

# **Music Genre Classification: Classifying the genre of a song or piece of music from audio signal feature engineering**

## **Introduction:**

Music is something that almost every individual can relate to. With the wide variety of tunes that exist, it can be daunting when one also considers the number of genres that exist. For example, the Free Music Archive contains over 160 genres [1]. The basis of our topic stems from our passion and interest towards music.

In general, music information retrieval serves as the underpinning of modern music software and collection systems. Many music services such as Spotify and iTunes rely on accurately classifying music metadata, such as artist, album, and genre [2]. Accurate descriptions of these fields allow for users to easily search music collection, and furthermore inform recommendation systems such as the one that Spotify uses in the auto generation of daily music playlists.

Despite the need for accurate genre classification, it is not always the case that genre can be so cut-and-dry. Additionally, these music services provide their own measures and features in terms of examining and ultimately classifying music. The problem arises when we consider how the classification is based. Fundamentally, a genre can be a subjective and somewhat controversial topic. What one person might classify as a country song another might classify as a rock song. In some cases, the genre of a piece might be classified solely based on the artist or where they are from. Our aim is to essentially bypass this by classifying solely based on information contained within the audio. By basing our music solely on audio, that is, .wav and .mp3 files, we aim to evaluate and create our own models by considering and optimizing unique parameters of each model.

In this work, we evaluate the performance of machine learning classifiers (support vector machine classifiers, k-nearest neighbors, multinomial logistic regression, convolutional neural networks) [2] [3] trained solely on features extracted from the audio data of music files to classify genre. We evaluate the performance of additional features on our models and observe the effect that different datasets and genres have on the model performance.

## **Background and Motivation:**

An ongoing debate in ethnomusicology, and perhaps somewhat more so in the public and blogospheres is whether or not music is universal [4] [5]. The result of this universality – or perhaps lack thereof – is the creation and perpetuation of multiple genres of music. What exactly separates genres is not always clear, however.

The problem becomes particularly acute when we are presented with new genres. For example, what exactly is “Chiptune”? When presented with this genre, one might be led to believe that it is solely composed of Alvin and the Chipmunk tunes, and maybe Alvin and the Chipmunk

tribute bands. Chiptune is actually a type of electronic music based off of synthetic video game music produced in the 1980s [6]. However, if we were presented with a song, either by a friend or a music recommendation system, that only listed the name of the song, the artist, and the sub-genre, how would we know in this case what this type of music would be? The problem is especially acute when we are unfamiliar with either the artist or the genre. In our chiptune example, would a lover of electronic music unfamiliar with the genre even consider to try listening to the piece?

As another example, consider the case where the artist is known but the song is new or unfamiliar. For example, consider reggae influenced song “D’yer Mak’er” by Led Zeppelin, an English rock band. If we have never listened to the song before, but know that Led Zeppelin was a rock band, we might expect the song to be typical 70s classic rock. But, is it something different? Perhaps it’s closer to reggae? Or something in between?

Herein lies the problem on relying on music file metadata solely as the basis for a recommendation system. The result is a lack of clarity into what the top-level genre of a piece of music is, and whether the genre listed for the piece (or artist) is the actual genre for that piece of music.

The solution to bypass this issue is to instead base the genre solely from the audio of the piece of music. Humans are quite good at doing just this. On average, human genre classification based on the audio alone is around 76% [7]. An even better solution would be to train a machine learning model that can classify the genre of a piece of music based solely on the audio alone.

### **Literature Review:**

For the purposes and background of our study, we found several papers that we thought stood out in terms of music information retrieval. The first of these papers we wish to discuss here is, “Music Genre Classification using Machine Learning Techniques” by Hareesh Bahuleyan at The University of Waterloo [8]. This paper offers our basis of comparing machine learning models using hand-built features from audio sources to convolutional neural nets (CNNs) trained on frequency-spectra images. In our research, we are taking audio files and essentially using a rolling-window (short-time) Fourier Transform to transform the time data of a certain audio file into frequency data. This paper offers insight for us as it shows us how to take audio files in the time domain and then convert it into the frequency domain. We realized, after reading this paper, that we can utilize the spectrogram as the spectrogram can take frequency to visualize specific characteristics of music. The time domain alone is limited to features such as amplitude, and tempo. The frequency domain allows us to visualize things such as intensity for frequency bands, pitch, and other spectral features.

The next paper we gained insight from is, “Unsupervised Learning of local features for music classification”, by Jan Wulfinf and Martin Riedmiller at the University of Freiburg [9]. This study provided us with our inspiration towards utilizing a linear SVM for classification. Alternatively,

in 3-D spaces, the data follows a clustering approach such as a k-means approach as suggested in [9]. For this reason, unsupervised learning may also be utilized.

In [10], a study was performed to build a Convolutional Neural Network (CNN) to not only recognize genre but also recognize artist and beat detection. This literature dives a little deeper into supervised learning and unsupervised learning, namely which aspects of the model perform better when training the data set in either fashion. It undertakes the problem of training the entire data in a real-world scenario as it is not practical to train the whole of dataset in a strictly supervised fashion. Although using unsupervised pre-trained data improves convergence speed in all aspects and in certain instances improves accuracy of recognizing the artist, our study on the other hand is utilizing the supervised approach. Moreover, our approach utilizes other statistical analysis as to how the model performs when compared to other models as we are only recognizing a genre and our dataset differs from the dataset utilized in their study.

### **Approach and Methodology:**

We inform our attribute selection and feature engineering given the process described in [8]. Attributes are not formed by the metadata of the audio file, but rather formed through feature engineering of the raw audio signal. We use the LibROSA Python library to extract the audio signal from either .mp3 or .wav files (depending on the dataset). We can then use the same library to present the audio signal in both the time and frequency domains.

The time domain features include the central moments of the amplitude, the zero-crossing rate, the root mean square energy, extracted using LibROSA as described in [8]. Intuitively, the use of these features makes sense. We might consider the average amplitude to differ among genres, and the zero-crossing rate be a useful substitute for evaluating timbre. The root means square energy is also a good discriminant; we might believe metal to have a greater root means square energy than folk music.

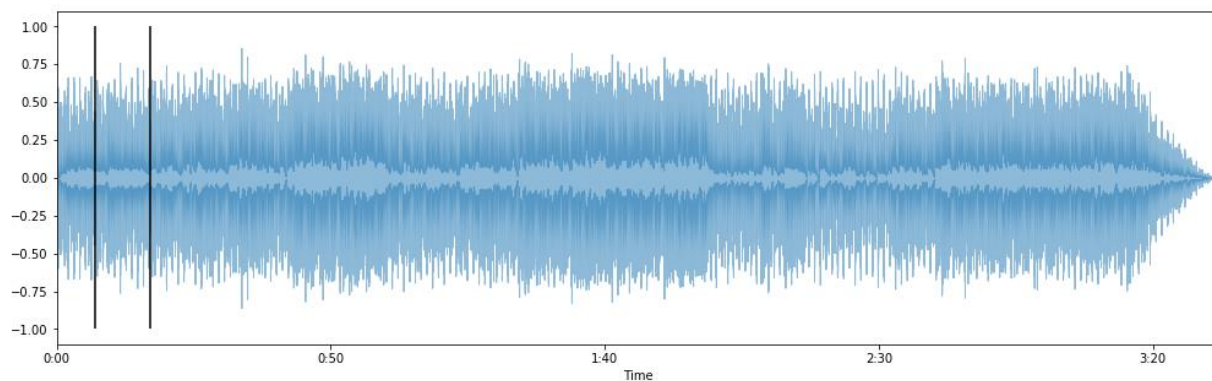
Where we differ in the approach to feature engineering from the time domain in [8] is in the use of tempo. In [8], the tempo is measured as the mean tempo of the aggregate tempo for each 10-second audio sample. The tempo of a piece of music is not necessarily constant, and the mean tempo of a piece does not reflect time-signature changes or changes in tempo. In fact, changes of tempo might be a useful discriminant for genre, as we might consider jazz music to feature many instances of variable tempo. For this reason, we propose altering our measure of tempo, by including the central moments of tempo to better reflect this variance. In addition, as we will be considering the entire length of pieces of music, we include an additional feature measuring the length of the music.

For feature engineering in the frequency domain, we follow the same features as suggested in [8]. These include the mel-frequency cepstral coefficients, chroma feature vectors, spectral centroid, spectral bandwidth, spectral contrast, and spectral roll off. We calculate these using the LibROSA library. For each of these, we include the mean and standard deviation. While we

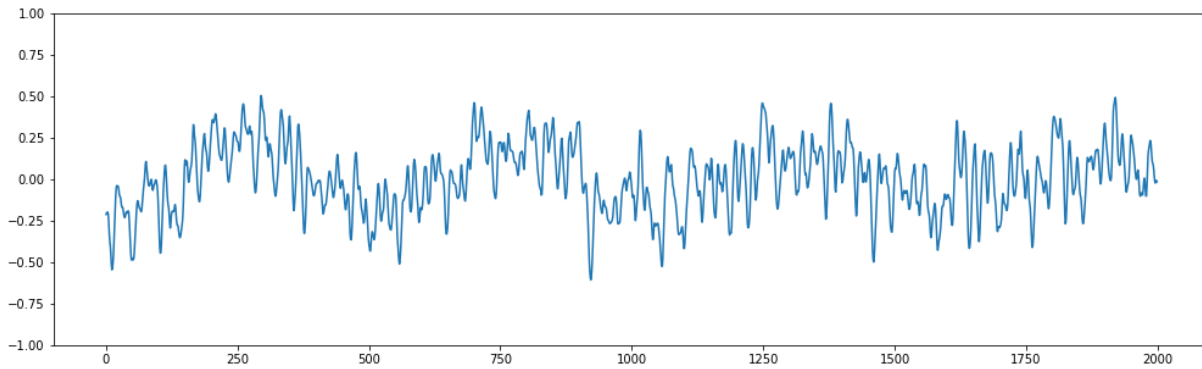
do not differ in our selection of these, we do wish to make some comments on their applications and limitations. As described in [8], chroma features are a vector which correspond to the total energy of the audio signal in each of the twelve possible pitches corresponding to the western music scale (i.e. C, C#, D, D#, etc.). Intuitively, we expect this to be a useful metric for discriminating among the key of music pieces. For example, there could be a greater prevalence in minor keys with drop-d tuning for rock and metal. However, this also highlights an important limitation in the use of this feature. For pieces of music that do not follow the western music scale, this metric will not work now. For our use, it is fine to include this metric as the majority of our samples consist of western popular music. For training on the free music archive dataset and testing against our own music collections, we will need to check to determine the prevalence of pieces that do not follow the 12-pitch scale.

Lastly, for the CNNs, we will be training on spectrograms of the audio signal, which is a 2D representation of the frequency over time of the piece. Color is used in the spectrogram to encode the amplitude of each frequency bin. We believe that this approach will have the greatest accuracy given sufficient data sample size, but to the detriment of model explainability and simplicity [2].

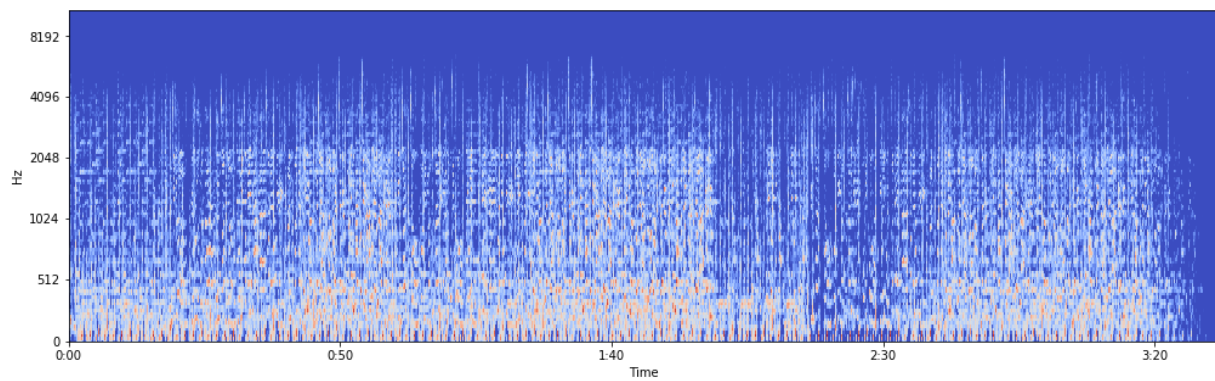
The evaluation process begins with the processing of the individual audio files in our datasets. Each audio file consists of either an independent .mp3 or .wav file. We extract the time domain data from these files using the LibROSA Python library [11]. An example of a plot of the extracted waveform for a song is shown in Figure 1. The time domain attributes can be calculated using LibROSA from this waveform alone. Next, we perform a short-time Fourier transform (STFT), also using the LibROSA library, to get the frequency domain of the signal. From the STFT, we can calculate the remainder of the spectral features with the LibROSA library and compute the log-mel spectrogram, which we use to train the CNN models. An example of a spectrogram computed from the same input audio file is shown in Figure 3.



