Machine Learning Engineer Nanodegree

Capstone Project

Manish Kumar September 15th, 2019

I. Definition

Project Overview

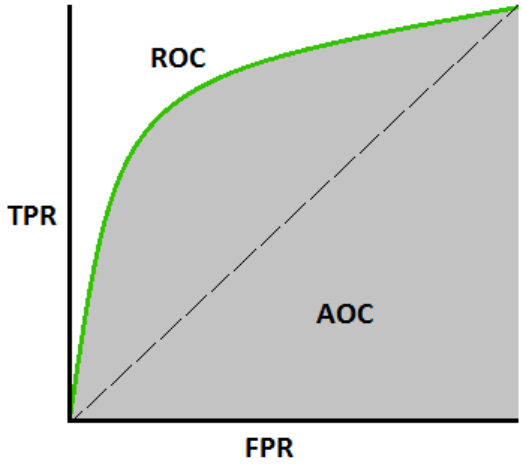
The commercialisation of the Internet and its entry into daily life along with the switch from analog to digital and the invention of the personal computer were the beginnings of the digital and technological changes that are now seen particularly within the music industry in the 21st century.
Few years ago, it was inconceivable that a person would listen to the Various Artists of choice on their morning commute. But, the glory days of Radio DJs have passed, and musical gatekeepers have been replaced with Machine Learning algorithms, continously finding and curating new tracks and unlimited streaming services.
While an OTT music subscriber has access to all kinds of music, algorithms still struggle in some areas. Without enough data about listening patter of the user, how would an algorithm know if the listener will like a new song or a new artist And, how would it know what songs to recommend to a new user. Music being an 18 Billion Dollars industry, is growing as more free subscribers are converting to a paid user for the convenience of auto music curation.

Problem Statement

In this regard; at the 11th ACM International Conference on Web Search and Data Mining (WSDM 2018) presented a Kaggle Challenge to build a better music recommendation system using a donated dataset from KKBOX, Asia's leading music streaming service, holding the world's most comprehensive Asia-Pop music library with over 30 million tracks.
KKBOX uses a collaborative filtering based algorithm with matrix factorization and word embedding in their recommendation system but believe new techniques could lead to better results.
In this project, I will try to predict the chances of a user listening to a song repetitively after the first observable listening event within a time window was triggered.
If there are recurring listening event(s) triggered within a month after the user's very first observable listening event, its target is marked 1, and 0 otherwise in the training set. The same rule applies to the testing set.

Metrics

In Machine Learning, performance measurement is an essential task. So when it comes to a classification problem, we can count on an AUC Curve. When we need to check or visualize the performance of the multi - class classification problem, we use AUC (Area Under The Curve) ROC (Receiver Operating Characteristics) curve. It is one of the most important evaluation metrics for checking any classification model's performance. Higher the AUC Value, better the model is at predicting 0s as 0s and 1s as 1s. In this case, Higher the AUC, better the model is at distinguishing between repeatability of a song.



$$FPR = 1 - Specificity = \frac{FP}{TN + FP}$$

$$TPR\ /Recall\ /\ Sensitivity = \frac{TP}{TP + FN} \qquad Specificity = \frac{TN}{TN + FP}$$

An excellent model has AUC near to the 1 which means it has good measure of separability. A poor model has AUC near to the 0 which means it has worst measure of separability. In fact it means it is reciprocating the result. It is predicting 0s as 1s and 1s as 0s. And when AUC is 0.5, it means model has no class separation capacity whatsoever.

II. Analysis

Data Exploration

From KKBOX we have training data set consisting of information of the first observable listening event for each unique user-song pair within a specific time duration. Metadata of each unique user and song pair is also provided.
The train and the test data are selected from users listening history in a given time period. The train and test sets are split based on time, and the split of public/private are based on unique user/song pairs.
Number of Unique Songs in Training Dataset: 359966
Number of Unique Songs in Testing Dataset: 224753

Number of Unique Users in Training Dataset: 30755
Number of Unique Users in Testing Dataset: 25131

Number of Unique Artists in Training Dataset: 40582
Number of Unique Artists in Testing Dataset: 27563

Number of Languages in the Training and Testing Dataset: 10
Number of Genres in Training Dataset: 572

Number of Genres in Training Dataset: 501

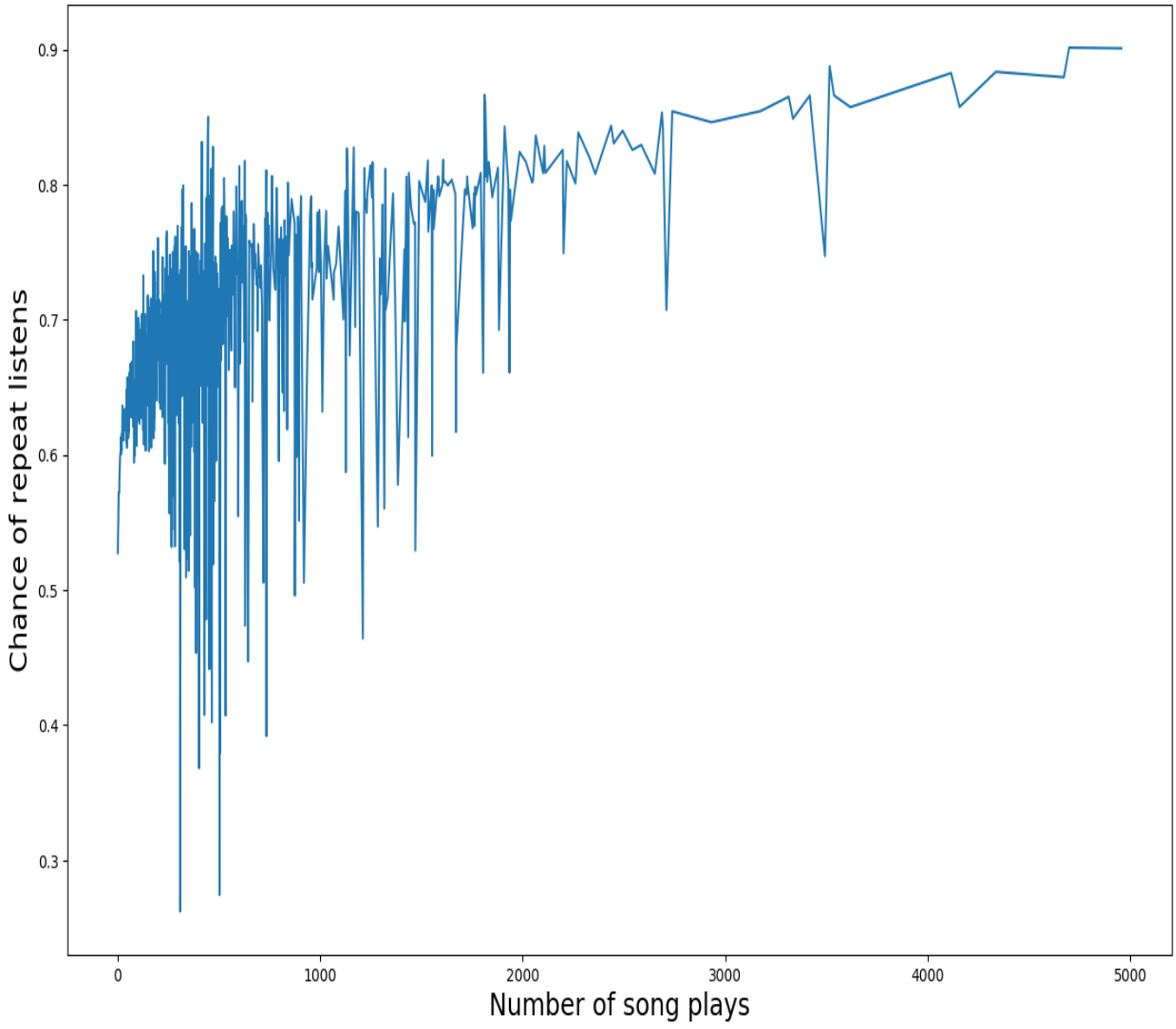
The Dataset has been taken from the WSDM - KKBox's Music Recommendation Challenge

