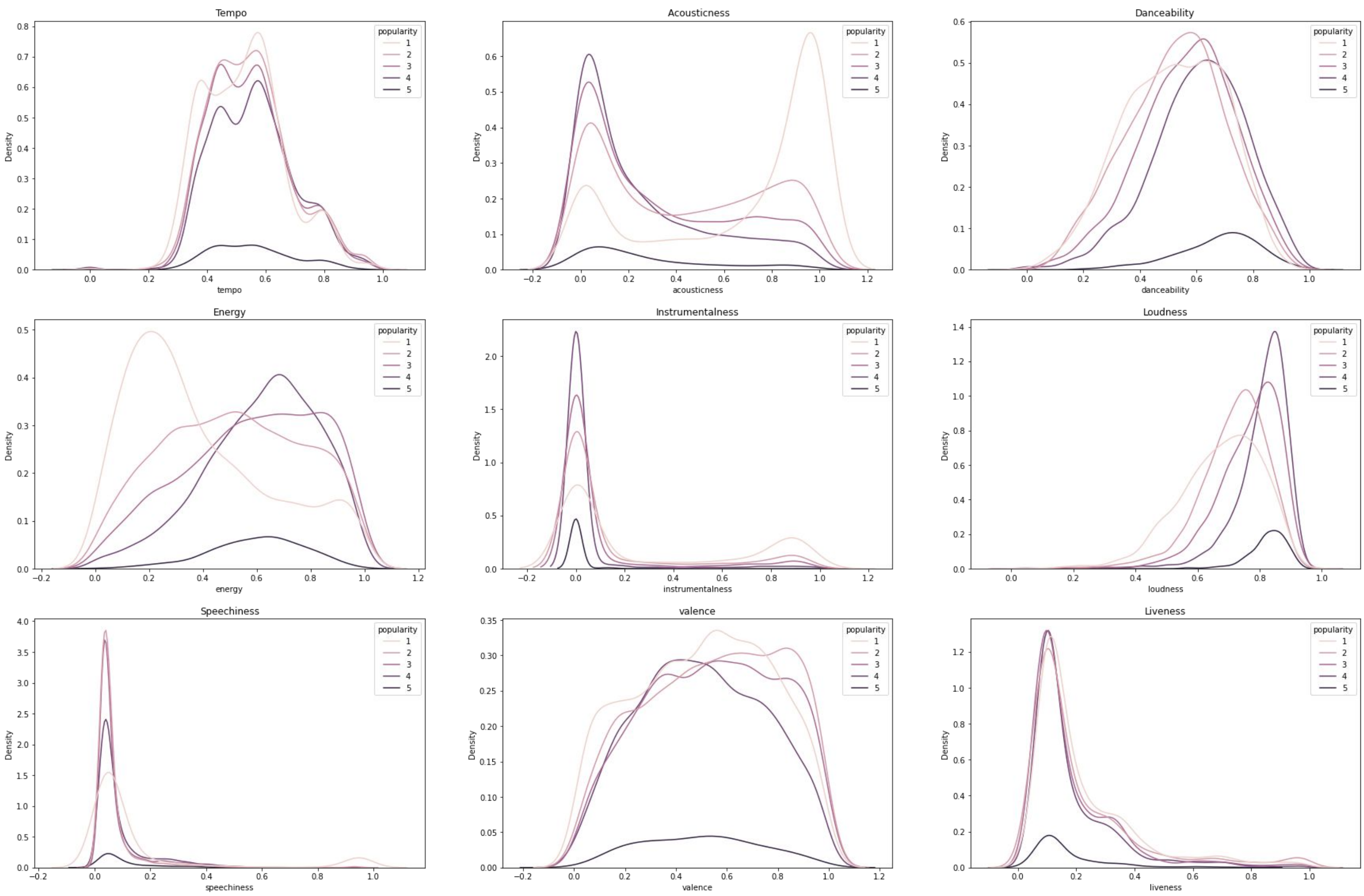
Plot showing the density of different factors wrt to its presence ratio in the song.
Here some key factors to note is

1. Speechiness has its pick in a range btw 0-0.25 that means maximum songs lie in this region,also we can say songs with speechiness <0.33 have nearly no lyrics and more music
2. The peak for loudness and tempo lies in the range 0.5-1