**Figure 1: Amplitude waveform extracted from .mp3 using LibROSA of Rick Astley's *Never Gonna Give You Up*. [12]**



**Figure 2: Zoomed in view of waveform between vertical cursors in Figure 1.**



**Figure 3: Log-Mel Spectrogram of signal from Figure 1.** Heatmap color scale – red refers to greater signal intensity. We utilize .png images produced from these plots to train our CNN.

Our feature engineering completed, we can begin training our models for SVM classifiers, kNN classifiers, logistic regression, and k-means. For the CNN, the spectrograms are sufficient to train on - no additional attributes will be needed for training the model. Our data processing and modeling pipeline is summarized in Figure 4.

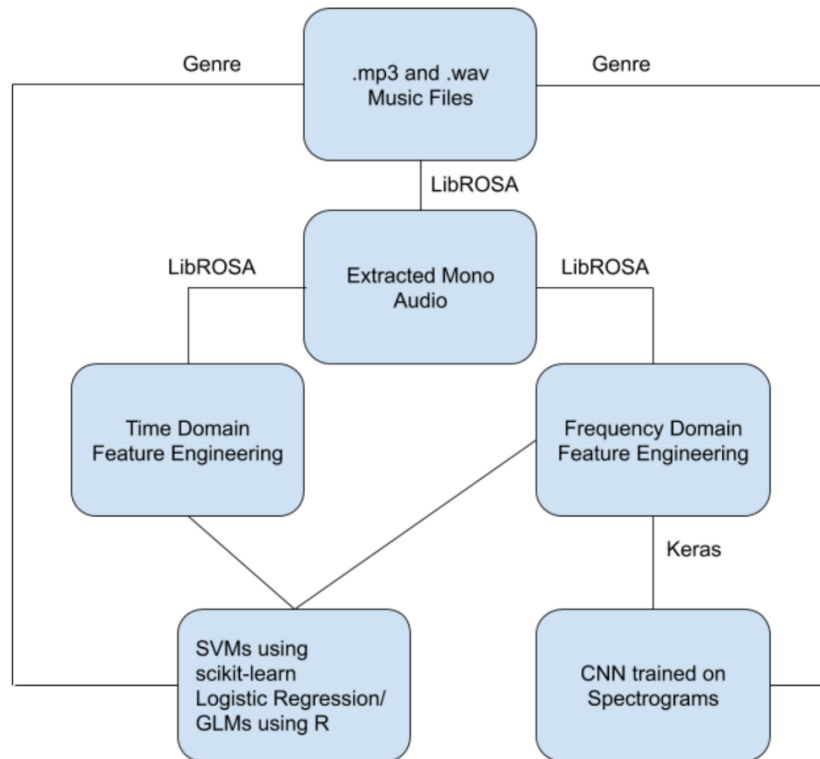


Figure 4: Data processing, feature engineering and modeling pipeline.

Our data come from two separate sources, the canonical GTZAN dataset for learning from audio music files, and the FMA dataset, a collection of pieces scraped from the Free Music Archive. The GTZAN dataset was initially created in 2002 for use in learning from audio files [13]. Several papers related to learning from audio files rely on comparing their model performance based on training from this dataset. The dataset consists of approximately 1.2GB worth of audio files (.wav); 1000 audio tracks each 30 seconds long. These data consist of 10 genres, with each genre having 100 tracks. The GTZAN dataset is problematic, however, for training on. First, the dataset consists of a limited number of audio tracks for each genre, especially when considering in proportion to the total population of audio tracks. Each audio track is also limited in size to exactly 30 seconds. While this certainly cuts down on memory and computational requirements for training on this data, it is not necessarily representative of the population. Consider for instance, classical music, or even progressive rock, for which pieces of music might last for several minutes (or even an hour). A 30 second sample from this piece of music is not necessarily representative of the overall piece, especially if it consists of multiple movements, tempo changes, and changes in dynamics. Second, the corpus consists solely of the researcher's own personal music collection from the early 2000s. Additional criticisms of the GTZAN dataset for building learning models from audio data are presented in [14].

To evaluate the validity of these criticisms and debias the data from a researcher's personal music tastes, we look at training models on the FMA dataset [1]. The entire FMA dataset consists of roughly 917 GiB of full-length music audio data from 106,574 tracks, 16,341 artists, and 14,854 albums, arranged in a hierarchical taxonomy of 161 genres. Due to computing

resource constraints, we utilized a smaller subset of the corpus which consists of 8000 pieces limited to 30 seconds in length balanced across 8 genres.

After performing the feature engineering for each dataset, we had to identify the individual files that the attributes came from with their respective genre. For the GTZAN dataset, this was as simple as performing string parsing on the file name, as the genre was included in the filename. For the FMA dataset, more work was involved. It was necessary to look up the file ID in the track metadata for the entire corpus, and then use string parsing to extract a vector of genre IDs. Following this, we utilized a genre lookup table which allowed us to collapse the list of genres to a single top-level genre that matches the top level genres on the Free Music Archive. We found post-feature engineering that 5 files in the FMA dataset were corrupted, so we removed these as we had no feature engineering results for them. This left us with 7995 instances in the FMA dataset. The breakdown of the number of pieces per genre is shown in Figure 5. One issue in comparing results from the two datasets can also be seen in Figure 5, namely that not all of the genre types overlap.

FMA		GTZAN	
Pop	1000	rock	100
Instrumental	1000	reggae	100
Hip-Hop	1000	pop	100
Rock	999	metal	100
International	999	jazz	100
Folk	999	hiphop	100
Experimental	999	disco	100
Electronic	999	country	100
Name: genre, dtype: int64		classical	100
		blues	100
		Name: genres, dtype: int64	

Figure 5: Left – Number of songs per genre in FMA dataset. Right – Number of songs per genre in GTZAN dataset. For the FMA dataset, 5 files were corrupted resulting in missing data during the feature engineering process. As a result, these songs were dropped from the dataset during modeling.

Figure 6 shows an example of some instances from the constructed CSV file of our features post feature engineering for the FMA dataset. We utilized the summary statistics for each attribute to ensure that ranges of values were what we expected. For example, we expected the mean tempo to be positive and between 0 and 300 bpm, and the amplitude to be between -1 and 1 with arbitrary units.



	names	x_mean	x_stddev	x_kurtosis	x_skew	azt_mean	azt_stddev	azt_kurtosis	azt_skew	rms_mean	rms_stddev	rms_kurtosis	rms_skew	tempo	mean_dynamic_tempo	stddev_dynamic_tempo	kurtosis_dynamic_tempo	skew_dynamic_tempo	length	mfc_mean	mfc_stddev	mfc_kurtosis
0	fma_instr004 U004549.mp3	-5.540000e-05	0.094732	1.405296	0.109443	0.036622	0.015446	0.368767	0.007812	0.081343	0.048523	-0.508646	0.400302	129.199219	116.057249	13.523365	-1.446340	-1.446340	29.976576	-4.190333	78.650864	10.942378
0	fma_instr000 U000002.mp3	3.131910e-04	0.166016	3.161952	-0.008733	0.086366	0.068404	3.496868	1.825609	0.141235	0.087223	1.256372	1.295190	166.706669	151.252396	24.971430	-0.169339	-0.169339	29.976576	-1.235037	44.951669	7.116079
0	fma_instr043 U043842.mp3	5.900000e-06	0.096036	0.683987	-0.072567	0.030740	0.011371	3.341493	1.193829	0.094412	0.030400	0.230696	0.321780	112.347147	111.863700	3.952476	-0.691430	-0.691430	30.002696	-4.814893	68.705233	8.471082
0	fma_instr024 U024749.mp3	6.840000e-06	0.002337	4.743158	0.010256	0.150333	0.045382	0.262682	0.407067	0.002062	0.001943	0.297424	0.661104	126.040018	127.832444	5.219731	11.308176	11.308176	29.976576	-20.694630	120.823732	13.810281
0	fma_instr031 U031390.mp3	3.070000e-07	0.095171	0.776142	0.063080	0.018225	0.011794	33.303192	4.037206	0.088375	0.033313	-0.408662	0.473642	123.046875	123.287199	0.814549	7.374971	7.374971	30.002696	-5.528642	77.496249	9.437461
	mfc_skew	chroma_mean	chroma_stddev	chroma_kurtosis	chroma_skew	spectra_centroid_mean	spectra_centroid_stddev	spectra_centroid_kurtosis	spectra_centroid_skew	spectral_bandwidth_mean	spectral_bandwidth_stddev	spectral_bandwidth_kurtosis	spectral_bandwidth_skew	spectral_contrast_mean								
-2.513902	0.381312	0.294891	-0.476253	0.849955	862.814190	196.541081	4.678505	0.575661	1235.504267	174.546616	5.394988	1.574932	24.180353									
-0.488627	0.637587	0.207228	-0.821492	0.118739	1842.291742	700.993537	1.136937	1.221557	1748.564847	299.211384	0.372977	0.391007	20.072165									
-1.469262	0.509667	0.253366	-0.765644	0.305868	875.664850	320.523096	3.363020	1.693295	1323.632559	372.352087	1.238664	1.223722	21.867556									
-3.708112	0.691191	0.192113	-0.665367	-0.178773	2328.595021	563.682060	0.286272	0.750030	2208.625286	361.567081	-0.501479	0.513081	18.372763									
-2.172769	0.567915	0.230850	-0.841282	0.361873	637.609551	232.021707	25.650214	3.509963	1181.232092	213.339511	7.426956	1.819323	22.790993									
	spectral_contrast_stddev	spectral_contrast_kurtosis	spectral_contrast_skew	spectral_rolloff_mean	spectral_rolloff_stddev	spectral_rolloff_kurtosis	spectral_rolloff_skew	spectral_flatness_mean	spectral_flatness_stddev	spectral_flatness_kurtosis	spectral_flatness_skew											
8.453981	-0.408714	0.417988	1542.897371	454.278915	9.500656	0.801681	0.000094	0.000235	218.012832	12.258998												
12.262117	1.607168	1.661760	3709.932382	1143.797297	-0.349026	0.461345	0.000956	0.001826	45.821934	5.722371												
8.736492	0.394949	1.021540	1737.392125	1053.824911	3.188415	1.893030	0.000118	0.000236	77.033202	7.394355												
8.992594	-0.565770	0.617952	4919.098463	1225.944539	-0.639003	0.254632	0.010913	0.016450	26.958460	4.130177												
7.954318	-0.094249	0.647226	1153.325856	582.813426	11.549484	2.480383	0.000082	0.000251	218.690556	13.352615												

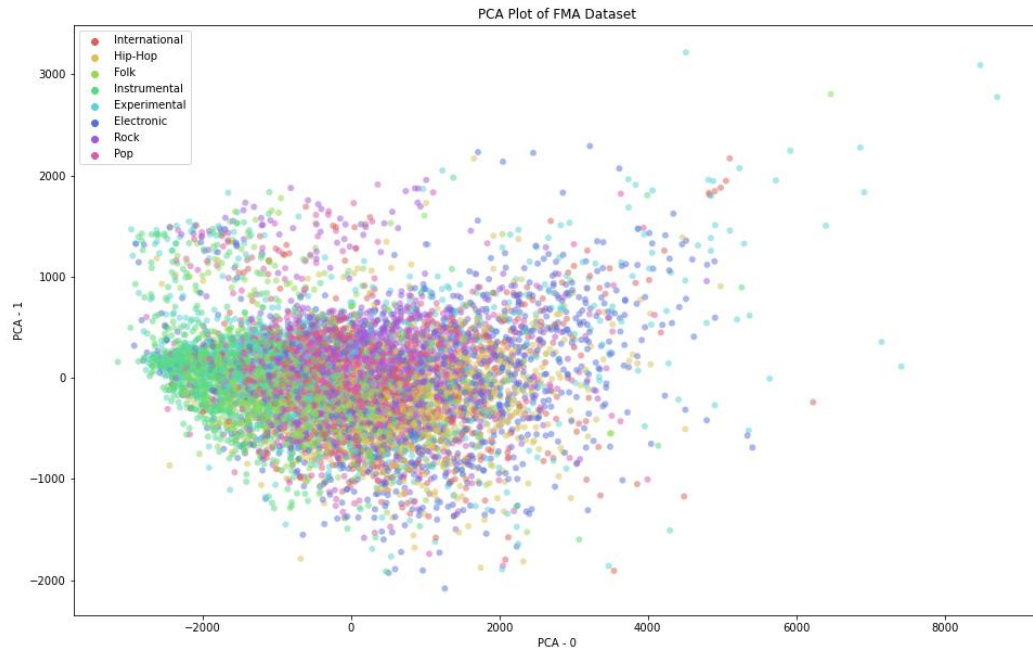
**Figure 6: Pandas DataFrame Head for the FMA dataset.** Example of the finalized .csv file loaded in a Pandas DataFrame following feature engineering, data cleaning, and genre joining for use in modeling.

