Predicting Music Genre Based on Audio Features

Neha Ramana, Aarushi Buddhavarapu, and Oviya Muralidharan

Link: https://www.kaggle.com/datasets/vicsuperman/prediction-of-music-genre

Summary

1. Can we identify any patterns or trends in the relationship between music loudness and energy based on its genre?

Yes, there tends to be a positive correlation between the loudness and energy of a song and this is observed across all genres.

2. Are certain audio features more influential in determining specific music genres than others?

Yes, we found that there were six features that were most influential in predicting music genres, namely, acousticness, danceability, energy, instrumentalness, loudness, and speechiness.

3. How accurately can machine learning algorithms classify music genres based on audio features such as acousticness, danceability, energy, instrumentalness, loudness, tempo, and valence?

Supervised models, like Decision Tree Classifiers and Random Forest Classifiers, tend to perform better in terms of accuracy of genre classification within our dataset.

Motivation

The ability to accurately predict music genres based on audio features has practical applications in recommended systems, music streaming platforms, and marketing strategies. Furthermore, understanding the relationship between music characteristics such as energy, danceability, and acousticness, and genre classifications can provide insights into cultural trends and audience preferences.

Data setting

The dataset contains audio features for a collection of music tracks, including attributes such as popularity, acousticness, danceability, energy, instrumentalness, loudness, tempo, and valence. The dataset provides an opportunity to explore the relationship between these features and music genre classification.

The datasheet doesn't provide much information about the data given - the source of information is listed as "audio features provided by Spotify", however, collection methods and the timeframe aren't provided. The fact that this data's origins are not well documented may imply that the data that is gathered may be unfactual or biased to some extent. Any predictions made from models trained on this data set have the potential to not be applicable to other music data sets with similar attributes. With further exploration, few columns are labeled with units or ranges of values they fall within, making it difficult to understand the meaning of each value and the analysis we do with it. For example, there is no available definition of instrumentalness and loudness and they aren't associated with a unit in this dataset. Additionally, the context of the obtained_date column isn't explained - if the values were the date that the data was obtained by the data creator, or if it is something else. We will have to do additional research to see what these terms mean in a music analysis context. The last point to bring up is that certain attributes have values that aren't logical within this context - they don't make sense. For example some values in the duration_ms column are negative and the date_obtained values are all the 3,4,or 5 of April. Overall, the data does provide some interesting inconsistencies that can be resolved through further research into music analysis and the sourcing of the data itself.

Method

Data preprocessing

1) Download the csv file
2) Duplicate the file into another tab
3) In the second tab remove the NaN, '?', and empty values
4) Rename the columns on both tabs into easier to understand names
5) Upload the filtered data set into shared Jupyter Notebook

For the 1st question

- We would plot the energy vs. loudness on a line plot, for each genre using seaborne.
- Imported pearsonr function from scipy library (scipy.stats module) for statistical analysis using correlation coefficients to validate results.

For the 2nd question - we would use feature selection to determine this.

- Employ feature selection to choose a subset of features that increase the quality of our model.

For the 3rd question: Employ and compare 3 machine learning models to determine the best model at predicting music genres.

We split the dataset into 80% training and 20% testing dataset. For all three of the ML models, we used the same training and testing data splits.

Algorithm 1: DecisionTreeClassifier

Use a DecisionTreeClassifier model to classify the music genres based on the different features.

- Create the Decision Tree Classifier model with max_depth parameter set to 7.
- Use recursive feature elimination to choose a subset of audio features
  - We used n_features_to_select=6
- Fit the model to training data and visualize the decision tree
- Test the model on testing data
- Evaluate model using accuracy

Algorithm 2: Random Forest

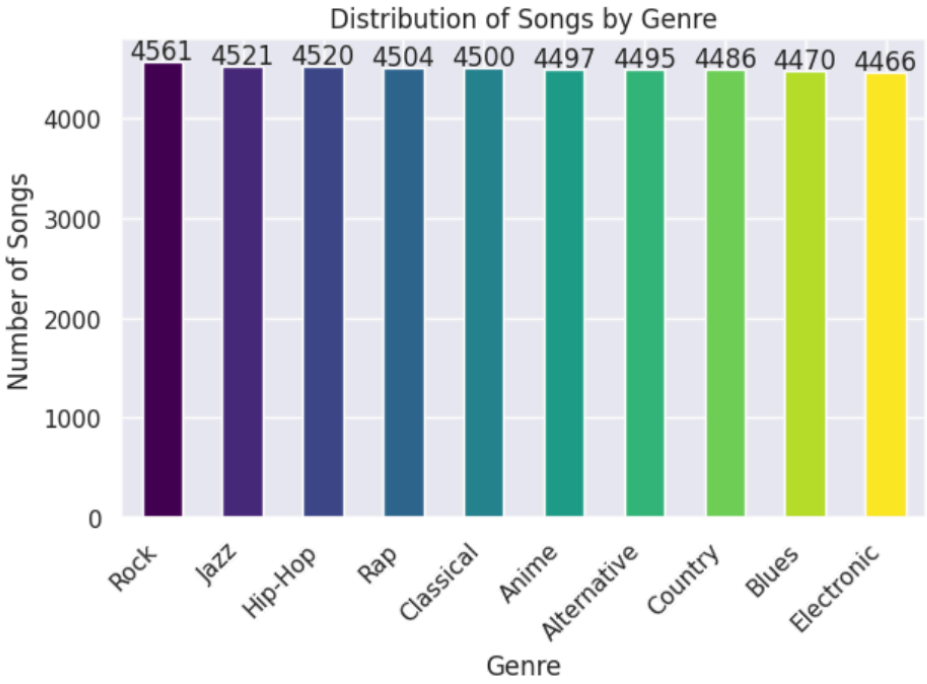
- Create the Random Forest Classifier model and use cross validation to optimize performance of the model by tuning hyperparameters such as max_depth, and n_estimators
- Fit the models to the training dataset and determine best model
- Evaluate model with testing dataset

Algorithm 3: K-Means Clustering Algorithm

- Initialization - create 10 clusters and fit the k-means model to the training data
- Assign the Genres with a for loop that iterates through each cluster
- Create an array of predicted genres for training data based on the assigned clusters
- Evaluate model with testing dataset

