# Project 2: Music Genre Classifier
# Karim Sasa, 9/16/2024

### Kaggle Dataset:

https://www.kaggle.com/datasets/insiyeah/musicfeatures - Note I used data.csv

### Introduction to the Problem:

I listen to different types of music everyday as a way to relax my nerves. Everyone has a type of music they prefer, but what classifies a music genre besides the person making the song? As I love to listen to music, I'm not completely sure what type of genre I like to listen to. I decided to look at this Kaggle dataset that has a total of ten different genres of music which include: blues, classical, country, disco, hiphop, jazz, metal, pop, reggae & rock. I want to mainly discover what classifies the genre of music is it the tempo, beats, chroma stft, etc. I'd like to take all these features to determine what impacts the result the most.

### Introduction to the Data :

The data is from a Kaggle dataset, the reference is listed at the top of this document, and it gathered info from 1000 different songs with the genre provided for each song. There are 30 features listed about each of those songs, and they list the information about the filename, tempo, beats Rythmic, chroma_stft, rmse, spectral_centroid, spectral_bandwidth, Roll-off, zero_crossing_rate, mfcc1 to mfcc20, and labels. As someone who doesn't know what most of these features mean I have created a data dictionary below with the information provided by the kaggle dataset with the descriptions of what each of these features describe about the song. I only included mfcc1 because the others would be defined the same.

| Feature Name | Type | Description | % Missing |
|---|---|---|---|
| filename | Categorical | The filename for each audio file in the dataset. | 0% |
| tempo | Numeric | The speed at which a passage of music is played (beats per minute). | 0% |

| | | | |
|---|---|---|---|
| beats | Numeric | The rhythmic unit in music. | 0% |
| chroma_stft | Numeric | Short-Time Fourier Transform of the chroma, representing the harmonic content. | 0% |
| rmse | Numeric | Root Mean Square Error of the signal, representing the energy of the signal. | 0% |
| spectral_centroid | Numeric | Indicates the center of mass of the spectrum, i.e., the frequency around which most of the energy is concentrated. | 0% |
| spectral_bandwidth | Numeric | Wavelength interval in which a radiated spectral quantity is not less than half its maximum value. | 0% |
| rolloff | Numeric | The frequency below which a specified percentage of the total spectral energy lies. Typically used to distinguish between speech and music. | 0% |
| zero_crossing_rate | Numeric | The rate at which the signal changes sign from positive to negative or vice versa. | 0% |

| | | | |
|---|---|---|---|
| mfcc1 | Numeric | Mel-frequency cepstral coefficients, which represent the short-term power spectrum of the sound. | 0% |
| unique values | Categorical | Unique classification labels for the music genres (e.g., blues, classical, rock, etc.). | 0% |
| Label | Categorical | The classification label of the genre associated with the audio file. | 0% |

## Pre-processing step:

Upon first look, the data looks pretty clean, but there were only a few preprocessing steps I had to take. First, I removed irrelevant data that won't contribute to the classification algorithm because I don't need 19 other mfcc files, so I only kept mfcc1, and I also removed the file name because that won't contribute to the model. I also checked for duplicate values, and there were 13 entries of duplicated data. I removed those 13 entries as they had the same exact tempo value as some other entries, and the model wouldn't benefit from these similar entries. All the data seems to be correct, so there is no need for type conversion. Now for syntax errors, there are no typos or whitespace. The standardization step is not required as all the data types converted are lowercase already lowercase. The scaling step isn't essential, as my data doesn't really have large values, and it's not required for the random forest model, the same goes for normalization. I also had no missing values either. I did however use a label encoder from Sklearn to make it so the non-numerical data turns into numerical data for the model to understand.

