**MUSIC RECOMMENDER SYSTEM**

**-BY TEJAS RAJ**

ABSTRACT

Recommender systems have become a very important part of the modern entertainment where the users are constantly looking for new content that is similar to others that has been consumed by the user. Recommender systems are used to recommend new movies and tv shows by video streaming platforms like Netflix, YouTube and lately even Facebook for their Facebook watch feature. Another use for recommender systems has been in the music industry by apps like Spotify and Apple Music to give music recommendations to their users based on the type of songs they listen to. For this project, I have tried to create a simple music recommender system that will firstly classify an input music into one of the 10 genres and then give you song recommendations from that genre.

*Keywords – Recommender systems; features; variables; observations; regression.*

INTRODUCTION

Recommender systems in the music industry are generally used in 2 ways – to classify similar songs together and give recommendations based on these similarities (Spotify, Apple Music), or as a product by itself to identify a song (Shazam, Snapchat). We will be using this system in the first way – to give song recommendations based on similarities with an input song.

Dataset:

The dataset that we will be using for this system is called GTZAN dataset. It is a data set containing thousand 30 sec .wav files that are divided into 10 different genres – 100 files per genre. The 10 genres that we will be classifying the songs into are – blues, classical, country, disco, hiphop, jazz, metal, pop, reggae and rock. As we can see that we already have the genres for all the songs in our dataset, this is a case of supervised learning.

Steps Involved

To do this there are broadly 3 steps that we need to complete. We will look at all these steps in more details as we go ahead, but here is a small introduction of all the steps that will be involved in creating such a recommender system –

* Feature Extraction – the files that we have in our dataset is in .wav format and to be able to run a classification on function on these files, we will have to extract the describable features from each of the files before we can start classifying them. We will be extracting a bunch of features from each file to be able to better capture the independent qualities of each file. We will store these features in a csv file.
* Training

We will try a few different classification models to test and see which model gives us the best accuracy for our dataset. We will also try different iterations of some of these models to see which one can give us the best accuracy. The dataset will be divided into training and testing data so that we can check for the fit of the model.

* Prediction

Once we have trained the model and selected the one which gives us the best overall fit for our dataset, we can then proceed with giving an input file that the model can then classify into one of the 10 genres. Based on the genre that the input file is classified into we will get recommendations for songs from that genre.

RESEARCH PROBLEM

This program is unfortunately theoretical at the moment as the dataset that we have does not contain songs, they only contain snippets of music from that genre. Due to this, the recommendations will not be real songs but just music that describes a particular genre. This problem could be tackled with more time to create a new dataset from scratch with publicly released songs along with their respective genres.

This recommender system gives recommendations based solely on the genre that the input song is classified into. This would be a more robust system if and additional classification step could be added, like also taking into account the number of streams a song has and other songs that users have listened to after listening to the input song. This would also require more time to create a dataset from scratch and finding the individual streams for a song.

RESEARCH OBJECTIVES

The objective of this research is to create a classifier model that can accurately classify songs into their respective classes or genres, and then give. We have already spoken briefly about the steps that will be involved in this research. Following are the objectives behind performing each of these steps –

* Feature Extraction

To train a model to classify songs into genres, we will first need to extract usable features from each file. There are various features that can be extracted from .wav files but the following are the ones that I have extracted to use as the independent variables in the classification algorithm –

* + Chroma Frequencies - chroma\_stft
  + Root Mean Square - rmse
  + Spectral Centroid - spectral\_centroid
  + Spectral Bandwidth - spectral\_bandwidth
  + Spectral Rolloff - rolloff
  + Zero Crossing Rate - zero\_crossing\_rate
  + Mel Frequency Cepstral Coefficients - MFCC

These variables are explained further ahead in this report. The dependent variable is the music genre that each file belongs to.

* Training

There are various classification algorithms that can be used to perform this task of classifying songs into genres. The important thing however is to chose a model that provides the best fit for our data. For this, we will be training various classification models and find the testing and training accuracy for each model. We can then choose the algorithm that provides the best fit for our data. The classification algorithms we will be using are –

* + Logistic Regression
  + Decision Tree Classifier
  + KNeighbours Classifier
  + Linear Discriminant Analysis
  + Gaussian Naïve Bayes
  + Support Vector Machine
* Prediction

Once we have a trained model with the best fit for our data, the next step is to extract features from the input song and then using the trained model predict the genre of the input song. Based on this predicted genre, we can filter through the original files to list all the songs that have the same genre as the prediction.

