Using Data to Find the Next Hit Album - Final Report

Record labels want to release albums that will be successful, but trying to determine what acts will be well-received is subjective and imprecise. Music blogs and reviewers play a critical role in the promotion and discovery of new groups. By understanding trends in what reviewers respond positively to, record labels can make more informed decisions about what artists have the highest chance of being critically acclaimed.

To solve this problem, we will look at the scores assigned to albums by music reviewers, and use a variety of attributes from the songs to attempt to predict these scores. Steps will include collecting the data from multiple sources (more below), performing EDA to determine what musical attributes are most relevant, and attempting to develop a model to determine what albums will be better received by critics. Two aspects in particular will be given attention: first, to the music's genre, to help understand what traits are most popular in different types of music; and second, to the context of time, looking at if certain traits become overused and lose their popularity over time, or are cyclical and have a resurgence after a period of unpopularity.]

The data to answer this problem lends itself well to answering a second question: based only on the musical elements of an album, can we teach a computer to predict what genre of music it is?

Our data comes from two sources. First, there is a Kaggle dataset on music reviews from Pitchfork.com, one of the most widely used music review websites. Second, we'll use the API of Spotify, a music streaming service, to access data on the individual songs and albums in the reviews from the Pitchfork data.

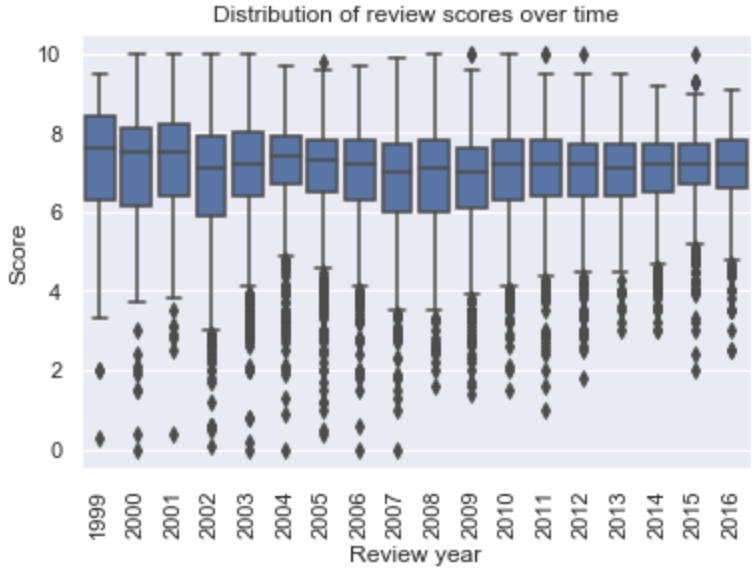
The Pitchfork dataset contains several variables useful for our problem. It contains information linked to roughly 17,000 album reviews, dating from the website's founding in 1996 (?) up through 2017. The most important piece of information from this dataset is the review score, a numeric evaluation of the album from 0 to 10. This score is going to be what we focus on modelling and predicting. The dataset also contains information about the album's release year and genre which we will use to help study trends over time and compare reviews within the same genre.

Our primary source of data comes through the extremely popular music streaming service Spotify. The Spotify API allows for access of musical attributes of all songs, including features such as key, time signature, and duration, along with other features determined by Spotify such as energy, acousticness, danceability, and many more. The package spotipy allows to access the spotify API through Python. Through this package, we can search for the albums contained in Pitchfork review dataset, find those albums in Spotify, access the song data for each song on the album, and then construct a variety of features to use in analysis based on the song data.

Using data from different sources created several challenges. In order to make use of the Pitchfork review data, we had to find the reviewed album in the Spotify and access it via the API. However, the only pieces available to join these datasets were the artist's name and the album name, which had several problems. First, there were differences in many album names between the two data sources. For example, Pitchfork would label any EPs (shorter collections of songs) by putting EP at the end of the album name, a notation Spotify does not use. Albums which were live, remastered, or rereleased often had different naming notations between the two sources. To handle this, text manipulation is done on the Pitchfork album names. Regular expressions are used to remove troublesome phrases like EP, as well as text such as 'remastered' or 'rerelease'.

