OPEN IIT DATA ANALYTICS 2021

Team Number: 39

INDEX

Problem Statement

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

Objective: We 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
Very High	5	10
High	4	8
Average	3	6
Low	2	4
Very Low	1	2

Data Description

Acousticness: Acousticness is a confidence measure of whether the song is acoustic or not. The values range from 0, that signifies less likely to be acoustic to 1 signifying extremely likely to be acoustic.

Danceability: Danceability measures how suitable the song is as a dance song.

Energy: The energy of the song is the measure of intensity of the song.

Explicit: Explicitness is a categorical feature that describes whether the song contains explicit language or not.

Instrumentalness: Instrumentalness measure the extent of vocals in the song. A song with very few vocals would have a high instrumentalness score and vice versa.

Key: Key is a musical term that roughly means the chords of the song. There are 12 possible keys in music and the data assigns an integer value to these keys.
Liveness: Liveness the measure of likelihood the song was performed in front of a crowd. It detects the presence of audience noises in the recording.
Loudness: Measures the perceived loudness of the song. The less negative the numeric value is, the louder it will sound.
Mode: Mode describes the musical scale of the song. It is a categorical feature and categories the music as 'major' or 'minor' scale.
Release Date: This contains the release date of the song in dd-mm-yyyy format.
Speechiness: Speechiness measures the amount of spoken words in the song.
Tempo: Contains the tempo of the song measured in beats per minute
Valence: Valence is an abstract feature that weighs the musical positivity of the track.
Year: Contains the year of release of the song.
Duration-min: The length of the song in minutes
Popularity: Categorical feature, classifying song popularity in 5 levels.

Work

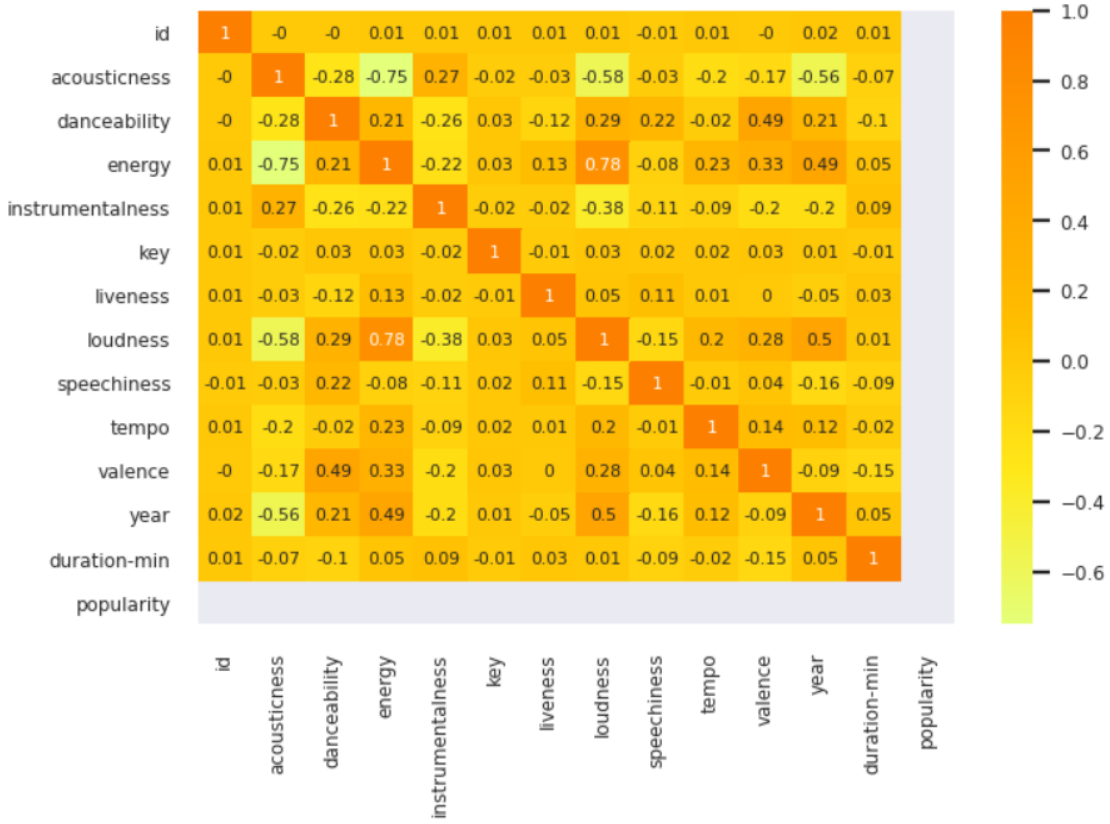
- The initial things we noticed from the problem statement were:
    1. The data we needed to predict was ordinal categorical, ranging from very low to very high.
    2. The accuracy of the model wasn't the most important factor as we had another condition on the predictions, i.e., we should not predict a track lower than what it turned out to be or we would not be able make a successful bid and we should also not predict a value too high otherwise we risk making a loss, so we need to focus more on whether the difference in prediction v/s actual results is positive or negative to maximise revenue.

