6202 – Machine Learning 1

Final Project Proposal

Bradley Reardon, Salim Haruna

6/13/21

**Music Genre Classification**

**Table of Contents**

1. Introduction
2. Dataset Overview
3. Network and Models
4. Experimental Setup
5. Results
6. Conclusion
7. Citations

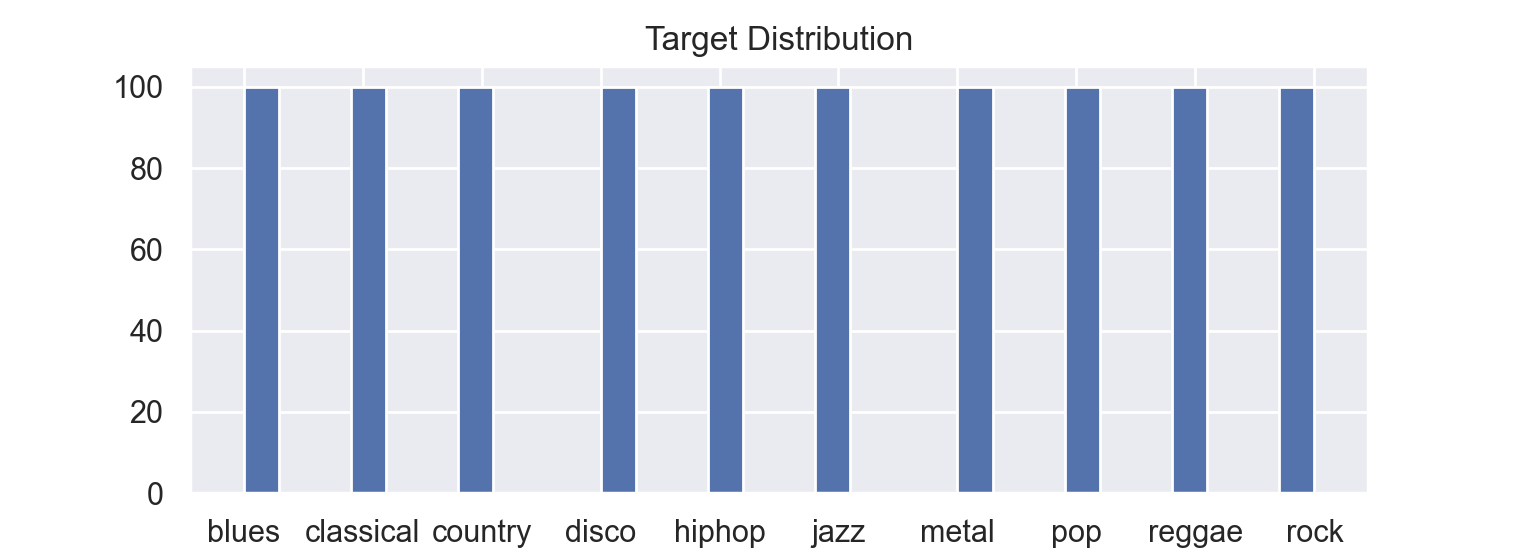
**Introduction**

Picture this: you’re sitting at a bar having a great time with your friends and this amazing song starts playing. It is nothing like you’ve ever heard before. You think to yourself, “I wonder what genre of music this is?” but you’re not even sure where to begin searching for that information. Well, worry no further. With the help of our music genre classifier, all you need is the name of the song (.mp3 file in the case of this project demo) and a few moments to spare. With the click of a button, you can find out the genre of any song you come across from here on out.

We chose to work on music genre classification as we feel it is increasingly difficult for music streaming platforms to properly classify all of the unique genre-bending music being released today. We want to help provide a solution for this issue and believe this can be solved with the help of machine learning, and more specifically with the help of a neural network. Using a training dataset found on Kaggle, we trained a multilayer perceptron classification network to classify songs originally stored as an .mp3 file. Additionally, we created an interactive graphical user interface that allows a user to select the desired .mp3 file they would like to classify into a music genre.