One limitation is that using Spotify's API is only able to return so many potential matches, and so obscure artists might not be returned correctly because searching using their name does not return a match before the Spotify search limitation. To combat this, matching albums between the two sources is done using a combination of two methods. First, we search by album name, and check to see if the artist name also matches. Second, we search by artist name, and then look through the artist's discography on Spotify. These two different methods succeed on different albums, so by using both methods and combining the results, the number of matches is maximized. As a result of all the preparations and methods, we end up successfully matching around 12,000 out of 17,000 albums.

Having successfully cleaned and combined the Pitchfork and Spotify data, the analysis can begin! When exploring the data, several interesting patterns and trends emerged. Since we are most interested in the review scores, we can examine how those scores are distributed over time, and find an interesting pattern. Over time, while the average score hasn't changed much, but the distribution has clearly shrunk, indicating that Pitchfork has slowly shifted away from using more extreme scores.



This visual pattern is backed up by statistical inference. To study this, the data is grouped annually, and the standard deviation of review scores for each year is calculated. These standard deviations are then split into two groups: one for years between 1999-2007, and one for music from 2008-2016. A t-test can then be run to test the null hypothesis that the standard deviation for review scores is the same in the two time periods versus the alternative hypothesis that the standard deviation is different. Running this test produces a T score of 4.81, with a p-value of 0.00031, allowing us to reject the null hypothesis. The variance in review scores has definitely changed over time.

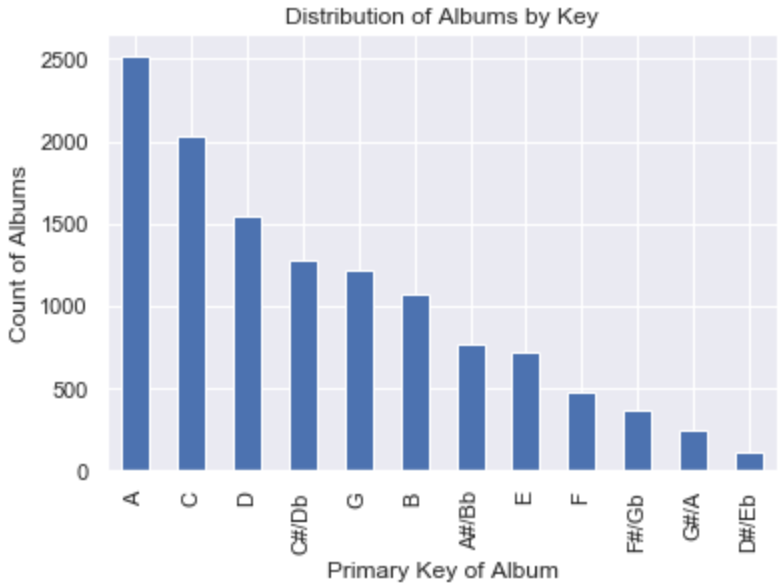
Taking a step further, we can look at the musical attributes that describe these albums reviewed through Pitchfork's history. The level of analysis for the data is the album, not individual songs, so all data has been aggregated to that level. From this aggregation, for each audio feature from Spotify, there are two primary features calculated; for each feature the mean and the standard deviation are calculated. The mean is used to see what sounds albums tend to have, while the standard deviation is useful to see how much variety is on the album - albums with high standard deviation in a feature have a lot of variance between the songs on the album, while low standard deviations for a feature signify that songs on the album are similar to each other, or lack variety, in regards to that feature.

Two such features with interesting trends are the energy and loudness of albums. The two are related - the loudness is a measurement of the overall loudness of each track measured in decibels, while the energy is a more nuanced measurement which quantifies the speed, volume, and general 'noisiness' of the track. Looking at how these two audio features have trended over time reveals an interesting insight:

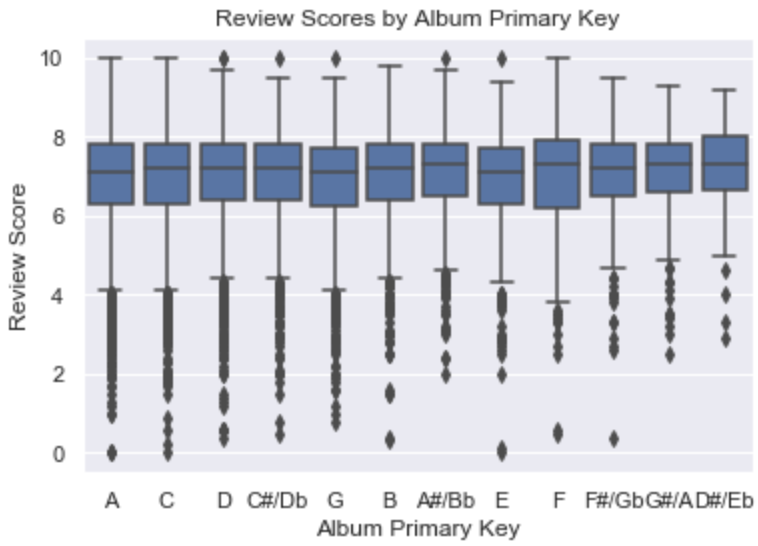
By using the similar process of splitting the data into two time periods and running a t-test comparing each of the values between the two periods. Each test ends up producing very large t-scores, ranging from values of 7.21 to 10.81. These trends are clearly substantial.

From the 90s up until the present, music has consistently been getting louder and more energetic, visible in the upward trend for the average energy and loudness. Not only this, but simultaneously, the variance of these features has been shrinking. In other words, not only has music been growing louder and more intense, but there has been less variety in regards to these features on albums. Not only is music getting louder and more energetic, but albums have had fewer slow songs to balance them out.

After looking at these trends in music, we can start to see how individual aspects of the music is related with the review score. One critical component in this is the key that the music is composed in. When it comes to composing music, not all keys are created equal - musicians have primarily favored a select few keys:



Critics have generally responded to each key equally. However, it is noteworthy that the less common tonalities have garnered a less extreme reaction. In the boxplot below, the boxes are arranged to match the frequency of the keys as seen in the plot above. When looking at the boxes, music composed in the less common keys has been reviewed similarly to music in the more common keys, but there are fewer albums with extremely low scores or with perfect high scores:



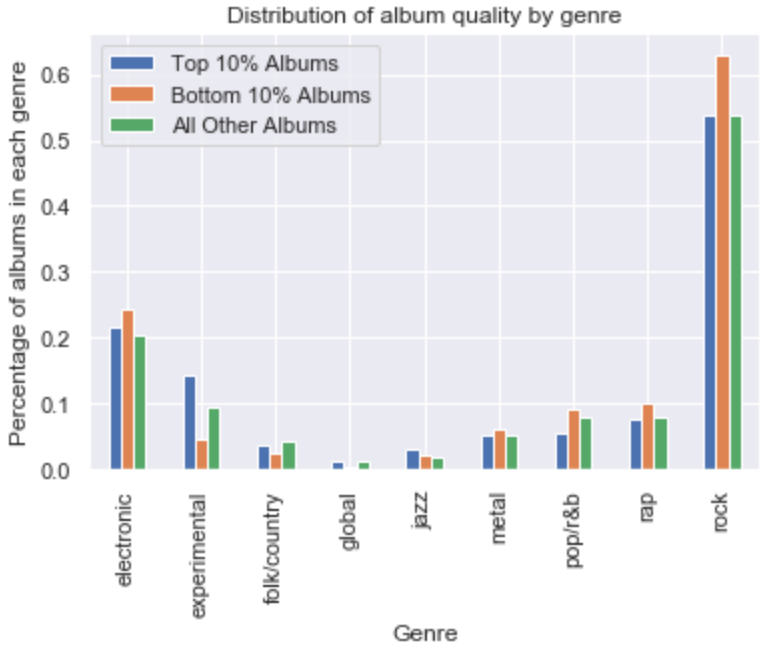
Conducting the F test with data from all keys generates an F-score of 2.58, with a p-value of 0.0028. From this, we can reject the null hypothesis and conclude that key can have an impact on review score. But we can go further than this. Looking at the counts of each key, it seems that roughly half of the keys are considerably more popular than the other half. We split the data into these two groups, with the cutoff of 'common keys' being any key with at least 1,000 albums composed primarily in that key, and any key with fewer than 1,000 albums categorized as an 'uncommon key'. First, since we're now looking at two groups (common versus uncommon keys), we can once again use a t-test to compare the two groups. From this, we obtain a T-score of -3.15 with a p-value of 0.0016, so it is clear the two groups are indeed different.