Exploratory Visualization

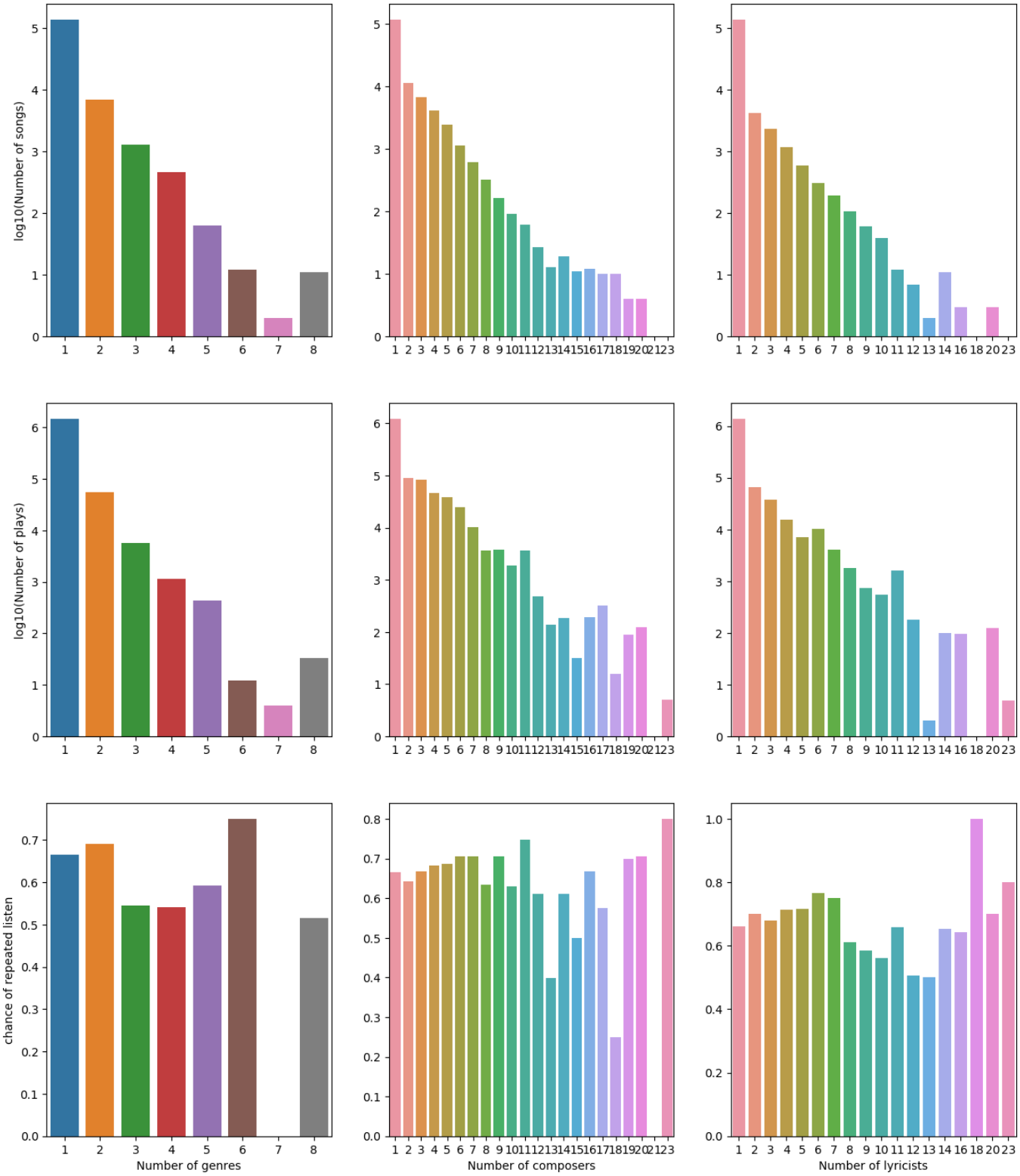
In this section, you will need to provide some form of visualization that summarizes or extracts a relevant characteristic or feature about the data. The visualization should adequately support the data being used. Discuss why this visualization was chosen and how it is relevant. Questions to ask yourself when writing this section:

- *Have you visualized a relevant characteristic or feature about the dataset or input data?*
- *Is the visualization thoroughly analyzed and discussed?*
- *If a plot is provided, are the axes, title, and datum clearly defined?* As part of exploratory data analysis(EDA) to see what data can reveal beyond the formal modelling, following plots were obtained. This exploration was done using the Data Exploration Notebook checked in the GitHub Repository.