This is the resulted preprocessed data frame:

| | tempo | beats | chroma_stft | rmse | spectral_centroid | spectral_bandwidth | rolloff | zero_crossing_rate | mfcc1 | label |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 103.359375 | 50 | 0.380260 | 0.248262 | 2116.942959 | 1956.611056 | 4196.107960 | 0.127272 | -26.929785 | 0 |
| 1 | 95.703125 | 44 | 0.306451 | 0.113475 | 1156.070496 | 1497.668176 | 2170.053545 | 0.058613 | -233.860772 | 0 |
| 2 | 151.999081 | 75 | 0.253487 | 0.151571 | 1331.073970 | 1973.643437 | 2900.174130 | 0.042967 | -221.802549 | 0 |
| 3 | 184.570312 | 91 | 0.269320 | 0.119072 | 1361.045467 | 1567.804596 | 2739.625101 | 0.069124 | -207.208080 | 0 |
| 4 | 161.499023 | 74 | 0.391059 | 0.137728 | 1811.076084 | 2052.332563 | 3927.809582 | 0.075480 | -145.434568 | 0 |

## Data Understanding/Visualization

For visualizing the questions I had for my data I used Seaborn, and I used boxplots, and counplots. For each visualization, it helped me to understand each feature, and some of the visualizations lead to some interesting insights. Honestly, all the visualizations surprised me, but that's just because I thought some genres would be a lot more distinct, but I go more in detail in each visualization story.

### Visualization 1: Distribution of Genres (Target Variable): Story

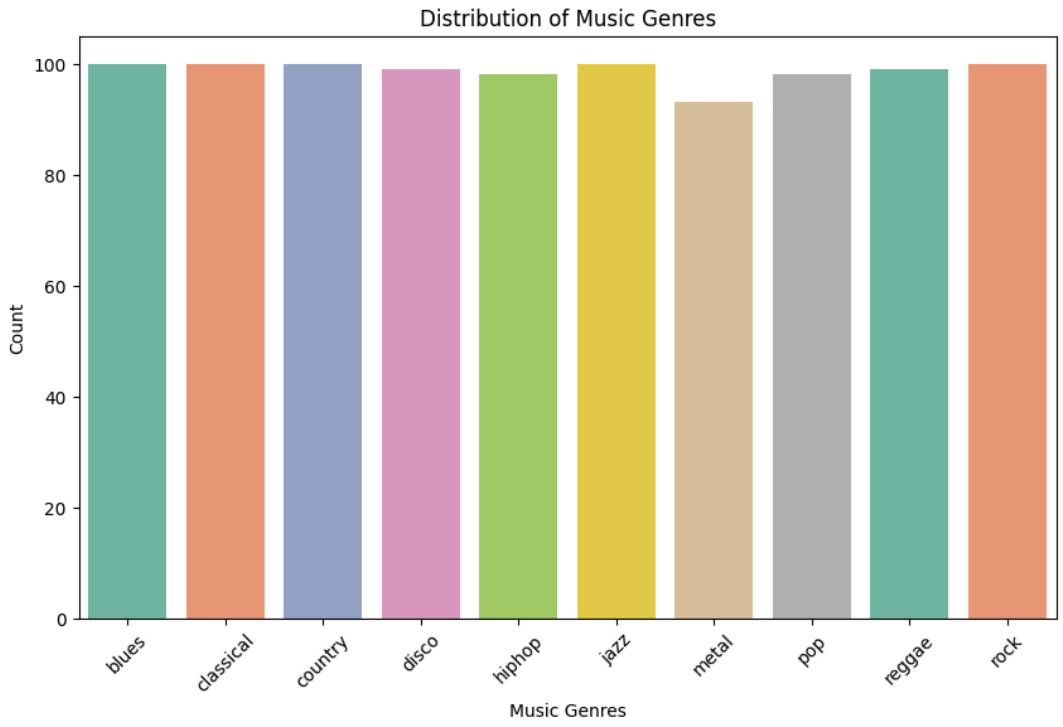
To answer my overarching question of which features determine a genre the most, I believe it's important to first understand the distribution of my target variable, which in my case is the music genres. I created a figure that shows the distribution of genres in the dataset, which includes 10 genres: blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, and rock.