Next, we use F tests to compare only the common keys between each other and to compare only the uncommon keys between each other. When running the test on the common keys, we generate an F score of 1.08 with a p-value of 0.366, so we cannot reject the null hypothesis for these keys. However, running the test on the uncommon keys generates an F score of 2.81 with a p-value of 0.0156, so we can reject the null hypothesis for the uncommon keys. We interpret this to say that among the popular keys, the key does impact review score, but among the less popular keys, the key can impact review score.

One final way to examine the data is to break it into categories. Conceptually, albums that are considered to be all-time greats should be different from albums which are considered to be horrible. To conduct this analysis, the 10th and 90th quantiles for review score are computed, and from these quantiles the data is split into three categories - albums which are in the top ten percent, the bottom ten percent, and the albums which fall in between. As it turns out, using the audio features from Spotify, these three groups are actually quite similar - much like how there were no discernable relationships between the audio features and the review

scores, there are not any glaring differences in the distributions of the audio features across these three groups.

However, one interesting pattern that exists lies in the genres of the albums in these three different categories, as seen below:



Rock music is by far the most common genre of music reviewed by Pitchfork - just over 50% of all albums reviewed are tagged as being ‘rock’. However, an even higher proportion of albums are rock albums. Thus, it seems that it is much more common to write a bad rock album than any other genre. On the other side of the coin, experimental music tends to do comparatively very well, as there are many more experimental music albums in the top ten percent than average, and very few experimental music album in the bottom ten percent.

Since we’re interested in the difference between more than two groups, we’ll use the F test again. For each genre, we’ll run an F test comparing the number of albums in the top 10%, bottom 10%, and middle, under the null hypothesis that genre causes no difference between the number of albums in each quality category.

The results match the same conclusion we drew from the visualization. Some genres have a very large F score - Experimental and Rock music have F scores of 39.04 and 20.31, indicating there is a tremendous difference in quality categorization for these genres. Most other genres have relatively high F scores, although there are a few that we’re less likely to reject the null for; Rap and Jazz with F scores of 3.87 and 3.57, leading to p-values of 0.0209 and 0.0281 which would lead to rejecting the null at a confidence of 95% but not 99%, and Metal, with an F score of 0.96 and a p-value of 0.3833, we can’t reject the null for any conventional level of confidence.

Having done all of the analysis described above, we have a solid understanding of how the data has been collected, cleaned, and a variety of different trends and patterns within the data. This is an excellent foundation to begin applying machine learning algorithms to our data.

There are two topics we will address using machine learning algorithms. The first topic is using musical data to predict numeric review scores, from which we hope to learn what aspects of music are the most important for critical acclaim. The second topic is attempting to classify albums into genres using their musical attributes. In both topics, we will start by using relatively more simple ML algorithms, and then see if we can improve the models by tuning hyperparameters or selecting other models.

Our regression problem is as follows: using all of the information available about an album, what score do we predict critics to review the album with?

To answer this question, we will start with using linear regression. To help improve our model, we will make some transformations to the data. We will square the review scores and attempt to predict this score squared value, which is done so that the variable we are trying to predict follows a more normal distribution. Additionally, we will scale all of our variables so that they have a mean of 0 and standard deviation of 1. These changes help the models run more smoothly, but also help us interpret the results more easily when all of the variables are set to the scale and thus are easier to compare. Finally, for the regression problem, we will separate the data by genre and create models for each genre independently, so that differences between genres are properly captured.