Our basis for selecting the best model (and overall model comparison) was accuracy. This is due to the overall cost of a misclassified piece of music not being high. A rock piece misclassified as hip-hop does not have serious consequences. In fact, it's possible that it contains elements of hip-hop in it (as an example, consider Rage Against the Machine). What we are interested in is the performance and effect of our newly selected attributes, length, spectral flatness, and dynamic tempo, on the models that we build, and whether or not a difference exists in model performance depending on the dataset that it was trained on. To do this, we will consider t-tests on the accuracy of the model. We will check our data to see if there are any hidden groupings or clusters through visual inspection and k-means. Our machine learning models that we will evaluate are multinomial logistic regression, support vector machine (SVM) classifiers, and k-nearest neighbors (kNN). Lastly, we are interested in the performance of CNNs trained on the images of spectrograms from each dataset. For the CNNs, given our limitations in computing resources and small sample size for the GTZAN dataset will be mostly qualitative. For k-means and multinomial logistic regression, we use R. For kNNs and the SVMs we use Scikit-learn. For the CNNs, we use Tensorflow with the Keras API.

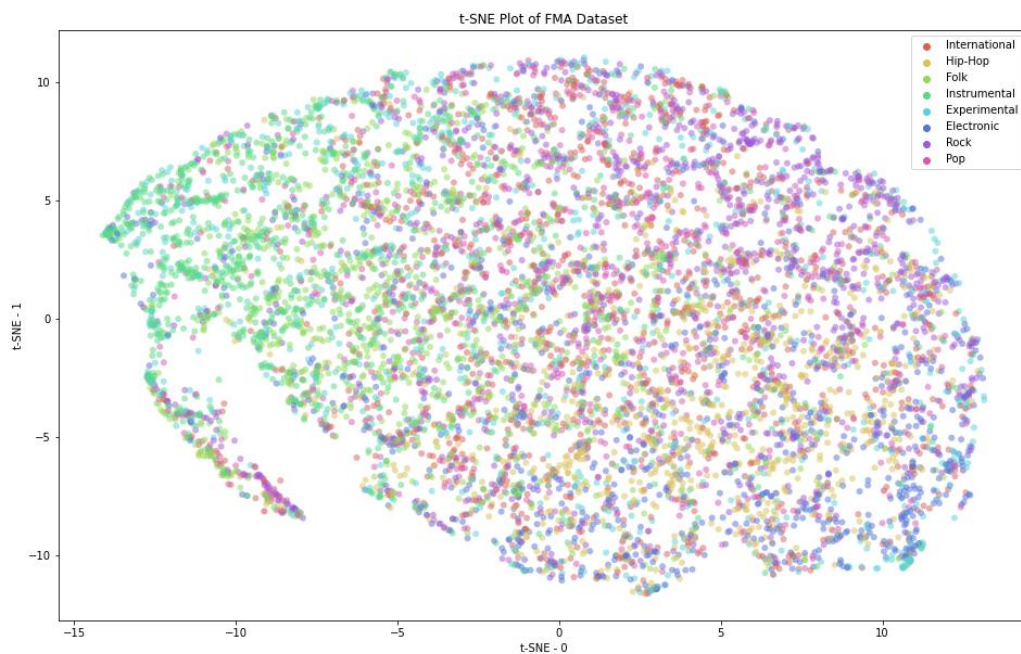
## Results

We conducted exploratory data analysis by first visualizing the data with scatter plots of the first two components from Principle Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE) using Scikit-learn. The purpose of these visualizations is to determine if there are any groupings of the data. In Figure 7, we show the PCA plot for the FMA dataset. The PCA plot seems to lend some credence to Instrumental and Experimental genres being distinct from the rest. In the t-SNE plot shown in Figure 8, we do not see any distinct groupings of the data. However, there does appear to be some spatial separation between Folk and Instrumental versus the rest of the genres.





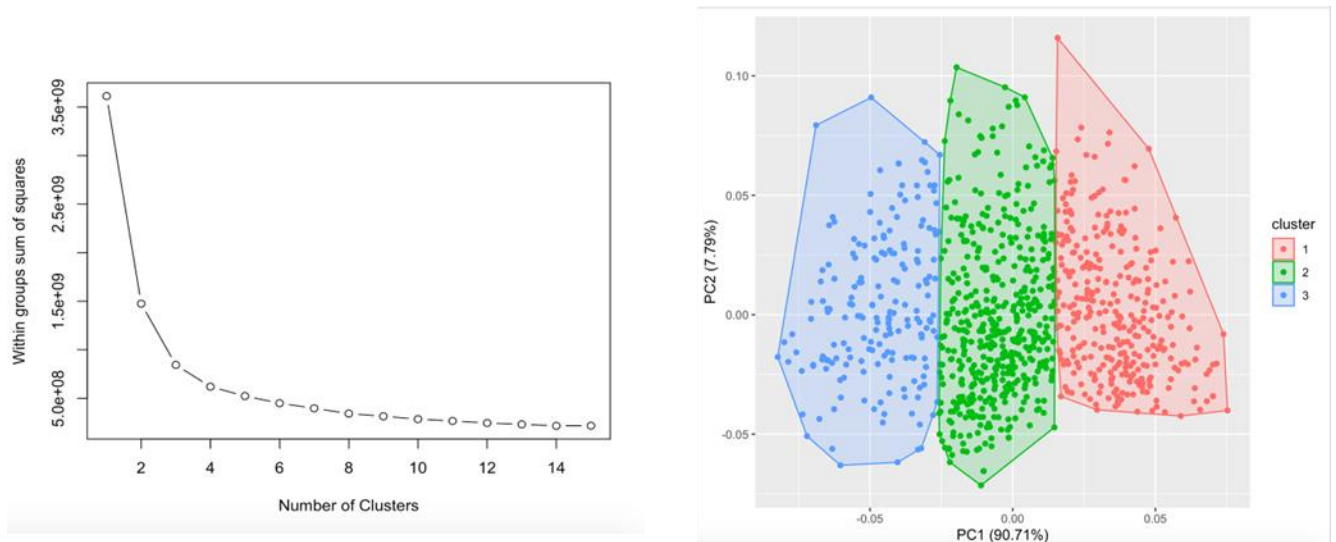
**Figure 7: PCA Plot of the first two principle components for the FMA Dataset.** Folk and instrumental music seem to be clustered to the left side of the plot, while Rock and Pop are more centered. Beyond this, there is not much to suggest distinct spatial groupings or clusters.



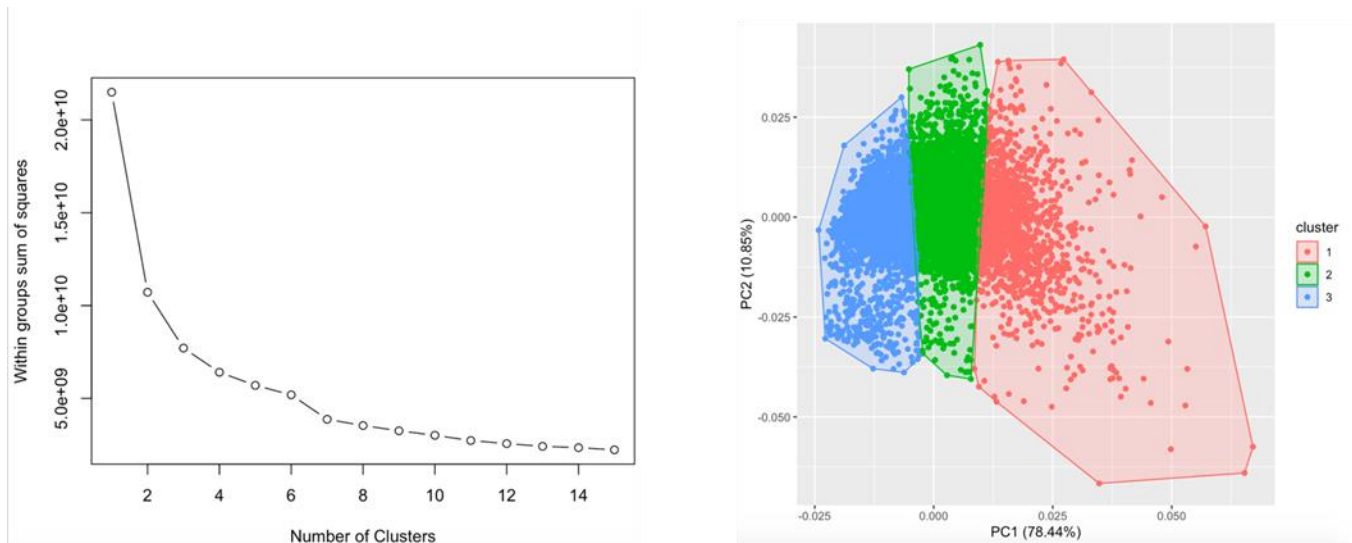
**Figure 8: T-SNE Plot of the FMA Dataset.** Again, folk and instrumental music appear on the left side of the plot, but there is not much to suggest distinct groupings or clusters.

We turn our attention now to utilizing k-means clustering to tease out hidden groupings in the data. We use R and the following libraries: Stats, dplyr, ggplot2, and ggfortify. To find the optimal value for  $k$ , we use within sum of squares, or WSS, which is a measure to explain the homogeneity within a cluster. We determine the optimal value of  $k$  to be the kink in the plot of WSS (elbow method) against the number of clusters. For the GTZAN dataset, as shown in Figure

9, the optimal number of clusters appears to be 3. Figure 10 shows the results for the FMA dataset, for which 3 groupings also appear to be optimal.



**Figure 9: Left – Plot of WSS against k for GTZAN. Right – GTZAN PCA Plot with Clusters overlaid.** Using the elbow method to select the optimal value of k suggests k=3. Overlaying the clusters on the PCA plot for the GTZAN dataset suggest weak support for three distinct clusters.

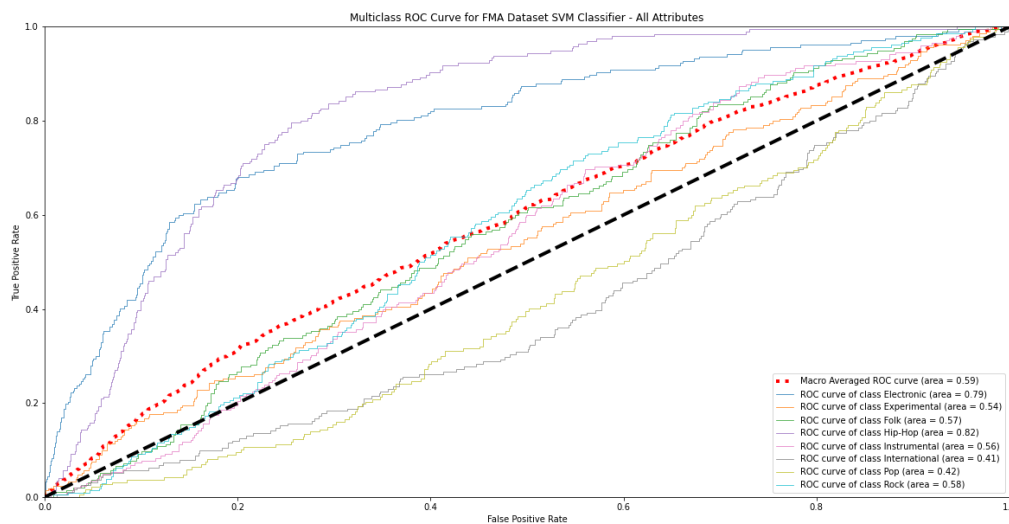


**Figure 10: Left – Plot of WSS against k for FMA. Right – FMA PCA Plot with Clusters overlaid.** Using the elbow method suggests k = 3. Weaker support using the elbow method might suggest k = 7. Overlaying the three clusters on the PCA plot for the FMA dataset suggests even weaker support than there was for the GTZAN dataset for three distinct clusters.

Following this, we proceeded with modeling the data with SVM classifiers in Scikit-learn. For both the GTZAN and FMA datasets, the data was split between training and testing sets with an 80% to 20% ratio, respectively. For the FMA dataset, an initial SVM classifier was fit on the training data, leaving the Scikit-learn default hyperparameters. Initial results from 10-fold cross validation yielded an average accuracy of 0.3458 +/- 0.0313, where the error is estimated as twice the standard deviation from the cross-validation accuracy. Grid search was performed using Scikit-learn's GridSearchCV method to attempt to optimize the classifier. The number of

cross-validation folds was left at the default, 5, for each testing run. The parameter grid was chosen to optimize for C and gamma hyperparameters, with values of 0.5, 1, 10, and 100 for C, and 0.000001, 0.0005, 0.0001, 0.0005, 0.001, and 0.005 for gamma. From this, the best SVM classifier had an average training accuracy of roughly 0.371. The weak performance of the classifier following optimization from grid search gave support to transforming or normalizing the dataset so that the values for attributes would be roughly in the same range and order of magnitude.