Here loudness is plotted in the X axis and energy in Yaxis and the distribution curve of both has been displayed wrt to popularity.(Note : the densest colour represents highest popularity)

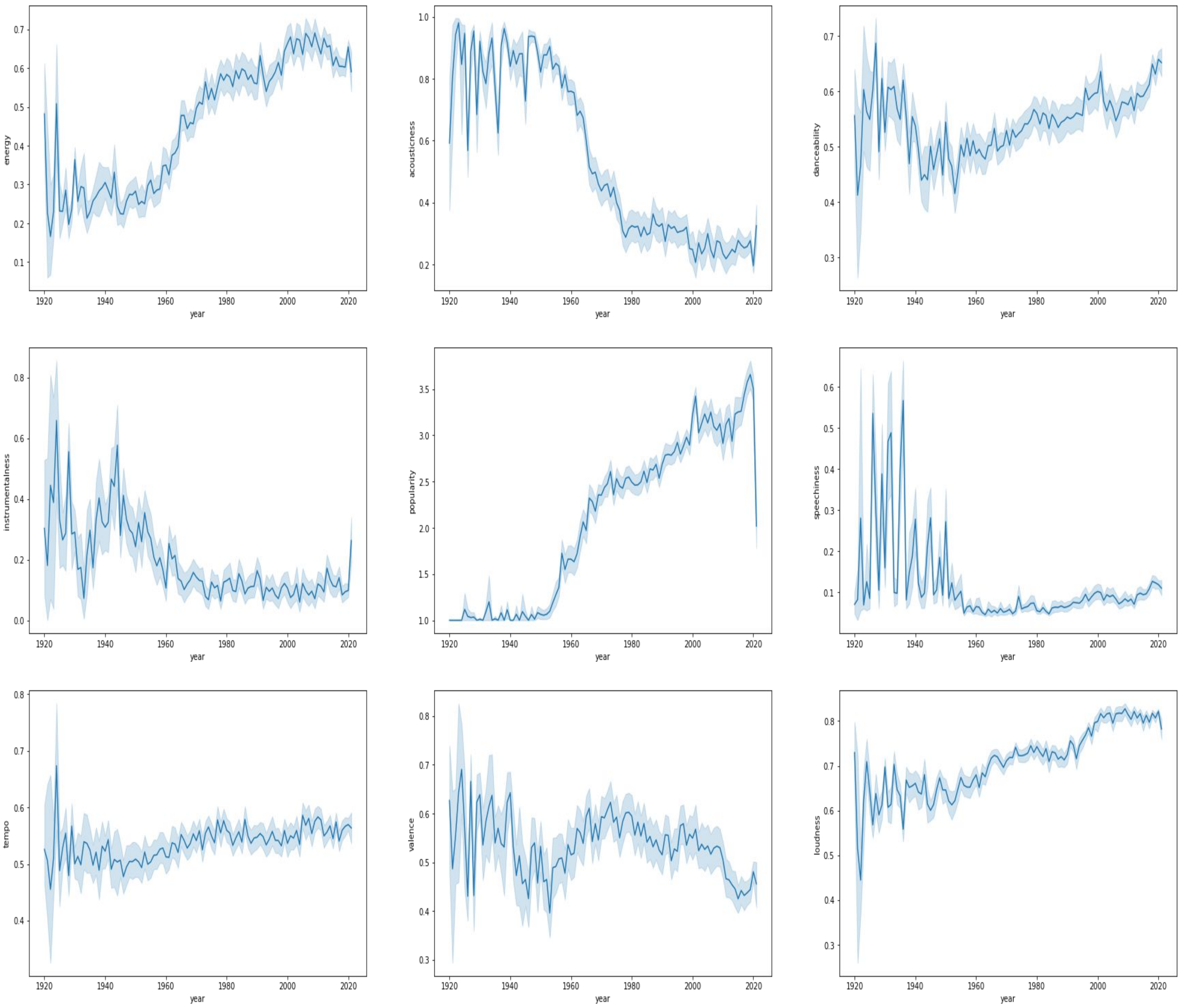
1. We can interpret from the graph that when it comes to popularity,energy is more of a normal distribution whereas loudness is rightly skewed
2. We can also notice that there is a positive linear correlation between energy and loudness





Some prominent graphs to note from above are with respect to ‘high popularity’ curve

1. It can be seen on the graph the distributions of variables speechiness, acousticness liveness,and instrumentalness are left-skewed with valued tending to be closer to 0.
2. danceability ,loudness and energy are right skewed whereas valence follows nearly normal distribution.