The purpose of my visualization is to see how balanced each genre is in the dataset. From the bar plot, you can observe that all genres are pretty balanced, and they would've all been equal if I hadn't removed the duplicates.

### Relevance to the Model

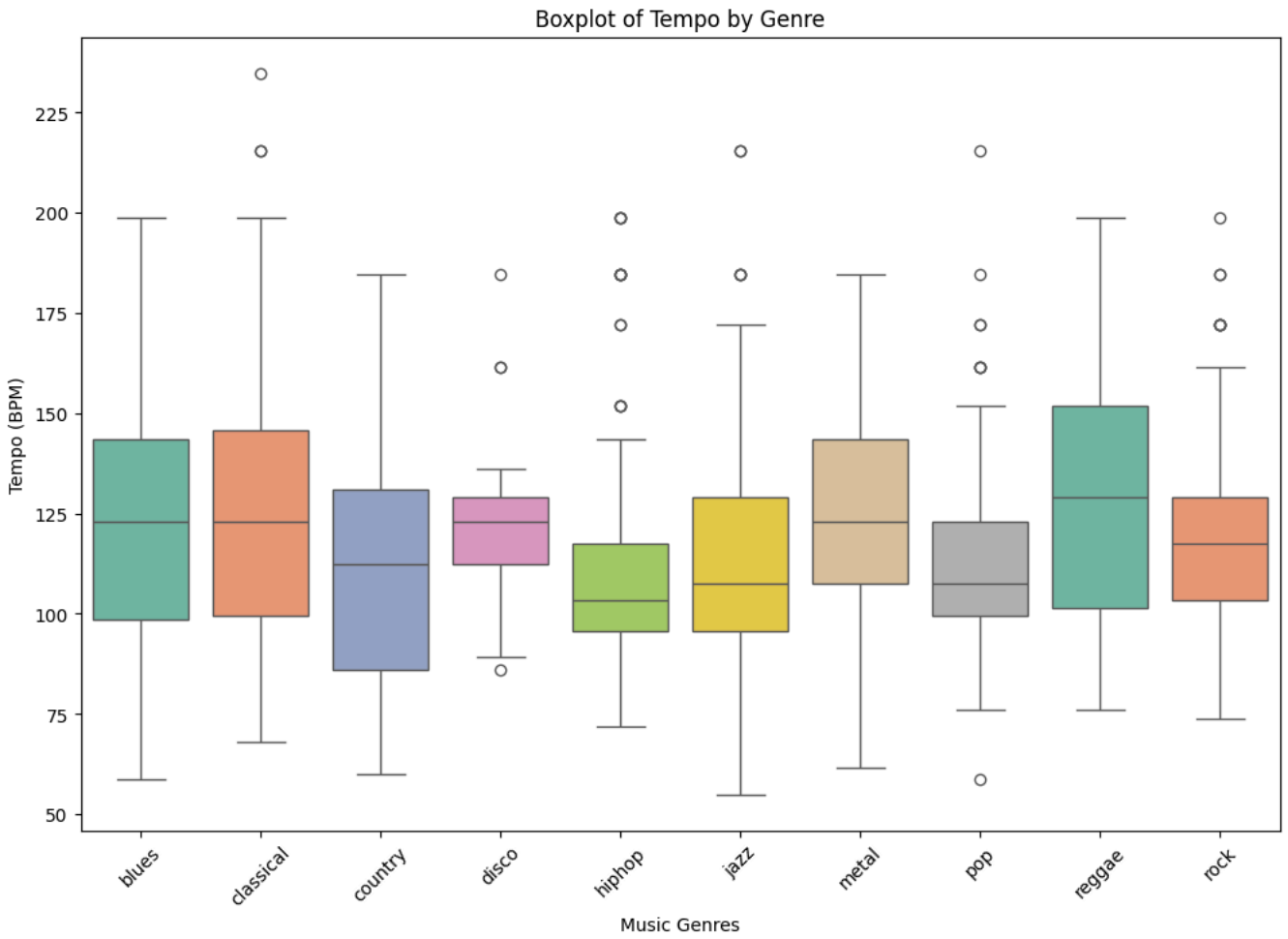
I checked for this just in case, because imbalances can affect the performance of the classification model. For example, if metal and pop dominated the dataset, the model may favor these genres and perform less accurately for hypothetical underrepresented genres like classical and jazz. If it were unbalanced, we would need to balance the dataset or adjust the class weights to ensure fair prediction across all genres.