- The next obvious thing to do was to check for missing values in the dataset, the result to which came out to be no. There weren't any missing values in either the training dataset or the test dataset. We also checked for duplicate rows in the dataset and we found none.

- We began going through the data and noted the features i.e:
  'id', 'acousticness', 'danceability', 'energy', 'explicit',
  'instrumentalness', 'key', 'liveness', 'loudness', 'mode',
  'release_date', 'speechiness', 'tempo', 'valence', 'year',
  'duration-min', 'popularity'

- The next thing we did was to encode the ordinal data into numeric substitutes.

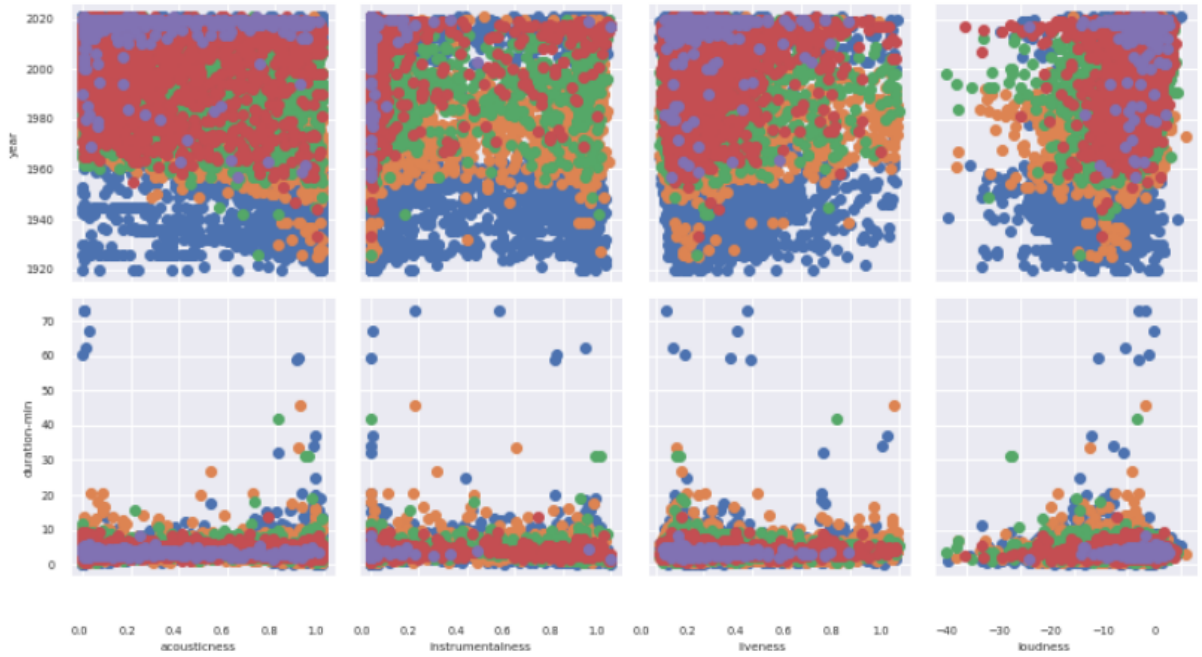
- Next we plotted a correlation heatmap to see the relation of features with each other:



From this heatmap we notice some important correlations between the features:

1) Energy is highly correlated with acousticness and loudness. Acoustic songs tend to be less energetic while loud songs tend to be highly energetic.
2) Valence of the song is fairly related to the danceability of the song; which makes sense as lively and positive songs tend to be suitable for dancing.
3) In recent years, more and more energetic and loud songs are being produced as shown by the positive correlation between the year and these parameters.

- We plotted pair plots comparing different features against one another based on their popularity and observed the following graphs:

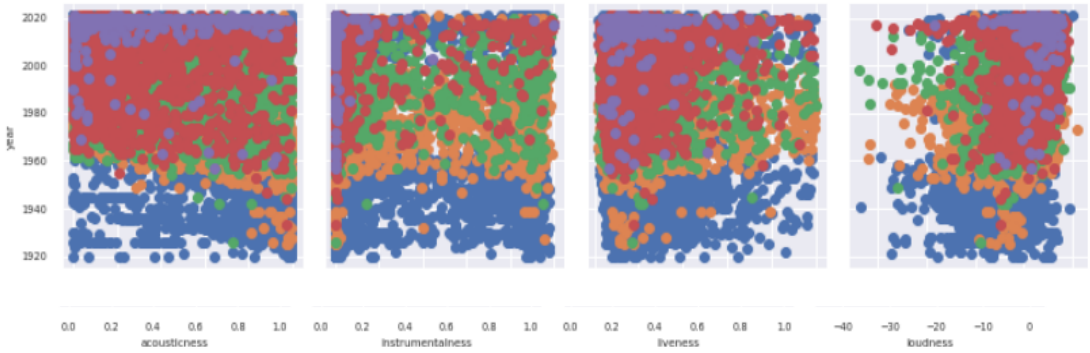


- The major observations from above graphs are:
    1. When the duration-min of the song is high, like above 40, it had always performed poorly in terms of popularity
    2. Above duration-min of 50, the songs have always been very less popular before the years 1940-1950, the chances of the song being hit were very very less. There were just a few hits which were highly popular, and few were lowly popular and most of them were very lowly popular. This could be due to lack of connectivity at those times compared to the times following it.

- We extracted the seasons out of the release-dates and made it a separate feature to see if it indicates some relation with popularity. Our logic behind this was, it may have been possible that, say during Christmas time, the majority of the people might be listening to a certain type of music, say happy, warm and during summers they may be listening to more energetic music.

- We attempted judging the correlation of the season of release with the popularity of the song but found it did not have much impact.

- From the year graphs,



- we could see that popularity of the songs showed a different trend in 4 major time periods:

    - Before 1960, the songs rarely gained some popularity and very high popular songs were nill.
    - Between 1960 and 1980, a few rare very popular songs started to appear and the overall popular songs gained their number.
    - Between 1980 and 2000, this growth further continued, probably due to improvement in communication and broadcast facilities such as radios.
    - After 2000, the growth boomed tremendously, obviously due to the advent of the internet.

So the next thing we did was to convert the year feature from numerical to categorical, based on the zone they lied:

1. Old_days: before 1960
2. Mid_days: 1960 to 1980
3. New_days: 1980 to 2000
4. Mod_days: after 2000

- The next idea we had was to divide songs into categories, like genres, based on their parameters like loudness, speechiness, energy etc. These new features measure the likelihood of each song falling into one of these categories found by certain algebraic operations on appropriate features.