1. Since 1960 we can see a surge in the intensity of energy in songs
2. The steep decline in acousticness shows the YoY increase in demand for electronic music in comparison to acoustics
3. We can see from the graph the YoY,musicians have moved to increased loudness in their tracks.
4. Previously songs used to contain both words and music, eventually now as the score of speechiness can be observed <0.33 ,we can say that songs have relatively more music .

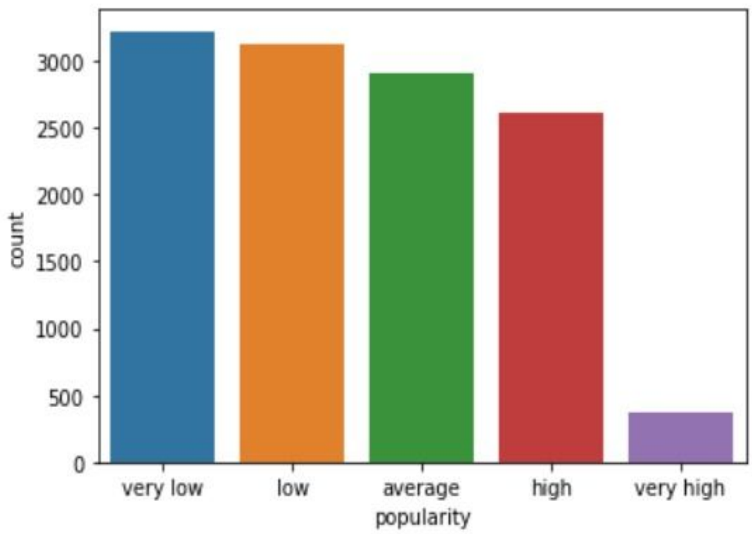
6. ANALYSIS

- **FEATURE ENGINEERING :**

- It is important that we pass scaled values to SVM and Neural Networks
- Since scaling would not make any sense in the case of 'Year' feature, we have binned the years (from 1920-2020) in buckets of 10 years and replaced the 'Year' feature by 'Decade' feature.
- As an example, the years between 1920-1930 have been replaced by Decade 0.
- We have been given tempo which is expressed as beats per min and we have been given duration of songs in minutes. A new feature has been created by multiplying these features
- Total beats = Tempo * Duration

- **OVERSAMPLING:**

As we can observe from the histogram below, the classes are imbalanced. To deal with this class imbalance, we have made use of Oversampling (by SMOTE) for our models.



very low	3222
low	3118
average	2912
high	2606
very high	369

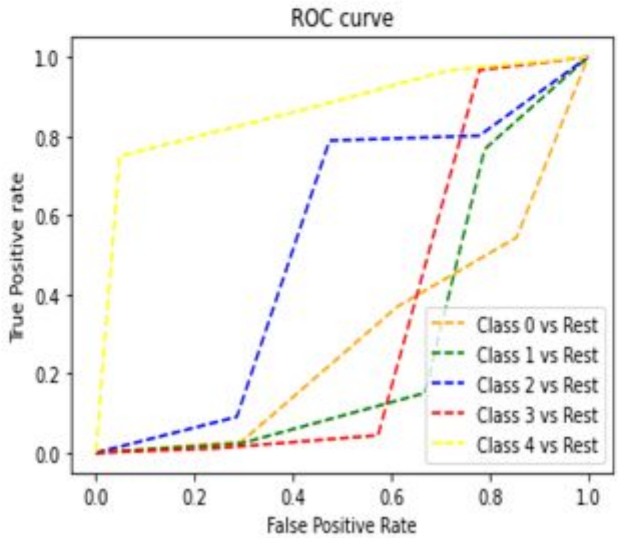
ANALYSIS

• MODELS - SUPPORT VECTOR CLASSIFICATION

SVM is a supervised machine learning algorithm which uses a technique called the kernel trick to transform your data and then based on these transformations it finds an optimal boundary between the possible outputs. It is a memory efficient algorithm.