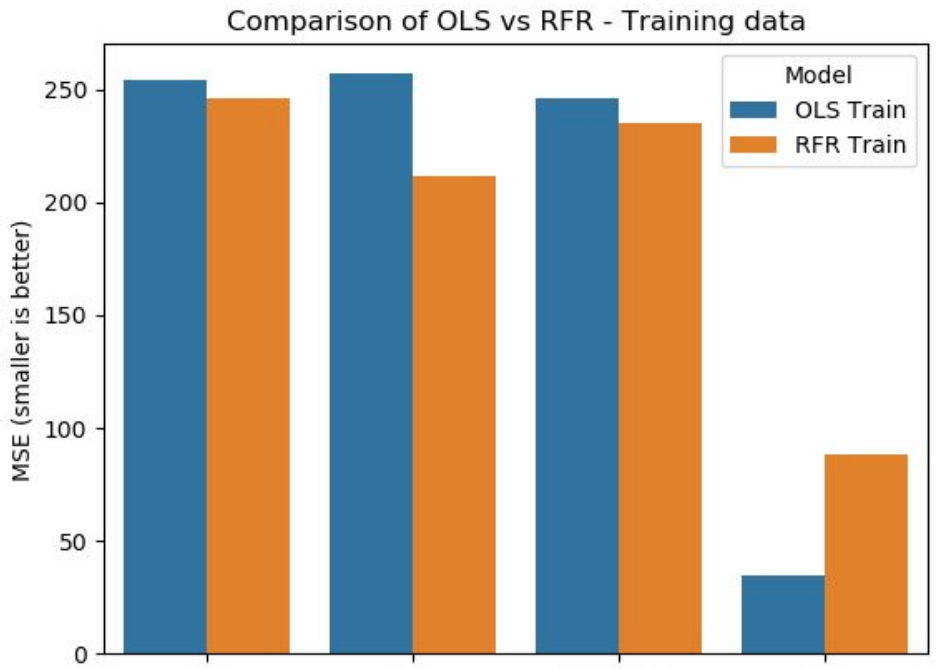
For the linear regression models, there are a few elements we will focus on. First, we can check the performance of each genre's model by looking at the F-score and adjusted R squared to see how well the model handles the data. Second, we can look at the coefficients and p-values associated with each variable in the model to see what lessons can be learned about the importance of individual musical attributes (ie, for each genre, which musical attributes are found to be significant and in what direction).

For the overall performance of our linear regression models (plural because there is one model for each genre), our results are mixed. Looking first at the overall F-score, which checks to see if at least one variable is statistically significant, nearly every model is significant at alpha = 0.05, and some have very high F scores. However, for all models, the adjusted R squared is very small - the highest, metal, has a value of 0.11, meaning only a tiny amount of the variance in review scores is successfully explained by our model. These adjusted R squared values are smaller than 0.1 for every other genre, showing how limited the predicted powers are for these models.

While the models overall do not have much predictive power, we can still learn from them. There are a few variables for some genres with sizeable coefficients. From these, lessons can be learned about what critics respond positively to across different genres of music.