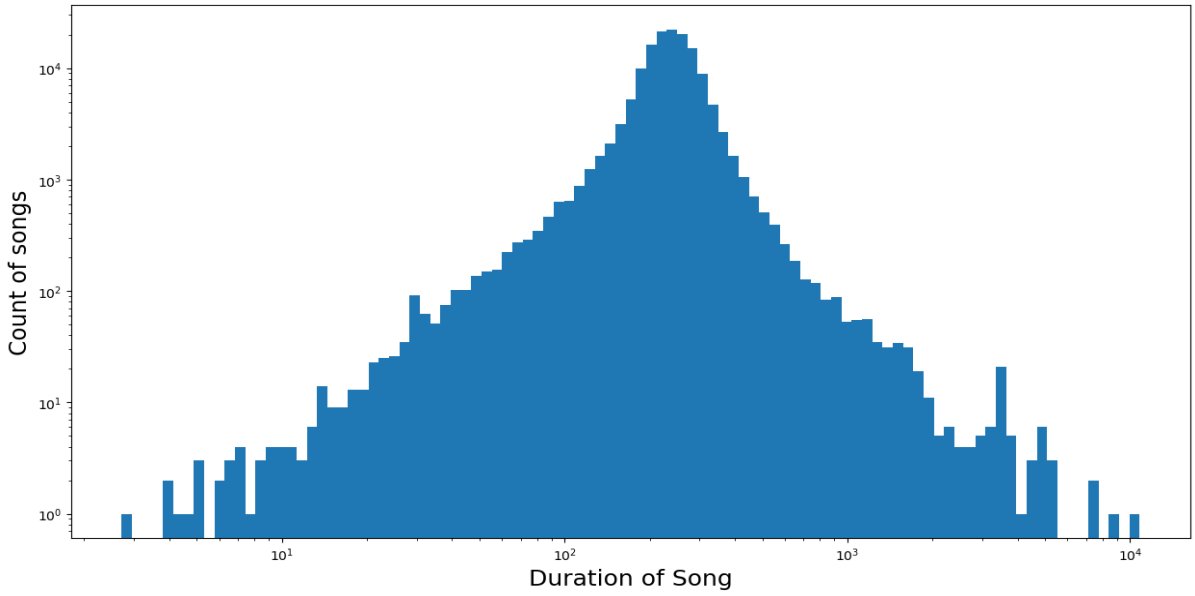
Plotting Number of Plays VS Repeatability:



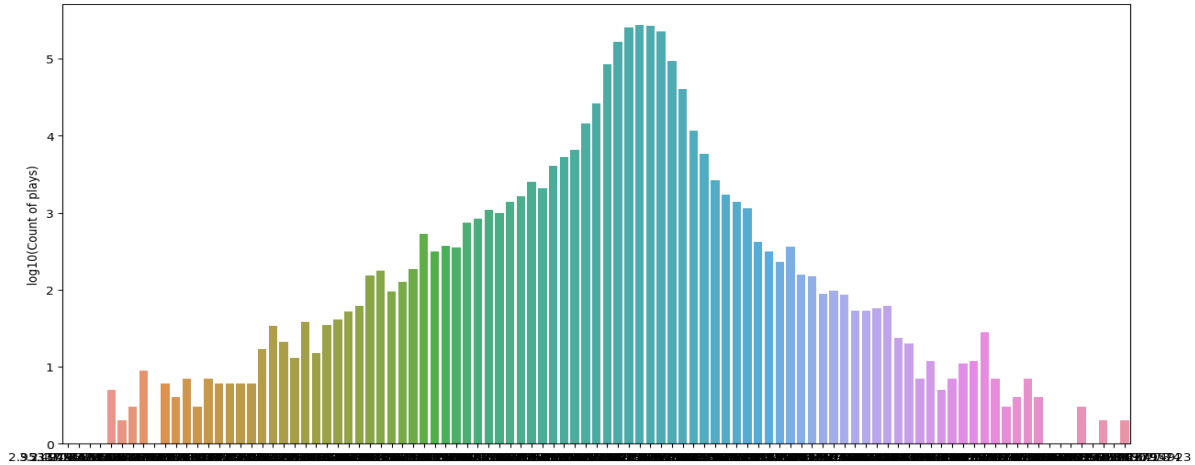
Plotting Genre,Composer,Lyricist Verses Repeatability of the Song:

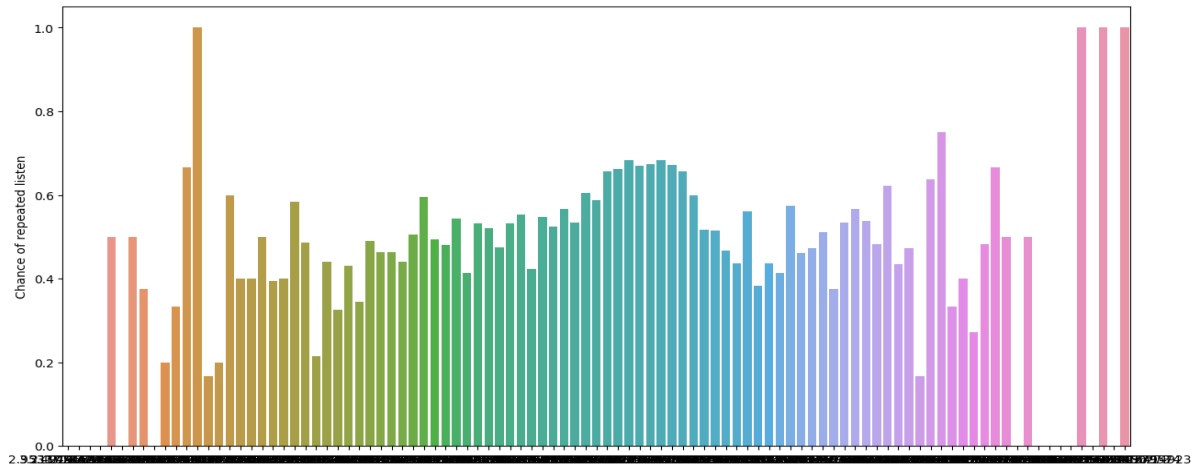


Plotting Count Verses Duration of the Song:

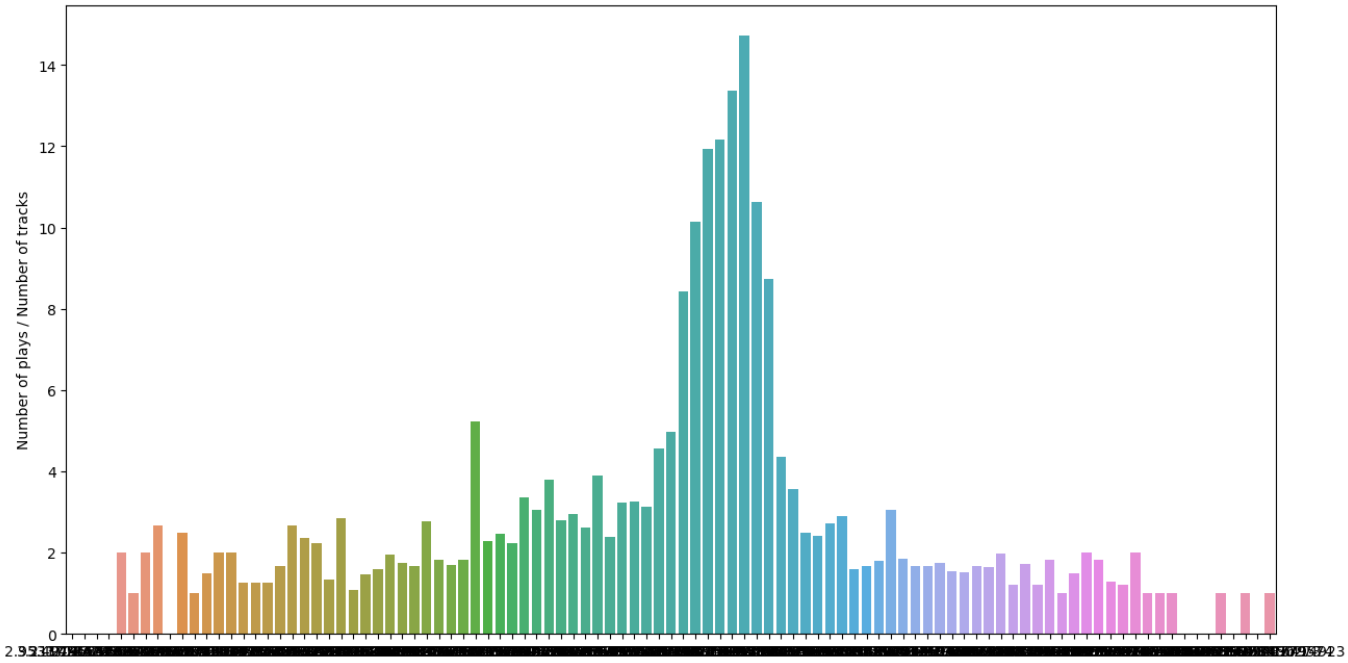


Plotting Count Verses Repeatability of the Song:

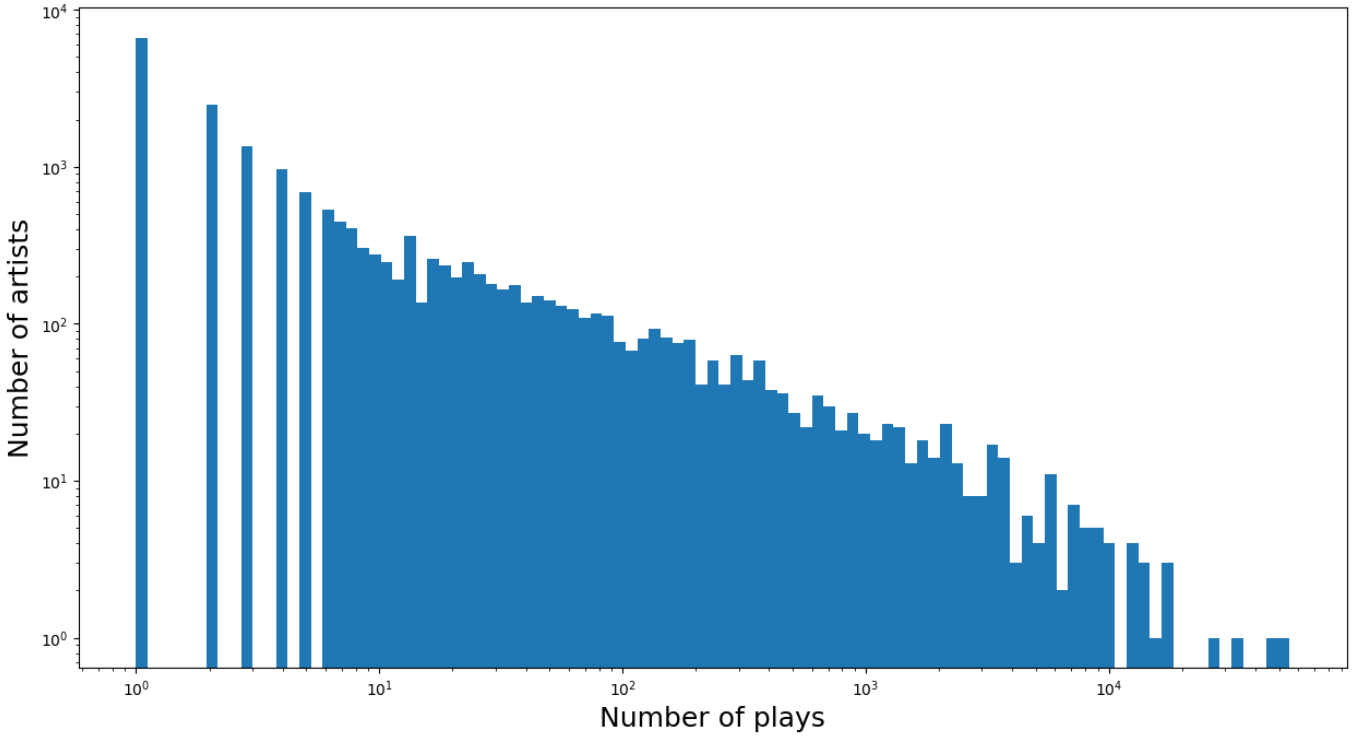




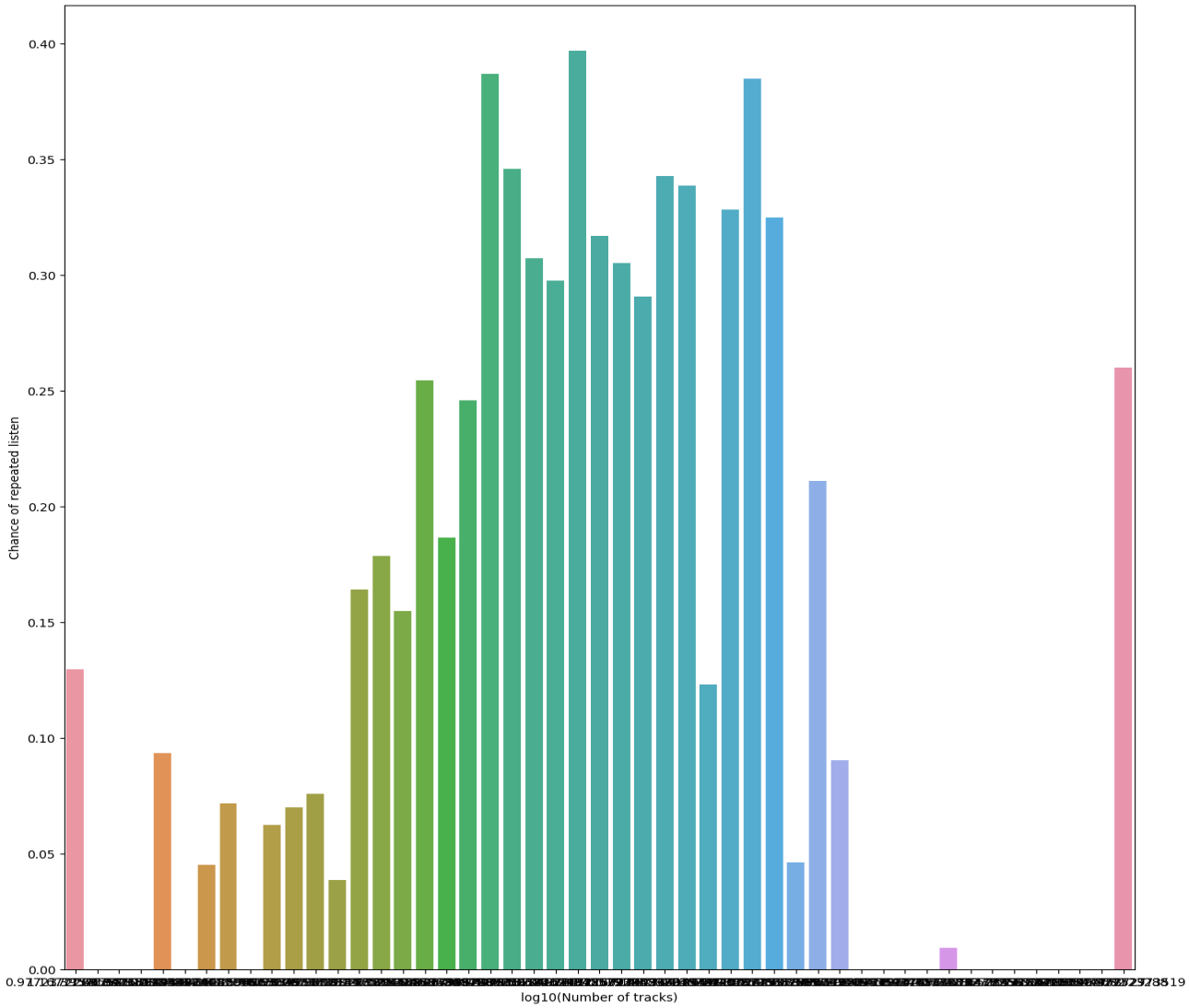
Plotting Count Verses Number of Plays of the Song:



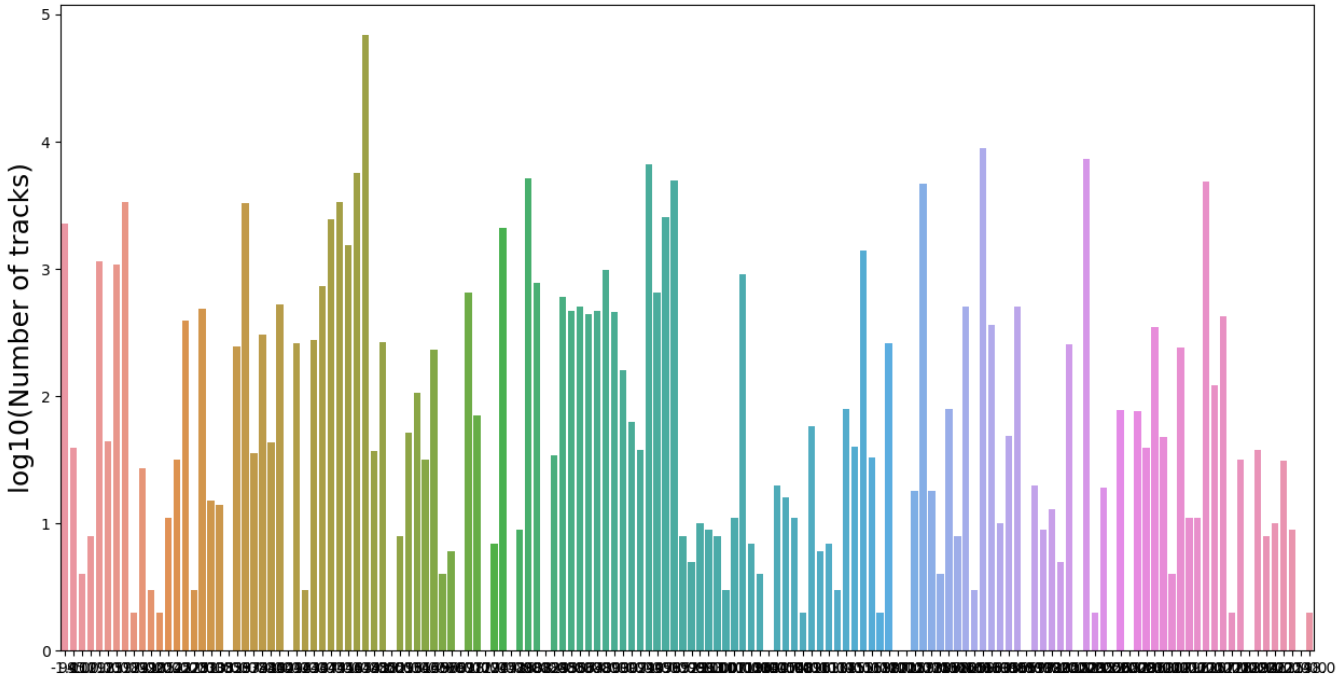
Plotting Artists Verses Number of Plays of the Song:

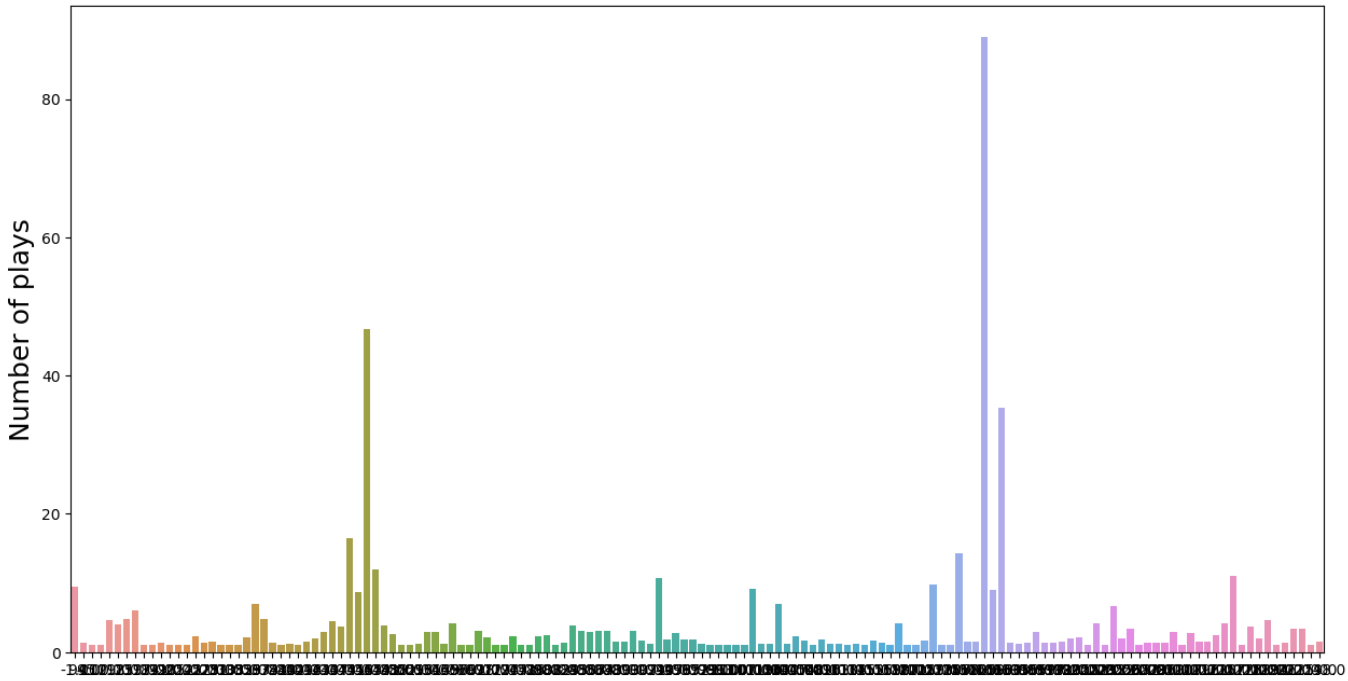


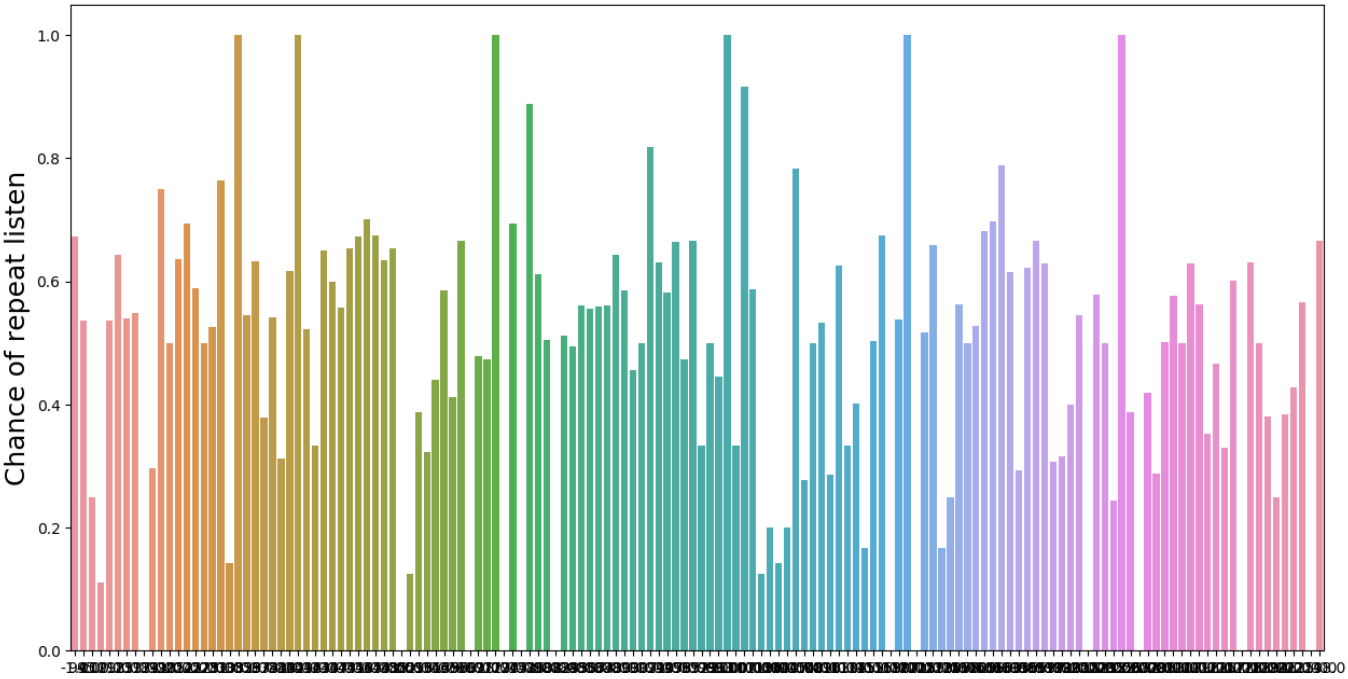
Plotting Repeatability Verses Number of Plays of the Song:



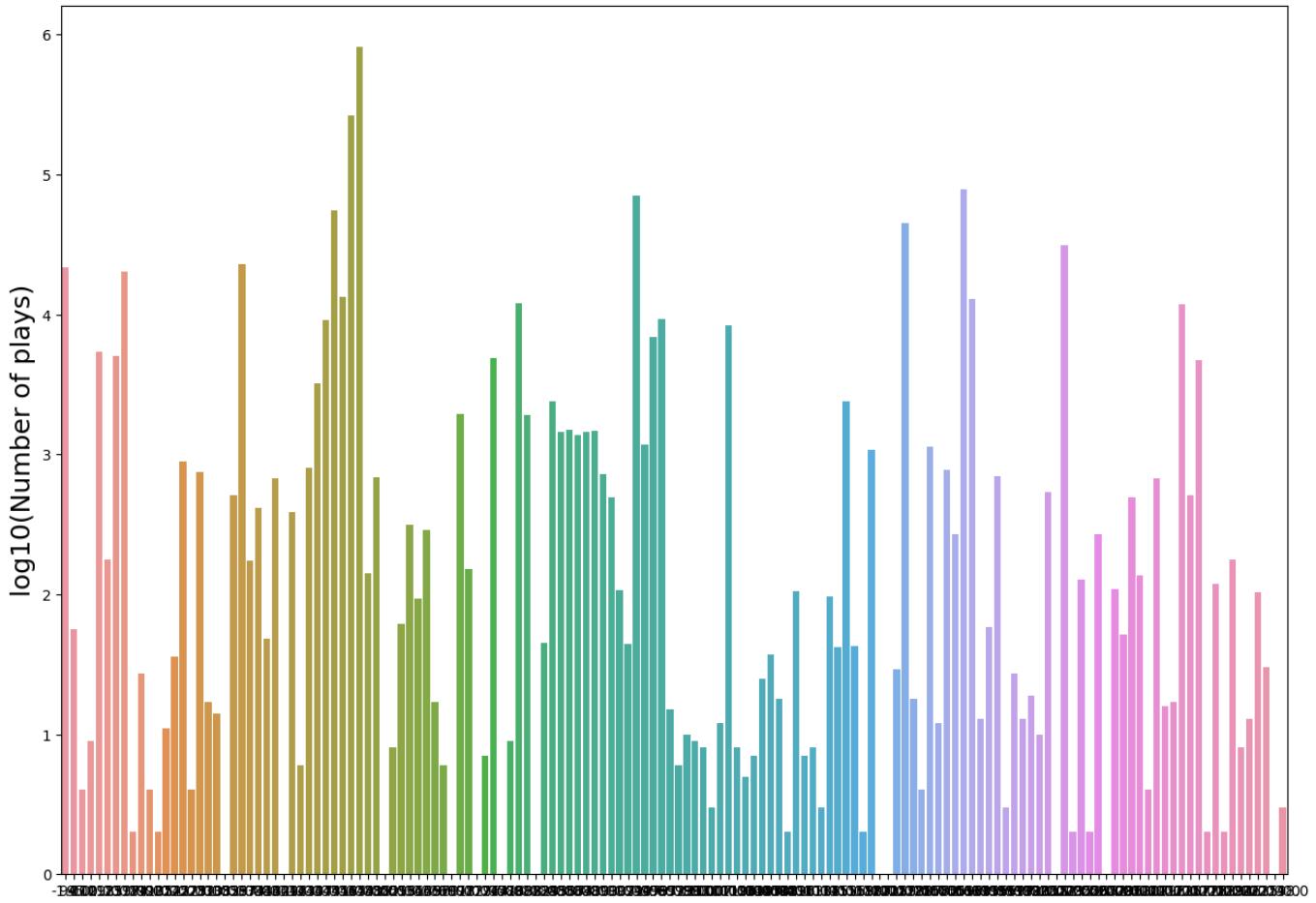
Plotting Track,Plays,Repeatability Verses Genre of the Song:




Number of plays
0
20
40
60
80


Chance of repeat listen
0.0
0.2
0.4
0.6
0.8
1.0

Plotting Plays Verses Genre of the Song:



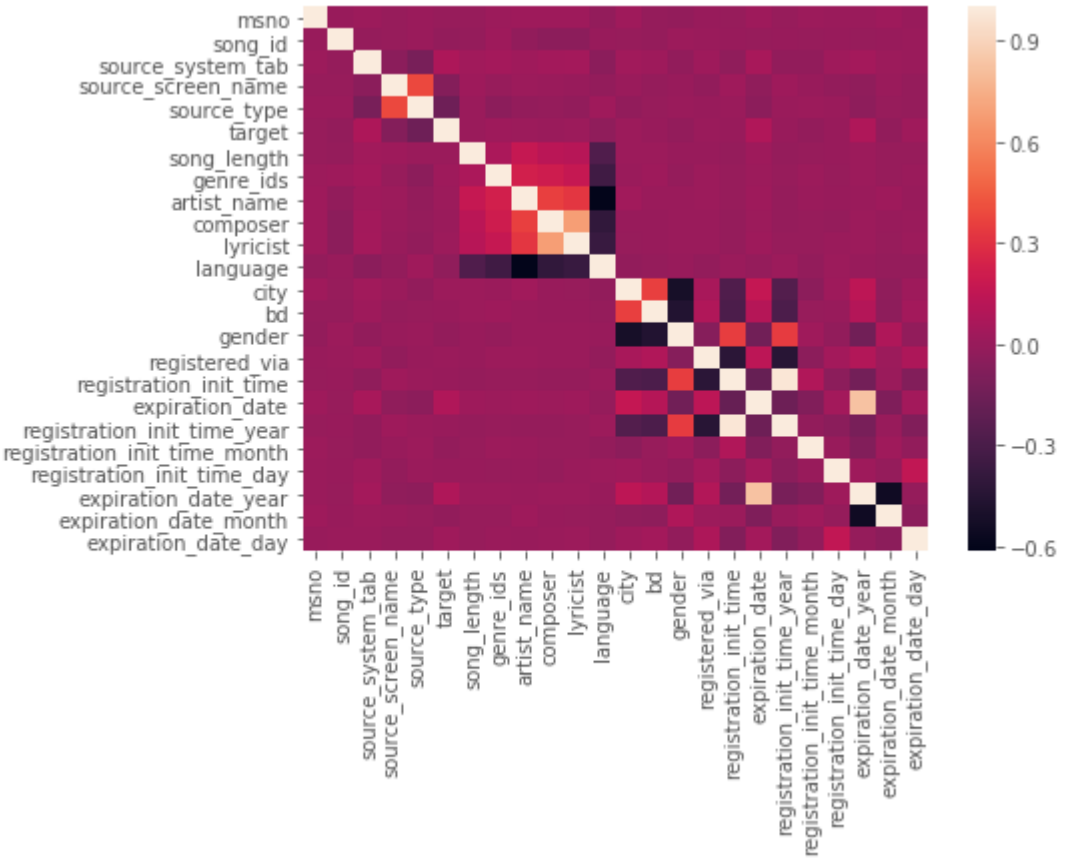
Algorithms and Techniques

The Algorithms that I intend to use in this project is XGBOOST and CNN. 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. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. XGBoost is an algorithm that has recently been dominating applied machine learning and Kaggle competitions for structured or tabular data. XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. The implementation of the algorithm was engineered for efficiency of compute time and memory resources. In my case of predicting the Repeatability of the song, I will use this Algorithm to compute precision value. CNN (Convolutional neural networks) are comprised of two very simple elements, namely convolutional layers and pooling layers. Although simple, there are near-infinite ways to arrange these layers for a given problem. Fortunately, there are both common patterns for configuring these layers and architectural innovations that can be used to develop very deep convolutional neural networks. Studying these architectural design decisions developed for state-of-the-art classification tasks can provide both clarity and intuition for how to use these designs when designing a deep convolutional neural network model.