Even though this distribution might seem expected, it provides a clear understanding of the dataset's structure.




**Visualization 2: Tempo Distribution Across Music Genres: Story**

I was curious to find what features could distinguish each music genre, so I decided to examine the distribution of tempo across the different genres using a boxplot. Tempo, which is measured in beats per minute (BPM), is one of the key characteristics that defines the pacing and style of a song.




From the boxplot, I noticed some interesting details:

1. **Smaller IQRs in Disco, Hiphop, Pop, and Rock**: You can see that in these genres they have smaller interquartile ranges (IQRs) compared to the others. This means that the tempo of most of these songs in these genres is more consistent. For example,disco has the most narrow range of tempo values, indicating that songs in this genre tend to follow a more regular and predictable beat. It could suggest that tempo is a defining feature for these genres, especially disco, which has a median tempo similar to the upper bounds of other genres.

2. **Outliers**: Several genres show outliers with notably high or low tempo values, such as the outliers in classical and pop. I believe that these outliers might

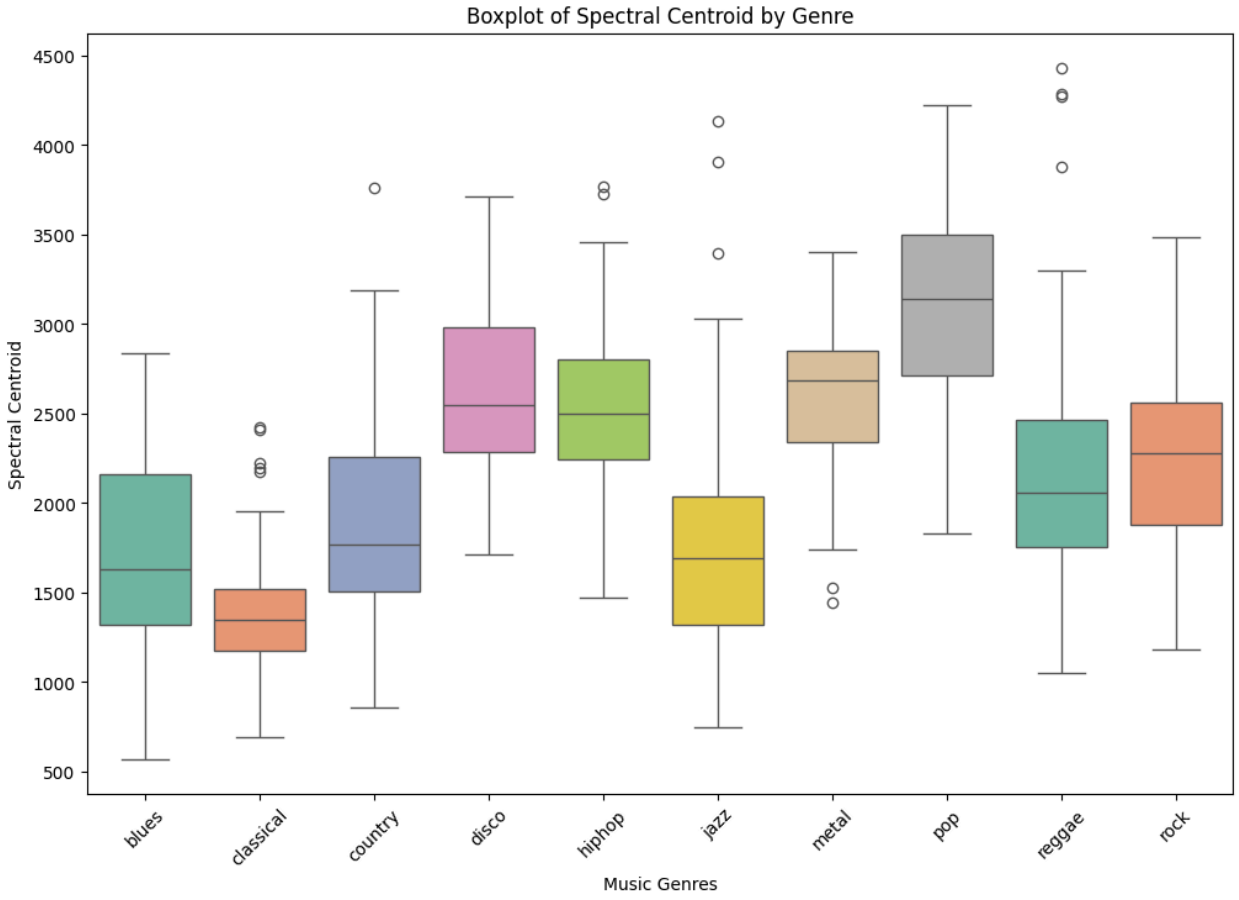
represent specific subgenres or unusual songs that deviate from the typical tempo of their genre.

