Song Genre and Hit Prediction

EE 660 Course Project

Project Type: Design a system based on Real-World Data

Shyam Sundar Ravikumar

ssraviku@usc.edu

12/06/2019

1. Abstract:

Automatic Genre prediction has been one of the first steps toward Categorizing music. Equally important at the end of Artists and Music Production companies is the need to predict if a song will be a Hit or not so as to invest money into producing that song. In this work, I aim to provide a comparison between different models used for Genre prediction and also provide an answer to the question “Can a track be classified as a hit or non-hit based on just the audio features?”. For Genre prediction, I investigate the performance of five different classifier models on a dataset constructed from the AudioSet provided by Google and an accuracy of 0.534 is reported for this multiclass classification of Genre. On the other hand, a set of five classifier models are trained on a dataset built by extracting audio features of songs listed as Billboard hits and non-hits from the Million Songs dataset and Billboard data to predict if a song can make it to the Billboard Top 100, and a trained Random Forest classifier evaluates to an accuracy of 0.773 for the two-class classification.

2. Introduction:

Machine Learning, at present, is at the prime of its research on domains ranging from Deep Learning to Vision to Speech and Natural Language Processing with a lot of capital pouring in on Research. Music Streaming Companies such as SiriusXM, Spotify, Shazam, etc. undertake Classification of Music as the primary step towards building products such as Recommender system that suggests the next song, Categorizing similar music, or take a step further and provide real-time lyrics for a song when played.

2.1. Problem Type, Statement and Goals:

In this report, I have presented a detailed description of my investigation through machine learning algorithms, about one such audio-analysis problem that most of the companies perform to cater the needs of the people, that is predicting the Genre of a song, from among a list of seven candidates and as an add-on to this Classification problem, I have also tried to determine if the song can make it to the Billboard Top 100 or not. This is a pretty interesting problem as Genre-based Playlists, 'Hits' and 'Top 50 region-wise' are some of the most sought after features provided by the music streaming platforms and the fundamental step that needs to be taken is Genre and Hit Prediction. Getting these predictions right is not trivial, because Music genres are hard to describe due to their inherently subjective nature and estimation can only be done by selecting the right features with a fair amount of pre-processing.

2.2. Literature Review:

- *Genre Prediction:*

  Gaussian Mixture Models and K-Nearest Neighbors were some of the supervised learning methods that were used along with Frequency domain features such as Mel-Frequency Cepstral coefficients (MFCC) and spectral roll-off to classify the Genre of songs as proposed in [1]. Moving on down the line, Hidden Markov models were used in [2], Support Vector Machines (SVM) in [3], and SVM with AdaBoost in [4]. With the onset of Deep Learning techniques, Convolutional Neural Networks were used on Spectrogram images of the audio signals for classification. The first module of this project is focussed on performing Genre classification with the above-mentioned algorithms and also with more recent and advanced models such as Random Forests and Gradient Boosted trees and evaluate their performance.

- *Hit Prediction:*

  There has been a considerable amount of work done earlier in the field of Song Hit prediction. [5] classified a few hits from non-hit songs on the UK based songs chart, and following this, [6] predicted song popularity based on Youtube view count and audio features from the dataset curated by Echo Nest which was acquired by Spotify. In 2016, [7] used audio features and metadata of tracks from the Echo Nest to classify hits. The second module of this project aims to test different machine learning models on a custom made dataset to classify hits from the non-hit songs and evaluate their performance.

2.3. Prior or Related Work:

Prior and Related Work - None

2.4. Overview of Approach:

This project can be broadly classified into two independent modules:

- *Prediction of Genre*:

  A custom dataset was constructed with features extracted from 40540 10-second audio clips downloaded from Youtube and the data was pre-processed before training machine learning models on the data and different models were trained and validated on a set of data to select the best performing model for which the evaluation metrics used were primarily Accuracy and F1-score. The model with the highest scores was chosen and the performance was tested on the unknown data (test set).

- *Hit Prediction*:

  A custom dataset was constructed with song data from Million songs dataset and Billboards for the last 20 years (1990 - 2019), and features pertaining to the song data were extracted using the Spotify API, to get a final dataset with 13 features. These data were pre-processed and models were trained and evaluated on them to classify the Hits. The evaluation metrics used were Accuracy, F-Score, Precision, and Recall. The models were validated on the Validation set and the best performing model was chosen to test performance on the test dataset.

3. Implementation:

This section aims to provide a detailed description of the Implementation of the two modules of this project.