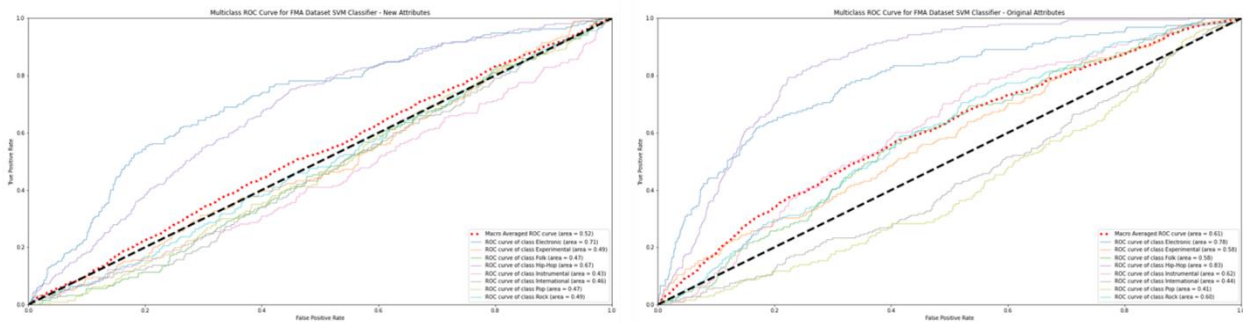
The data for the FMA dataset was transformed using Scikit-learn's in-built StandardScaler method. Grid search was performed to optimize hyperparameters. After some initial experimentation, the parameter grid was selected to try different values for C and gamma. These are, C: 0.5, 1, 10, 50, 100, and gamma: 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, and 1. We note that in the case of overlap in the parameter selection between the un-transformed and transformed data, the model on the transformed data outperforms the model trained on the untransformed data. The best model from the grid search had a C of 10 and a gamma of 0.01, with an average classification score of approximately 0.5206.



**Figure 11: Multiclass ROC Curve of the SVM Classifier for the FMA Dataset – All Attributes.** The black dashed line is the ROC for a random result, red dashed line is the macro averaged ROC for all genre classes. The area under the curve (AUC) is 0.59 for the macro averaged ROC. International and Pop have the lowest AUC of all genres.

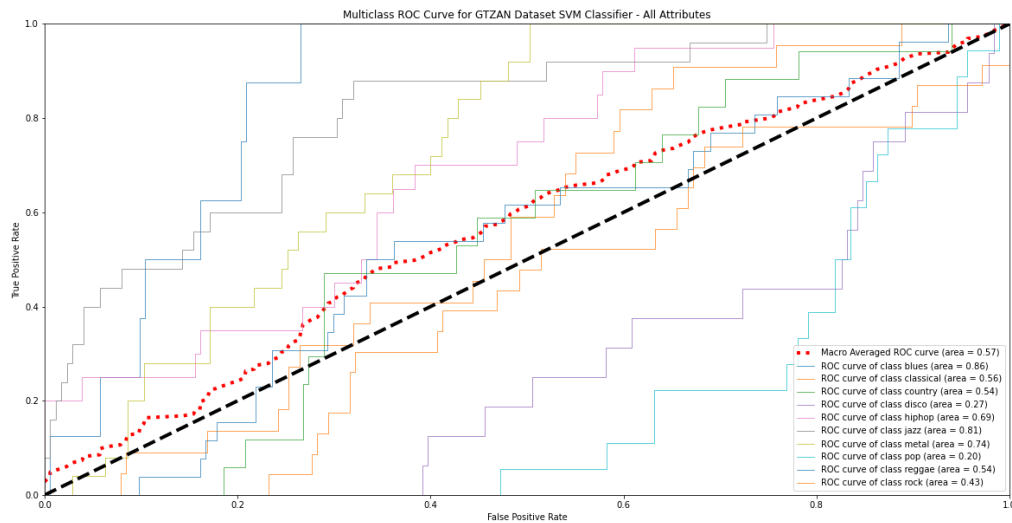
We were interested in seeing if the inclusion of our new attributes – dynamic tempo, length, and spectral flatness, had a measurable effect on model performance. We were also interested in whether these attributes would be sufficient to allow for classifying genres on only these attributes (thereby reducing the number of attributes required). To do so, we made SVM classifier models that used only the new attributes, and a separate classifier that only used all the attributes apart from the new attributes. Both SVM classifiers were optimized using grid search with the same parameters and cross-validation folds as in the model for the transformed data with all the attributes. The accuracy for the model with only the new attributes is 0.2856, and the accuracy for the model with all the attributes except the new attributes is 0.5078. T-

tests were performed for the significance of the difference in the accuracy of the model with all the attributes and the model with the attributes except the new attributes.

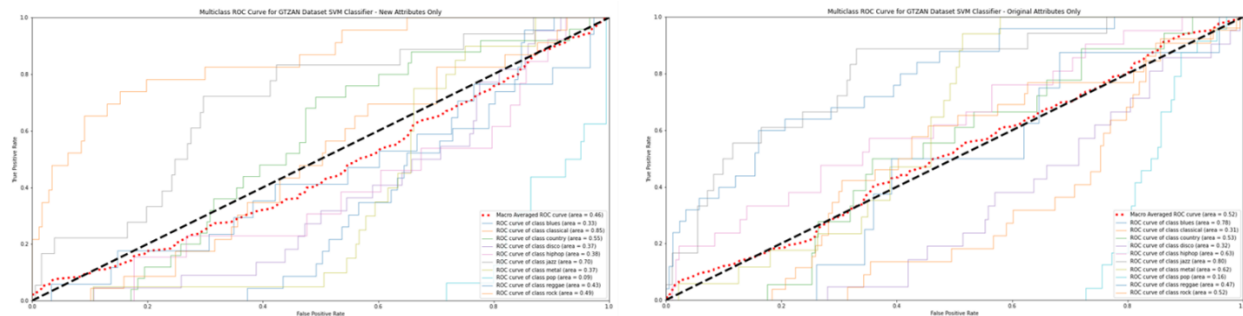


**Figure 12: Left – Multiclass ROC Curve of the SVM Classifier for the FMA Dataset – New Attributes Only. Right – Multiclass ROC Curve of the SVM Classifier for the FMA Dataset – Attributes less the New Attributes.** In each plot, the red dashed line is the macro average of the multiclass ROC, while the black dashed line represents the ROC for a random classifier. The SVM utilizing just the new attributes performs slightly better than random, while the SVM for all the attributes with the exception of the new is similar in its macro averaged ROC to the one shown in Figure 11.

SVM models were also trained on the GTZAN dataset. The data was split across an 80-20 testing, training split. The data was scaled using Scikit-learn's in-built StandardScaler method. For each SVM model, grid search was used with the same parameter grid as for the FMA data. The results of the GTZAN SVMs and FMA SVMs on the held-out test data is shown in Table 1.



**Figure 13: Multiclass ROC Curve of the SVM Classifier for the GTZAN Dataset – All Attributes.** The ROC curves show a more stepped or jagged appearance, which is expected given the GTZAN's smaller size. The AUC for the macro-averaged ROC curve is 0.57, which is similar to the AUC in Figure 11.



**Figure 13: Left – Multiclass ROC Curve of the SVM Classifier for the GTZAN Dataset – New Attributes Only. Right – Multiclass ROC Curve of the SVM Classifier for the GTZAN Dataset – Attributes less the New Attributes.** When using the SVM built only on the new attributes, the macro averaged ROC is worse than random. On the attributes less the new, the macro averaged ROC is slightly better than random.

**Table 1: SVM Model Summaries on Held-Out Test Data.** Overall, the SVM built using all attributes from the GTZAN dataset had the highest accuracy, precision, recall, and f score.

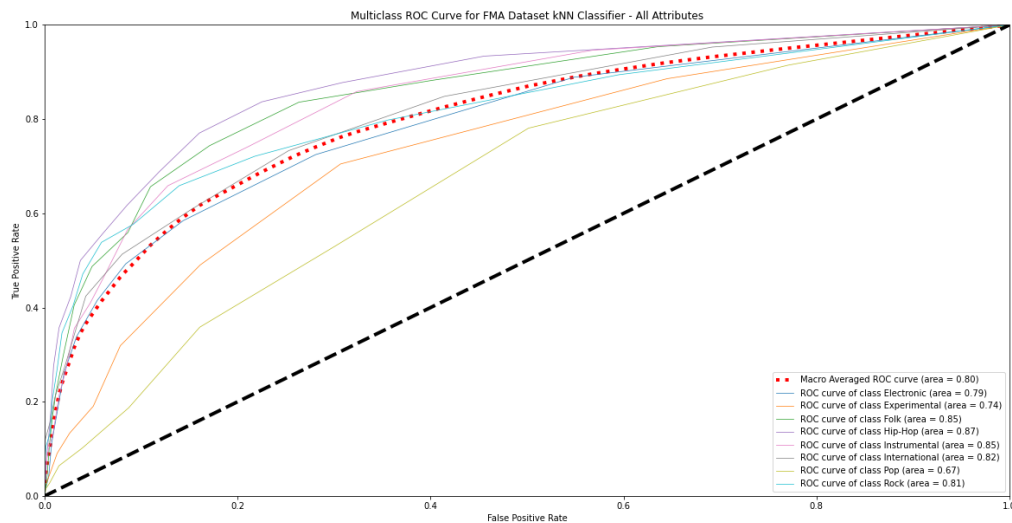
Model	Accuracy	Precision	Recall	F1 – Score
FMA – All Attributes	0.53	0.52	0.53	0.53
FMA – New Attributes	0.29	0.27	0.28	0.27
FMA – Attributes less new Attributes	0.52	0.51	0.52	0.51
GTZAN – All Attributes	0.69	0.69	0.71	0.68
GTZAN – New Attributes	0.43	0.46	0.45	0.44
GTZAN – Attributes less new Attributes	0.66	0.67	0.66	0.65

With the SVMs in place we proceeded to look at the result from kNN classifiers using Scikit-learn. For both the FMA and GTZAN datasets, we use the same transformed 80-20 train/test split as in the SVMs. We repeat the same models as before: kNN with all attributes, new attributes only, and attributes less the new attributes. For each model, we perform grid search with 5-fold cross validation. In this instance, we optimize on the choice of  $k$ , and the leaf size. For  $k$ , we try  $2^0$ ,  $2^1$ ,  $2^2$ ,  $2^3$ ,  $2^4$ ,  $2^5$ ,  $2^6$ ,  $2^7$ ,  $2^8$ , and  $2^{10}$  for the FMA dataset. The leaf size values tested are 1, 5, 10, 15, 30, 45, 60, 100, 250, and 500. For the GTZAN dataset, the maximum value of  $k$  in the grid search is changed to  $2^9$  to account for the lower number of samples. The maximum value of the leaf size is also limited to 250. Results for the kNN classifiers performance on the held-out test data are shown in Table 2. ROC curves are shown in Figures 14 – 17.

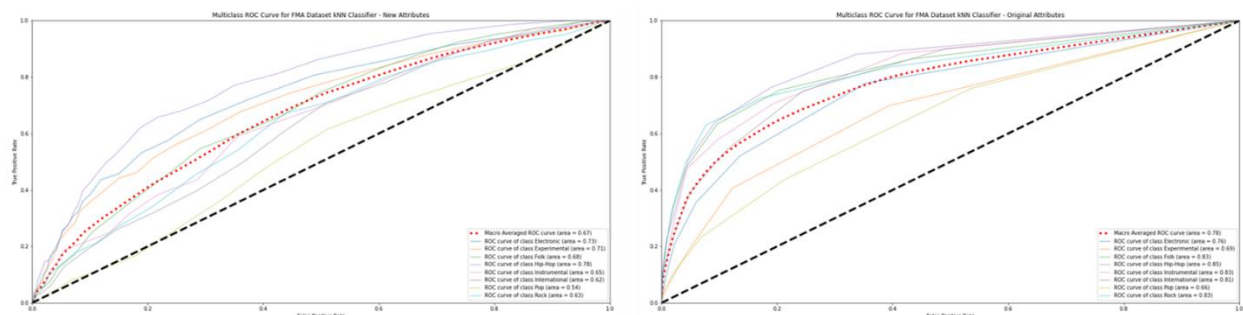


**Table 2: kNN Model Summaries on Held-Out Test Data.** Overall, the kNN classifier built using all attributes less the new attributes for the GTZAN dataset had the highest accuracy, precision, recall, and f score.