**Relevance to the Model**

This visualization suggests that tempo could play an important role in distinguishing genres like pop, disco, rock and hiphop, which have more defined tempo ranges.

**Visualization 3: Boxplot of Spectral Centroid by Genre: Story**

Doing some further research on what features could make music distinct, I came across details about the spectral centroid. Basically, it's a feature related to the "brightness" of a sound, and the higher values indicate a high-pitched sound, and lower values represent darker, bass-heavy sounds. I made another boxplot, to explore the distribution of spectral centroid across different music genres to see how this feature might help distinguish between them.



My observations:

1. **Classical Music's Low Spectral Centroid**: As I would expect, classic music shows much lower spectral centroid values compared to other genres as the “brightness” of the songs is generally low. Also, the relatively small interquartile range (IQR) I believe shows a more consistent quality for this genre.
2. **Jazz, Blues, and Country**: These three genres have relatively lower spectral centroid values as well, and it suggests that they lean towards lower-frequency, less bright sounds. The IQR for jazz is decently narrow with fewer songs that deviate significantly from the median value.
3. **Metal, Pop, and Disco**: Upon first glance, you can see that these genres show much higher spectral centroid values, reflecting the known “brighter” sound profiles typically found in metal guitars, pop vocals, and disco's use of synthetic, high-pitched instruments. The high variability in pop’s music's spectral centroid values shows that pop songs can have both bright and darker qualities in tone, suggesting that this genre has a diverse range of sound.

**Relevance to the Model**

Since genres like classical have low and consistent spectral centroid values, I think it’ll be easily distinguished by this feature. In comparison, genres like pop and rock, which have higher variability, might require more details on the features to have a more accurate classification.

Metal and disco, with their brighter sounds and higher spectral centroid values, are likely easier to classify based on this feature alone.

## Modeling:

I looked over a few of the models, and decided to use the Random Forest classifier. First off, Random Forest is known as a reliable model for classification tasks and works well with datasets like mine that have somewhat complex numerical features such as tempo, beats, chroma stft, and others. My main reason for choosing Random Forest is its ability to handle complex interactions between features, and it doesn’t need that much “tuning”. Its main strength is that it reduces the risk of overfitting by building multiple decision trees and averaging their results, which is helpful in maintaining a balance between both bias and variance.