**Dataset Overview**

The training dataset we chose to work with is a [music genre classification dataset](https://www.kaggle.com/harish24/music-genre-classification) found on Kaggle. This dataset contains 27 features (filename, rmse, chromasft, central\_spectroid, central\_bandwidth, rolloff, zero\_cross, and 20 mfcc for the various frequencies within each .wav (Mel frequency, the way in which humans perceive sound)) and the label which contains 10 classes (blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, rock). There are 1000 rows, each row containing the feature and target data of a single .wav file. The target balance was perfect with each class comprising 10% of the dataset *(Figure 1)*.



*Figure 1: Target Distribution*

**Network and Models**

The artificial neural network we chose to use is the [MLPClassifier](https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html) from the sklearn package. A multilayer perceptron (*Figure 2*) is a perceptron model that contains multiple layers that begins with input data being fed into the first layer, and each succeeding layer being fed the output from the preceding layer as input data. The advantages of using an artificial neural network over traditional machine learning models are their adaptability of coefficients through backpropagation, their ability to approximate unknown functions, and their ability to deal with every type of problem (regression, classification, clustering, time-series). As stated in [this](https://www.sciencedirect.com/topics/computer-science/multilayer-perceptron) section of Advances in Computers by S. Abirami and P. Chitra, “The backpropagation algorithm is a form of steepest-descent algorithm in which the error signal, which is the difference between the current output of the neural network and the desired output signal, is used to adjust the weights in the output layer, and is then used to adjust the weights in the hidden layers, always going back through the network towards the inputs. Thus, although the neural network operates on the input signals to give an output in an entirely feedforward way, during learning, the resulting error is propagated back from the output to the input of the network to adjust the weights.”

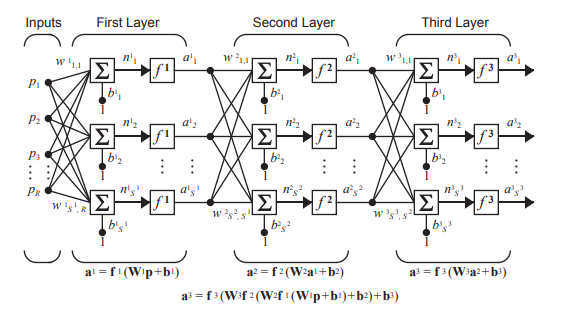
We conducted performance optimization of our network with the help of [GridSearchCV](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html) which, as described via the documentation page, is an “exhaustive search over specified parameter values for an estimator.” We included all of the possible parameter options within our hyperparameter space (barring all possible hidden-layer sizes considering there are infinite possibilities) and iterated through each parameter combination. This is a very computationally expensive approach, so in order to limit the number of times we needed to run this code, we used the *best\_params\_* attribute to return the parameter combination that resulted in the best performance. The following code shows the optimal MLPClassifier parameters for our specific project:

*MLPClassifier(*

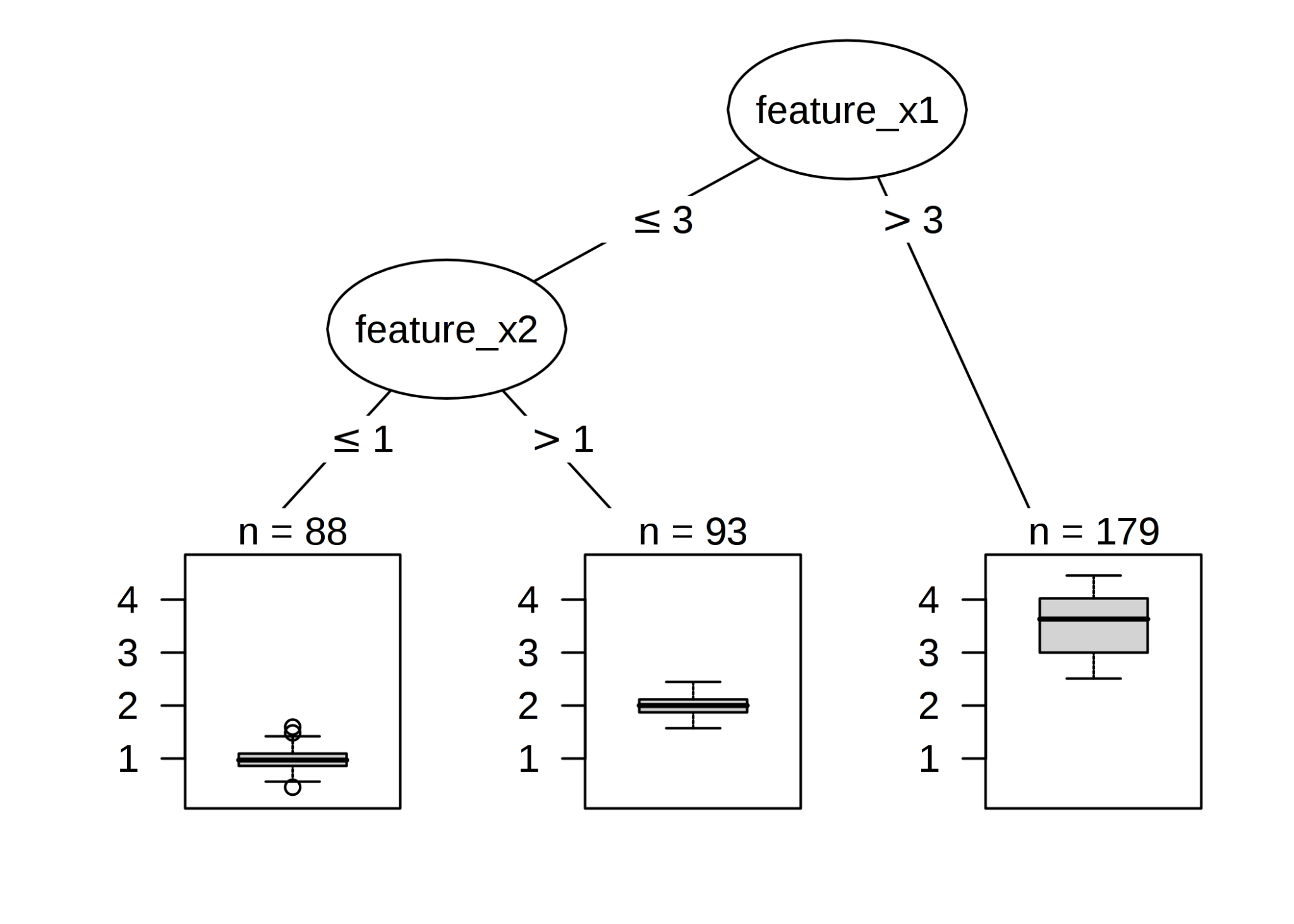
*hidden\_layer\_sizes=(60,100,60),max\_iter=10000,learning\_rate='invscaling,solver="adam", activation='tanh', alpha= 0.0001*

*)*

In order to determine if our artificial neural network would outperform a more traditional machine learning model, we also ran our data through a decision tree *(Figure 3)* model using DecisionTreeClassifier from the sklearn package. In machine learning, a decision tree is a type of supervised learning used for classification where the data is continuously split into categories based on feature parameters until the model has enough information to classify the input data.



*Figure 2: Multilayer Perceptron Example*

**

*Figure 3: Decision Tree Example*

**Experimental Setup**

Since the target data in our dataset was originally in string form, we used a label encoder to transform the targets into integer form. Additionally, we standardized our feature data to ensure all features were on the same scale and were cross-comparable. Using the *train\_test\_split* function from the sklearn package, we split the music genre dataset into train and test sets with a 4:1 split respectively. After training and testing our network with the optimal features, we were able to consistently receive both F1 and accuracy scores of ~65%. In order to test our model on unseen data, we needed to first convert the chosen .mp3 file into a .wav file and extract the features from said .wav file. The functions to do so, *convert\_mp3\_to\_wave* and *wav\_features*, require the use of the [librosa](https://librosa.org/doc/latest/index.html) and [pydub](https://github.com/jiaaro/pydub) packages and can be found in the [Code folder](https://github.com/breardon7/Music-Genre-Classification/tree/master/Code) of our github repository. The *wav\_features* function creates a new dataframe consisting of the same features as the training dataset. Each row in this new .wav dataset represents a single byte of the .wav file wavelength, and the number of rows varies depending on the wavelength of the .mp3 that was converted. The non-zero mean of each feature is calculated and the single row dataset is then fed into the network and the row receives its own class label that determines the .mp3 genre as predicted by the network.