  After some trial and error we defined them as follows:

  g0: Highly acoustic and instrumental. Low danceability, energy tempo, valence. “Slow & Somber Acoustics”

  g1: Highly instrumental and valent. Mid-tempo, mid-energy. Low acoustic ness and speechiness. “Happy & Danceable Instrumentals''

  g2: Highly instrumental. Low valence, speechiness. “Sad Instrumentals”

  g3: Highly valent. Speechy. Low instrumentality. “Upbeat Songs With Cheerful Vocals''

  g4: Highly instrumental, danceable, fast. Low acousticness. “Fast & Danceable Instrumentals”.

g5: High energy, valent, and fast. Relatively high liveness. Low acoustic ness and instrumentalness. 'Fast, Upbeat & Cheerful"

g6: Highly acoustic. Mid-high danceable. Speechy. Low energy. "Slow Dance".

g7: Highly valent and instrumental. Low tempo and speechiness. "Happy & Slow"

g8: High energy, tempo and instrumentalness. Low acoustic ness and speechiness. "Happy & Upbeat Instrumentals''

- We also noticed that the songs seemed to have similar tempos lying closely around three different values of 70, 100, 150. These being the tempos of classically slow, medium and fast paced songs. We defined 3 different features that measured the deviation from these tempos.

- We also observed a similar trend as that of year in the duration of the songs in minutes:
    - Below ~ 10-minute songs: There was a high possibility of finding very popular tracks there. In fact, 3 and a half minutes was the sweet spot for a song to get popular.
    - Above 10 but below 22-minute songs:
    - Above 22 but below 50-minute songs:
    - Above 50 minute songs:

We made the divisions accordingly and converted them into an ordinal feature.

- Next, we split the data into X_train and y_train(the training datasets) and X_test and y_test(the cross-validation datasets), where the test data here are the cross-validation sets made from the given training dataset.

NOTE:

WE DIDN'T ONE-HOT ENCODE AND NORMALIZE THE DATA IN THE EDA PROCESS AS DIFFERENT MODELS BEHAVE DIFFERENTLY WITH ONE-HOT ENCODED AND NORMALIZED DATA, SO WE DECIDED TO IMPLEMENT THEM ON THE SPECIFIC MODELS BASED ON THE MODEL'S REQUIREMENTS.

Implementing the Models:

1. XGBoost

We implemented XGBoost on the training set after applying the required preprocessing. Through this, we attempted to get a baseline for our predictions further.

XGBoost gave us an accuracy of

a. **0.62580** on the train set
b. **0.62552** on the validation set.

2. Random Forest

Next we tried implementing Random Forest Classifier after performing the required pre-processing and obtained an accuracy of

a. **0.58388** on the training set
b. **0.57196** on the cross-validation set

which was much below the baseline.

3. Neural Network with kappa loss

Next, we implemented a neural network with the kappa loss function which seemed reasonably fitting with our problem statement:

We obtained the following accuracy for this model:

a. Train set: **0.5671**
b. Cross-validation set: **0.5605**

This came out to be much better than the earlier models.

4. Neural Network with class weights

We also tried to implement a neural network with class weights (higher weights for higher prediction) just as an attempt to see if we can try something more.

For this model, we obtained an even better accuracy of:

a. Train set: **0.6187**
b. Cross-validation set: **0.5856**

5. Assembly of the final model

Finally, it was time to assemble the final model. We used the weighted average of the predictions from our individual models to construct an ensemble classifier.

For Assembly, we obtained an even better accuracy of:

a. Train Set: **0.7390**
b. Cross-validation set: **0.6321**

For Assembly, we also employed a customised metric as follows:

$$\frac{Total\ predictions\ greater\ than\ or\ equal\ to\ actual\ value}{Total\ Data\ points)}$$