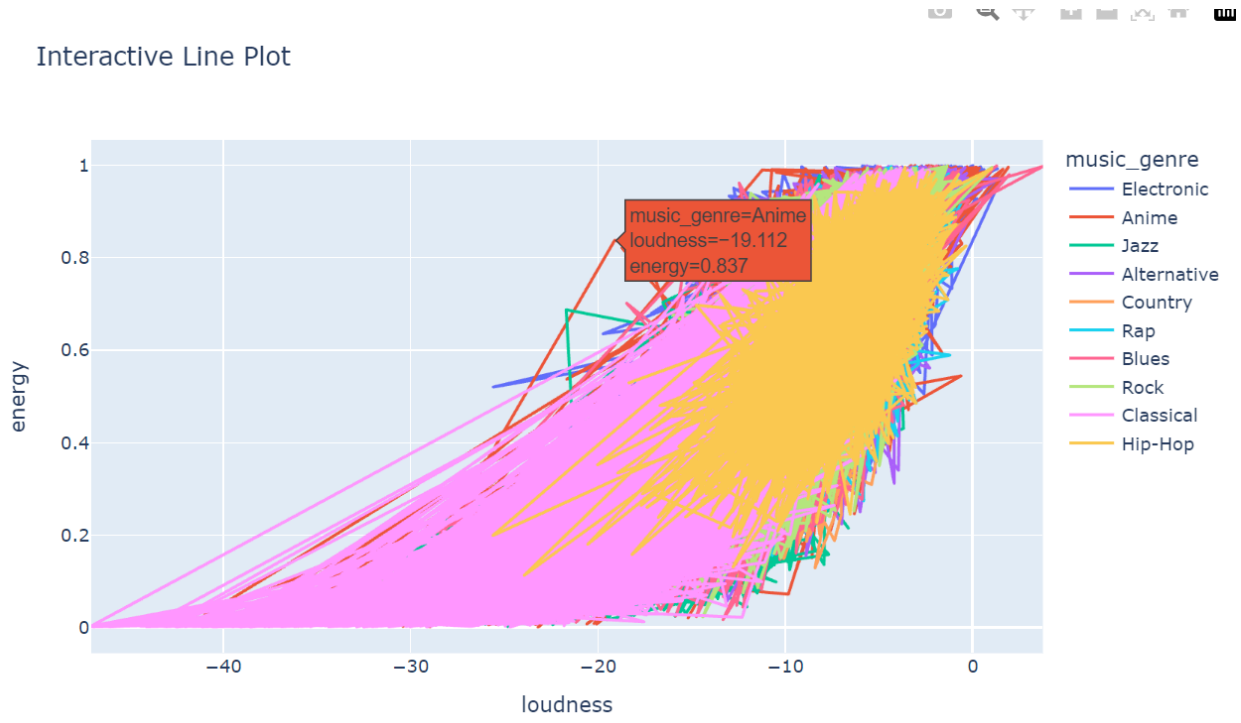
Results

We were first interested in seeing if the distribution of the 10 music genres in our dataset was evenly distributed. The graph below illustrates the distribution of songs by genre within the dataset. Each bar in the plot represents a specific music genre, with the height of the bar corresponding to the number of songs attributed to that genre. To enhance visual clarity and distinguish between different genres, a color map is applied, assigning a unique color to each genre bar. Additionally, each bar is annotated with the count of songs it represents, providing a clear indication of the quantity of songs within each genre category.



Research Question 1:

Our results suggest that there is a positive correlation between music loudness and energy across various genres. Regardless of the genre, as the loudness of the music increases, so does its energy level. This finding could potentially inform music production and consumption practices, highlighting the importance of loudness as a factor contributing to the perceived energy of a musical piece, regardless of its genre classification. To enhance the visualization of our data, we utilized an interactive line plot generated with Plotly. This dynamic visualization tool offers improved usability and flexibility, allowing users to zoom in and out to specific areas of interest, highlight particular data points, and isolate plot segments for individual genres. Additionally, the interactive features enable us to overlay multiple plots, facilitating comparisons between different genres.



To validate our results, we used statistical analysis by determining the correlation coefficients using the pearson function from the scipy library. The correlation between loudness and energy across various music genres is consistently strong and positive, indicating that as the loudness increases, there is a strong tendency for higher energy in the music across genres such as Anime, Classical, Jazz, Rock, Electronic, Alternative, Blues, Country, Rap, and Hip-Hop.

```
Pearson Correlation Coefficients for each Genre:
{'Electronic': 0.7588257497879368, 'Anime': 0.8682681816355308, 'Jazz':
0.7832900311057576, 'Alternative': 0.7653912810484661, 'Country':
0.7637553169207268, 'Rap': 0.6919763332023111, 'Blues': 0.740384971666163,
'Rock': 0.7560740010449117, 'Classical': 0.8065667147889829, 'Hip-Hop':
0.6696522717603414}
```

Research Question 2:

This section was focused on determining whether certain audio features were more important in predicting music genres and we found that yes, there are certain features that matter more. Based on our results, we can see that 'acousticness', 'danceability', 'energy', 'instrumentalness', 'loudness', 'speechiness' were the most important in predicting music genres. We determined this by creating an RFE object with the Decision Tree classifier as the estimator and set it to initially select 3 features. RFE recursively removes less important features until the desired number of features is reached. After creating our model, we manually modified the n_features_to_select to

increase the number of features selected and observed the resulting accuracy score. We determined that 6 features was the optimal value by increasing the number of features until there was no significant change in the accuracy scores of the testing and training data. This indicates that the choice of 6 features strikes a balance between capturing relevant information from the training data and ensuring the model's ability to generalize to new, unseen data. We also printed the features selected by RFE, to visualize which features are considered important by the algorithm.