Using the [PyQT5](https://pypi.org/project/PyQt5/) package, we created a GUI that allows the user to select their desired .mp3 file they want to determine the music genre of. To do this, run the [main.py](https://github.com/breardon7/Music-Genre-Classification/blob/master/Code/main.py) and wait for the window to open. Once open, click the *Select mp3 File* which will open your directory where you can then navigate to the desired .mp3. Select the .mp3 and wait for the file to be run through the process mentioned above. Once finished, the GUI will display a classification report for both the *MLPClassifier* and *DecisionTreeClassifier,* a music genre classification, and a plot showing the monophonic waveform of the .mp3 file.

**Results**

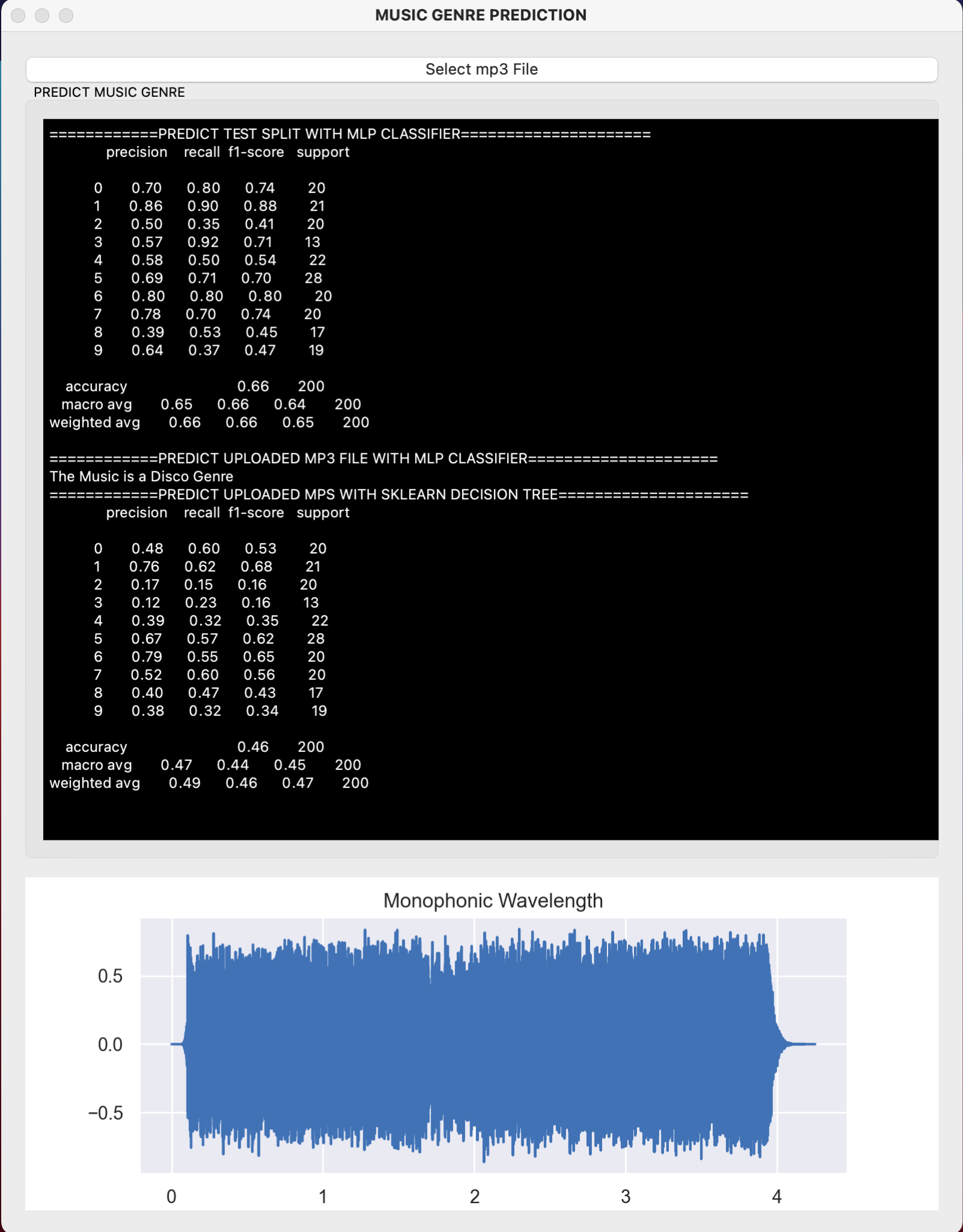
In order to arrive at the final results, we must first look at the results returned from converting an .mp3 file into a .wav file and conducting feature extraction. In *Figure 4*, we see the dataset that is returned after an .mp3 file is selected and pipelined through the *convert\_to\_wav* and *wav\_features* functions. As mentioned earlier, each row represents a single byte of the .wav file wavelength.