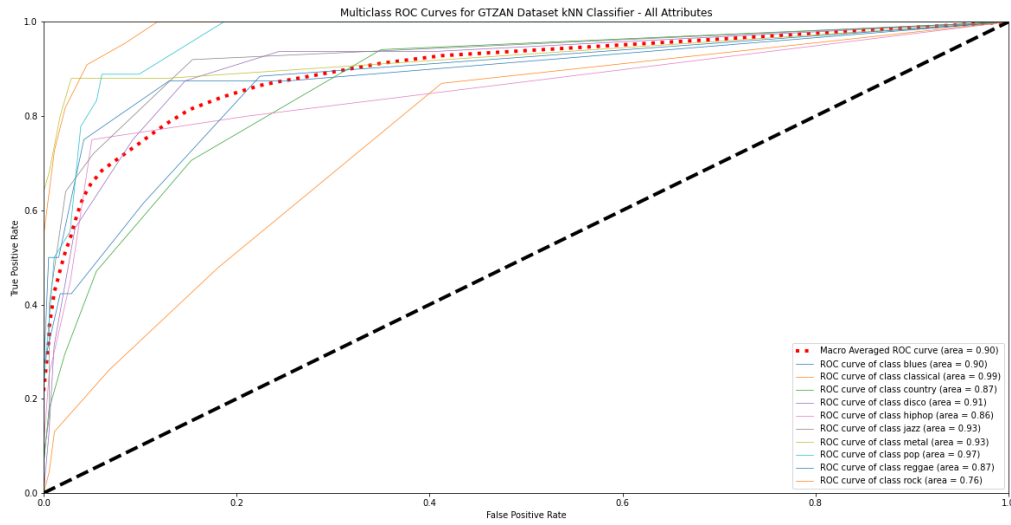
Model	Accuracy	Precision	Recall	F1 – Score
FMA – All Attributes	0.45	0.44	0.45	0.43
FMA – New Attributes	0.27	0.25	0.27	0.26
FMA – Attributes less new Attributes	0.46	0.45	0.46	0.45
GTZAN – All Attributes	0.57	0.59	0.59	0.55
GTZAN – New Attributes	0.32	0.3	0.33	0.29
GTZAN – Attributes less new Attributes	0.60	0.62	0.62	0.59



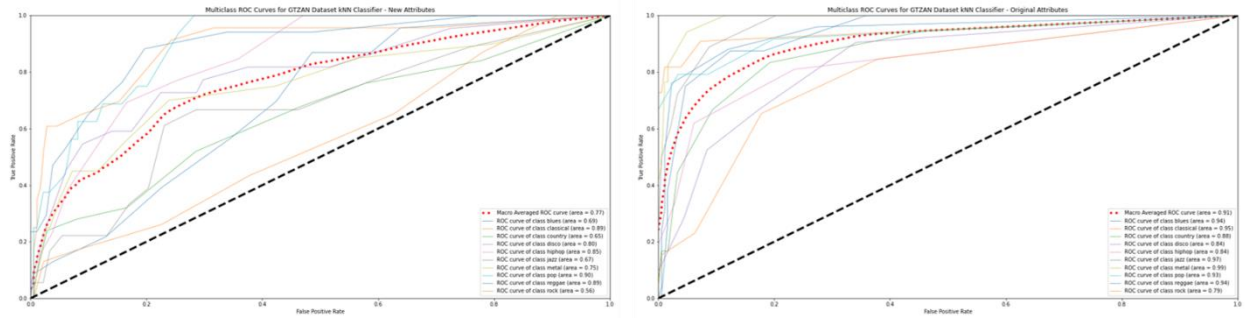
**Figure 14: Multiclass ROC Curve of the kNN Classifier for the FMA Dataset – All Attributes.** Direct comparison between ROC curves for the SVM classifiers and the ROC curves for the kNN classifiers should be limited due to a difference in the way Scikit-learn calculates the ROC for SVM classifiers and kNN classifiers. For the kNN model, Pop and Experimental have the lowest AUC.



**Figure 15: Left – Multiclass ROC Curve of the kNN Classifier for the FMA Dataset – New Attributes Only. Right – Multiclass ROC Curve of the kNN Classifier for the FMA Dataset – Attributes less New Attributes.**



**Figure 16: Multiclass ROC Curve of the kNN Classifier for the GTZAN Dataset – All Attributes.** The AUC of the macro averaged ROC curve is 0.9, which is higher than the AUC of 0.8 for the kNN for the FMA Dataset shown in Figure 14.



**Figure 17: Left – Multiclass ROC Curve of the kNN Classifier for the GTZAN Dataset – New Attributes Only. Right – Multiclass ROC Curve of the kNN Classifier for the GTZAN Dataset – Attributes less New Attributes.**

We performed inference on the average accuracy results from the cross-validation of the best model chosen from the grid search in each case. We use the altered form of the standard t-test for cross-validation results as described in [15]. Our main interest is in determining if the new attributes that we have added, length, central moments for dynamic tempo and spectral flatness, had a statistically significant effect on the accuracy of the models. Our results are shown in Table 3. We find that there is no statistically significant difference between models using all the attributes versus models that are trained without the new attributes. We do, however, find a statistically significant difference between the SVM and kNN classifiers for either dataset. Lastly, a statistically significant difference exists for model accuracy between the datasets.



**Table 3: Inference for the Significance of Differences in Model Cross Validation Average Accuracy**

Dataset, Test	P-Value
FMA, SVM All Attributes = Attr. Less new	0.217723
FMA, kNN All Attributes = Attr. Less new	0.280234
GTZAN, SVM All Attributes = Attr. Less new	0.958231
GTZAN, kNN All Attributes = Attr. Less nes	0.587463
FMA, SVM = kNN	0.003685
GTZAN, SVM = kNN	0.002413
SVM, FMA = GTZAN	0.000332
kNN, FMA = GTZAN	0.000351

Multinomial logistic regression was performed in R. The process was the same for both the FMA and the GTZAN dataset. We used an 80-20 train/test split on each respective dataset. We utilized libraries such as nnet, and caret. We performed 10-fold cross validation.

For multinomial logistic on the GTZAN dataset, we found the residual deviance was 1873.233, which performed better than the null model. Additionally, the AIC (Akaike's Information Criterion) was 2701.33. The confusion matrix for this model is shown below in Figure 18.

pred1	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock
blues	11	0	5	1	0	3	1	0	3	1
classical	0	13	0	0	0	0	1	0	0	2
country	2	0	9	3	0	3	0	4	1	4
disco	0	0	0	5	0	0	0	3	0	1
hiphop	0	0	0	1	15	0	4	1	1	3
jazz	0	2	0	1	0	15	0	0	2	0
metal	2	0	0	2	1	1	12	0	0	0
pop	0	0	0	2	2	0	0	11	0	0
reggae	1	1	1	1	1	0	0	2	13	2
rock	2	0	0	4	0	0	3	0	1	3

**Figure 18: Confusion Matrix for the Multinomial Logistic Regression Classifier on the Held-out Test Data for the GTZAN Dataset.**

In terms of the confusion matrix, we wanted to record the misclassification error. Using R, the misclassification error was 43.39%. After seeing this misclassification error, we thought that it would be best to analyze the F1 score as an overall metric. The overall F1 score was 0.533. Namely, this spurred from the overall precision = 0.573, the overall accuracy = 56.61%, and the overall recall = 0.564. However, after analyzing these results, we concluded that our dataset and overall logistic regression model should be based on accuracy. Hence, accuracy was our main basis of analysis.

## Logistic Regression: GTZAN Data

### Results

TP: 107

Overall Accuracy: 56.61%

Class	n (truth) ②	n (classified) ②	Accuracy	Precision	Recall	F1 Score
1	18	25	88.89%	0.44	0.61	0.51
2	16	16	96.83%	0.81	0.81	0.81
3	15	26	87.83%	0.35	0.60	0.44
4	20	9	89.95%	0.56	0.25	0.34
5	19	25	92.59%	0.60	0.79	0.68
6	22	20	93.65%	0.75	0.68	0.71
7	21	18	92.06%	0.67	0.57	0.62
8	21	15	92.59%	0.73	0.52	0.61
9	21	22	91.01%	0.59	0.62	0.60
10	16	13	87.83%	0.23	0.19	0.21

Overall F1 Score = 0.553, Overall Precision = 0.573 , Overall Accuracy = 56.61% , Overall Recall = 0.564

Figure 19: Cross Validation Summary Results of the Multinomial Logistic Regression Model for the GTZAN Dataset.

While performing the k-fold cross validation, our output summarized penalized multinomial regression with roughly a 0.540 accuracy. Accuracy vs. Weight Decay can also be shown below:

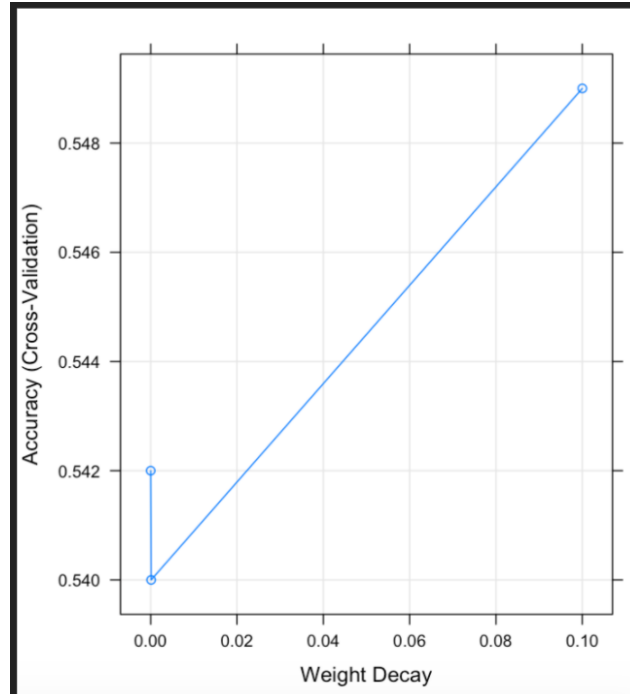
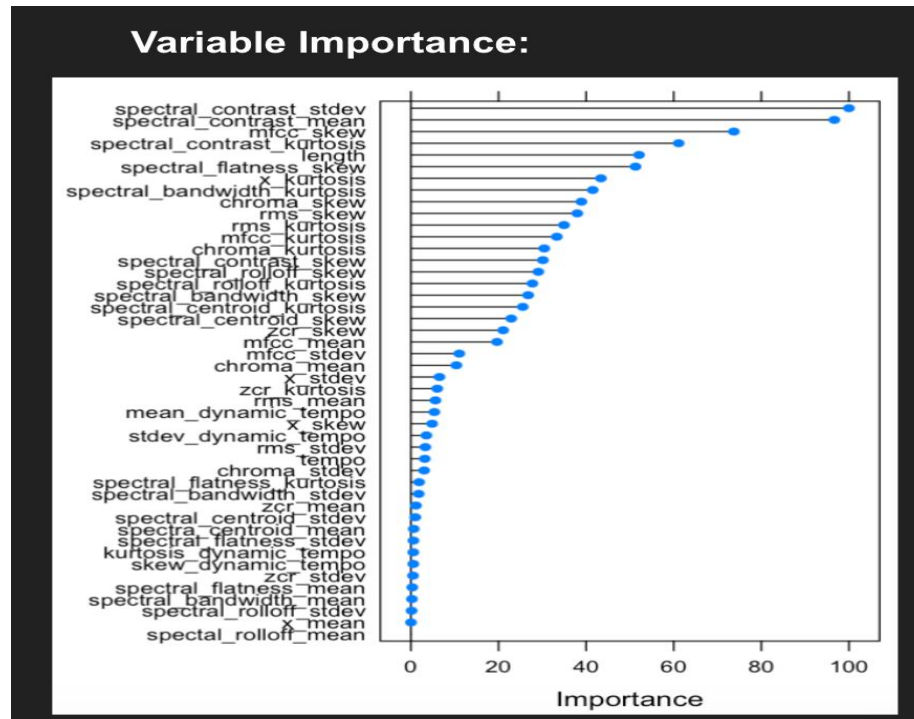


Figure 20: Cross Validation Accuracy of the Multinomial Logistic Regression Model for the GTZAN Dataset against Weight Decay.

In terms of our overall conclusions, we thought that it would be ideal to analyze variable importance along with scatter plots against probability. The results for this are shown below:



**Figure 21: Variable Importance for the Multinomial Logistic Regression Model for the GTZAN Dataset.** Importance is measured in arbitrary units. The standard deviation of the spectral contrast, along with the mean spectral contrast seem to have the highest importance for the model.

For the FMA dataset, we noticed that the residual deviance was much higher relative to the GTZAN dataset at 21027.87. This performed better than the null model. Additionally, the AIC for this dataset was 21671.87. After recording a summary of the model, we then analyzed the confusion matrix depicted below for the testing data of FMA:

pred1	Electronic	Experimental	Folk	Hip-Hop	Instrumental	International	Pop	Rock
Electronic	53	10	0	21	3	19	15	11
Experimental	10	46	6	3	23	19	13	10
Folk	9	19	110	4	29	23	36	11
Hip-Hop	40	25	12	134	8	29	41	21
Instrumental	24	36	19	13	86	10	21	16
International	22	19	16	5	7	77	16	11
Pop	8	18	13	16	7	15	23	13
Rock	24	31	15	16	20	15	50	113

**Figure 22: Confusion Matrix for the Multinomial Logistic Regression Classifier on the Held-out Test Data for the FMA Dataset.**

In terms of the confusion matrix, we wanted to record the misclassification error once again. Using R, the misclassification error was 60.07%. After seeing this misclassification error, we again realized that it would be best to analyze the F1 score as an overall metric. The overall F1 score was 0.3825. Namely, this spurred from the overall precision = 0.3838, the overall accuracy

= 39.93%, and the overall recall = 0.4025. However, after analyzing these results, we concluded that our dataset and overall logistic regression model should be based on accuracy. Hence, accuracy again was our main basis of analysis for the FMA dataset similar to the analysis of the GTZAN dataset.

## Logistic Regression: FMA Data:

### Results

TP: 642

Overall Accuracy: 39.93%

Class	n (truth) ②	n (classified) ②	Accuracy	Precision	Recall	F1 Score
1	190	132	86.57%	0.40	0.28	0.33
2	204	130	84.95%	0.35	0.23	0.28
3	191	241	86.82%	0.46	0.58	0.51
4	212	310	84.2%	0.43	0.63	0.51
5	183	225	85.32%	0.38	0.47	0.42
6	207	173	85.95%	0.45	0.37	0.41
7	215	113	82.46%	0.20	0.11	0.14
8	206	284	83.58%	0.40	0.55	0.46

Figure 23: Cross Validation Summary Results of the Multinomial Logistic Regression Model for the FMA Dataset.