3.1. Datasets:

- *Prediction of Genre*:

  This dataset was constructed based on data derived from the AudioSet dataset provided by Google, and this has the information (YoutubeID) of those 10-second Youtube video clips that would fit in different genres. The primary goal here is to extract the 10-second clips from Youtube which fall under one of the seven candidate genres which are:

➔ Pop
➔ Hip-Hop
➔ Rock
➔ Rhythm Blues
➔ Reggae
➔ Vocal
➔ Techno

40540 10-second clips from these seven genres are downloaded as .mp4 files from Youtube using a package 'youtube-dl' and these .mp4 files are converted using another package 'ffmpeg' into .wav raw audio files [11].

● *Hit Prediction*:

This dataset was constructed based on data derived from the Million Songs Dataset and a script written to fetch data about songs that were featured in the Billboard Hot 100 between the years 1990 to 2019. In addition to these, data of songs that were not chart-toppers were also collected so as to maintain the balance between the Hits and Non-Hits classes. The combination of these datasets was then analyzed, and repeating occurrences of the same song data were removed to get a dataset with 3700 songs which had details of the Track name and the Artist who composed it.

3.2. Feature Extraction and Preprocessing:

● *Prediction of Genre*:

Each of the raw audio files is sampled at 22050 Hz and necessary features are extracted. The first step of this process is pre-emphasis filtering where a signal's frequencies are boosted before they are transmitted. Essentially, the noise in the audio signal is filtered thus improving the Signal to Noise Ratio (SNR) of the signal with a pre-emphasis coefficient of 0.95. The pre-emphasized signal is then processed to extract features in time-domain and Fourier transformed to extract the frequency-domain features:

- Time Domain:
  - Signal mean
  - Signal standard deviation
  - Zero Crossing Rate: Number of times a signal changes sign
  - Tempo: Pace at which the music flows. Unit: Beats per minute
- Frequency Domain:
  - Mel-Frequency Cepstral Coefficients (MFCC):

    The Mel-Frequency Cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of the log power spectrum on a nonlinear scale of frequency known as the Mel Scale. MFCCs are coefficients that collectively make up an MFC. They are obtained by taking the Fourier transform of a signal and the powers of the spectrum are mapped to the Mel Scale and Discrete Cosine Transform is applied onto the log of the powers of the Mel Spectrum. The amplitudes of the spectrum are the MFCCs. In our case, we take the mean and standard deviation of 20 filter banks of the MFC.
  - Spectral Centroid:

    Indicates the center of mass of the spectrum. In other words, it is the Frequency around which maximum energy of the signal is centered at. The centroid is calculated by taking a Fast Fourier transform with a Hanning window size of 2048 and the number of samples between successive frames (hop length) is taken to be 512. Mean and standard deviation of these centroids are taken for each of the signals.
  - Spectral Rolloff:

    It is the frequency below which a specified percentage of the total spectral energy lies. Rolloffs are calculated with the same metrics mentioned above. Mean and standard deviation of these are taken for each of the signals.
  - Chromatic Features:

    Chroma-based features, which are also referred to as "pitch class profiles", are a powerful tool for analyzing music whose pitches can be meaningfully categorized into twelve pitch categories (C, C♯, D, D♯, E, F, F♯, G, G♯, A, A♯, B). Mean and standard deviation for each of the 12 pitch classes are taken for each of the signals.

The table below summarizes the features of the Genre Prediction dataset after they have been extracted.

Feature	**Type**
Signal (Mean and Std. Dev)	Real int values
Zero Crossing Rate	Real int values
Tempo	Real int values
Mel-Frequency Cepstral Coefficients (MFCC) (20 coefficients)	Real int values
Spectral Centroid	Real int values
Spectral Rolloff	Real int values
Chromatic features (12 pitch classes)	Real int values
Label for Genre	Categorical (0-6)

Table [1]: Dataset feature description for Genre Prediction

There are no missing values in this dataset because the features are obtained by processing the audio files. Since there are a total of 73 features, not all models perform well with a reduced feature space, but then Logistic Regression with L1 regularization and XGBoost take care of selecting the best features for classification with which a good performing model can be trained.

- *Hit Prediction*:

With the Track and the Artist name from the dataset, the Spotify API is called to extract features about the song. The features[8] are:

➔ Danceability:

It describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity.

➔ Energy:

It represents a perceptual measure of intensity and activity. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.

➔ Key:

The estimated overall key of the track. Integers map to pitches using standard Pitch Class notation

➔ Loudness:

It is the overall loudness of a track in decibels (dB). Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude).

➔ Mode:

Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived.

➔ Speechiness detects the presence of spoken words in a track.

➔ Acousticness is a confidence measure from 0.0 to 1.0 of whether the track is acoustic

➔ Instrumentalness:

Predicts whether a track contains no vocals. Closer this value is to 1.0, the greater the likelihood, the track contains no vocal content.