**How the Random Forest Works**:

- Random Forest is basically builds a series of decision trees on random subsets of the data. Each tree is trained on a different subset, and the final prediction is made by averaging the predictions of all trees (for regression) or taking the majority vote (for classification).
- Overall, this method reduces variance and provides better generalization of the data.

**Cons:**

- Computational Complexity - With large sets of data it can be costly
- Memory Usage - It simply uses a lot of memory
- Prediction Time - It takes a while to make predictions
- Hard to Interpret - Might always not be clear on what the results show
- Overfitting - It reduces overfitting yes, but still it can suffer from it.

**What Classification Metrics Did I use?**

For evaluating the performance of my Random Forest models, I only used the classification report as my primary evaluation metric. The classification report provides a detailed breakdown of how the model performed on each class (music genre), including metrics like precision, recall, F1-Score, and support.

**Why?**

I used the classification report because it provides a detailed evaluation of how well the model performs on each genre in general. It includes the precision which is how accurate predictions are, recall how well the model captures actual instances of each genre, and the F1-score which is essentially a balance between precision and recall. With these metrics, they helped me to understand not only the overall performance but also how well the model distinguishes between different music genres.

Now onto our first model:

Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.48 | 0.35 | 0.41 | 34 |
| 1 | 0.79 | 0.95 | 0.86 | 20 |
| 2 | 0.39 | 0.53 | 0.45 | 34 |
| 3 | 0.43 | 0.36 | 0.39 | 33 |
| 4 | 0.64 | 0.56 | 0.60 | 32 |
| 5 | 0.55 | 0.53 | 0.54 | 34 |
| 6 | 0.49 | 0.92 | 0.64 | 25 |
| 7 | 0.56 | 0.54 | 0.55 | 26 |
| 8 | 0.58 | 0.37 | 0.45 | 30 |
| 9 | 0.32 | 0.24 | 0.27 | 29 |
| accuracy | | | 0.51 | 297 |
| macro avg | 0.52 | 0.54 | 0.52 | 297 |
| weighted avg | 0.51 | 0.51 | 0.50 | 297 |

Note 0 - 9 represent blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, and rock respectively.

The first model run is not too bad, but not good either as it has an overall average of a 51% accuracy. Though, the model was able to predict classical, and hiphop pretty well, and I believe that's the case due to hip hop and classical music's distinguishing features

To try to improve the score, I'm going to mess around with the hyperparameters, and I will say I did use external resources to find out how to do this, and they are referenced below. Basically, the code finds the best parameters for the data, and then gives the accuracy score for the data.

```python
param_grid = {
    'n_estimators': [100, 300, 500, 700, 900],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
}

rf_model = RandomForestClassifier(random_state=42)

grid_search = GridSearchCV(
    estimator=rf_model,
    param_grid=param_grid,
    cv=5,
    n_jobs=-1,
    verbose=2
)

grid_search.fit(X_train, y_train)

print("Best parameters found: ", grid_search.best_params_)

best_rf_model = grid_search.best_estimator_
y_pred = best_rf_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
```

✓ 1m 41.0s

However, the results:

```
...                precision    recall  f1-score   support

           0       0.50      0.38      0.43        34
           1       0.80      1.00      0.89        20
           2       0.39      0.59      0.47        34
           3       0.43      0.36      0.39        33
           4       0.62      0.56      0.59        32
           5       0.55      0.50      0.52        34
           6       0.53      0.92      0.68        25
           7       0.56      0.54      0.55        26
           8       0.61      0.37      0.46        30
           9       0.29      0.21      0.24        29

    accuracy                           0.52       297
   macro avg       0.53      0.54      0.52       297
weighted avg       0.52      0.52      0.51       297
```

As you can see the results have slightly improved by a percent, and with this I'm not sure how to improve any further, my models did not work out very well, but it may be due to me removing the other 19 mfcc columns. The high precision, recall, and f1-score

for the classical music genre shows that that the model is able to accurately identify classical music while also correctly capturing the most instances of it, which makes it highly effective at distinguishing classical tracks from other genres!

