Jared Baca
CS-677 Data Science with Python
Joshua Enxing
25 September 2023

Music Genre Classification

CS-677 Term Project Report

Project Overview

The goal of this project was to use machine learning tools available in Scikit Learn to build a classifier that would determine the genre of a piece of recorded music based on features of the audio signal. The model was trained using the GTZAN Dataset, which is comprised of 1000 audio files representing ten music genres.

About the Dataset

The feature matrix of the GTZAN dataset contains ten primary features that represent three fundamental aspects of music: timbre, rhythm, and harmony. The features are as follows:

- **Chroma STFT** (Short Term Fourier Transform) - The range of pitches within the segment of music.
- **RMS** (Root Mean Square) – A measure of the overall loudness
- **Spectral Centroid** – "Center of gravity" of the sound's frequency content
- **Spectral Bandwidth –** The range of variance around the spectral centroid
- **Spectral Roll-Off –** The frequency below which 85% of the total spectral energy lies
- **Zero Crossing Rate** – Rate at which the signal changes from positive to negative; represents the overall "noisiness" of the audio file
- **Harmony –** The individual musical pitches (frequencies) present
- **Perceptr –** Unclear what this value represents
- **Tempo –** number of beats per minute
- **Mel-Frequency Cepstral Coefficients** (MFCC's) – Coefficient that provide a "compact representation of the spectral envelope" (Tzanetakis), most commonly used in speech recognition.

Many features include both a mean and variance value, and there is a total of twenty MFCC's, bringing the overall number of features close to fifty. The response vector contains labels for ten distinct genres: blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, and rock.

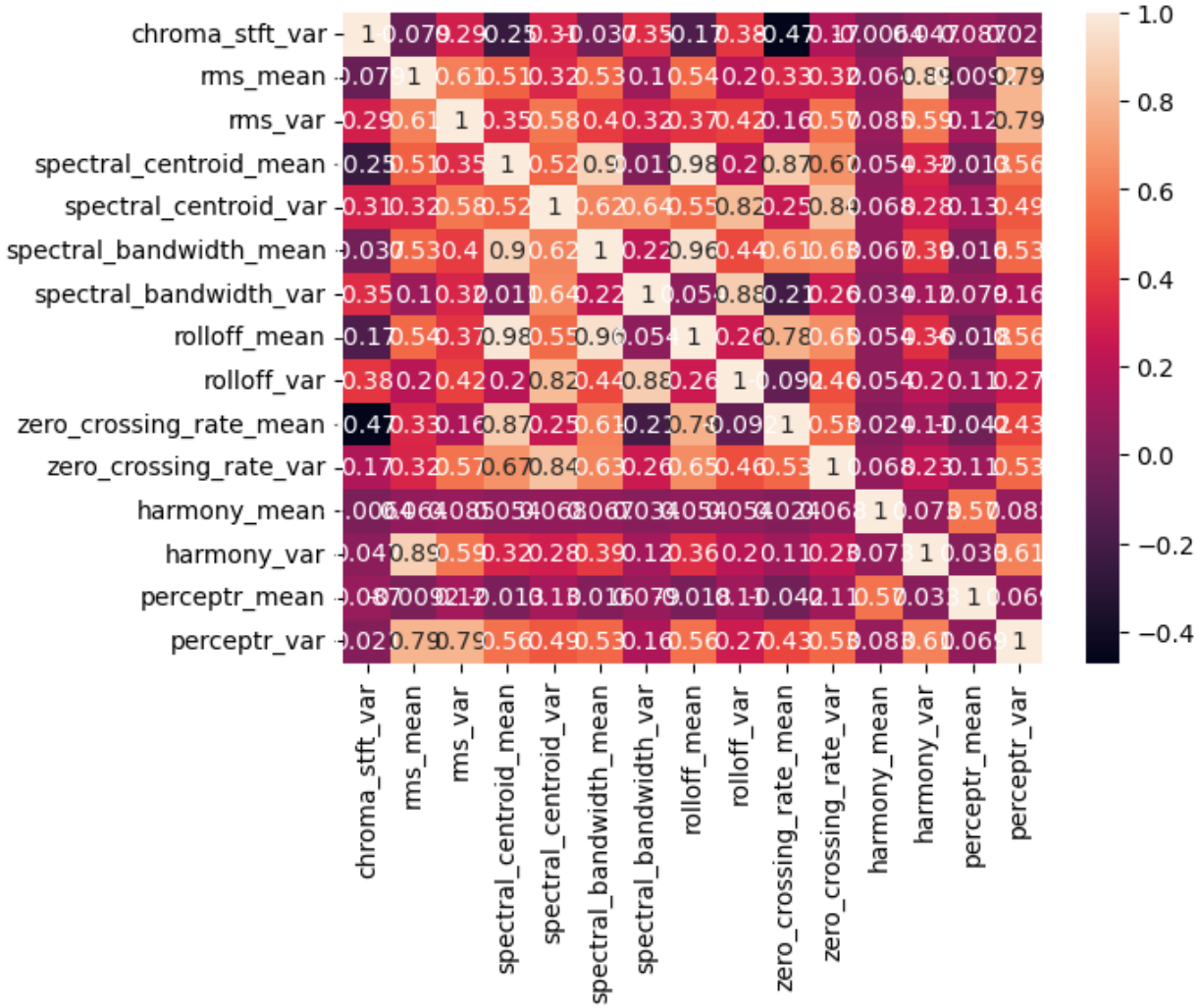
The spectral features proved to be the most significant in determining the genre. These include spectral centroid, spectral bandwidth, and spectral roll-off, as well as the Mel-Frequency