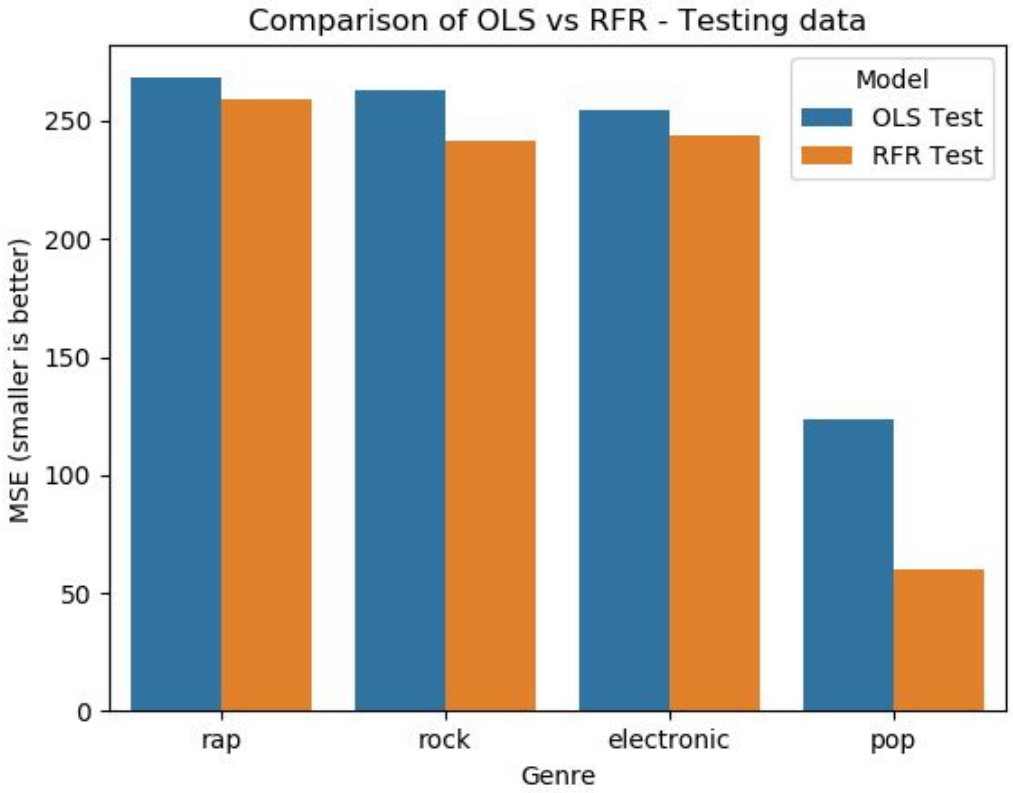
Genre	Positive Attributes	Negative Attributes
Rock	• Acousticness • Instrumentalness • Valence	• Danceability • Minor Key
Metal	• Instrumentalness	• Danceability
Jazz	• Speechiness	• Composed in the key E
Folk/Country	• Acousticness • Minor Key	
Experimental	• Longer duration • Albums with variety in acousticness	
Electronic	• Instrumentalness • Albums with variety in danceability	

Having fit OLS models to each genre, we'll now compare how OLS does with a more sophisticated machine learning algorithm - Random Forest Regression. First, for each genre, we'll use sklearn's GridSearchCV to test a wide variety of hyperparameters on our training data, selecting the set which does the best for each genre. Then, to compare the predictive power of the RFR models with the OLS models, we'll calculate a measure of performance in the Mean



Squared Error (MSE), and compare. Since MSE is a measurement of model error, smaller values indicate a better model.

On the training data, RFR outperforms OLS on three genres, but doesn't do as well on pop music. From this, we could either choose to use the RFR for all genres, or use OLS on pop music and RFR for the other genres. I would lean towards using RFR for all genres, as conceptually it seems more likely that OLS' very low MSE on pop music is a quirk of the training data, and may not necessarily do as well on data the model wasn't trained on. Looking at the test data, RFR ends up doing better on three genres, but does worse on electronic music. Again, I'm inclined to believe that, given a large volume data our model hasn't seen, RFR would outperform OLS in all genres.