One surprising result was the lack of importance of tempo of song in predicting music genres. The absence of tempo among the selected features may be due to its high correlation with other features, its relative importance compared to other features, or its limited relevance to genre classification in the dataset. Additionally, the Decision Tree model's binary decision-making process may have struggled to capture the nuanced variations in tempo across genres effectively.

```
Selected Features: ['acousticness', 'danceability', 'energy', 'instrumentalness', 'loudness', 'speechiness']
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=7)
```

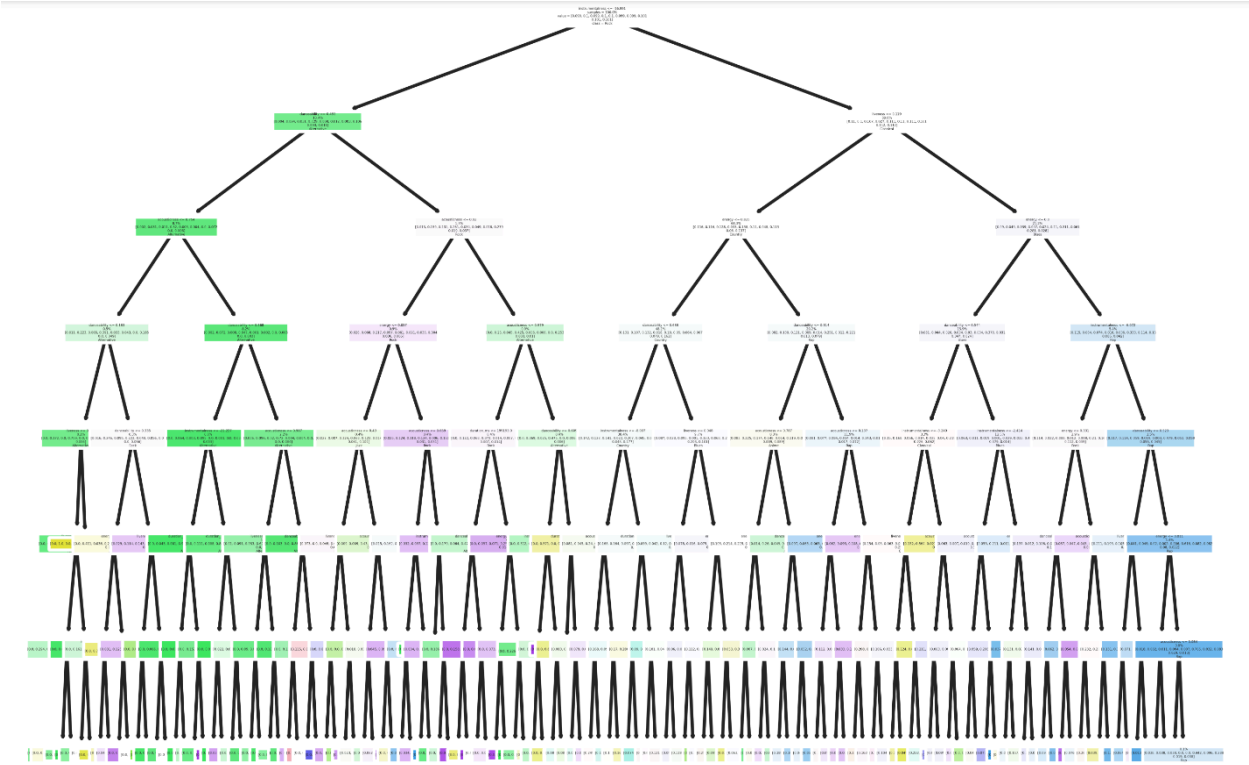
Research Question 3

1. Decision Tree Classifier

The first way that we predicted music genres based on certain audio features was using the Decision Tree Classifier algorithm. Based on the results, the model achieved an average accuracy score of 39% to 40% when evaluated with the testing data with different max_depth values. The max_depth parameter that we ended up using was a value of 7 which was determined by manually changing the values and observing the difference in accuracies of the training and test data. A max_depth of 7 was determined as optimal for our purposes since this is the value that gave a maximum accuracy score for the test data, while still rendering a training accuracy score close to this value. This ensures that the model was not overfitting to the training data, which would provide an accuracy score for training much greater than testing, or underfitting, which would just render very low accuracy scores for both. We then employed recursive feature elimination with the Decision Tree classifier model, which served as the estimator to select 6 features, as mentioned in the previous section.

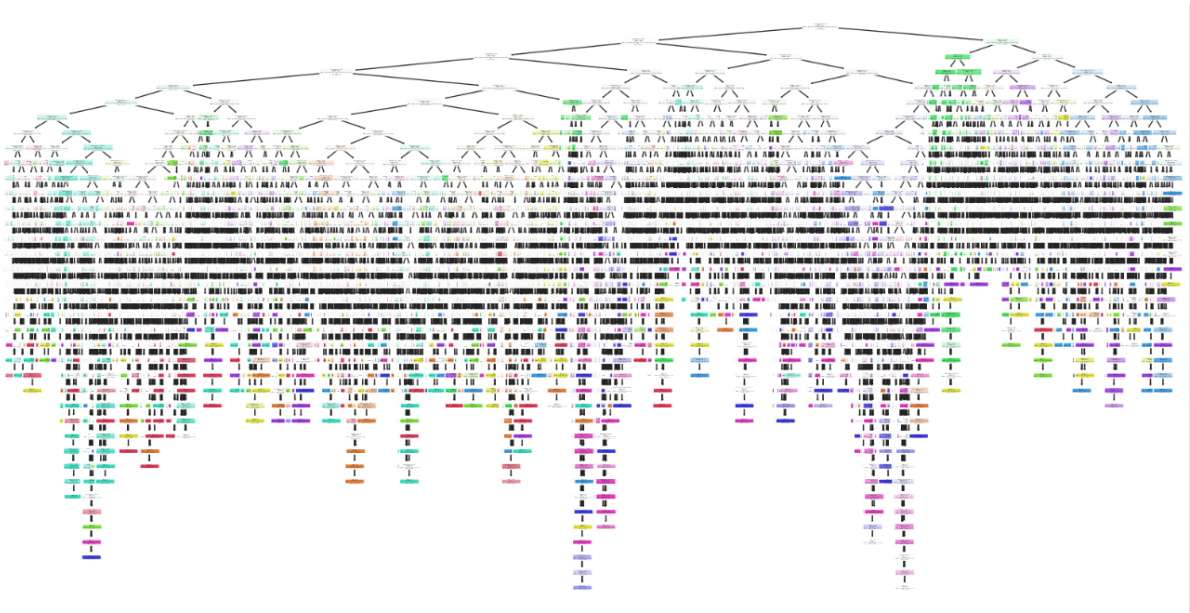
We also displayed a visualization of The Decision Tree which shows the decision-making process of the model for predicting music genres based on audio features, employing the model with a max_depth parameter set to 7. The tree structure is made of nodes representing decision points on specific features, with internal nodes signifying subsequent decision points and leaf nodes indicating final predicted genres. Features closer to the root are deemed more influential. The paths from the root to leaf nodes illustrate the decision logic for different feature combinations. A max_depth of 7

strikes a balance between model complexity and generalization. The visualization offers transparency in understanding feature importance and decision paths, providing an interpretable framework for predicting music genres.