Cepstral Coefficients. These features represent the frequency content of each audio file and correspond to the timbre or overall tone of the music. A thorough mathematical explanation of the features in the dataset is beyond the scope of this project summary, but a detailed explanation can be found in the paper *Musical Genre Classification of Audio Signals* by George Tzanetakis.

Methodology

Preprocessing and Feature Engineering

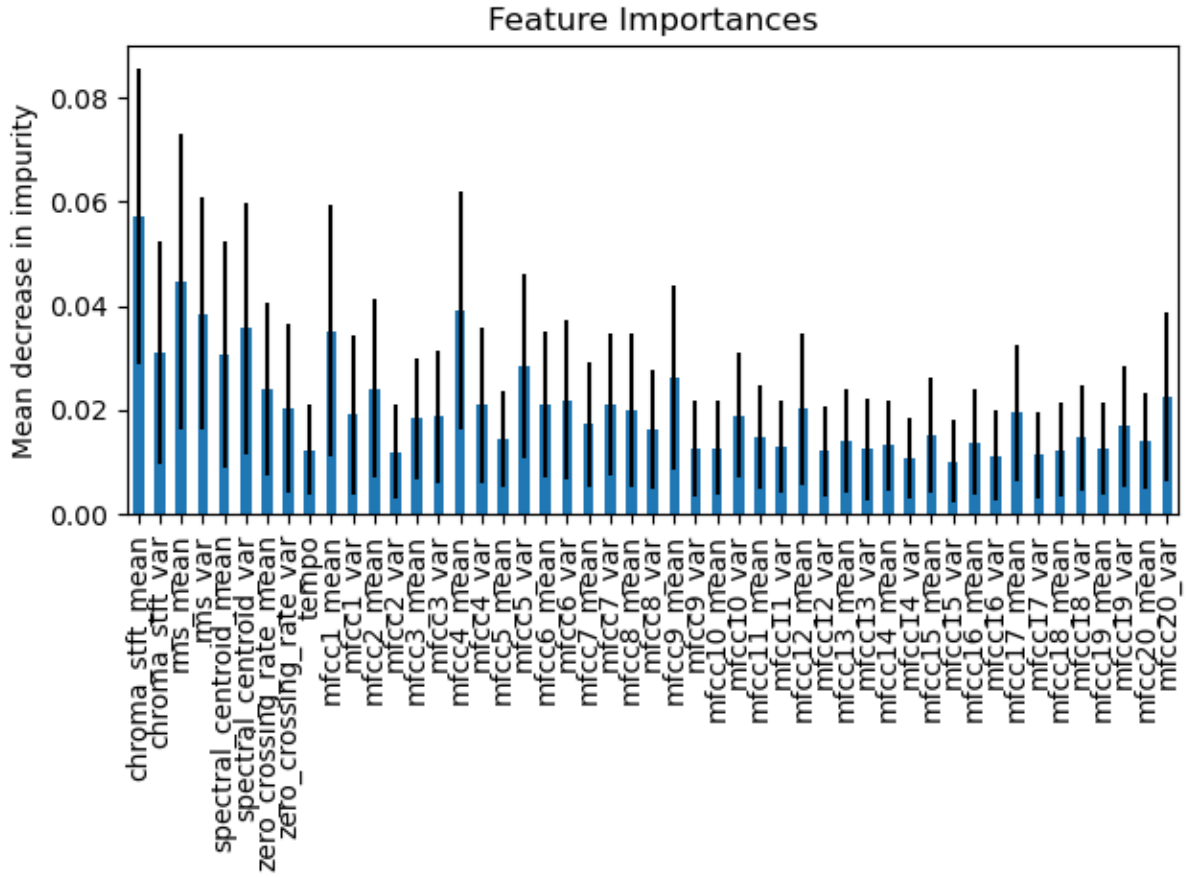
In order to determine which of the fifty features were most significant, I first generated the following correlation matrix.