➔ Liveness:

Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live.

➔ Valence:

A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track.

➔ Tempo:

In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.

➔ Year and month of song release

The table below summarizes the features of the Hit Prediction dataset after they have been extracted.

Feature	**Type**
Danceability	Real int values
Energy	Real int values
Key	Multiclass Categorical (0-11)
Loudness	Negative Real int values
Mode	Binary Categorical (0,1)
Speechiness	Real int values
Acousticness	Real int values
Instrumentalness	Real int values
Liveness	Real int values
Valence	Real int values
Tempo	Real int values
Year	Multiclass Categorical (1990-2019)
Month	Multiclass Categorical (1-12)

Table [2]: Dataset feature description for Hit Prediction

Spotify API could not return all the features for all songs resulting in missing data. These missing values were found and the entire row was eliminated from the dataset and after this, there were a total of 3439 songs with the complete feature space. Since there are a total of only 14 features, and all of them are significant for the classification problem, reduction in dimensionality affects the performance of the models, so there is no method such as Principal Component Analysis (PCA) or any form of Discriminant Analysis (LDA, QDA, Fisher’s LDA) that is employed.

3.3. Dataset Methodology:

- *Prediction of Genre*:

  The dataset is split into three different subsets in the ratio of 80:12:8 as Training, Validation, and Test sets and will have a total of 32432, 4864, 3244 data points respectively. The feature values were scaled using a MinMaxScaler to confine them within the range of [min. val., max. val.] by fitting the scaler on the Training data and then transforming the validation and test datasets. The models were trained on the training data, and in specific, Logistic Regression, Support Vector Machine, Random Forest and the Gradient Boosted XGBoost models were cross-validated to estimate the best set of hyperparameters, which when chosen, give better Accuracy and F-score metrics on the validation dataset, and with these metrics, the best performing model is chosen to predict the Genre of the unseen Test set.

- *Hit Prediction:*

  In this problem, the dataset is split into three different subsets in the ratio of 90:5:5 as Training, Validation, and Test sets and will have a total of 3090, 172, 174 data points respectively. The feature values were scaled using a MinMaxScaler to confine them within the range of [min. val., max. val.] by fitting the scaler on the Training data and then transforming the validation and test datasets. Models such as K-Nearest Neighbors, Logistic Regression, Support Vector Machine, Random Forest and XGBoost are cross-validated to estimate the best hyperparameters and models are trained and evaluated on the validation set with these parameters to visualize the model and select the best performing one. This model is chosen to be tested on the unseen test set to predict the possibility of the particular song getting featured in the Billboard Hot 100.

3.4. Training:

- *Models trained*:

  ➔ Naive Bayes' Classifier:

  The most important assumption that Naive Bayes' makes is that the classifier is 'Naive' meaning it assumes independence of all the features for classification inherently. Although this is not true in most cases, the classifier works fairly well, and it is because of the independence assumption, all features are given equal weights during classification. The Naive Bayes' Classifier works on the basis of Bayes' theorem which defines the posterior probability with the likelihood, the prior and evidence. This is suitable for discrete data, but for continuous data, another assumption is made in which the continuous feature values are associated with a distribution, and in this problem, since we are dealing with continuous data, I have assumed a Gaussian distributed feature space. The Naive Bayes' is a simple classifier that was tested to see how well it can classify the seven different genres, and it turns out that the independence assumption did not work well for this problem and the model was not able to classify 59% of the dataset, which can be inferred from the accuracy of the model, which was 0.412. Potential reasons for over/underfitting can be attributed to the class conditional probabilities becoming 0 or it can also be because of class imbalance. In our problem, with a large enough dataset, it is highly unlikely for the class conditional probabilities to go to 0 and we have established earlier that there are enough data points in each of the classes for a class imbalance problem to arise, so we can eliminate the two probable reasons for over/underfitting in this problem. The confusion matrix for the classification using Naive Bayes is shown below.