## Storytelling - Post Modeling:

At the start of the project, my main question was what defines a music genre in terms of its audio features, such as tempo, spectral centroid, and beats. After performing an initial analysis, I trained a Random Forest model and evaluated its performance.

Here's what I learned from the process:

**Initial Results**: First off, the model was able to distinguish genres like classical and hip-hop pretty well, and I assume it's because these genres have distinct audio features that set them apart. For example, classical music had a more precise and lower spectral centroid value, while hip-hop seemed to have more of a precise tempo-range.

**Challenges with Other Genres**: Some genres just did not work out in the model. The ones that didn't work out were: rock, county, and disco. I think the problem with these genres is that they have too many overlapping features, and this might have confused the classifier. This suggests that the audio features I selected might not be sufficient to fully differentiate between these genres, and I probably should've kept more of the mfcc values.

**If I were to Start Over**: After I used the code for hyperparameter tuning, the model's performance only improved slightly. I was disappointed, but it once again it makes me believe that the model might benefit from incorporating additional features, such as the other MFCC coefficients that were removed during preprocessing, but I really didn't think they were important at first. Removing those features may have caused the model to lose some of the important information that might  distinguish certain genres.

**Was I able to answer my initial question?**
Well, with the features given in this dataset it's about 80% precise for only the classical genre, so I think I can say the classical music could be classified more often than other genres with these features in my dataset, but it just shows that I need more overall data to train my model on.

Overall, doing this project has taught me a lot about how music genres can be classified based on their audio characteristics, but some need more characteristics than others. Also, I think predicting music genres, especially in this day and age is difficult. While my model didn't achieve extremely high accuracy, it gave me insights into which features are important for genre classification, and I do see where improvements can be made.

## Impact

For my music genre classification project, I believe that there are several potential impacts that can be both positive and negative. First off, my dataset contains audio features for only 1,000 songs, which really isn't enough, and when you think about each genre it's only 100ish songs for each genre. This could limit the model as it may not perform well on songs from less common genres of music that blend-in multiple genres. Additionally, there can be cultural biases in the dataset, as it may over-represent Western music genres and under-represent every other genre, leading to misclassification or underperformance with more diverse music contexts.

Another issue I could see happening is the over-reliance on this genre classification model. What I mean by this is streaming services or record labels could depend on this to classify their genre. I do believe that the automated genre classification could help improve music recommendations, but I think defining music to one genre is not always the case. It might restrict certain artists or genres to a limited set of features, and it can affect how their music is marketed and perceived by listeners. To rephrase, artists whose songs may contain multiple genres might be unfairly categorized, and this could limit their exposure due to people looking for songs to listen to.

For ethical concerns, there could be a concern about how the model may influence music discovery. The songs that are promoted based on the model's genre prediction may create a situation where the listeners are repeatedly exposed to similar music. This could reduce exposure to new or unconventional genres, potentially ruining creativity and diversity in musical experiences.

Lastly, the use of this model by music streaming platforms could have an economic impact. By making incorrect playlists and recommendations, these platforms might affect the visibility and reach of specific artists or genres, and this could influence what type of genres that may become popular or commercially successful. With this, it raises some questions about fairness in how different types of music are promoted.

In conclusion, while the main goal of this project demonstrates the potential for genre classification, it is important to recognize its limitations and ensure that it is developed and applied responsibly. It's important to recognize both the cultural diversity of the music and ethical concerns.

## References:

Data Pipeline Resource:
https://www.beautiful.ai/player/-NoO5oPOTi1lShJwrDCs/2-The-Pipeline-Defining-the-Problem-and-Understanding-Data

Random Forest Resource:
https://www.ibm.com/topics/random-forest

Gridsearch and Hyperparameter Tuning Resource:
https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74

Pros and Cons of Random Forest:
https://www.geeksforgeeks.org/what-are-the-advantages-and-disadvantages-of-random-forest/