As expected, the most highly correlated features are those that represent the spectral envelope of the audio file: spectral centroid, spectral bandwidth, and spectral roll-off. These three together give an idea of where the audio segment resides in the frequency spectrum. A high spectral centroid, for example, suggests music with a larger concentration of high frequencies, while a low value would suggest music that has more low frequency (bass-heavy) content. Since it is considered best practice to avoid highly correlated features, I subsequently dropped spectral bandwidth and spectral roll-off, keeping only spectral centroid. This change had no measurable impact on the accuracy of the model. To further narrow the focus of the feature set, I dropped Harmony and Perceptr, which also did not lead to a noticeable loss of accuracy.

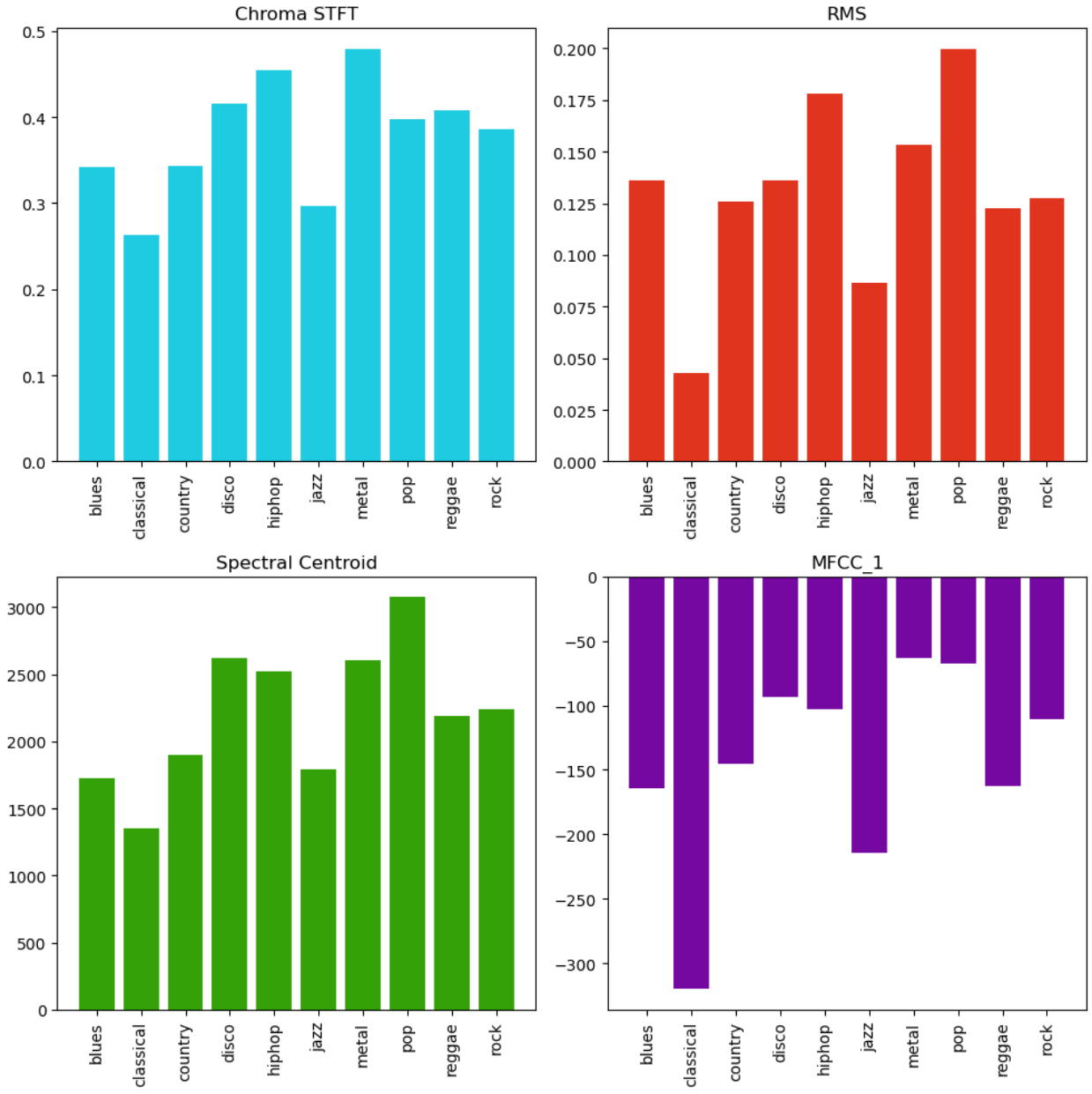
After removing these features, I scaled the data using the Standard Scaler in SciKit Learn's preprocessing module. The choice to scale the data was based on the fact that one of the models I planned to test was K-Nearest Neighbors. Scaling the data is required for KNN, and did not have an adverse impact on the accuracy of the other models, so I opted to work with the scaled data throughout.