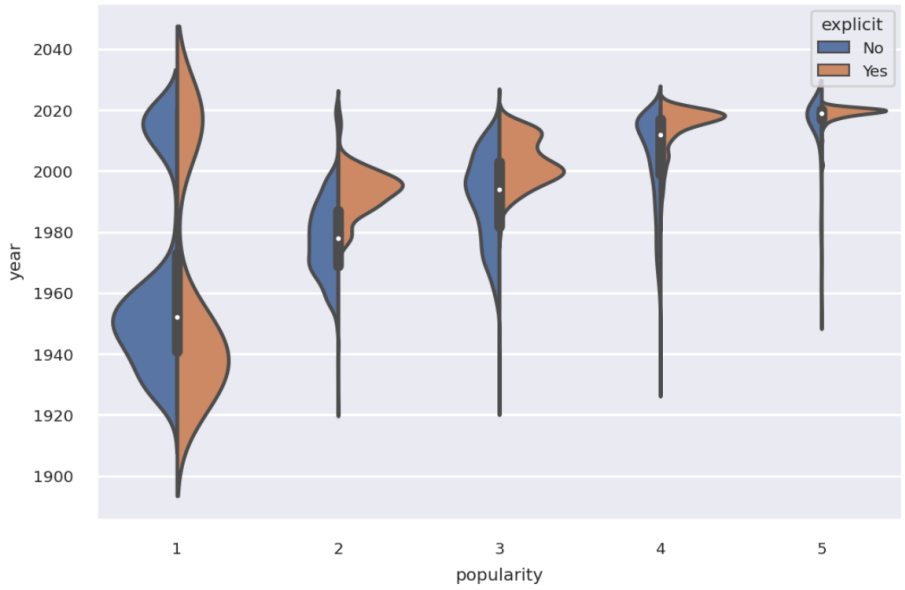
This metric tells us about the total number of successful bids possible. (bidding price higher than or equal to actual price of song)

a. Train set ratio of successful bids: **0.829**
b. Cross validation set ratio of successful bids: **0.756**

Some interesting insights:

We plotted quite a handful of graphs to analyse and understand the data thoroughly and make and erase features accordingly. In the process there were a few graphs that were the most important and have been shown in the list below.

1. Popularity through years of explicit and inexplicit tracks



The above graph gives us an idea of how popularity of explicit songs varied over the years 1920-2021.

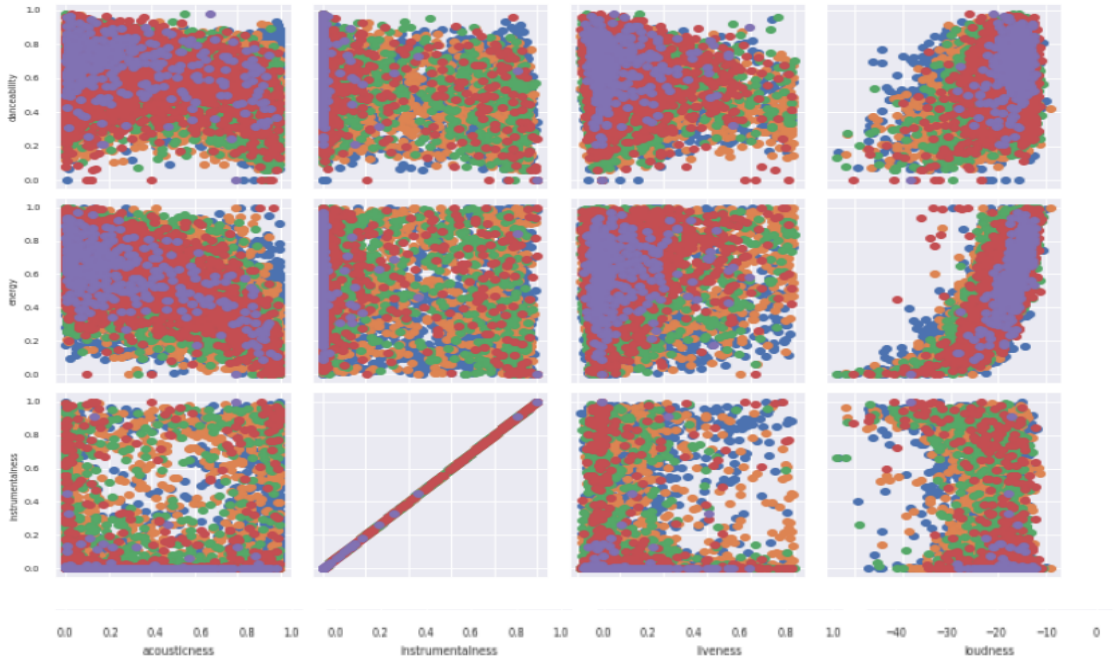
Interpretations:

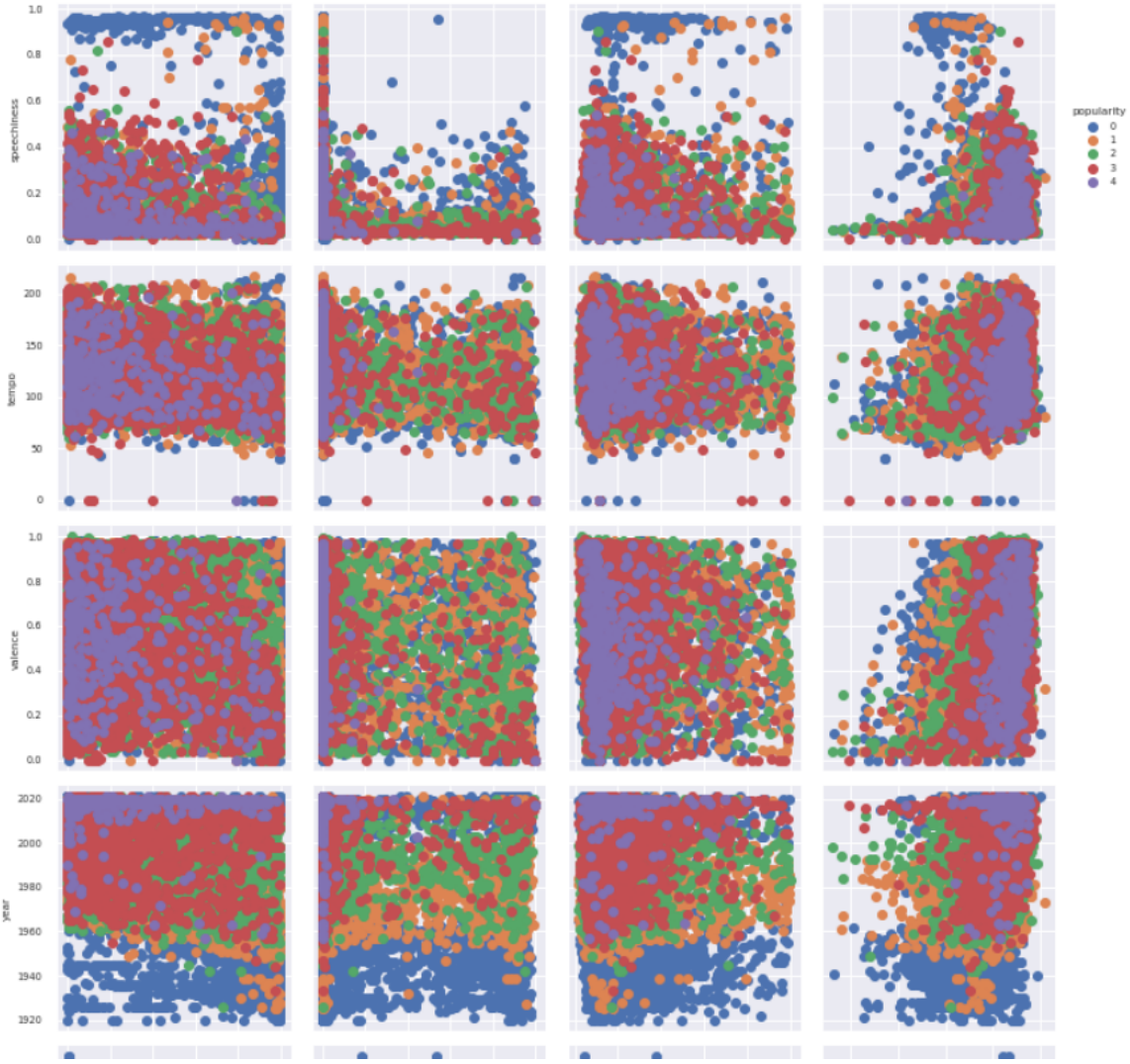
1. Firstly, we notice the graph overshoots into ~2050s which is just done by the module to make the violin plot, though it's irrelevant as we have data only until 2021.
2. Considering songs from average to highly popular as popular songs, we observe that explicit songs started getting popular from around 1985.
3. Though they started getting popular since 1980s, the actually popular songs(popular and highly popular) started peaking since around 2010 and it's only since around 2015, that
4. Though they started getting really popular around 2015, they started making it to "highly popular", or in other words, started topping the billboards.