ANALYSIS

* Feature extraction

We have already discussed the features that we will be extracting from each file. Following is a little explanation about what they each are –

* + Chroma Frequencies – this records the entire spectrum of the 12 semitones existing in the musical octave
  + Root Mean Square – calculates the rms for every frame in the audio sample.
  + Spectral Centroid – this tells us the location of the centre of mass for an audio. It is obtained by calculating the weighted mean of the frequencies.
  + Spectral Bandwidth – this is a measure of the shape of the signals present in an audio.
  + Spectral Rolloff – computes the rolloff frequency contained in every frame of an audio sample.
  + Zero Crossing Rate – it is the rate at which the signals in an audio change from positive to negative and back.
  + MFCC – Mel Frequency Cepstral Coefficients or MFCC are a set of 10-20 features that record the overall shape of a spectral envelope. It captures the descriptors of the human voice. We are capturing a set of 20 features in MFCC.

These are all functions available in the librosa package which is widely used for audio information retrieval for music and audio analysis.

These features once retrieved from each file is then saved in a csv file with 2 additional columns – filename which contains the names of the songs and label which captures the genre of each song.

* Training

Once we have the data for each file, we can then proceed to train the model. For this, the dataset is first randomly divided into two parts, training and testing, in 80-20 ratio. The model is first built using the training dataset and then the testing dataset is used to validate how accurately the model can predict on a new dataset.

We can also check for underfitting and overfitting based on the accuracy of these two datasets.

* + Overfitting occurs when the prediction accuracy on the training dataset is high but the testing accuracy is very low. This signifies that the model is not very well generalised to adapt to new data.
  + Underfitting is when the training and testing accuracy are both very low. This usually signifies that the model was not able to pick up the important characteristics of the training dataset and hence is not able to predict well on the test dataset.

Figure 1 shows a visual representation of underfitting, overfitting and a good fit.

Now that we have pre processed the data and we have an understanding of how to choose the right algorithm for our data, let us train the model and compare the results.

* + Logistic Regression – this is one of the most commonly used algorithms for binary classifications. It estimates the relationship between the independent variables and the dependent variable.

Before running the algorithm, we should first standardise the data. Standardising the data ensures that all the variables are on a common scale, hence ensuring better comparability between different variables which may initially have been recorded in a different format.

After running the algorithm on the standardised data, we get a **training accuracy of 0.71 and test accuracy of 0.66.**

* + Decision Tree Classifier – this a type of non-parametric supervised learning. This algorithm predicts the value of the dependent variable based on a set of rules or decisions that the algorithm learns from the data. A feature of the decision tree is that we can manually set the depth of decision tree. The depth determines the complexity of the decision rules and hence the fit of the model.

In figure 2 we can see the training and testing accuracy at various depths. To ensure that the algorithm has learned enough from the data, we will choose a depth which has at least 0.65 accuracy. The difference between training and testing accuracy is the least at a max depth of 7, where we **have training accuracy of 0.66 and test accuracy of 0.42.**

* + KNeighbours Classifier – KNN is a form of non-parametric classification algorithm in which the prediction is made based on the number of nearest neighbours from the data point for which the class needs to be predicted. ‘K’ ascertains the number of neighbours we are taking into account. The nearest neighbours are determined by finding the Euclidean distance between the new data point to be classified and the observations closest to it. Suppose we decide to set K as 3. The algorithm will then compute the three observations closest to the new data point and then based on the majority class of those three observations determine the class of the new data point.

In figure 3, we can see that with KNN Classifier, we get a **training accuracy of 0.65 and a test accuracy of 0.28** with k = 2 neighbours.

* + Linear Discriminant Analysis – LDA focuses mainly on reducing the dimensions of a variable from a dataset while still retaining usable information from the data. To explain in easy terms, LDA tries to represent all the variables in a particular class as one. It then makes predictions based on the probability of a new input observation being in one of the classes and the class with the highest probability is the predicted class for the input observation.

After training the model using LDA algorithm, we get a **training accuracy of 0.67 and a test accuracy of 0.68.**

* + Gaussian Naïve Bayes – like LDA, Naïve Bayes algorithms also predicts classes based on the probability of the data point being a part of one of the classes, with the class with the highest probability being choses as the predicted class. However, a Gaussian Naïve Bayes is used under two circumstances – when the data is continuous and when it is assumed that the data follows a Gaussian, or a normal distribution.