2. Random Forest Classifier:

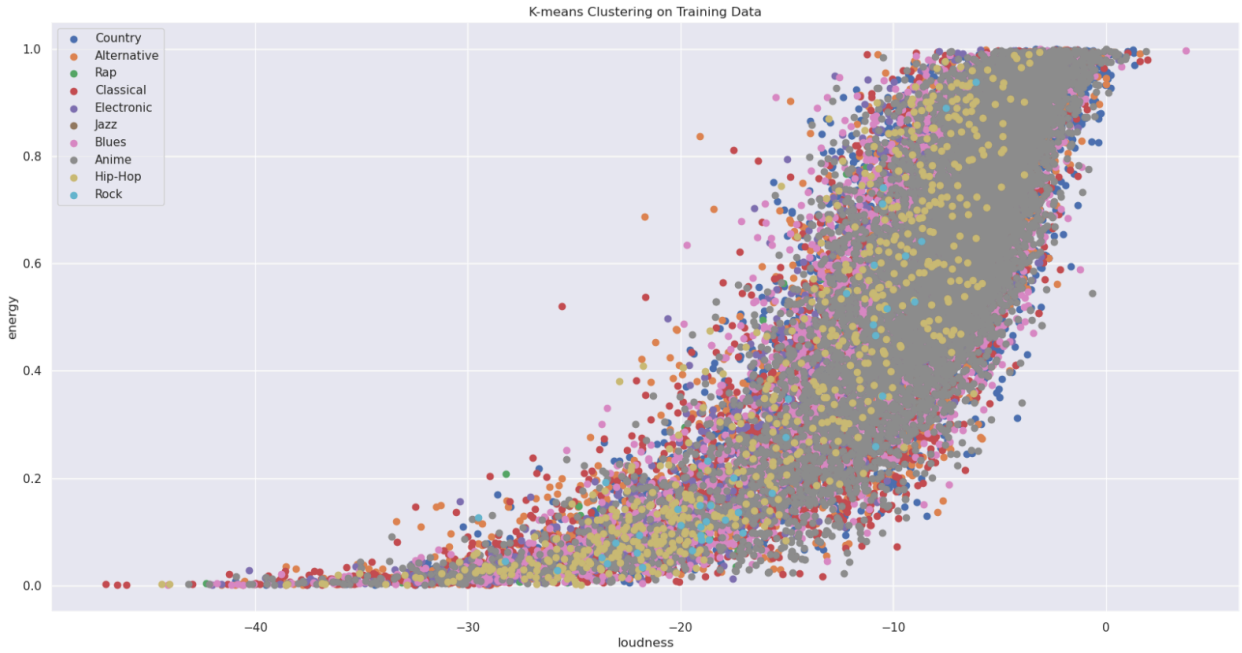
We employed a Random Forest Classifier to predict music genres based on a set of acoustic features. The Random Forest Classifier achieved an average accuracy of approximately 42% to 43% across multiple runs. This indicates that the model's ability to predict music genres based on the given features is fair. To gain insight into the decision-making process of the Random Forest Classifier, we visualized one of the decision trees in the forest. The decision tree provides a graphical representation of the rules learned by the model to classify music genres. Further investigation could explore strategies to improve the classification accuracy of the Random Forest Classifier. This may involve refining feature selection, tuning model hyperparameters, or experimenting with alternative machine learning algorithms. Additionally, incorporating additional data sources could enhance the model's ability to capture the nuances of music genres. In conclusion, our Random Forest Classifier analysis provides valuable insights into the predictive capabilities of machine learning algorithms in the context of music genre classification. While the model exhibits some predictive ability, its performance underscores the importance of understanding both the strengths and limitations of machine learning models in real-world applications.



3. K-Means Clustering:

We applied K-means clustering, an unsupervised machine learning technique, to categorize music genres based on their acoustic features. The model clustered the data into groups, with each cluster representing a distinct genre category. We associated the following genres with each cluster based on the model's clustering results - Cluster 0: Country, Cluster 1: Alternative, Cluster 2: Rap, Cluster 3: Classical, Cluster 4: Electronic, Cluster 5: Jazz, Cluster 6: Blues, Cluster 7: Anime, Cluster 8: Hip-Hop, and Cluster 9: Rock. We evaluated the model's performance by computing the accuracy of genre predictions on the training data, considering all audio features. The accuracy score averaged approximately 11% to 12%, indicating that our model's ability to classify music genres based on the provided features is limited. Although our visualization only depicts the relationship between two acoustic features (loudness and energy), the accuracy assessment takes into account all audio features used in the model. Future research could explore alternative clustering algorithms or incorporate additional features to improve classification accuracy and better capture the diverse characteristics of music genres. Overall, our K-means clustering analysis provides valuable insights into the acoustic profiles of different music genres. However, further refinement and exploration are necessary to enhance the accuracy and robustness of genre classification models.

To note, in the previous iteration, as seen in our recording, this figure did not have a legend making the data more difficult to interpret.



Impact and Limitations

The lack of documentation surrounding this data's origins suggests potential inaccuracies or biases in the gathered data. Consequently, predictions derived from models trained on this dataset may not be transferable to other music datasets with similar attributes. When assessing the validity and performance of datasets and machine learning models, it's crucial to employ a variety of metrics beyond accuracy alone. Metrics such as precision, recall, F1 score, confusion matrix, ROC curves, AUC-ROC scores, and regression metrics like MAE, MSE, and R-squared provide a comprehensive evaluation of model performance. Additionally, cross-validation techniques ensure robust estimates of model performance by partitioning data into subsets for training and testing, enhancing generalization to unseen data and enabling informed decision-making for iterative improvements in analyses.

Challenge Goals

- New library
    - We believe that our project will meet this challenge goal because of the need to validate some of our results using statistical testing using the pearson function from scipy library.
    - We also made our line plot interactive using the plotly library to allow for better visualization of our data, especially when comparing different genres.
- Machine learning

- Because this data set has many features that are related to music, having many types of ML models would provide insight as to which type of model predicts the musical genre the best. Understanding which type of learning (supervised vs unsupervised) is also beneficial within this circumstance, to see if human selected features perform better than clustering algorithms.