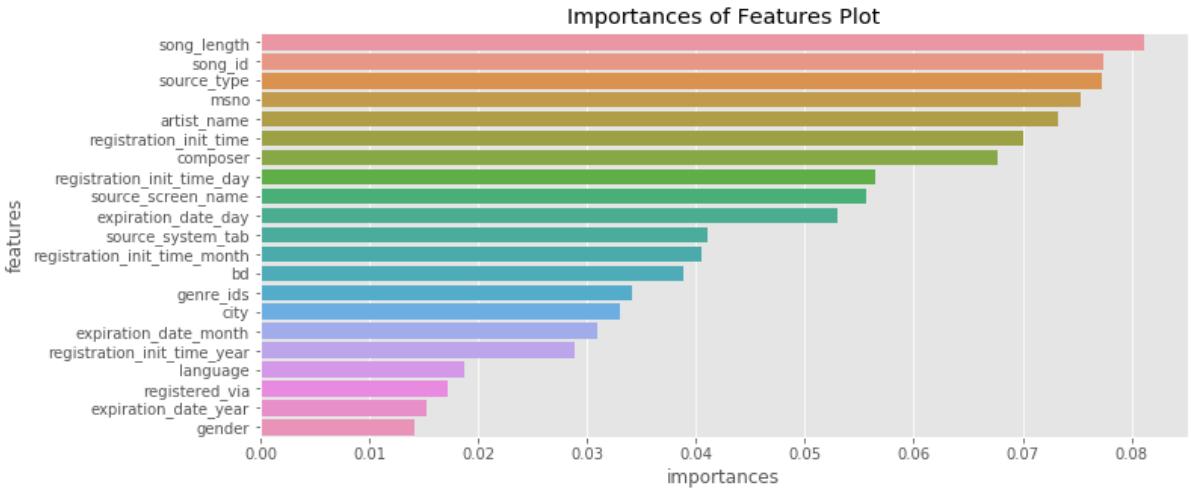
Benchmark

In this section, you will need to provide a clearly defined benchmark result or threshold for comparing across performances obtained by your solution. The reasoning behind the benchmark (in the case where it is not an established result) should be discussed. Questions to ask yourself when writing this section:

- *Has some result or value been provided that acts as a benchmark for measuring performance?*
- *Is it clear how this result or value was obtained (whether by data or by hypothesis)?* To create a Benchmark model as mentioned above, I would be using XGBOOST algorithm.
  The notebook used for this can be accessed here. It starts with some data Preparation:
- Replacing NAs
- Merging Datasets(train, songs, members)
- Creating new features (registration_init_time_days, registration_init_time_months, registration_init_time_years and expiration_date)
- Dropping correlated columns.
  Finding CoRelation between the features



Feature Importance



Dropping the less important columns(< 0.04).
Then using a XGBOOST Classifier with following parameters: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1, gamma=0, learning_rate=0.1, max_delta_step=0, max_depth=15, min_child_weight=5, missing=None, n_estimators=300, n_jobs=1, nthread=None, objective='binary:logistic', random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None, silent=None, subsample=1, verbosity=1) ## III. Methodology _(approx. 3-5 pages)_

Data Preprocessing

In this section, all of your preprocessing steps will need to be clearly documented, if any were necessary. From the previous section, any of the abnormalities or characteristics that you identified about the dataset will be addressed and corrected here. Questions to ask yourself when writing this section:

- *If the algorithms chosen require preprocessing steps like feature selection or feature transformations, have they been properly documented?*
- *Based on the **Data Exploration** section, if there were abnormalities or characteristics that needed to be addressed, have they been properly corrected?*
- *If no preprocessing is needed, has it been made clear why?*

Implementation

In this section, the process for which metrics, algorithms, and techniques that you implemented for the given data will need to be clearly documented. It should be abundantly clear how the implementation was carried out, and discussion should be made regarding any complications that occurred during this process. Questions to ask yourself when writing this section:

- *Is it made clear how the algorithms and techniques were implemented with the given datasets or input data?*
- *Were there any complications with the original metrics or techniques that required changing prior to acquiring a solution?*
- *Was there any part of the coding process (e.g., writing complicated functions) that should be documented?*

Refinement

In this section, you will need to discuss the process of improvement you made upon the algorithms and techniques you used in your implementation. For example, adjusting parameters for certain models to acquire improved solutions would fall under the refinement category. Your initial and final solutions should be reported, as well as any significant intermediate results as necessary. Questions to ask yourself when writing this section:

- *Has an initial solution been found and clearly reported?*
- *Is the process of improvement clearly documented, such as what techniques were used?*
- *Are intermediate and final solutions clearly reported as the process is improved?*

IV. Results

(approx. 2-3 pages)

Model Evaluation and Validation

In this section, the final model and any supporting qualities should be evaluated in detail. It should be clear how the final model was derived and why this model was chosen. In addition, some type of analysis should be used to validate the robustness of this model and its solution, such as manipulating the input data or environment to see how the model's solution is affected (this is called sensitivity analysis). Questions to ask yourself when writing this section:

- *Is the final model reasonable and aligning with solution expectations? Are the final parameters of the model appropriate?*
- *Has the final model been tested with various inputs to evaluate whether the model generalizes well to unseen data?*
- *Is the model robust enough for the problem? Do small perturbations (changes) in training data or the input space greatly affect the results?*
- *Can results found from the model be trusted?*

Justification

In this section, your model's final solution and its results should be compared to the benchmark you established earlier in the project using some type of statistical analysis. You should also justify whether these results and the solution are significant enough to have solved the problem posed in the project. Questions to ask yourself when writing this section:

- *Are the final results found stronger than the benchmark result reported earlier?*
- *Have you thoroughly analyzed and discussed the final solution?*
- *Is the final solution significant enough to have solved the problem?*

V. Conclusion

(approx. 1-2 pages)

Free-Form Visualization

In this section, you will need to provide some form of visualization that emphasizes an important quality about the project. It is much more free-form, but should reasonably support a significant result or characteristic about the problem that you want to discuss. Questions to ask yourself when writing this section:

- *Have you visualized a relevant or important quality about the problem, dataset, input data, or results?*
- *Is the visualization thoroughly analyzed and discussed?*
- *If a plot is provided, are the axes, title, and datum clearly defined?*

Reflection

In this section, you will summarize the entire end-to-end problem solution and discuss one or two particular aspects of the project you found interesting or difficult. You are expected to reflect on the project as a whole to show that you have a firm understanding of the entire process employed in your work. Questions to ask yourself when writing this section:

- *Have you thoroughly summarized the entire process you used for this project?*
- *Were there any interesting aspects of the project?*

- *Were there any difficult aspects of the project?*
- *Does the final model and solution fit your expectations for the problem, and should it be used in a general setting to solve these types of problems?*

Improvement

In this section, you will need to provide discussion as to how one aspect of the implementation you designed could be improved. As an example, consider ways your implementation can be made more general, and what would need to be modified. You do not need to make this improvement, but the potential solutions resulting from these changes are considered and compared/contrasted to your current solution. Questions to ask yourself when writing this section:

- *Are there further improvements that could be made on the algorithms or techniques you used in this project?*
- *Were there algorithms or techniques you researched that you did not know how to implement, but would consider using if you knew how?*
- *If you used your final solution as the new benchmark, do you think an even better solution exists?*

Before submitting, ask yourself. . .

- Does the project report you've written follow a well-organized structure similar to that of the project template?
- Is each section (particularly **Analysis** and **Methodology**) written in a clear, concise and specific fashion? Are there any ambiguous terms or phrases that need clarification?
- Would the intended audience of your project be able to understand your analysis, methods, and results?
- Have you properly proof-read your project report to assure there are minimal grammatical and spelling mistakes?
- Are all the resources used for this project correctly cited and referenced?
- Is the code that implements your solution easily readable and properly commented?
- Does the code execute without error and produce results similar to those reported?