Using this algorithm, we get a **training accuracy of 0.42 and a test accuracy of 0.45.**

* + Support Vector Machine – an SVM works by building a hyperplane that best divides the entire datapoint into its classes. The input data is then classified into one of these classes based on similarity to the observations in each class.

Using SVM, we get a **training accuracy of 0.28 and a test accuracy of 0.28.**

Now that we have found the training and test accuracies for all the algorithm, it is now time to choose one which provides the best fit for this dataset. Figure 4 plots these accuracies in a graph. We can reject Gaussian Naïve Bayes and SVM for having a very low accuracy. Decision Tree Classifier and KNeighbours Classifier have a very low test accuracy when compared with training accuracy which suggests overfitting in both these algorithms. We are hence left with Logistic Regression and LDA which both offer a suitable fit with a relatively high accuracy for both training and testing dataset. We can therefore select the algorithm with higher training accuracy, which is Logistic Regression.

* Prediction

Now that we have trained a model to classify the songs into their classes or genres, we can accept an input file that will be classified into one the genres using the model that was trained using the Logistic Regression algorithm. We can safely say that this prediction will be correct 66% to 71% percent of the time based on the accuracy scores we have generated earlier for this algorithm.

Once the algorithm finishes classifying the new input file, we get a predicted genre, ‘pop’ in my case. All that is left to do now is to filter the original csv file containing the names of all songs and their genres based on the predicted genre and then list the names of all songs that are present in that genre.

GRAPHS AND CHARTS

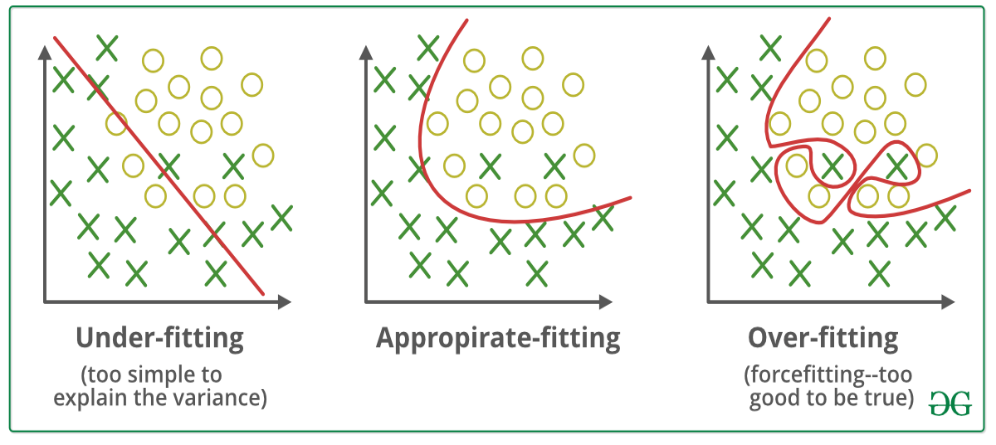


Figure 1

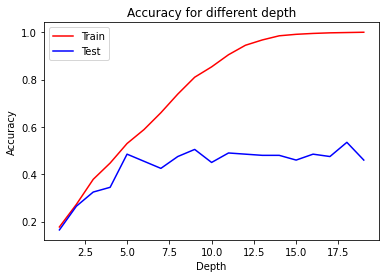


Figure 2

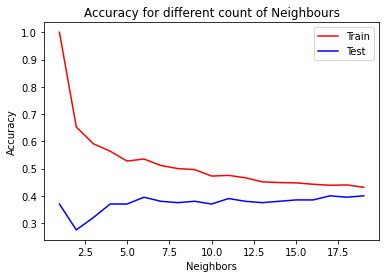


Figure 3

Figure 4

CONCLUSION

I will conclude by talking a little about classification algorithms and their importance when it comes to analysing data. Classification algorithms are a very important tool when it comes to analysing data. These algorithms can give us a much needed understanding of the dataset and can also help with classification predictions. Conducting this project has given me a lot of insight into how various classification algorithms work and under what circumstances different algorithms must be used.

This project also gives us a very good understanding of how to check the fit of a model for your data. Selecting the right algorithm that fits your dataset is paramount when it comes to effective predictions.

REFERENCES

1. <https://towardsdatascience.com/music-genre-classification-with-python-c714d032f0d8>
2. <https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/>
3. <https://scikit-learn.org/stable/modules/tree.html>
4. <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html>
5. <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>
6. <https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html>
7. <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>
8. <https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html>