Plan Evaluation

The evaluation of our proposed work plan indicates a generally accurate estimation of the tasks involved in completing our machine learning models. While we ultimately achieved our objectives, there were notable deviations from the original plan. Notably, we did not implement cross-validation for hyperparameter tuning in both decision trees. It's important to note that although we did explore the make_blobs() function for k-means clustering as initially planned, we found that it was not suitable for our dataset. The function's random selection of cluster centers did not align with the structure of our data, leading us to pursue alternative approaches. Despite these deviations, we successfully completed our ML models. Another thing to note is that we changed our challenge goal of “Result Validity” to “New Library”. The reason for this was to ensure that we were meeting the challenge goal requirement as part of the project and due to challenges with validating the results of our machine learning models. Moving forward, we can use this experience to refine our planning process, ensuring better alignment between proposed work plans and actual implementation, while also remaining adaptable to unforeseen circumstances.

Testing

To validate the dataset splitting functionality, we employed a structured testing strategy. Initially, we split the dataset into training and testing sets, allocating 80% of the data for training and 20% for testing. This split was chosen to ensure a balance between model training on a substantial portion of the data and evaluating its performance on unseen samples. Subsequently, we trained machine learning models using the training data and assessed their performance using the testing data. The evaluation focused on various performance metrics such as accuracy, precision, and recall, which provided insights into the model's ability to generalize to new data. We included assertions within the code to verify if all the data was being considered in our dataset such as mean, min, max, as well as verifying that our code did not consider cells with NaN using .isnull() in the assert statements. While smaller data files were not utilized for testing, the comprehensive evaluation of the model's performance on the entire dataset split validated its effectiveness in accurately partitioning the data and assessing model performance. Overall, this systematic testing approach, combined with the use of assertions, ensured the correctness and reliability of the dataset splitting functionality, providing confidence in the validity of the reported results.

Collaboration

We consulted our classmates during code review sections to get advice on how to approach certain problems with our code.

For learning more about the ML models through ChatGPT, we used prompts such as:

- “What types of unsupervised models could we use for music genre predictability?”
- “Explain K-Means clustering”
- “What libraries should we use for a visualization of a random forest?”

To answer some of the sections in this report, we used prompts such as:

- “What other metrics apart from accuracy can be used to assess validity and performance?”
- “This is the section I have written. How can I make this better?”

Additionally, we all used documentation resources to figure out the basic structure of our ML models as well as what libraries should be used in our code.

In [1]:
```
!pip install seaborn
!pip install plotly
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.feature_selection import RFE
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, mean_squared_error
from sklearn.tree import plot_tree
from scipy.stats import pearsonr
from sklearn.dummy import DummyClassifier
from scipy.stats import ttest_rel
import plotly.express as px
sns.set_theme()

music_data = pd.read_csv('music_genre.csv')
cleaned_music_data = music_data.replace('?', np.nan)
cleaned_music_data = cleaned_music_data.dropna()
print(cleaned_music_data)

assert len(cleaned_music_data) == 45020 # Number of rows in the dataset
assert music_data.isnull().sum().sum() == 90 # Number of missing values
```

```
Requirement already satisfied: seaborn in /opt/conda/lib/python3.10/site-packages (0.13.1)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /opt/conda/lib/python3.10/site-packages (from seaborn) (1.26.
3)
Requirement already satisfied: pandas>=1.2 in /opt/conda/lib/python3.10/site-packages (from seaborn) (2.1.4)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /opt/conda/lib/python3.10/site-packages (from seaborn) (3.
8.2)
Requirement already satisfied: contourpy>=1.0.1 in /opt/conda/lib/python3.10/site-packages (from matplotlib!=3.6.1,>=
3.4->seaborn) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.10/site-packages (from matplotlib!=3.6.1,>=3.4-
>seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python3.10/site-packages (from matplotlib!=3.6.1,>
=3.4->seaborn) (4.47.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /opt/conda/lib/python3.10/site-packages (from matplotlib!=3.6.1,>
=3.4->seaborn) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.10/site-packages (from matplotlib!=3.6.1,>=
3.4->seaborn) (23.2)
Requirement already satisfied: pillow>=8 in /opt/conda/lib/python3.10/site-packages (from matplotlib!=3.6.1,>=3.4->se
aborn) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in /opt/conda/lib/python3.10/site-packages (from matplotlib!=3.6.1,>=
3.4->seaborn) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/python3.10/site-packages (from matplotlib!=3.6.
1,>=3.4->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.10/site-packages (from pandas>=1.2->seaborn) (2
023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in /opt/conda/lib/python3.10/site-packages (from pandas>=1.2->seaborn)
(2023.4)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.10/site-packages (from python-dateutil>=2.7->matplo
tlib!=3.6.1,>=3.4->seaborn) (1.16.0)
Requirement already satisfied: plotly in /opt/conda/lib/python3.10/site-packages (5.19.0)
Requirement already satisfied: tenacity>=6.2.0 in /opt/conda/lib/python3.10/site-packages (from plotly) (8.2.3)
Requirement already satisfied: packaging in /opt/conda/lib/python3.10/site-packages (from plotly) (23.2)
       instance_id           artist_name                track_name  \
0          32894.0             Röyksopp      Röyksopp's Night Out
1          46652.0  Thievery Corporation          The Shining Path
2          30097.0       Dillon Francis                 Hurricane
3          62177.0             Dubloadz                     Nitro
4          24907.0           What So Not          Divide & Conquer
...            ...                   ...                       ...
50000      58878.0                BEXEY                  GO GETTA
50001      43557.0            Roy Woods       Drama (feat. Drake)
50002      39767.0               Berner  Lovin' Me (feat. Smiggz)
50003      57944.0            The-Dream         Shawty Is Da Shit
50004      63470.0    Naughty By Nature            Hip Hop Hooray

       popularity  acousticness  danceability  duration_ms  energy  \
0            27.0       0.00468         0.652         -1.0   0.941
1            31.0       0.01270         0.622     218293.0   0.890
2            28.0       0.00306         0.620     215613.0   0.755
3            34.0       0.02540         0.774     166875.0   0.700
4            32.0       0.00465         0.638     222369.0   0.587
...           ...           ...           ...          ...     ...
50000        59.0       0.03340         0.913         -1.0   0.574
50001        72.0       0.15700         0.709     251860.0   0.362
50002        51.0       0.00597         0.693     189483.0   0.763
50003        65.0       0.08310         0.782     262773.0   0.472
50004        67.0       0.10200         0.862     267267.0   0.642

       instrumentalness key  liveness  loudness   mode  speechiness  \
0               0.79200  A#     0.115    -5.201  Minor       0.0748
1               0.95000   D     0.124    -7.043  Minor       0.0300
2               0.01180  G#     0.534    -4.617  Major       0.0345
3               0.00253  C#     0.157    -4.498  Major       0.2390
4               0.90900  F#     0.157    -6.266  Major       0.0413
...                 ...  ..       ...       ...    ...          ...
50000           0.00000  C#     0.119    -7.022  Major       0.2980
50001           0.00000   B     0.109    -9.814  Major       0.0550
50002           0.00000   D     0.143    -5.443  Major       0.1460
50003           0.00000   G     0.106    -5.016  Minor       0.0441
50004           0.00000  F#     0.272   -13.652  Minor       0.1010

                    tempo obtained_date  valence music_genre
0                 100.889         4-Apr    0.759  Electronic
1      115.00200000000001         4-Apr    0.531  Electronic
2                 127.994         4-Apr    0.333  Electronic
3                 128.014         4-Apr    0.270  Electronic
4                 145.036         4-Apr    0.323  Electronic
...                   ...           ...      ...         ...
50000   98.02799999999999         4-Apr    0.330     Hip-Hop
50001  122.04299999999999         4-Apr    0.113     Hip-Hop
50002             131.079         4-Apr    0.395     Hip-Hop
50003   75.88600000000001         4-Apr    0.354     Hip-Hop
50004   99.20100000000001         4-Apr    0.765     Hip-Hop

[45020 rows x 18 columns]
```