Baseline Accuracy

- Features removed - ID and Release Data
- Trained with default parameters
- Kernel used – Linear
- Cross validation accuracy – 54.62%



After Applying SMOTE

- Features removed – ID, Release date
- Trained with default parameters
- Kernel used – Polynomial
- Cross validation accuracy – 57%

After applying SMOTE, Hyperparameter tuning, using pipeline

- Features removed – ID, Release date, Key, tempo
- Added new feature : Total Beats (Tempo × Duration)
- Using scalar transformation
- Cross Validation Accuracy – 64%

Confusion Matrix, Recall, Precision and F1 Score

```
[[278 192 182   0  39]
 [118 453  70   0  15]
 [173  19 444   0 169]
 [  2  85   3   0   1]
 [ 13 183  41   0 577]]
              precision    recall  f1-score   support

           0       0.48      0.40      0.44       691
           1       0.49      0.69      0.57       656
           2       0.60      0.55      0.57       805
           3       0.00      0.00      0.00        91
           4       0.72      0.71      0.71       814
```

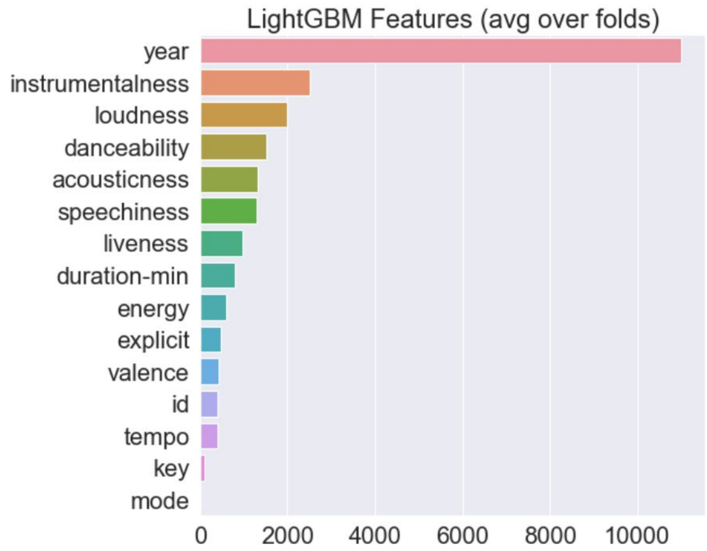
MODELS - Light Gradient Boosting Machine (LightGBM)

Light GBM is a fast, distributed, high-performance gradient boosting framework based on decision tree algorithm, used for ranking, classification and many other machine learning tasks.Since it is based on decision tree algorithms, it splits the tree leaf wise with the best fit whereas other boosting algorithms split the tree depth wise or level wise rather than leaf-wise

.

Baseline Accuracy :

- Removed ID and Release Date
- Trained with default parameter
- boosting_type : gdbt
- Cross Validation Accuracy :62.00%



After applying SMOTE:

- Removed ID and Release Date
- Trained with default parameter
- boosting_type : dart
- Cross Validation Accuracy : 62.09%

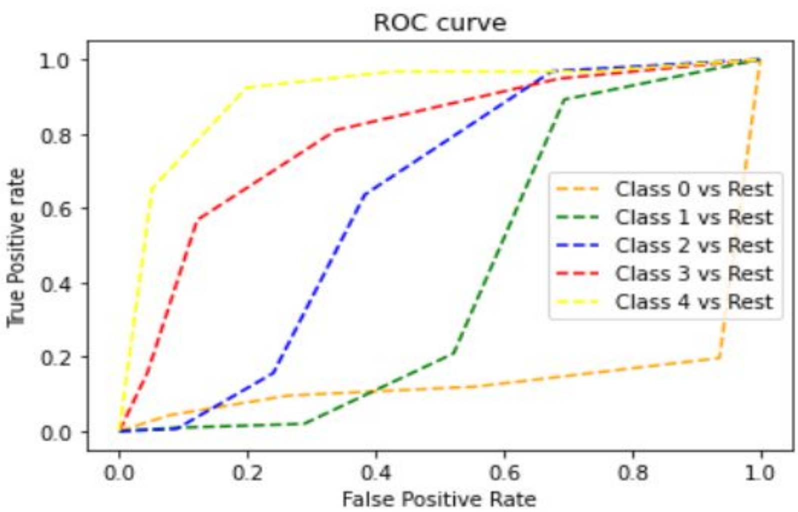
After applying SMOTE, feature selection and hyperparameter tuning:

- Removed ID, Release Date, explicit, mode, key
- boosting_type : dart
- Cross Validation Accuracy : 64.66%

lr	0.01
early_stop	30
num_leaves	15
lambda_l1	1.0
lambda_l2	1.0
num_boost_rounds	2000

confusion matrix

```
[[638  61  19  42  34]
 [ 83 526 146   8   7]
 [ 23 240 347 109   4]
 [ 35  94 164 278 107]
 [  3   0   4  25  60]]
```



	precision	recall	f1 score
0	0.816	0.80	0.810
1	0.571	0.683	0.622
2	0.510	0.480	0.495
3	0.602	0.410	0.488
4	0.283	0.652	0.604

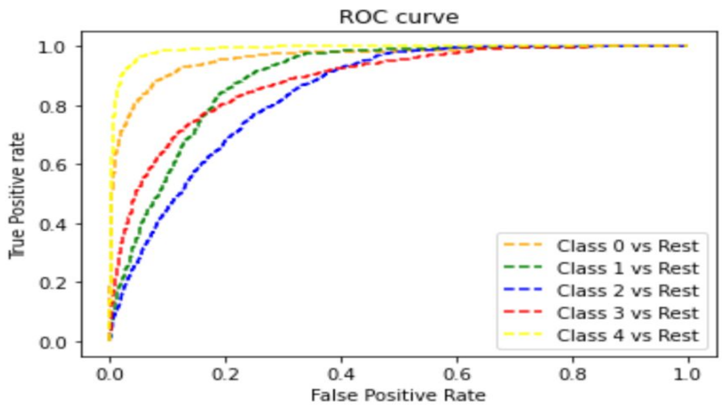
MODELS - eXtreme Gradient Boosting Machine (XGBOOST)

It is an efficient and scalable implementation of a gradient boosting framework. The package includes an efficient linear model solver and tree learning algorithm. It supports various objective functions, including regression, classification, and ranking. It has several features:

1. XGBoost can automatically do parallel computation on Windows and Linux, with OpenMP
2. XGBoost accepts sparse input for both tree booster and linear booster and is optimized for sparse input.
3. xgboost supports customized objective function and evaluation function

Baseline Accuracy :

- Removed ID and Release Date
- Trained with default parameter
- Accuracy : **62.75 %**



After applying SMOTE:

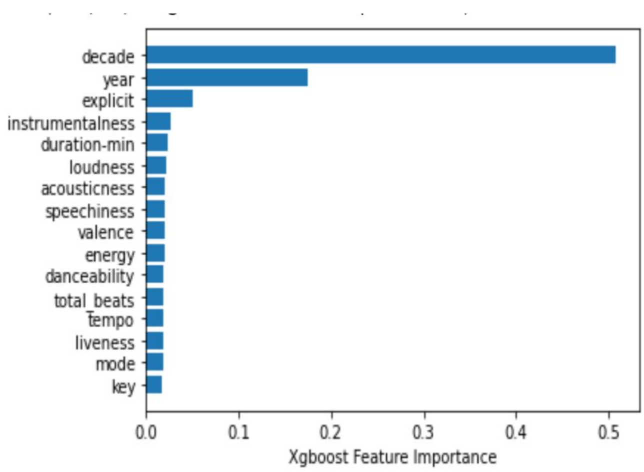
- Removed ID and Release Date
- Trained with default parameter
- Accuracy : 63.47 %

After applying SMOTE, feature selection and hyperparameter tuning:

- Removed ID, Release Date, explicit, mode, key
- added new features like "total_beats" and "decade".
- Tuned with learning_rate, n_estimators,max_depth, min_child_weight, colsample_bytree, gamma
- Accuracy : 69.86 %

{"Total_beats" = "min-duration" * "tempo"
And we obtained "decade" by binning "year"}