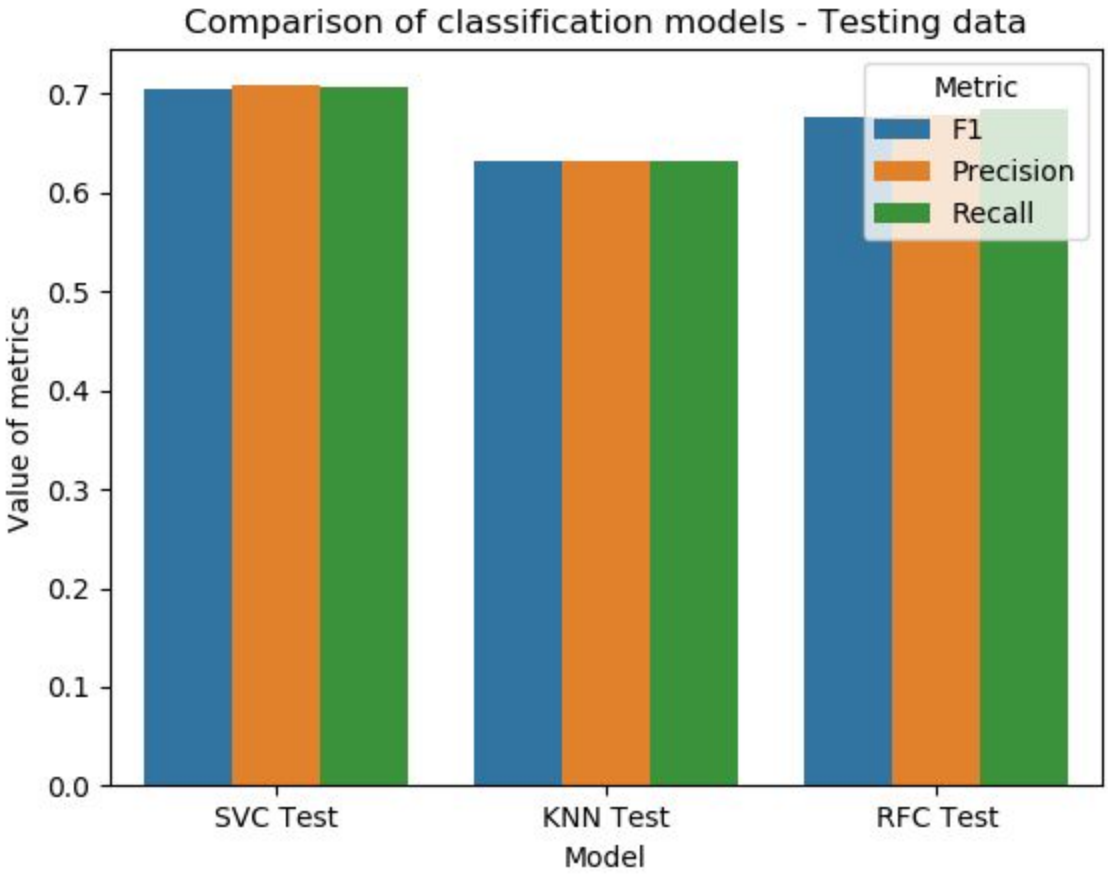


Next, we'll turn our attention to the classification problem. In this, we want to use all of our musical data to predict what genre an album falls into. We'll attempt a few different models and see what ends up handling this problem the best. The models that we'll use will be Support Vector Classification, K-Nearest Neighbors, and Random Forest Classification. For each of these, we'll use sklearn's GridSearchCV to test a wide variety of hyperparameters for each model, and establish what set of hyperparameters is the best for each model. Then, we'll compare the performance of our three model types to determine which is best suited for our task.

First, there is some data preprocessing necessary for this specific task. As was the case in the regression context, we'll scale all the numeric variables to be standardized to have mean 0 and standard deviation 1. The primary challenge created by the data is that while there are nine genres of music in the data, some genres have far more music than others. This creates a situation where, since a majority of the data is rock music, models can predict 'rock' far more than anything else, and end up correct not because it is finding insight in the data, but rather because most of the music in our data happens to be rock. In order to handle this, we'll keep only the top four genres, which each have at least 500 albums each, in for our analysis. Then, we'll sample 500 albums from each genre, so that we have the same amount of data for each genre. This is the data we'll use for the test/train data split, and for all of our classification.

To compare the three different models, we have a combination of metrics to use. Since we have four classes, we cannot create an ROC curve or use AUC. Instead, we'll look at accuracy, specificity, recall, and F score. After evaluating these metrics for each model, we'll make a selection about which has the overall best performance.

We'll use the testing data to compare models, as using random forest classification on the data it was trained on does not lead to accurate results.



On the test data (which the model was not fit on), support vector classification ends up doing the best of our three models, and so we will continue with SVC as our model of choice. To see how our selection ends up doing, we can look at a confusion matrix to see where the problems are arising. Ideally, we want to see very large numbers on the main diagonal, and zeroes everywhere else.

	Predicted Electronic	Predicted Pop/R&B	Predicted Rap	Predicted Rock
True Electronic	145	18	4	26
True Pop/R&B	33	91	9	61
True Rap	12	11	182	4
True Rock	17	37	2	148

From this, it is clear that the model handles some genres better than others. Pop/R&B gives the model the most trouble, which may be due to it being a broader label for a genre than the others. Outside of pop/R&B, the model has some trouble differentiating between electronic and rock albums, but has almost no trouble at all with correctly identifying rap music. But ultimately, it seems the model does a respectable job classifying albums into genres.