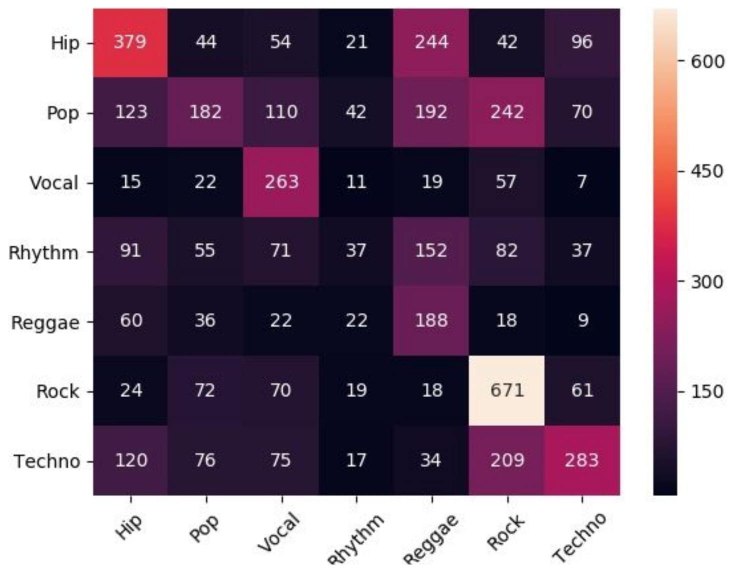


Figure [1]

➔ K-Nearest Neighbors:

The motivation for trying out K-Nearest Neighbors was the fact that classifying Hit songs is a Binary classification problem in which data points close to each other might hold the key to classifying the new data point to the right class. The supervised algorithm of Nearest neighbors works on the principle of classifying a data point by taking a majority vote from the classes of K-closest data points. The choice of ‘K’ nearest neighbors and the distance metric which is used to determine the closest points are dependent on the dataset, thus, to estimate these parameters, the training data for Hit prediction is cross-validated. Cross-validation suggested that choosing 31 neighbors calculated by the Manhattan (absolute) distance metric yields the best performing model with an accuracy of 0.622 and a precision score of 0.644 for the Hit class and 0.588 for the Non-hit class. The confusion matrix for the classification using K-Nearest Neighbors is shown below.

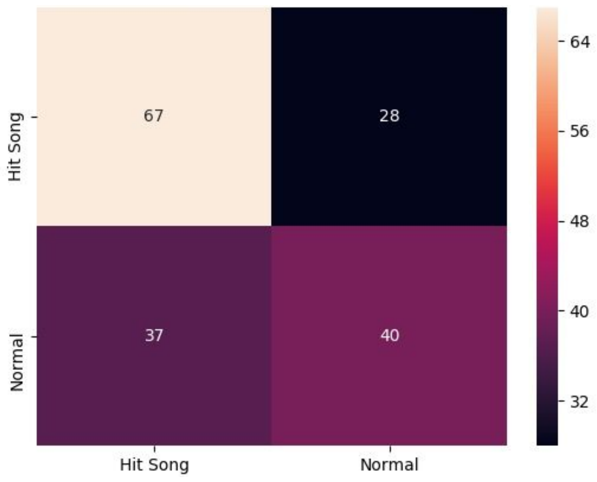


Figure [2]

➔ Logistic Regression:

Logistic Regression is a predictive modeling technique that uses the Sigmoid function to classify the binary/multiclass dependent variable. Logistic Regression relies on the assumptions that the target label is binary (0/1) and also there should be no high correlation among the predictors. In a multiclass scenario using logistic regression, two predictors are taken into consideration for the classification at once, and this method is known as One vs Rest Classification with cross-validation over 10 splits. This method is computationally intensive when we have a lot of features and the model can overfit to the data too. To prevent this, a new regularization parameter is added to reduce overfitting and to also select the features that are significant for classification. The regularization can be of three types: L1, L2 or a combination of both. L1 regularization adds “absolute value of magnitude” of coefficients as penalty term to the loss function while L2 regularization adds the “squared magnitude” of the coefficients as penalty term to the loss function. In our problem, we need the non-relevant features to be removed, so L1 regularizer which forces the coefficients to 0 is applied, thus eliminating some features. The results for Logistic Regression applied to Genre Prediction and Hit prediction are provided in the next section.

➔ Support Vector Machine:

An SVM classifies data points by finding a hyperplane (decision boundary) that maximizes the margin between the two classes. This hyperplane can be visualized as a line in a 2-dimensional space and as a plane, in a 3-dimensional that separates the classes. The vectors that define the properties of the hyperplane are known as support vectors. SVMs can find linear and nonlinear decision boundaries depending upon the choice of kernel function to classify more complex data. The most common kernel function used for classification is the nonlinear Radial Basis Function (RBF Kernel) which is the exponential of squared Euclidean distance between the feature vectors and is parameterized by γ (gamma) and C (inverse regularization coefficient). In our problem of multiclass classification in the case of predicting Genre, we make use of the method of One vs Rest classifiers where the class of a new data point is predicted by classifying with respect to the features individually (like a two-class classification problem) with class weights assigned based on the number of data points in each class and a majority vote is taken to assign a proper class. Cross-validation is done to estimate the best inverse L2 regularization coefficient for which the model classifies the data better. The results of SVM applied to Genre Prediction and Hit prediction are provided in the next section.