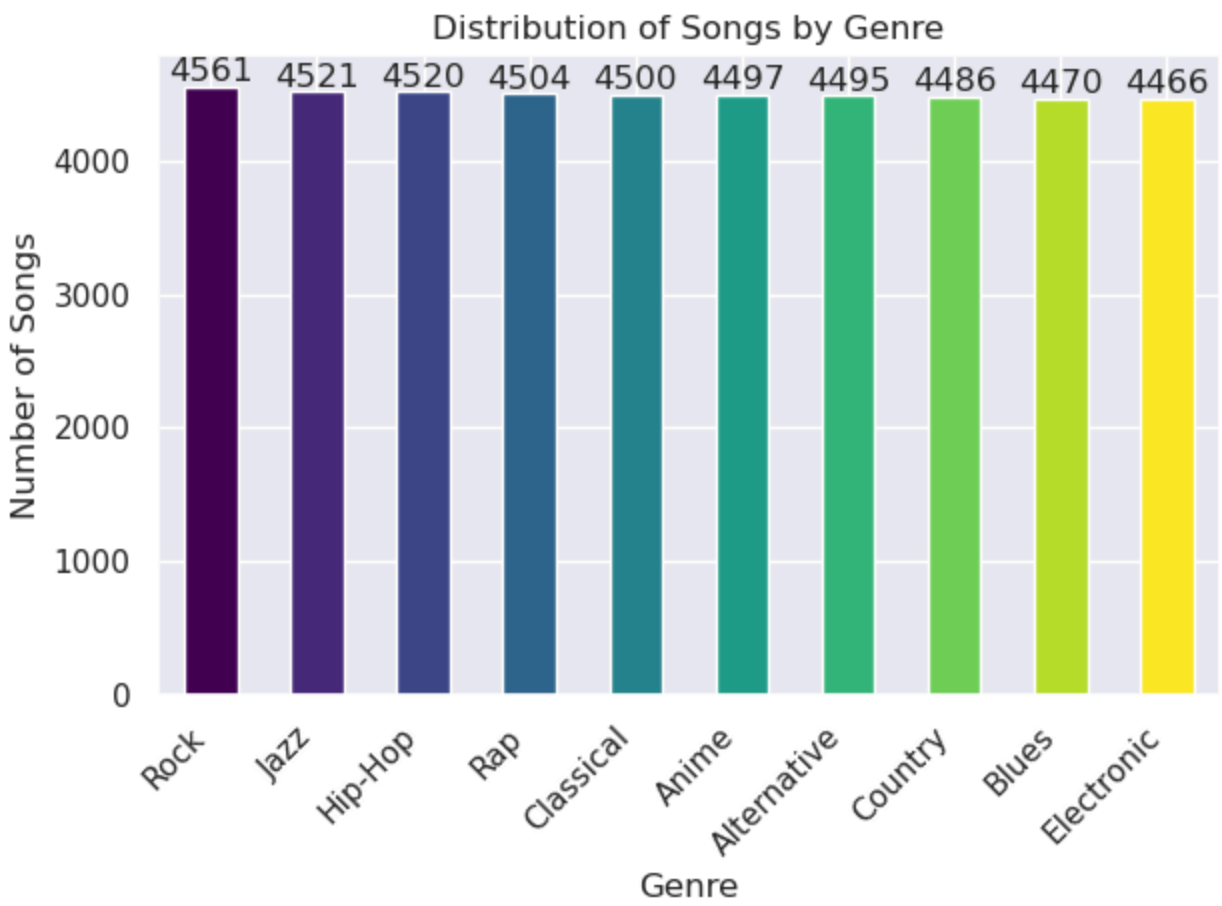
In [2]:
```
cleaned_music_data['tempo'] = pd.to_numeric(cleaned_music_data['tempo'], errors='coerce')
print('max', cleaned_music_data['tempo'].max())
print('min', cleaned_music_data['tempo'].min())
print('mean', cleaned_music_data['tempo'].mean())
```

```
assert cleaned_music_data['tempo'].max() == 220.276
assert cleaned_music_data['tempo'].min() == 34.347
expected_mean = 119.953
actual_mean = round(cleaned_music_data['tempo'].mean(), 3)
assert actual_mean == expected_mean
```

```
max 220.276
min 34.347
mean 119.95296059529099
```

In [3]:
```
numeric_features = ['acousticness', 'danceability', 'duration_ms',
                    'energy', 'instrumentalness', 'liveness', 'loudness', 'speechiness', 'tempo', 'valence']
X = cleaned_music_data[numeric_features]
y = cleaned_music_data['music_genre']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

In [4]:
```
# Distribution graph code
genre_counts = cleaned_music_data['music_genre'].value_counts()

plt.figure()
bars = genre_counts.plot(kind='bar', color=plt.cm.viridis(np.linspace(0, 1, len(genre_counts))))
plt.title('Distribution of Songs by Genre')
plt.xlabel('Genre')
plt.ylabel('Number of Songs')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability


# Annotate each bar with the count of songs
for bar in bars.patches:
    plt.annotate(format(bar.get_height(), '.0f'),
                 (bar.get_x() + bar.get_width() / 2, bar.get_height()),
                 ha='center', va='center',
                 xytext=(0, 5),
                 textcoords='offset points')

plt.tight_layout()
plt.show()
```



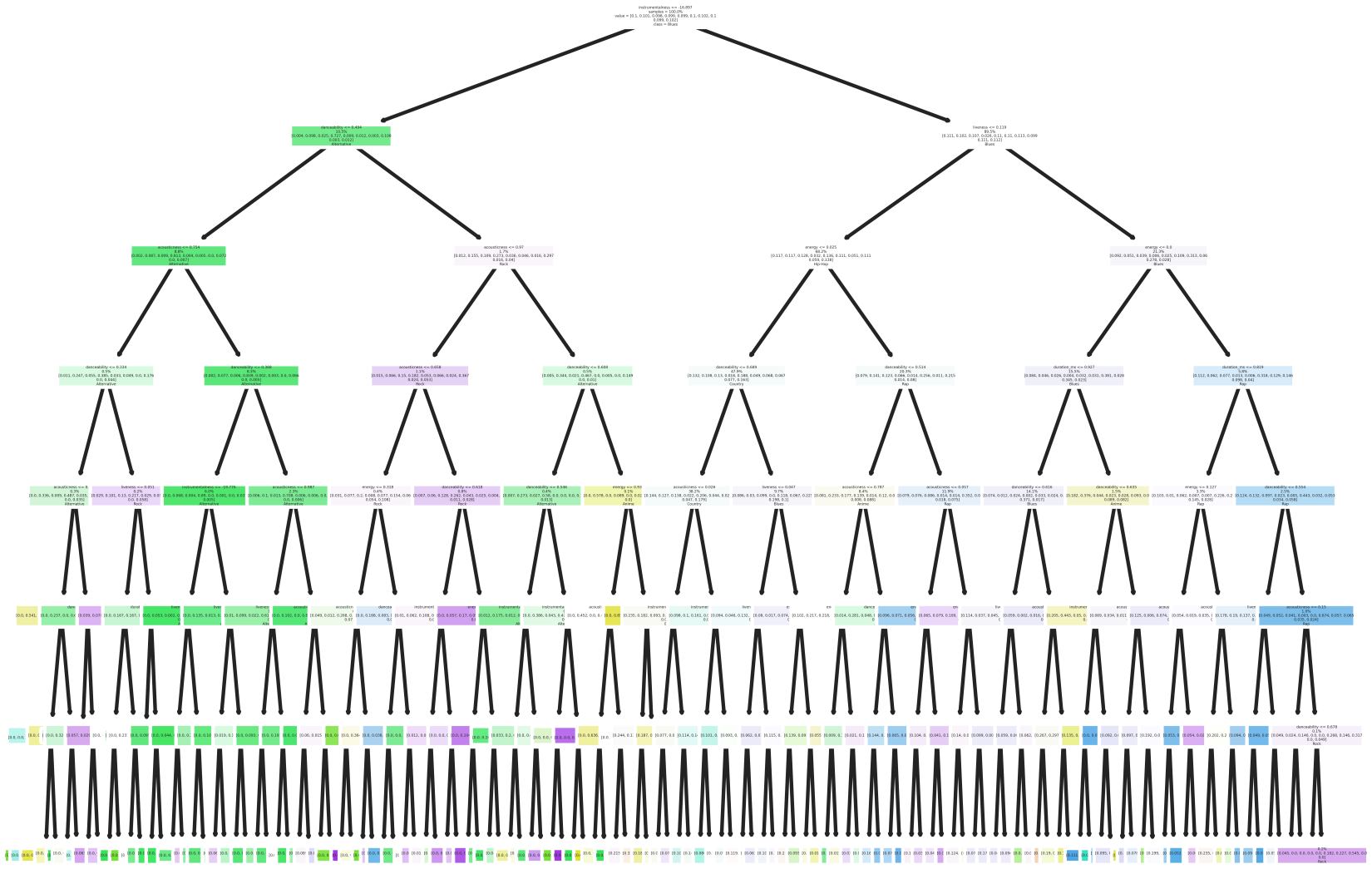
In [10]:
```
def pop_tempo_plot(cleaned_data):
    fig = px.line(cleaned_data, x='loudness', y='energy', color='music_genre',
                  title='Interactive Line Plot')
    fig.update_xaxes(title='loudness')
    fig.update_yaxes(title='energy')
    return fig.show()

pop_tempo_plot(cleaned_music_data)

genres = cleaned_music_data['music_genre'].unique()
correlation_coefficients = {}

for genre in genres:
    genre_data = cleaned_music_data[cleaned_music_data['music_genre'] == genre]
    pearson_corr, _ = pearsonr(genre_data['loudness'], genre_data['energy'])
    correlation_coefficients[genre] = pearson_corr

# Print the Pearson correlation coefficients for each genre
print("Pearson Correlation Coefficients for each Genre:")
print(correlation_coefficients)
```