While performing the k-fold cross validation, our output summarized penalized multinomial regression with roughly a 0.420 accuracy. Accuracy vs. Weight Decay can also be depicted below:

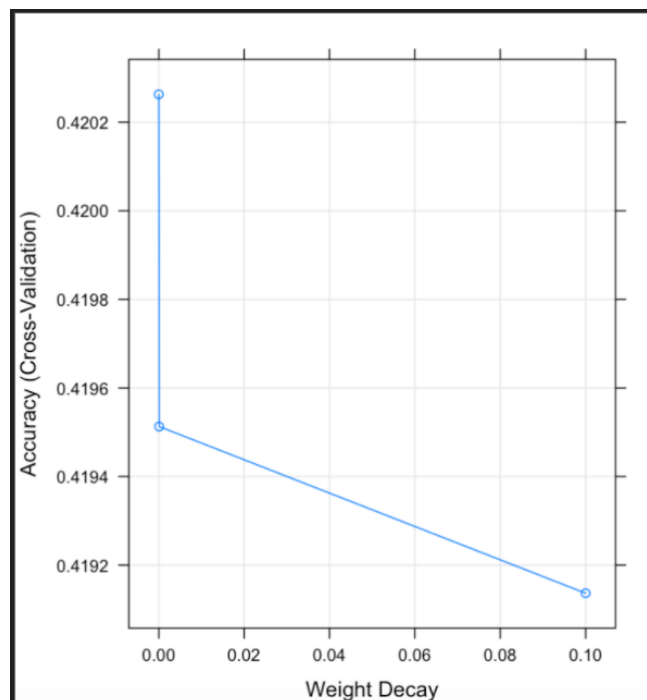
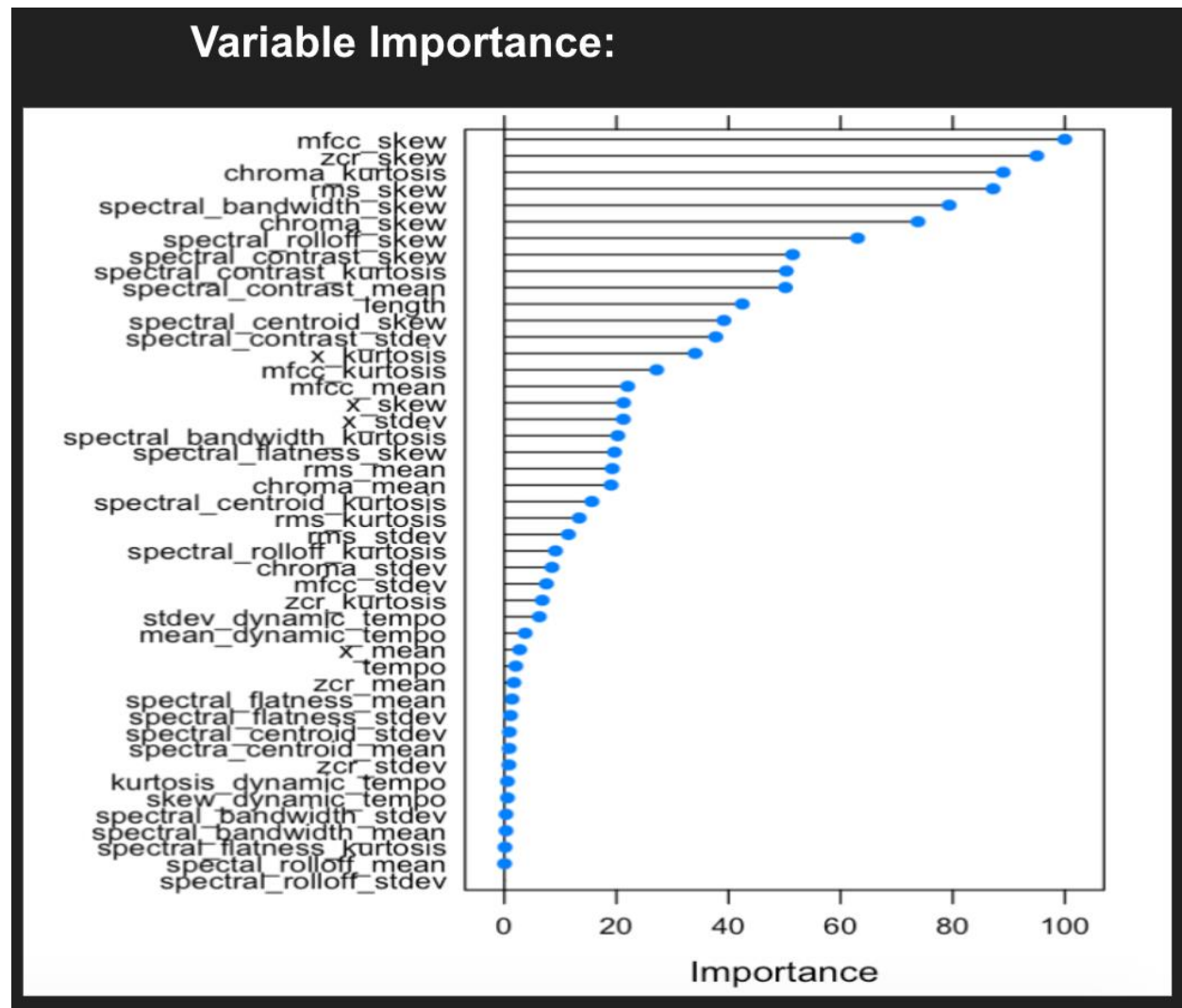


Figure 24: Cross Validation Accuracy of the Multinomial Logistic Regression Model for the FMA Dataset against Weight Decay.

In terms of our overall conclusions, we thought that it would be ideal to once again analyze variable. The results for this are shown below:



**Figure 25: Variable Importance for the Multinomial Logistic Regression Model for the FMA Dataset.** The skew of the mel-frequency cepstral coefficients along with the skew of the zero-crossing rate had the highest importance to the model. Importance is measured in arbitrary units.

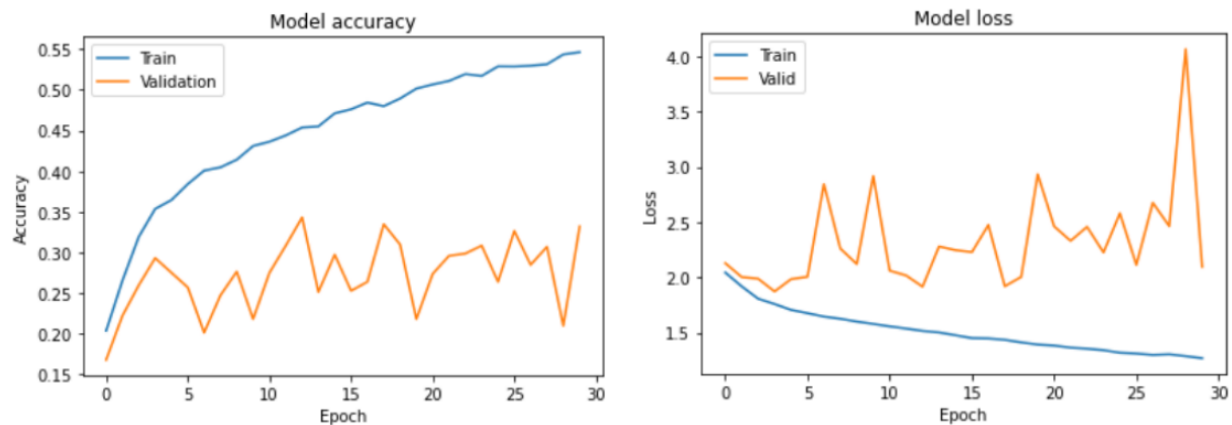
The summary of the results for all our machine learning models are shown in Table 4. Overall, the SVM classifier trained on the GTZAN dataset had the highest test accuracy of any model with 69% accuracy. For the FMA dataset, the most accurate model was also the SVM classifier, with an accuracy of 53%.

**Table 4: Model Summaries for Test Accuracy, Precision, Recall, and F-score.** The model with the overall highest accuracy, precision, recall, and f-score was SVM classifier built using the GTZAN dataset.

Model	Accuracy	Precision	Recall	F1 Score
FMA: SVM- Sklearn	0.53	0.52	0.53	0.53
FMA: kNN – Sklearn	0.45	0.44	0.45	0.43
FMA: Logistic – R	0.3993	0.3838	0.4025	0.3825
GTZAN: SVM – Sklearn	0.69	0.69	0.71	0.68
GTZAN: kNN – Sklearn	0.57	0.59	0.59	0.55
GTZAN: Logistic - R	0.5661	0.573	0.564	0.553

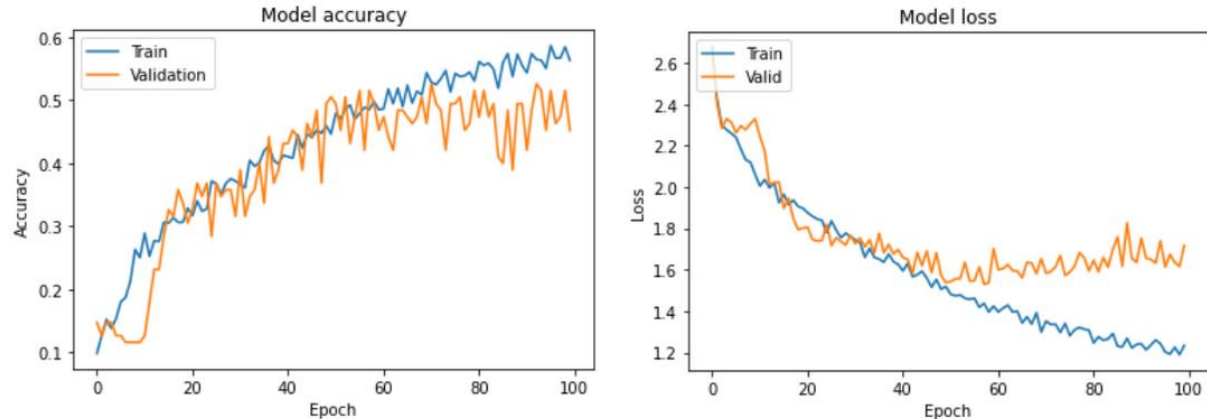
For the CNNs, separate models were trained on the FMA and GTZAN dataset. In each case, we considered a simple CNN model with two convolutional layers. We used rectified linear unit (ReLUs) as our activation function, categorical cross entropy for the loss, and Adam for the optimizer. Due to the limited sample size in either dataset, the training-validation split was taken as 90-10. Results for the accuracy and loss during training for either dataset can be seen in Figures 26 and 27.

## CNN Results - FMA



**Figure 26: Left – CNN Model Train and Validation Accuracy for the FMA dataset. Right – CNN Model Train and Validation Loss for the FMA dataset.** After 30 epochs, the validation accuracy never improves beyond 0.3.

## CNN Results - GTZAN



**Figure 27: Left – CNN Model Train and Validation Accuracy for the GTZAN dataset. Right – CNN Model Train and Validation Loss for the GTZAN dataset.** Training was left to continue for the full 100 epochs. Validation accuracy seems limited to around 0.45.

### Discussion

Our main consideration when evaluating classifiers was accuracy. We argue in favor mainly for accuracy due to the cost associated with the misclassification of a piece of music being low. A pop song misclassified as rock isn't necessarily detrimental; in some cases pop and rock have quite an overlap. Indeed, the reason for utilizing k-Means was to investigate if there were any hidden groupings or clusters different than those genre classifications suggested by the dataset. The result of our k-Means clusters for both the FMA and GTZAN datasets suggest that the optimal number of clusters is 3. Visually when these clusters are overlaid on PCA plots, the clustering is weak. However, it is interesting that ethnomusicologists classify music into one of three types: classical, folk, and pop [16]. Under this classification scheme, classical music is "court music", folk music is music of a local region or people, and pop is music that has been commoditized. This means that genres such as metal, pop, reggae, hip-hop, etc. all fall under the super-category of "pop". Under this classification scheme, for both of our datasets, the overwhelming majority of the samples would fall under "pop" music, with few samples making up folk or classical. For the GTZAN dataset – 9 out of 10 of our genres would become part of the "pop" super-category. Given the equal number of samples per genre in the GTZAN dataset, we would expect that when performing k-Means clustering we would see a ratio similar to this when producing the clusters. Instead, we see that there is a rough equivalence in the number of samples per cluster in k-Means. This suggests that the k=3 result from k-Means clustering is not necessarily a reflection of these super-categories from ethnomusicology, but possibly a spurious result.

The type of genre to classify on seems to play an important effect. We can see this in the comparison of the models built from the different datasets, which contain both a different number and different set of genres. The results from our hypothesis tests conducted on the SVM and kNN classifiers suggests a difference in model accuracy between the two datasets.