In addition to the correlation matrix, I plotted the feature importance, as shown below:



Taking the four most significant features, I then generated a bar chart to show the values of those features among the ten different classes, or musical genres. This provided a visual representation of some of the similarities and differences between the genres. For example, recordings in the Pop genre have noticeably higher values for spectral centroid and RMS (which

measures overall “loudness”) when compared to genres at the opposite end of the spectrum, most notably Classical and Jazz.



Model Selection

After reducing the size of the features set, I decided to apply five of the classification models covered in this course – KNN, Gaussian Naïve Bayes, Random Forest, Decision Tree, and a Voting Classifier comprised of the other four. I split the data into training and test sets, trained each of the models on the training set and generated predictions from the test set. The accuracy results were as follows:

KNeighborsClassifier accuracy: 0.7
GaussianNB accuracy: 0.55
RandomForestClassifier accuracy: 0.725
DecisionTreeClassifier accuracy: 0.495
VotingClassifier accuracy: 0.695

KNN and Random Forest had the highest accuracy scores, with RF performing best overall. With this in mind, I went back and added a cell in the notebook to evaluate the Random Forest Classifier and obtain the optimal hyperparameters using the Randomized Search CV class.

Feature Extraction

Once the decision to use the Random Forest Classifier had been made, the second phase of this project was to use the trained model to build a classifier that could be used on user-selected audio files not found in the original dataset. This required the ability to extract the same features used in the GTZAN dataset. The Librosa Python library made this possible, as it contains methods for extracting nearly all of the features in our feature matrix. This process was performed in a simple Python function. The function takes an audio file as input and extracts the desired features: chroma stft, rms, spectral centroid, zero-crossing rate, tempo, and MFCC's. It then organizes these values into a Pandas dataframe, scales the data, and returns the dataframe, which is then ready to be passed to the trained Random Forest Classifier.

In addition to placing this code in the Jupyter Notebook that was used for analyzing the data and training the models, I placed it in its own standalone Python file for the sake of portability. Using Pickle and Joblib, I exported the trained scaler and classifier files and imported them into the classifier.py file. I then placed the feature extraction code in its own file titled extract.py, which I imported into the classifier file as well. Separating these files and exporting the trained model allowed the classifier to be packaged as a standalone application without the dataset and the rest of the model training code. Lastly, I created a simple GUI using Tkinter that allows a user to upload an audio file, play back the file, and run the classification model to generate a prediction of the genre.

Results

The highest accuracy recorded during the tests was 75%, achieved by the Random Forest Classifier. However, when applied to files outside the dataset, the results achieved were noticeably less accurate. Still, the model performs slightly better than that of a coin flip. The results seem to align with those of Tzanetakis, who in his conclusion states that "[u]sing the proposed feature sets classification of 61% (nonreal time) and 44% (real time), has been achieved in a dataset consisting of ten musical genres."

Conclusion

While other research suggests that there may be more accurate ways to achieve music genre classification using neural networks and or other more sophisticated tools, this project illustrates how each of the different facets of musical composition can be represented as data, and how patterns can emerge from that data. Considering how complex and subjective music genre classification is, even when performed by the human ear, the performance of this classifier can be considered favorable.

References

Tzanetakis, George, and Cook Perry. *Music Genre Classification of Audio Signals*. IEEE, 2002. Print.

Pandey, Parul. "Music Genre Classification with Python." *Towards Data Science.* Dec 13, 2018. Web. <https://towardsdatascience.com/music-genre-classification-with-python-c714d032f0d8>.

McFee, Brian, et al. "Librosa: Audio and Music Signal Analysis in Python". *SciPy.* 2015, 2015. Print.

Cd. "Spotify Genre Classification Algorithm." *Towards Data Science.* April 22, 2021. Web. <https://towardsdatascience.com/spotify-genre-classification-algorithm-88051db23d42>.