➔ Random Forest:

Random Forest is an ensemble learning algorithm that works on the underlying concept of Bagging of Decision trees. We know that decision trees perform better than the previous models as they can fit complex nonlinear boundaries to classify the data and as we let the tree grow deep, it might even overfit. In a random forest, we perform the classification of multiple subsets that are repeatedly sampled from the original dataset. This process is known as Bagging, and each of these subsets is used to train individual decision trees, which give their prediction of a class based on the subset of features they were trained on and a majority vote is taken to predict the class to which that data point belongs to. In general, Decision trees and Random Forests are the models most prone to overfitting as they can completely classify data into multiple classes. To prevent this, the Random Forest Classifier is cross-validated to estimate the optimal number of estimators (trees) for which the classifier performs well. More the number of trees, then less is the possibility of overfitting. Another factor that is used to reduce overfitting is the minimum number of splits for each layer of the tree. Setting this number high restricts the tree from growing and thus cuts off the tree at a depth where the criteria are not satisfied, thus preventing overfitting the tree to the training data. This factor can be compared to the concept of tree pruning or setting the maximum depth up to which the tree can be grown. The results of Genre Prediction and Hit prediction with Random Forest classifiers are provided in the next section.

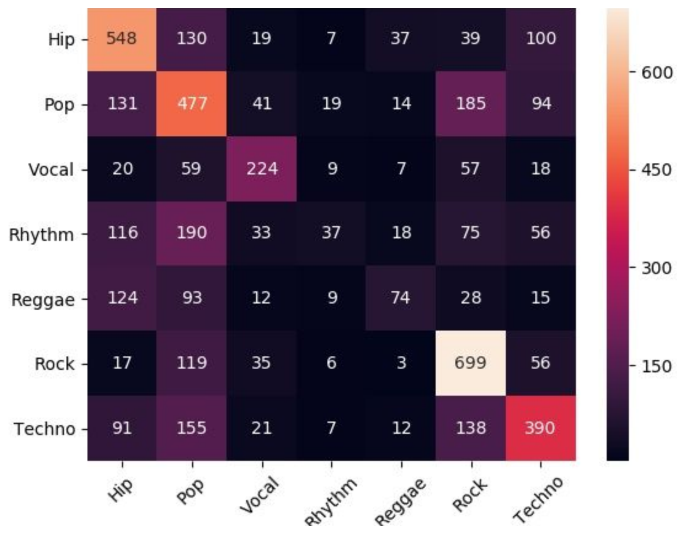
➔ Gradient Boosted Trees (XGBoost):

Boosting is a method of converting a set of weak learners into a strong learner. Initially, the set of data points that are considered for classification are given equal weights and the result of the first tree is analyzed and the weights are increased for the data points that were misclassified and reduced for those that were classified correctly. This process is repeated and the next tree improves the prediction based on the feedback from the aggregation of the previous trees. Thus we can say that the model learns from previous mistakes by focussing more on the observations or instances that were misclassified in a sequential manner. Gradient Boosting is a method where the gradient of a loss function is used to estimate the error in prediction. This error is calculated from the loss function and the gradient of this error indicates the direction in which the model parameters need to be changed so that the error is minimized. Extreme Gradient Boosting follows the same procedure of the Gradient Boosting algorithm, but instead computes the second-order gradients and also uses a combination of L1 and L2 regularization which provides more information on the direction of gradient propagation and minimization of the loss function. This makes the algorithm run faster while giving better results. But one obvious disadvantage of XGBoost is the overfitting problem. To prevent this in our case, the depth of the decision tree was explicitly set to 10 and the model was cross-validated on the training data to estimate the optimum number of trees required for correct classification and the objective function used was ‘multi:softprob’ which gives a matrix of probability values all the classes. The results of Extreme Gradient Boosting for Genre Prediction and Hit prediction are provided in the next section.

3.5. Model Selection and Comparison of Results:

- *Prediction of Genre*:

➔ Logistic Regression:



	Hip	Pop	Vocal	Rhythm	Reggae	Rock	Techno
Hip	548	130	19	7	37	39	100
Pop	131	477	41	19	14	185	94
Vocal	20	59	224	9	7	57	18
Rhythm	116	190	33	37	18	75	56
Reggae	124	93	12	9	74	28	15
Rock	17	119	35	6	3	699	56
Techno	91	155	21	7	12	138	390

Figure [3]

➔ Support Vector Machine:

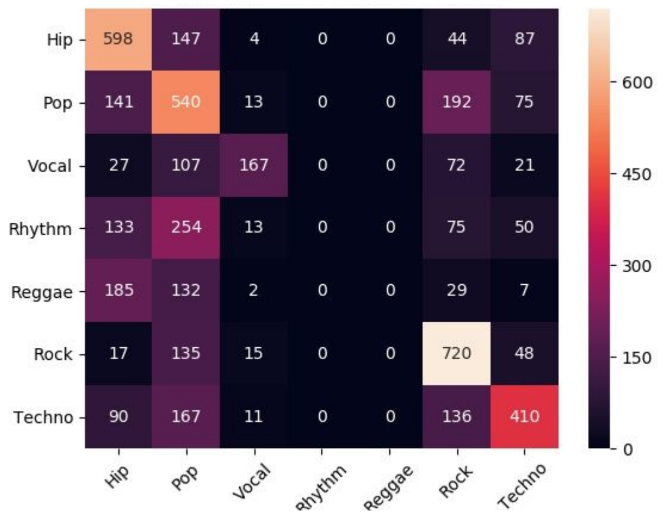


Figure [4]

➔ Random Forest:

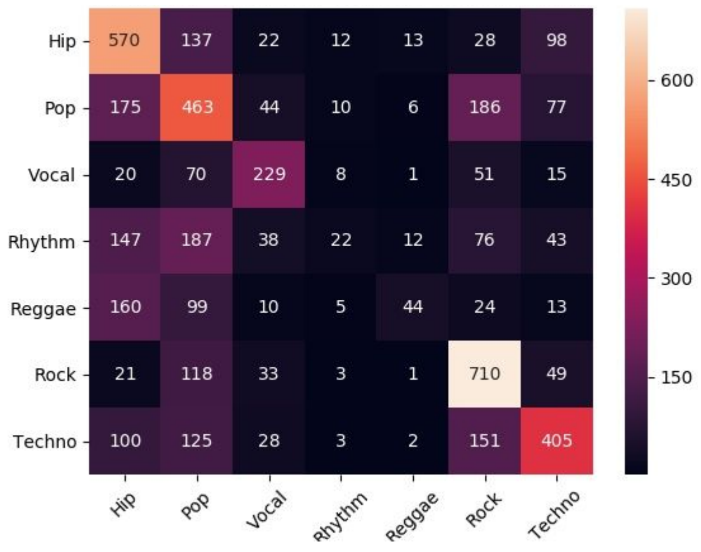


Figure [5]

➔ Gradient Boosted Trees (XGBoost):

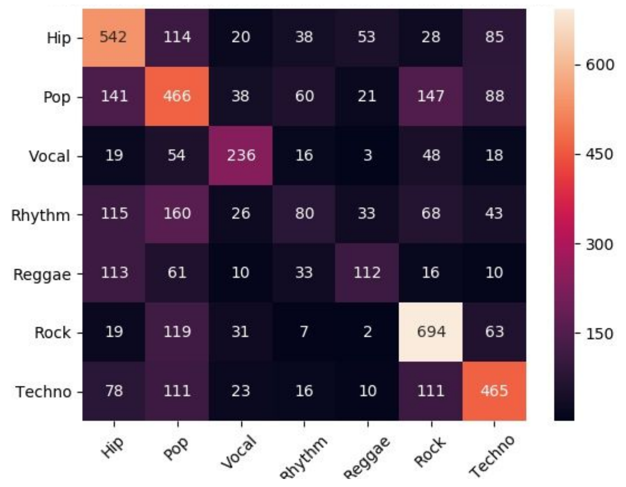


Figure [6]

Model	Accuracy	F-Score
Naive Bayes'	0.412	0.388
Logistic Regression	0.503	0.479
Support Vector Machine	0.501	0.452
Random Forest	0.502	0.469
XGBoost	**0.534**	**0.521**

Table [3]: Evaluation Metrics for Genre Prediction

XGBoost is chosen as the best model after the comparison of the metrics as shown in Table [3]. Detailed reasoning is provided in the next section.

- *Hit Prediction:*

➔ Logistic Regression:

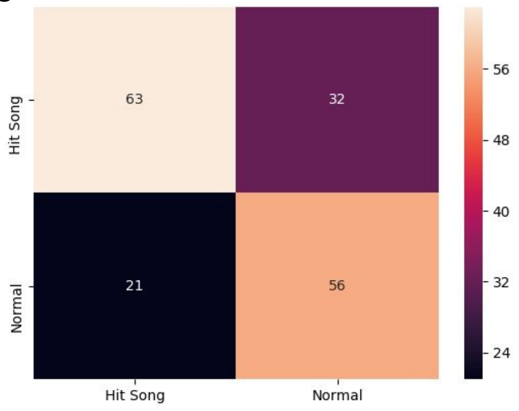


Figure [7]

➔ Support Vector Machine:

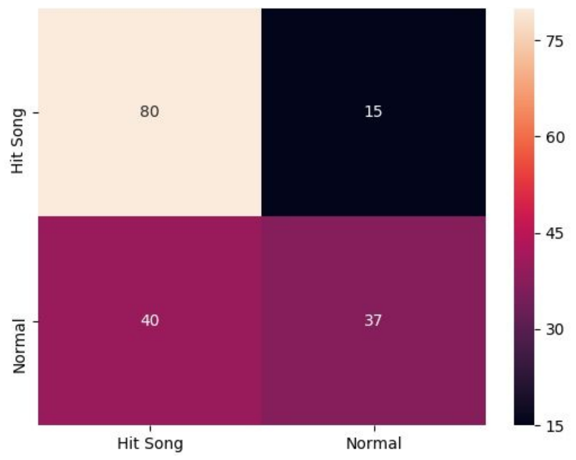


Figure [8]

➔ Random Forest:

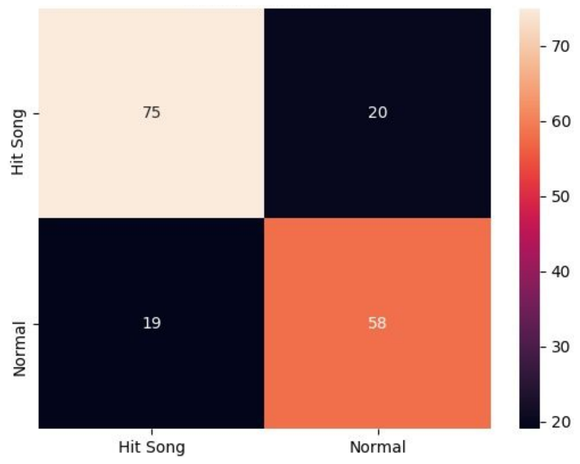


Figure [9]

➔ Gradient Boosted Trees (XGBoost):

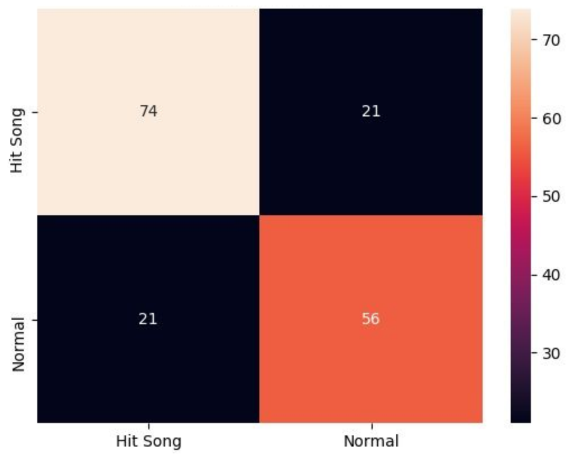


Figure [10]

Model	Accuracy	F-Score	Precision	Recall
KNN	0.622	0.552	0.588	0.519
Logistic Regression	0.692	0.679	0.636	0.727
Support Vector Machine	0.680	0.574	0.712	0.481
Random Forest	**0.773**	**0.748**	**0.744**	**0.753**
XGBoost	0.756	0.727	0.727	0.727

Table [4]: Evaluation Metrics for Hit Prediction

Random Forest is chosen as the best model after comparing the metrics as shown in Table [4]. Detailed reasoning is provided in the next section.

4. Final Results and Interpretation:

- *Prediction of Genre*:

From Table [3], we can conclude based on the Accuracy score, that the Gradient Boosted tree model with 500 estimators of maximum depth of 10 layers and the objective function as 'softprob' in a multiclass classification setting performed better than all the other models, as expected, with an accuracy of 54.3% on the unseen test dataset. As described earlier, a gradient boosted model can learn to minimize the loss and classify multiple classes by fitting complex nonlinear decision boundaries. This is the main reason for the increased performance of this model. The model is then tested on the unseen Test dataset (out of sample data) and the evaluated metrics in comparison to the Baseline model are mentioned below. In order to improve the performance of classification, it is imperative to extract more features regarding the songs as we can see from Figures [3] through [6], a larger part of the accuracy drop is a direct effect of the classes Rhythm Blues and Reggae being misclassified as Pop and Hip-Hop. Another method that can be used for better classification is a Spectrogram based Image classification problem using a Convolutional Neural Network which has been experimented by many researchers. This method can prove to be effective as the classification is done on spectrogram image data.