5. We notice another interesting thing, that the number of explicit songs has also altered throughout the years and for some reason, it suddenly dipped gradually around the 1980s. From the above data, we observe that the total number of explicit songs suddenly dipped around 1980 and since then it boomed up in popularity.

   Upon some research, we observed that there was a very tense environment in the 1980s around the topic of explicit songs which might very well be the reason for the dip. In 1985, the Parental Advisory tag was started to be implemented. Though a lot of people debate on its effectiveness, it can be deduced, looking at the fact that there wasn't quite a network available as of now to make explicit songs popular, the tag must have only helped the teenagers and other people find explicit tracks more easily which must have led to the rapid increase in popularity since 1985.
6. Another interesting point to observe is that before the 1980s, there were quite a lot of explicit songs which suggest a similar number of audience, but almost all the songs were very less popular. Almost all of them lied in the lowly popular zone. So it might be possible that though there was a huge audience for those songs, there weren't records of it's popularity as maybe people were reluctant in those times to admit it.

2. Pair Plots:





Final Results:

After training the ensemble model and running the test data through it we create the result file. The result file contains the popularity labels as predicted by our model.

Number of songs of each category predicted :

	Popularity	Number of songs
1	Very Low	1170
2	Low	1205
3	Average	907
4	High	707
5	Very HIgh	11

Calculated Revenue in train Set = 148808 units

Predicted Revenue in train Set = 139740 units ; Absolute difference: 9068 units

Calculated Revenue in Cross Valid Set = 36838 units

Predicted Revenue in Cross Valid Set = 33546 units ; Absolute difference: 3382 units

S.no	Model Name	Accuracy on validation set
1	XGboost	0.62552
2	Random Forest	0.57196
3	Neural Network	0.5856
4	**Ensemble Model**	**0.6321**

ANNEXURE

APPENDIX

SOFTWARE & LIBRARIES STACK

➢ Python 3.6 - Language of choice
➢ Pandas - for Handling files
➢ Numpy - for complex numerical analysis
➢ Seaborn - plotting advance visualizations
➢ Matplotlib - plotting visualizations
➢ Sklearn - for making machine learning models

REFERENCES

https://pandas.pydata.org/pandas-docs/stable/
Pandas offer data structures and operations for manipulating numerical tables and time series.

https://docs.scipy.org/doc/
Numpy is a library used for computing scientific/mathematical data. Other usages are in:
1) Numerical Analysis 2) Linear algebra 3)Matrix computations

https://matplotlib.org/3.1.1/contents.html
Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter.

https://www.geeksforgeeks.org/ml-feature-scaling-part-1/
Feature scaling is the method to limit the range of variables so that they can be compared on common grounds. It is performed on continuous variables.

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
Train_test_split is a helpful function for partitioning data which splits out your data into a training set and a test set.

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.
Linear Regression.html It is used to show the linear relationship between a dependent variable and one or more independent variables.

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.Random ForestRegressor.html
Random Forest is an extension of bagging that in addition to building trees based on multiple samples of your training data, it also constrains the features that can be used to build the trees, forcing trees to be different. This, in turn, can give a lift in performance.

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Lab elEncoder.html
Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important preprocessing step for the structured dataset in supervised learning.

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.One HotEncoder.html
One-hot Encoding is a representation of categorical variables as binary vectors.This first requires that the categorical values be mapped to integer values.Then, each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1.

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squ ared_error.html
Mean squared error is a risk function, corresponding to the expected value of the squared error loss. It is always non – negative and values close to zero are better. The MSE is the second moment of the error (about the origin) and thus incorporates both the variance of the estimator and its bias.