Likewise, for every one of our models – even the CNN – the smaller sized GTZAN dataset with more genres performed better than the larger sized FMA dataset. Some of this difference might be contributed to the type of music contained within each dataset. For the GTZAN dataset, the music samples consisted of pieces within the researcher’s own personal music collection from the early 2000s. Samples in the pop classification are dominated by female vocals and Britney Spears (the King of Pop was not represented at all) – likewise the inclusion of reggae as a top-level genre was questionable and biased away from international music. In contrast, the FMA dataset does not consist of any researcher’s own personal music collection, but rather a random sample of 8000 pieces of music from the 8 top-level genre classifications of the Free Music Archive. The lack of researcher selection bias and variety of music (not just from the 90s and early 2000s) led us to believe that the FMA constructed classifiers would perform better. Our results suggested otherwise. We think that this may be a result of both the top-level genres that the Free Music Archive classifies its music into and due to the type of music included in the archive. As evidence, consider that many works in the Free Music Archive are intentionally multi-genre pieces. While we focused on the equal-weighted single top-level genre pieces of the archive, it’s possible that this may have had an effect (misclassification of pieces). Also consider that these works are creative-commons licensed, intentionally outside of the commercial music sphere. The top-level genres also seem problematic. Consider the genre “instrumental”. It is not entirely clear what denotes an instrumental genre. There are instrumental pieces in rock, hip-hop, classical, etc. Are all of these pieces classified as instrumental in the FMA classification system? These issues may have contributed to classifiers built on the FMA dataset having poorer performance.

Our inclusion of additional attributes to augment those suggested in [8] had little effect on model accuracy or performance, but at least was not deleterious to the model as suggested by our hypothesis tests on cross-validation average accuracy between the SVM and kNN models. While our models built on just the new attributes produced accuracy better than a random classifier, it had the worst performance of any model that we considered. Still, further work is required in this area to determine the optimal number and type of attributes to classify on. While the inclusion of the central moments for all calculated attributes was influenced by [8], the argument for using the central moments in every case is weak.

Utilizing logistic regression provided meaningful output in terms of our analysis of these two datasets. For example, we concluded that Country and Rock had the lowest accuracy within the GTZAN dataset and that the model accuracy was higher for the Classical genre. In terms of the FMA, dataset, the model accuracy was high for Electronic. The misclassification error was more apparent in the FMA dataset as well. The accuracy that we received from logistic regression was not as apparent as the accuracy we received with SVM; however, the models provided quite conveyable output that further emphasizes our genre classification pipeline.

Our overall best classifier in terms of test accuracy was the SVM classifier trained on the GTZAN dataset, with a test accuracy of 69%. This is relatively close to human classification of music genres based on listening to individual pieces of music with no prior knowledge. Human accuracy in this case is roughly 76% [7].

The accuracy for our CNN model trained on the GTZAN dataset was somewhat surprising. We did not expect to get an accuracy in the 40% range given the small sample size of the dataset. Indeed, in [8], a dataset with 40,000 samples was used to bring their CNN classifier into the 60% range of accuracy. We had intended to build our final CNN classifier using the entire corpus of the FMA dataset – training it on an 80-20 train/test split of all 106,000 spectrograms. We fell short of completing this when it became clear that our other models built on the FMA dataset were systematically producing worse results than the GTZAN dataset despite its larger sample size. Training on a subset of the FMA dataset would have also required a significant rework for the genres to classify on, as a decision would have to be made when classifying on a multi-genre piece. Further work is needed in this area as the CNN models trained on the spectrograms are promising.

## Conclusion

We evaluated the performance of several different machine learning models for music genre classification based on feature engineering from music audio files. For all our models, kNN, SVM, multinomial logistic regression, and CNNs performed better than classification at random. We found that our best classifier was a SVM classifier trained using the GTZAN dataset, with accuracy of approximately 69%, approaching that of human classification.

We attempted to build our classifiers by utilizing additional attributes produced through feature engineering: length of the piece, dynamic tempo, and spectral flatness. We found that utilizing these additional attributes produced no significant difference in the accuracy of the model constructed for either a SVM or kNN classifier than one constructed using the attributes as suggested in [8]. Still, we note that additional work needs to be done to evaluate this on the other models that we evaluated. Additional, further work can be done in this area to narrow down the attributes used.

We found that models constructed using the GTZAN dataset systematically performed better than those constructed using the FMA dataset. While we believe that this is mainly due to differences in the type of music included in the FMA dataset, more work needs to be done in this area to validate this. We believe that it is possible that the increased model performance from the GTZAN dataset could also be due to the genres being classified on. Further work is necessary in this area to determine the optimal number and type of genre to classify on.

Lastly, we built trained a CNN model using just images of the spectrograms of the pieces from each dataset and managed to get relatively high accuracy (approaching 45%) considering the small sample size with the GTZAN dataset. In order to get similar results with the FMA dataset, we believe that a better system of genres is necessary for that dataset. However, given the larger sample size of this dataset, once the correct genre classifiers are determined, the accuracy of a CNN trained on the FMA dataset should be higher than that of a CNN trained on the GTZAN dataset.

We might consider additional avenues in which this work can be expanded. First, the spectrogram generation and all spectral information was generated using the short time Fourier transform. We might consider instead using wavelet transformations. We also by no means have done an exhaustive analysis of all the classifiers that could have been used on this data. One such classifier could be random forest. Likewise, different CNN architectures than the basic one that we used could be tried as well.

## References

- [1] M. Defferrard, K. Benzi, P. Vandergheynst and X. Bresson, "FMA: A Dataset for Music Analysis," 2017.
- [2] J. Nam, K. Choi, S.-Y. Chou and Y.-H. Yang, "Deep Learning for Audio-Based Music Classification and Tagging: Teaching Computers to Distinguish Rock from Bach," *IEEE*, 2018.
- [3] I. H. Witten, f. Eibe and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques - 3rd Edition*, Morgan Kaufmann Publishers, 2011.
- [4] J. H. McDermott, A. F. Schultz, E. A. Undurraga and R. A. Godoy, "Indifference to dissonance in native Amazonians reveals cultural variation in music perception," *Nature*, vol. 535, pp. 547-550, 2016.
- [5] L. Shaver-Gleason, "Is Music a Universal Language?," Not Another Music History Cliche!, 4 January 2018. [Online]. Available: <https://notanothermusichistorycliche.blogspot.com/2018/01/is-music-universal-language.html>. [Accessed 6 May 2020].
- [6] N. Houston, "Music Made on Game Boys is a Much Bigger Deal than You'd Think," *Vice*, 4 November 2014. [Online]. Available: [https://www.vice.com/en\\_uk/article/8gdb7p/chipzels-complete-history-of-chiptune-939](https://www.vice.com/en_uk/article/8gdb7p/chipzels-complete-history-of-chiptune-939). [Accessed 6 May 2020].
- [7] K. Seyerlehner, G. Widmer and P. Knees, "A Comparison of Human, Automatic and Collaborative Music Genre Classification and User Centric Evaluation of Genre Classification Systems," *Lecture Notes in Computer Science*, vol. 6817, 2011.
- [8] H. Bahuleyan, "Music Genre Classification using Machine Learning Techniques," 2018.
- [9] J. Wulfinf and M. Riedmiller, "Unsupervised Learning of Local Features for Music Classification," 2012.
- [10] S. Dieleman, P. Brakel and B. Schrauwen, "Audio-based music classification with a pretrained convolutional network," 2011.
- [11] B. McFee, C. Raffel, D. Liang, D. P. W. Ellis, M. McVicar, E. Battenberg and O. Nieto, "Librosa: Audio and Music Signal Analysis in Python".
- [12] R. Astley, *Never Gonna Give You Up*, 1987.

- [13] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE*, 2002.
- [14] B. Sturm, "The State of the Art Ten Years After a State of the Art: Future Research in Music Information Retrieval," 2014.
- [15] R. Bouckaert and E. Frank, "Evaluating the Replicability of Significance Tests for Comparing Learning Algorithms," 2004.
- [16] P. Tagg, "Analysing Popular Music: Theory, Method and Practice," *Popular Music*, vol. 2, pp. 37-67, 1982.
- [17] T. Lidy and A. Rauber, "Evaluation of feature extractors and psycho-acoustic transformations for music genre classification," 2005.
- [18] M. Abadi, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015.
- [19] B. Rai, *Multinomial Logistic Regression with R..*
- [20] B. Ripley, "nnet: Feed-Forward Neural Networks and Multinomial Log-Linear Models," 25 February 2020. [Online]. Available: <https://www.rdocumentation.org/packages/nnet/versions/7.3-13>. [Accessed 30 April 2020].
- [21] F. Pedregosa, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.

## Appendix

### Time Logs:

#### Maher:

Activity	Date	Hours
Literature Review	3/2, 3/4-3/5	4
Python Jupyter Notebook data manipulation	5-March	2
Data Sourcing	3/1, 3/2, 3/6	3
Data Cleaning	3/5, 3/6	1.5
Proposal Writing	6-March	1.5
Programming and Team Meeting on Google Hangouts	4/7/20	2
Data Cleansing for Logistic Regression	4/7/20	3.5
Genre Merging in Python GTZAN	4/9/20	3
Genre Merging in Python FMA	4/10/20	2
Code Review During Day	4/11/20	4
Read and Analyze Papers	4/13/20	2
Programming and Team Meeting on Google Hangouts	4/13/20	3.5
Data Parsing	4/14/20	1
Code Review During Day	4/17/20	2
Logistic Modeling Reading	4/18/20	2
Logistic Modeling Code	4/18/20	3
Programming and Team Meeting on Google Hangouts	4/19/20	2
Logistic Model Graphs	4/21/20	2
Cross Validation in R and Summaries	4/21/20	1
KMeans w Rashid	4/22/20	2
Refactorizing	4/23/20	2
Analyze Output of Logistic Regression	4/22/20	3
Programming and Team Meeting on Google Hangouts	4/24/20	3.25
Code Review During Day	4/26/20	2
Presentation	4/28/20	3
Paper	4/29/20	2.5
Paper	5/1/20	2
Paper	5/3	3
Paper	5/5/20	2.25
Total:		70

**Rashid:**

<b>Activity</b>	<b>Date</b>	<b>Hours</b>
Code Review - R and Python	4/3/20	5
Cleaning Data GTZAN	4/3/20	5
Cleaning Data FMA	4/4/20	9
Programming and team meeting	4/7/20	2
Merging Data and coding GTZAN	4/10/20	3
Merging Data and coding FMA	4/11/20	5
K-means clustering GTZAN - R	4/12/20	3
K-means clustering FMA - R	4/12/20	5
Programming and team meeting	4/13/20	2
KNN algorithim GTZAN - R	4/15/20	4
KNN algorithim FMA - R	4/15/20	5
Assited in Logistic (Maher)	16-Apr	3
Programming and team meeting	4/17/20	1
Assisted in SVM (Stepen)	4/19/20	1
Proposal Writing	3/7/20 - 3/8/20	3.5
Litrature Review	4/21/20	3
Programming and team meeting	4/23/20	2
Presentation	4/28/20	3
Paper	On going	2
<b>Total</b>		<b>66.5</b>

### Activity

## CSV Data Screenshots – GTZAN

**Merged with genres:**

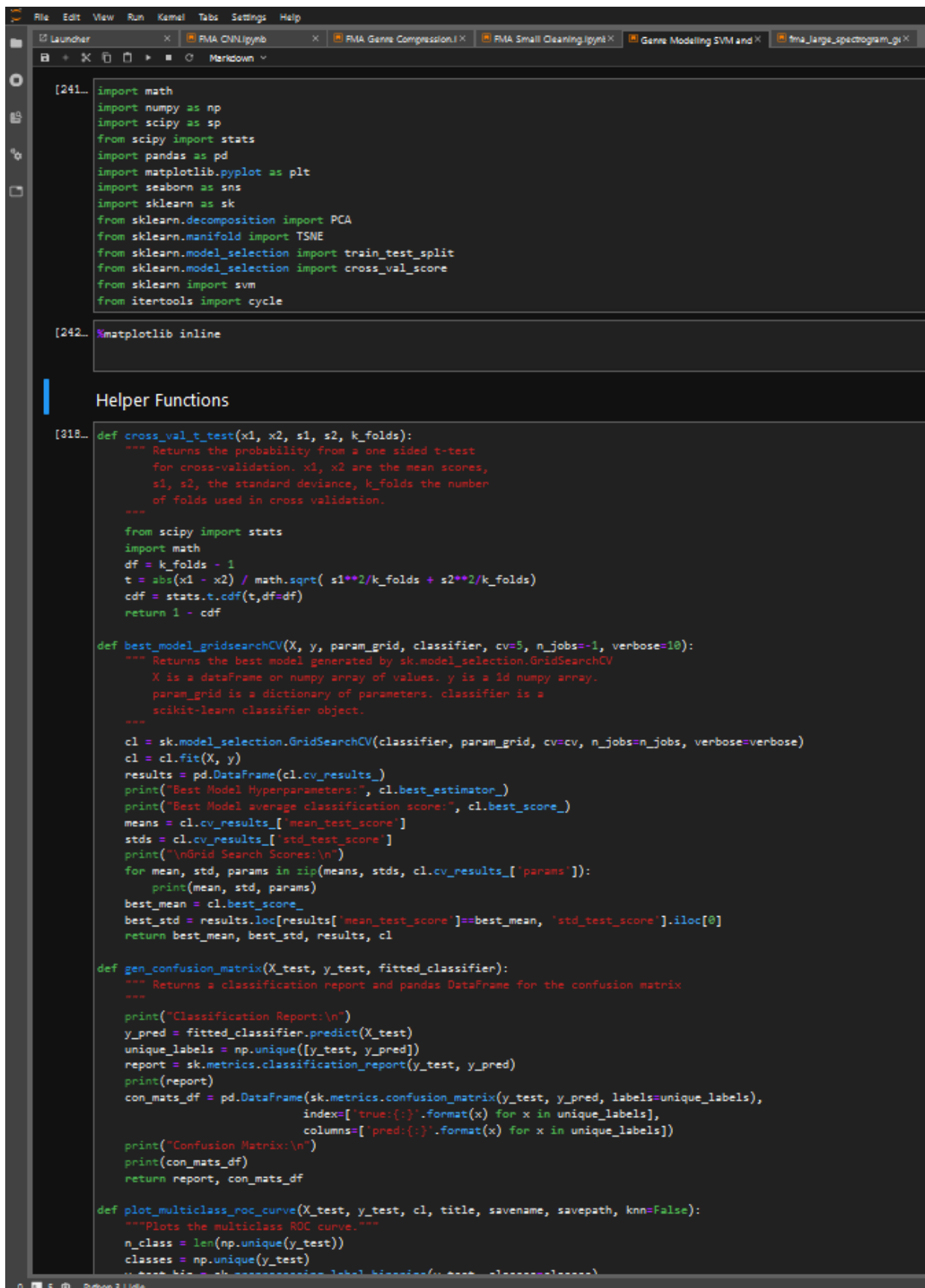
[illegible]