```
Pearson Correlation Coefficients for each Genre:
{'Electronic': 0.7588257497879368, 'Anime': 0.8682681816355308, 'Jazz': 0.7832900311057576, 'Alternative': 0.7653912810484661, 'Country': 0.7637553169207268, 'Rap': 0.6919763332023111, 'Blues': 0.740384971666163, 'Rock': 0.7560740010449117, 'Classical': 0.8065667147889829, 'Hip-Hop': 0.6696522717603414}
```

In [9]:

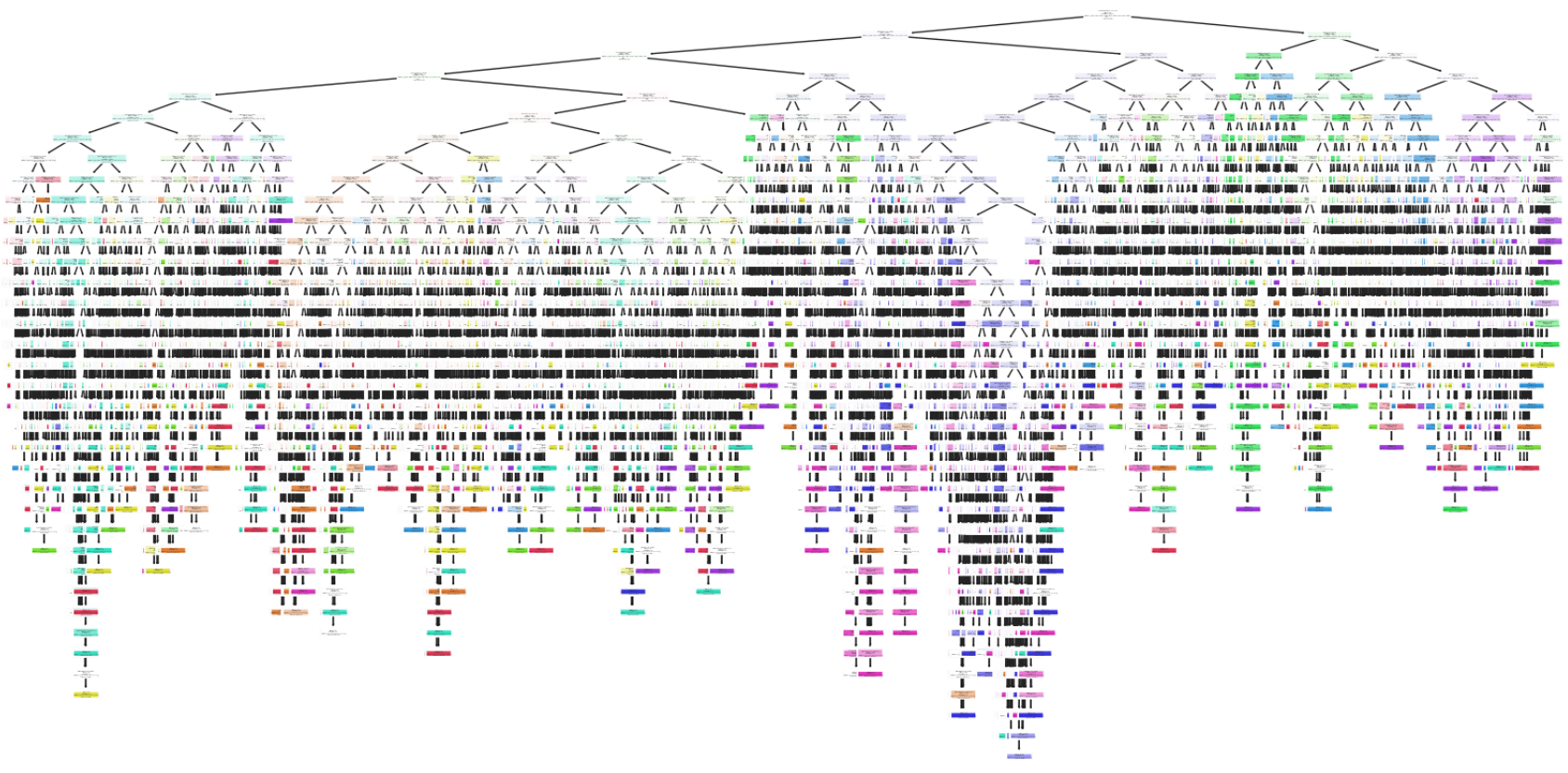
```python
# Model 1 - Decision Tree Classifer (with recursive feature elimination)

# Training the model on the training data and displaying selected features
clf = DecisionTreeClassifier(max_depth=7)

rfe = RFE(clf, n_features_to_select=6)  # Adjust the number of features to select
X_selected = rfe.fit_transform(X_train, y_train) # selected features

selected_feature_indices = rfe.support_
selected_features = [numeric_features[i] for i, selected in enumerate(selected_feature_indices) if selected]

print("Selected Features:", selected_features)

clf.fit(X_selected,y_train)

# Visualizing the Decision Tree
plt.figure(dpi=2000)
plot_tree(
    clf,
    feature_names=numeric_features,
    class_names=y.unique(),
    label="root",
    filled=True,
    impurity=False,
    proportion=True,
    rounded=True,
) and None # Hide return value of plot_tree

# Testing the model on the test data
train_predictions = clf.predict(X_selected)
predictions = clf.predict(rfe.transform(X_test))

# Evaluating the model
train_accuracy = accuracy_score(y_train, train_predictions)
accuracy = accuracy_score(y_test, predictions)
print("Training Accuracy:", train_accuracy)
print("Testing Accuracy:", accuracy)
```

```
Selected Features: ['acousticness', 'danceability', 'energy', 'instrumentalness', 'loudness', 'speechiness']
Training Accuracy: 0.3989060417592181
Testing Accuracy: 0.3802754331408263
```



In [7]:
```
# Model 2 - Random Forest
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

rf_classifier.fit(X_train, y_train)

predictions = rf_classifier.predict(X_test)

accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)

tree = rf_classifier.estimators_[0]  # Change the index to visualize a different tree

# Plotting the decision tree
plt.figure(figsize=(20,10))
plot_tree(tree, feature_names=numeric_features, class_names=y.unique(), filled=True, rounded=True)
plt.show()
```

```
Accuracy: 0.42358951577076853
```



In [15]:
```
# Model 3 - Unsupervised ML model using K-means clustering

kmeans = KMeans(n_clusters=10, random_state=42)
kmeans.fit(X_train)
cluster_labels = kmeans.labels_


clusters_mapping = {
    0: 'Country',
```

```
    1: 'Alternative',
    2: 'Rap',
    3: 'Classical',
    4: 'Electronic',
    5: 'Jazz',
    6: 'Blues',
    7: 'Anime',
    8: 'Hip-Hop',
    9: 'Rock',
}

for cluster in cluster_labels:
    if cluster not in clusters_mapping:
        print("Cluster label causing KeyError:", cluster)

y_pred_genres_train = np.array([clusters_mapping.get(np.where(kmeans.labels_ == cluster)[0][0], 'Unknown')
                                for cluster in cluster_labels])

# Calculate accuracy using y_train and y_pred_genres_train
accuracy_train = accuracy_score(y_train, y_pred_genres_train)
print("Accuracy on training data:", accuracy_train)

# Visualize clusters for training data
unique_genres = list(set(y_train))

# Visualize clusters for training data
plt.figure(figsize=(20, 10))
for i in range(10):  # Assuming you have 10 clusters
    plt.scatter(X_train['loudness'][cluster_labels == i], X_train['energy'][cluster_labels == i], label=clusters_mapp

plt.title('K-means Clustering on Training Data')
plt.xlabel('loudness')
plt.ylabel('energy')
plt.legend()
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
```

```
Accuracy on training data: 0.12136272767658818
```