Metric	Baseline Gradient Boosted Model	My Gradient Boosted model
Accuracy	0.59	0.534

Table 5: Performance comparison of XGBoost for Genre Prediction

- *Hit Predictor*:

  From Table [4], we can conclude based on all four metrics, that the Random Forest Classifier with 550 estimators and a minimum number of samples to be present in each layer of the tree of 10, performed better than the XGBoost and other models. The unseen Test dataset (out of sample data) was then tested on the Random Forest model, and the model was evaluated to get an accuracy of 72.7% and precision of 0.798 for the Hits class. Other evaluation metrics are shown in the table below. Evaluating the performance with other works will not prove efficient as this problem is based on a dataset that was constructed with songs until October 2019, and thus performance metrics can vary largely. One of the possible reasons for the better performance of Random forest with a bagging approach could be that since this is a two-class classification problem, randomly sampling data from the training set and training decision trees simultaneously predicts the correct class more accurately than a boosted tree. Once again, extracting more audio features for the songs can prove to be useful in reducing the number of False Positives in predictions. These audio features may include song duration, Genre, and the season during which the song was released. These features might seem irrelevant to the classification, but nothing can be commented about them further without actually using them in predictions.

Model	Accuracy	F-score	Precision	Recall
Random Forest	0.727	0.729	0.623	0.672

Table 6: Performance comparison of Random Forest for Hit Prediction

5. Summary and Conclusion:

The detailed analyses provided above helped me to understand the usage of the models in specific cases and how to overcome the shortcomings and finally demonstrated the performance of different models and of them, the eXtreme Gradient Boosting algorithm emerges as the go-to model for Genre prediction and Random Forest classifier for predicting the journey of a Song to the Billboard Top 100. I would like to work further on this interesting problem to improve the performance of this model by taking the steps mentioned earlier about extracting more features to provide a model that can be more reliable.

6. References:

1. George Tzanetakis and Perry Cook, “Musical Genre Classification of Audio Signals”, IEEE Transactions on Speech and Audio Processing, 2002.
2. Nicolas Scaringella, Guido Zoia, D. Mlynek, “Automatic genre classification of music content: a survey”, IEEE Signal Processing Magazine, 2005.
3. Michael I. Mandel and Daniel P.W. Ellis, “Song-level Features And Support Vector Machines For Music Classification”, International Society for Music Information Retrieval, 2005.
4. Loris Nanni, Yandre M.G. Costa, Alessandra Lumini, Moo Young Kim, Seung Ryul Baek, “Combining visual and acoustic features for music genre classification”, 2016.
5. Yizhao Ni, Matt McVicar, Raul Santos-Rodriguez and Tijl De Bie, “An end-to-end machine learning system for harmonic analysis of music”, IEEE Transactions On Audio Speech And Language Processing, 2012.
6. Nicholas Paul Borg, George Hokkanen, “What Makes For A Hit Pop Song? What Makes For A Pop Song?”, 2011.
7. Pham, J., Kyauk, E. & Park, E, “Predicting Song Popularity”, 2015.
8. https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/
9. https://scikit-learn.org/stable/_downloads/scikit-learn-docs.pdf
10. https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052
11. Hareesh Bahuleyan, “Music Genre Classification using Machine Learning Techniques”, 2018 (Baseline for Genre Prediction).
12. Minna Reiman, Philippa Örnell, “Predicting Hit Songs with Machine Learning”, 2018.

7. Appendix:

```
/
├── main.py
├── dataset_constructor.py
├── train.py
├── validate.py
├── test.py
├── audio_utils.py
├── download_models.sh
├── data
│   ├── data_genre_training.csv
│   ├── data_genre_validation.csv
│   ├── data_genre_test.csv
│   ├── data_hit_training.csv
│   ├── data_hit_validation.csv
│   └── data_hit_test.csv
├── audio_samples
│   ├── 0_Hip_hop_music.wav
│   ├── 1_Reggae.wav
│   ├── 2_Rhythm_blues.wav
│   ├── 3_Rock_music.wav
│   ├── 6_Vocal.wav
│   ├── 7_Pop_music.wav
│   └── 12_Techno.wav
├── results
│   ├── eval_metrics_training.csv
│   ├── eval_metrics_test.csv
│   └── conf_matrices
│       ├── genre_naive_bayes_cm.jpg
│       ├── genre_logreg_cm.jpg
│       ├── genre_svm_cm.jpg
│       ├── genre_random_forest_cm.jpg
│       ├── genre_xgboost_cm.jpg
│       ├── hit_knn_cm.jpg
│       ├── hit_logreg_cm.jpg
│       ├── hit_svm_cm.jpg
│       ├── hit_random_forest_cm.jpg
│       └── hit_xgboost_cm.jpg
└── ipynbs
    ├── Music_Genre_Prediction.ipynb
    ├── Hit_Predictor.ipynb
    └── EE660_Project.ipynb
```