	P	D	C	H	G	S	H	O	S	U	V	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS	AT	AV	AW	AX	AY	AZ	BA	BB	BC	BD	BE	BF	BG	BH	BI	BJ	BK	BL	BM	BN	BO	BP	BQ	BR	BS	BT	BV	BW	BX	BY	BZ	CA	CB	CC	CD	CE	CF	CG	CH	CI	CJ	CK	CL	CM	CN	CO	CP	CQ	CR	CS	CT	CU	CV	CW	CX	CY	CZ	DA	DB	DC	DD	DE	DF	DG	DH	DI	DJ	DK	DL	DM	DN	DO	DP	DQ	DR	DS	DT	DV	DW	DX	DY	DZ	EA	EB	EC	ED	EE	EF	EG	EH	EI	EJ	EK	EL	EM	EN	EO	EP	EQ	ER	ES	ET	EU	EV	EW	EX	EY	EZ	FA	FB	FC	FD	FE	FF	FG	FH	FI	FJ	FK	FL	FM	FN	FO	FP	FQ	FR	FS	FT	FV	FW	FX	FY	FZ	GA	GB	GC	GD	GE	GF	GG	GH	GI	GJ	GK	GL	GM	GN	GO	GP	GQ	GR	GS	GT	GU	GV	GW	GX	GY	GZ	HA	HB	HC	HD	HE	HF	HG	HH	HI	HJ	HK	HL	HM	HN	HO	HP	HQ	HR	HS	HT	HV	HW	HX	HY	HZ	IA	IB	IC	ID	IE	IF	IG	IH	II	IJ	IK	IL	IM	IN	IO	IP	IQ	IR	IS	IT	IU	IV	IW	IX	IY	IZ	JA	JB	JC	JD	JE	JF	JG	JH	JI	IJ	JK	JL	JM	JN	JO	JP	JQ	JR	JS	JT	JV	JW	JX	JY	JZ	KA	KB	KC	KD	KE	KF	KG	KH	KI	KJ	KK	KL	KM	KN	KO	KP	KQ	KR	KS	KT	KV	KW	KX	KY	KZ	LA	LB	LC	LD	LE	LF	LG	LH	LI	LJ	LK	LM	LN	LO	LP	LQ	LR	LS	LT	LV	LW	LX	LY	LZ	MA	MB	MC	MD	ME	MF	MG	MH	MI	MJ	MK	ML	MM	MN	MO	MP	MQ	MR	MS	MT	MV	MW	MX	MY	MZ	NA	NB	NC	ND	NE	NF	NG	NH	NI	NJ	NK	NL	NM	NO	NP	NQ	NR	NS	NT	NV	NW	NX	NY	NZ	OA	OB	OC	OD	OE	OF	OG	OH	OI	OJ	OK	OL	OM	ON	OO	OP	OQ	OR	OS	OT	OV	OW	OX	OY	OZ	PA	PB	PC	PD	PE	PF	PG	PH	PI	PJ	PK	PL	PM	PN	PO	PP	PQ	PR	PS	PT	PV	PW	PX	PY	PZ	QA	QB	QC	QD	QE	QF	QG	QH	QI	QJ	QK	QL	QM	QN	QO	QP	QQ	QR	QS	QT	QV	QW	QX	QY	QZ	RA	RB	RC	RD	RE	RF	RG	RH	RI	RJ	RK	RL	RM	RN	RO	RP	RQ	RR	RS	RT	RV	RW	RX	RY	RZ	SA	SB	SC	SD	SE	SF	SG	SH	SI	SJ	SK	SL	SM	SN	SO	SP	SQ	SR	SS	ST	SV	SW	SX	SY	SZ	TA	TB	TC	TD	TE	TF	TG	TH	TI	TJ	TK	TL	TM	TN	TO	TP	TQ	TR	TS	TV	TW	TX	TY	TZ	UA	UB	UC	UD	UE	UF	UG	UH	UI	UJ	UK	UL	UM	UN	UO	UP	UQ	UR	US	UT	UV	UW	UX	UY	UZ	VA	VB	VC	VD	VE	VF	VG	VH	VI	VJ	VK	VL	VM	VN	VO	VP	VQ	VR	VS	VT	VV	VW	VX	VY	VZ	WA	WB	WC	WD	WE	WF	WG	WH	WI	WJ	WK	WL	WM	WN	WO	WP	WQ	WR	WS	WT	WV	WW	WX	WY	WZ	XA	XB	XC	XD	XE	XF	XG	XH	XI	XJ	XK	XL	XM	XN	XO	XP	XQ	XR	XS	XT	XV	XW	XX	XY	XZ	YA	YB	YC	YD	YE	YF	YG	YH	YI	YJ	YK	YL	YM	YN	YO	YP	YQ	YR	YS	YT	YV	YW	YX	YY	YZ	ZA	ZB	ZC	ZD	ZE	ZF	ZG	ZH	ZI	ZJ	ZK	ZL	ZM	ZN	ZO	ZP	ZQ	ZR	ZS	ZT	ZV	ZW	ZX	ZY	ZZ
0	0.0000000000	0.0000000000	0.0000000000	0.0000000000																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																		

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31	A32	A33	A34	A35	A36	A37	A38	A39	A40	A41	A42	A43	A44	A45	A46	A47	A48	A49	A50	A51	A52	A53	A54	A55	A56	A57	A58	A59	A60	A61	A62	A63	A64	A65	A66	A67	A68	A69	A70	A71	A72	A73	A74	A75	A76	A77	A78	A79	A80	A81	A82	A83	A84	A85	A86	A87	A88	A89	A90	A91	A92	A93	A94	A95	A96	A97	A98	A99	A100	A101	A102	A103	A104	A105	A106	A107	A108	A109	A110	A111	A112	A113	A114	A115	A116	A117	A118	A119	A120	A121	A122	A123	A124	A125	A126	A127	A128	A129	A130	A131	A132	A133	A134	A135	A136	A137	A138	A139	A140	A141	A142	A143	A144	A145	A146	A147	A148	A149	A150	A151	A152	A153	A154	A155	A156	A157	A158	A159	A160	A161	A162	A163	A164	A165	A166	A167	A168	A169	A170	A171	A172	A173	A174	A175	A176	A177	A178	A179	A180	A181	A182	A183	A184	A185	A186	A187	A188	A189	A190	A191	A192	A193	A194	A195	A196	A197	A198	A199	A200	A201	A202	A203	A204	A205	A206	A207	A208	A209	A210	A211	A212	A213	A214	A215	A216	A217	A218	A219	A220	A221	A222	A223	A224	A225	A226	A227	A228	A229	A230	A231	A232	A233	A234	A235	A236	A237	A238	A239	A240	A241	A242	A243	A244	A245	A246	A247	A248	A249	A250	A251	A252	A253	A254	A255	A256	A257	A258	A259	A260	A261	A262	A263	A264	A265	A266	A267	A268	A269	A270	A271	A272	A273	A274	A275	A276	A277	A278	A279	A280	A281	A282	A283	A284	A285	A286	A287	A288	A289	A290	A291	A292	A293	A294	A295	A296	A297	A298	A299	A300	A301	A302	A303	A304	A305	A306	A307	A308	A309	A310	A311	A312	A313	A314	A315	A316	A317	A318	A319	A320	A321	A322	A323	A324	A325	A326	A327	A328	A329	A330	A331	A332	A333	A334	A335	A336	A337	A338	A339	A340	A341	A342	A343	A344	A345	A346	A347	A348	A349	A350	A351	A352	A353	A354	A355	A356	A357	A358	A359	A360	A361	A362	A363	A364	A365	A366	A367	A368	A369	A370	A371	A372	A373	A374	A375	A376	A377	A378	A379	A380	A381	A382	A383	A384	A385	A386	A387	A388	A389	A390	A391	A392	A393	A394	A395	A396	A397	A398	A399	A400	A401	A402	A403	A404	A405	A406	A407	A408	A409	A410	A411	A412	A413	A414	A415	A416	A417	A418	A419	A420	A421	A422	A423	A424	A425	A426	A427	A428	A429	A430	A431	A432	A433	A434	A435	A436	A437	A438	A439	A440	A441	A442	A443	A444	A445	A446	A447	A448	A449	A450	A451	A452	A453	A454	A455	A456	A457	A458	A459	A460	A461	A462	A463	A464	A465	A466	A467	A468	A469	A470	A471	A472	A473	A474	A475	A476	A477	A478	A479	A480	A481	A482	A483	A484	A485	A486	A487	A488	A489	A490	A491	A492	A493	A494	A495	A496	A497	A498	A499	A500	A501	A502	A503	A504	A505	A506	A507	A508	A509	A510	A511	A512	A513	A514	A515	A516	A517	A518	A519	A520	A521	A522	A523	A52
--	----	----	----	----	----	----	----	----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-----

### Example Code Screenshot:



```

[241]_ import math
import numpy as np
import scipy as sp
from scipy import stats
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn as sk
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn import svm
from itertools import cycle

[242]_ %matplotlib inline

Helper Functions

[318]_ def cross_val_t_test(x1, x2, s1, s2, k_folds):
    """ Returns the probability from a one sided t-test
        for cross-validation. x1, x2 are the mean scores,
        s1, s2, the standard deviance, k_folds the number
        of folds used in cross validation.
    """
    from scipy import stats
    import math
    df = k_folds - 1
    t = abs(x1 - x2) / math.sqrt( s1**2/k_folds + s2**2/k_folds)
    cdf = stats.t.cdf(t,df=df)
    return 1 - cdf

def best_model_gridsearchCV(X, y, param_grid, classifier, cv=5, n_jobs=-1, verbose=10):
    """ Returns the best model generated by sk.model_selection.GridSearchCV
        X is a DataFrame or numpy array of values. y is a 1d numpy array.
        param_grid is a dictionary of parameters. classifier is a
        scikit-learn classifier object.
    """
    cl = sk.model_selection.GridSearchCV(classifier, param_grid, cv=cv, n_jobs=n_jobs, verbose=verbose)
    cl = cl.fit(X, y)
    results = pd.DataFrame(cl.cv_results_)
    print("Best Model Hyperparameters:", cl.best_estimator_)
    print("Best Model average classification score:", cl.best_score_)
    means = cl.cv_results_['mean_test_score']
    stds = cl.cv_results_['std_test_score']
    print("\nGrid Search Scores:\n")
    for mean, std, params in zip(means, stds, cl.cv_results_['params']):
        print(mean, std, params)
    best_mean = cl.best_score_
    best_std = results.loc[results['mean_test_score']==best_mean, 'std_test_score'].iloc[0]
    return best_mean, best_std, results, cl

def gen_confusion_matrix(X_test, y_test, fitted_classifier):
    """ Returns a classification report and pandas DataFrame for the confusion matrix
    """
    print("Classification Report:\n")
    y_pred = fitted_classifier.predict(X_test)
    unique_labels = np.unique([y_test, y_pred])
    report = sk.metrics.classification_report(y_test, y_pred)
    print(report)
    con_mats_df = pd.DataFrame(sk.metrics.confusion_matrix(y_test, y_pred, labels=unique_labels),
                              index=['true:{}'.format(x) for x in unique_labels],
                              columns=['pred:{}'.format(x) for x in unique_labels])
    print("Confusion Matrix:\n")
    print(con_mats_df)
    return report, con_mats_df

def plot_multiclass_roc_curve(X_test, y_test, cl, title, save_name, save_path, knn=False):
    """Plots the multiclass ROC curve."""
    n_class = len(np.unique(y_test))
    classes = np.unique(y_test)
    y_test_bin = sk.preprocessing.label_binarize(y_test, classes=classes)
  
```