*Figure 4: .wav file dataset after conversion from .mp3*

This dataset is passed through the *predict\_music\_genre* method (within the *predict\_music\_genre\_window* class)where the dataset feeds into the *MLPClassifier* network and *DecisionTreeClassifier* model. The GUI output, represented in *Figure 5*, returns a classification report for both classifiers, the music genre that the artificial neural network classified the .mp3 as, and the plot of the monophonic waveform representing the wavelength, amplitude, pitch and the frequency of the .mp3 file.

To our surprise, the multilayer perceptron consistently outperformed the decision tree model. On average, the *MLPClassifier* returned an accuracy score of 0.65 meaning it correctly classified the .mp3 files 65% of the time. As we can see in *Figure 5* based on the F1 scores being near or below 0.50, the *MLPClassifier* struggled with classifying some classes more than others, particularly with classifying classical, disco, pop, and reggae observations. The remaining classes were often classified with high accuracy, with F1 scores being greater than 0.70.



*Figure 5: GUI Output Example with results*

**Conclusion**

Given that the dataset we used to train the network and model was relatively small, we assumed that the multilayer perceptron might not outperform the decision tree model considering artificial neural networks often require larger datasets to perform well. This is why traditional models are still constantly used in industry, as many companies are not working with large enough datasets for artificial neural networks to be the optimal choice. Overall, the *MLPClassifier* outperformed the *DecisionTreeClassifier* with its accuracy score being 0.20 higher on average than that of the *DecisionTreeClassifier*. Additionally, the highest variance amongst *MLPClassifier* F1 scores was 0.40 while that of the *DecisionTreeClassifier* was 0.50, meaning that the *MLPClassifier* was more efficient in correctly classifying each class.

A difficulty we noticed is that songs might not always perfectly fit into one genre. This causes high single-label classification precision to be difficult to achieve consistently, and is due to modern music overlapping and being influenced by multiple genres. The features of a song that fits this description will fall into different genres throughout a single .mp3 file, and thus cause noise in the classification process.

A caveat we dealt with during this project is that our training data was limited to 10 core music genres. In an ideal setting, our training data would include observations from a much larger array of genres which would allow us to more precisely train a network and ultimately improve its classification ability. In the future, we would like to continue adding as much .wav genre data into our training dataset as possible to achieve higher accuracy when classifying songs into genres.

**Citations**

Abirami , S., & Chitra, P. (n.d.). *Multilayer Perceptron*. Multilayer Perceptron - an overview | ScienceDirect Topics. https://www.sciencedirect.com/topics/computer-science/multilayer-perceptron.

*GridSearchCV*. scikit. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.GridSearchCV.html.

Jiaaro. (n.d.). *jiaaro/pydub*. GitHub. https://github.com/jiaaro/pydub.

librosa. (n.d.). https://librosa.org/doc/latest/index.html.

*MLPClassifier*. scikit. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.neural\_network.MLPClassifier.html.

Natarajan, H. (2020, November 3). *Music Genre Classification*. Kaggle. https://www.kaggle.com/harish24/music-genre-classification.

**Appendix**