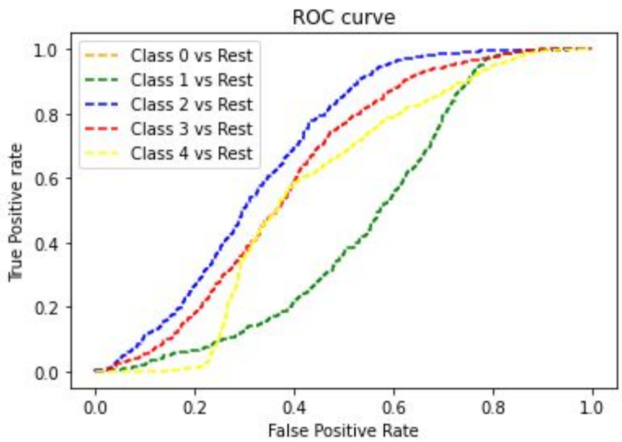
colsample_bytree	gamma	learning_rate	max_depth	min_child_weight	n_estimators
0.7	0.3	0.15	15	1	100



	precision	recall	f1-score
0	0.817	0.802	0.809
1	0.542	0.658	0.594
2	0.523	0.501	0.512
3	0.683	0.534	0.599
4	0.869	0.936	0.901
Accuracy			0.686
Macro Avg	0.687	0.686	0.683

- **MODELLS - Adaptive Gradient Boosting (ADABoost)**

AdaBoost algorithm, is a Boosting technique that is used as an Ensemble Method. Adaboost helps us combine multiple “weak classifiers” into a single “strong classifier”. The weak learners in AdaBoost are decision trees with a single split, called decision stumps.AdaBoost works by putting more weight on difficult to classify instances and less on those already handled well.
Features Used: acousticness, danceability, energy, explicit,, mode, Instrumentalness, liveness, loudness, speechiness, Tempo, valence, year, duration-min ,popularity



Baseline Accuracy:
- Features removed: ID and release_date
- The dataset was trained on default parameters.
- Accuracy: 59%

After Applying Smote:
- Features removed: ID and release_date
- The dataset was trained on default parameters.
- Accuracy: 57.4%

After applying SMOTE, feature selection and hyperparameter tuning:
- Features removed ID, release_date, explicit, mode, key
- Tuned learning_rate, n_estimators
- Added a new feature: total_beats
- Accuracy : 68.6%

Confusion Matrix

```
[[530  67  22  51   9]
 [ 68 408 137  12   0]
 [ 17 190 309  94   0]
 [ 11  64 117 401  55]
 [  1   3   7  25 624]]
```

Hyperparameter Tuning

n_estimators	learning_rate	Accuracy
500	0.1	60.88%
100	0.3	60.57%
100	0.4	61.01%
100	0.5	68.60%

Precision , Recall and F1 Score

	precision	recall	f1-score
1	0.845	0.781	0.812
2	0.557	0.653	0.601
3	0.522	0.507	0.514
4	0.688	0.619	0.652
5	0.907	0.945	0.926
accuracy			0.705
macro avg	0.704	0.701	0.701
weighted avg	0.709	0.705	0.706

MODELS – Random Forest Classifier

Random forests is a supervised learning algorithm. Random forests create decision trees on randomly selected data samples, get prediction from each tree and select the best solution by means of voting.
Mean **accuracy = ~50%(average)**. Features removed: id and release_date.

Handling Imbalanced Classes:
Tools used:
a. **SMOTE oversampling**- to oversample the dataset
b. **Cost-sensitive learning**-to bias towards those classes that have fewer examples in the training dataset

Tools Used	Mean Accuracy
Cost-sensitive learning	~50%
Default SMOTE oversampling	~68%

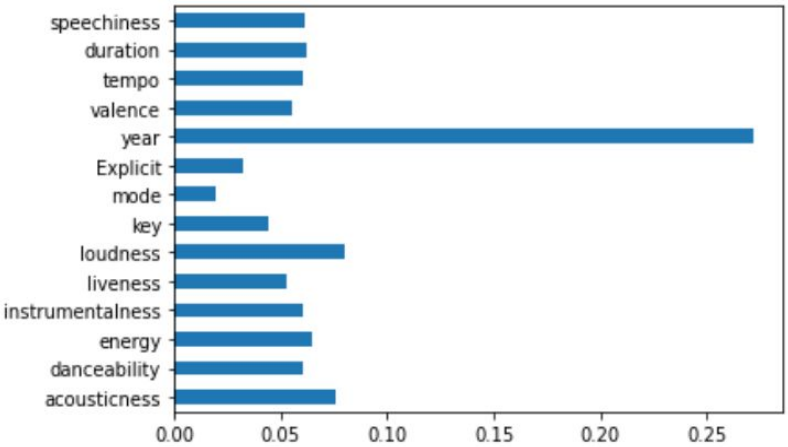
Hyperparamter tuning:

Estimators	Mean Accuracy
100	~68%
500	~70%
500-900	~70%(run time was increasing)

Oversampling strategy:

No. of examples in each class	Mean Accuracy
3700	~72%
4000	~74%
5000	~77%

Feature Importance Graph:



Features removed: key, mode, explicit

Checking for overfitting:
--Oversampling had resulted in overfitting
--**99% accuracy** on the train data
--Hence, applied **repeated stratified k-fold cross validation**
--No. of estimators = 100
--No. of splits=10 and repeats=3
*Further increase in hyperparameters resulted in high increase in run time

Model evaluated on	**Mean Accuracy**
Total data	~79%
Train data	~77.4%

Confusion Matrix, Precision, Recall and f1 score:

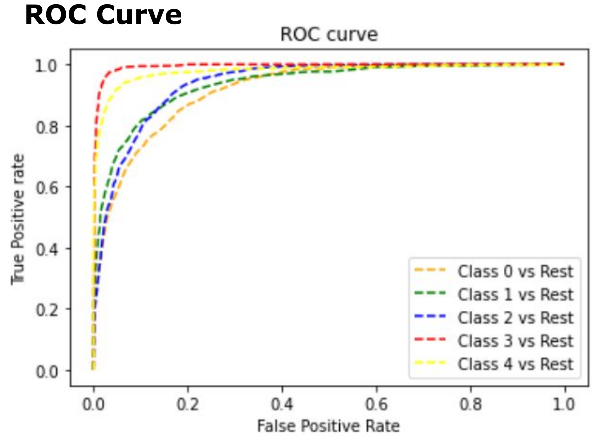
```
[[639 103 203   0  23]
 [147 718  85  66  10]
 [151  10 727   0  69]
 [  8  32   8 991   0]
 [ 28  41  68   8 865]]
```

```
              precision    recall  f1-score

           0      0.657     0.660     0.658
           1      0.794     0.700     0.744
           2      0.666     0.760     0.710
           3      0.931     0.954     0.942
           4      0.895     0.856     0.875

    accuracy                          0.788
   macro avg      0.788     0.786     0.786
weighted avg      0.792     0.788     0.789
```

ROC Curve



MODELS -Neural Network

Features removed : Explicit, Release date, Tempo, Key, Year

Features added : Total Beats (Duration-min * Tempo (Beats/min))

Scaling : Standard Scaler

Oversampling : By SMOTE

Model 1:

Number of hidden layers	3 (5->32->16->8->5)
Activation in hidden layers	relu
Activation in output layer	softmax
Loss function	Categorical cross entropy
optimizer	adam
batch size	5
epochs	2000
Cross validation accuracy	53%

Model 2 :

Number of hidden layers	5 (5->128->64->32->16->8->5)
Activation in hidden layers	relu
Activation in output layer	softmax
Loss function	Categorical cross entropy
optimizer	adam
batch size	5
epochs	2000
Dropout before final layer 5	0.2
Cross validation accuracy	65%

7. OVERCOMING OVERFITTING:

Due to the very nature of how SMOTE works, the models were all prone to overfitting. To minimize overfitting, we extensively tuned the regularization parameters of our models using Grid Search CV. This allowed us to find the best possible combination of parameters for the specific model. GridSearchCV performs a cross validation on each parameter of the grid and calculates the score. It is used to benchmark hyper parameters.

Further, to minimize the effect of overfitting on our results and to get an accurate idea of performance, we applied K-fold cross validation and have reported the cross validation scores throughout our report.

K-fold cross validation, a strong preventive measure against overfitting, was used to partition into k-subsets or “folds”. Then, the algorithm was iteratively trained on k-1 folds while the remaining folds were used as the test set (called the “holdout fold”).

.

8. CONCLUSION:

Random Forest proved to give the best cross validation accuracy. As can be observed from the ROC curve, accuracy on all classes is reasonably good. Neural Network showed promising performance and would certainly benefit from further hyperparameter tuning and training.

SMOTE lead to inherent overfitting which affected the performance of all the algorithms.

9. FUTURE WORK:

- Neural Network would benefit from additional training data.
- Data regarding the artist of the song would almost certainly improve predictive ability of models.
- Alternate methods of dealing with class imbalance could be looked into - such as manually assigning class weights in all the algorithms would eliminate the problem of overfitting.

THE END