MET CS677 Data Science with Python
Music Genre Classification

Alan Szmyt

April 27, 2023

For my final project, I explored music genre classification using different machine learning models by analyzing and comparing them against a small dataset of Spotify audio tracks.

For the dataset, I created a small dataset using Spotify's web API endpoints. I used their API to download track data for two genres: **Metal** and **Country**.

I gathered track data from two heavy metal playlists:

1. **https://open.spotify.com/playlist/37i9dQZF1DX9qNs32fujYe**
2. **https://open.spotify.com/playlist/37i9dQZF1DWWOaP4H0w5b0**

and two country playlists:

1. **https://open.spotify.com/playlist/37i9dQZF1DWZBCPUIUs2iR**
2. **https://open.spotify.com/playlist/37i9dQZF1DXadasIcsfbqh**

Dataset Description: The dataset consists of 200 country tracks and 250 heavy metal tracks.

There are 13 (continuous) features: $F = \{f_1, \ldots, f_{13}\}$ and a genre class label L (Metal: 0, Country: 1).

1. f_1: acousticness
2. f_2: danceability
3. f_3: energy
4. f_4: instrumentalness
5. f_5: key
6. f_6: liveness
7. f_7: loudness
8. f_8: speechiness
9. f_9: tempo
10. f_{10}: time_signature
11. f_{11}: valence
12. f_{12}: duration
13. f_{13}: popularity
14. L: genre (Metal: 0, Country: 1)

I created a python class *ModelAnalyzer* that uses the strategy design pattern to switch between machine learning models and datasets to perform training, testing, and gather analytics for different types of models. These analytics can then be compared to determine a suitable model for predicted music genre.

I chose to compare a Linear Support Vector Machine (SVM), a Random Forest classifier, logistic regression model, and k-Nearest Neighbors (kNN) classifier. I also used a Linear Regression model to plot the highest correlation features for each genre.

Cell 08

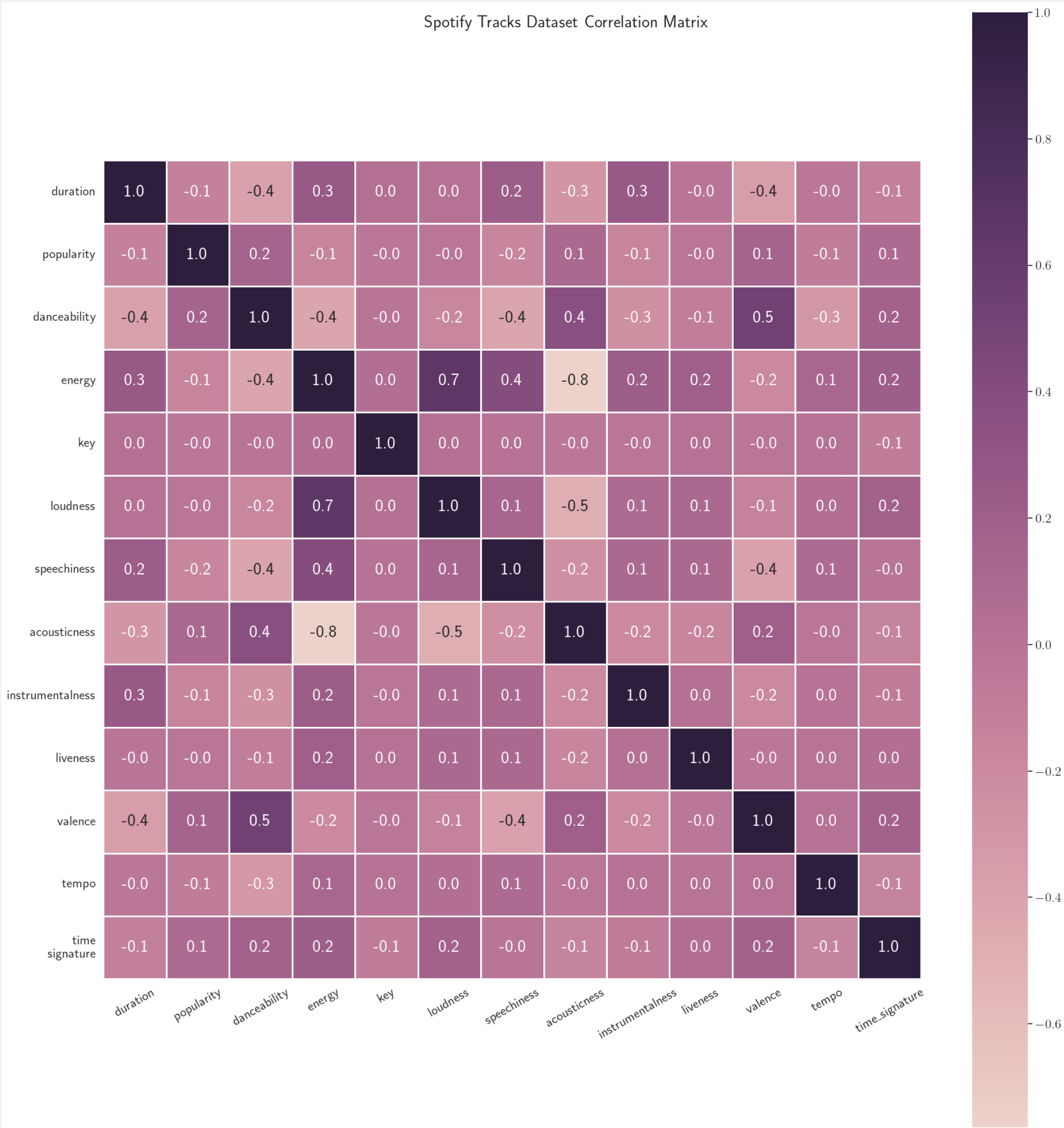
```python
# Machine learning model analyzer context class.
analyzer: ModelAnalyzer = ModelAnalyzer(
    dataset=tracks_dataset,
    model=LinearSVMModel(
        category_col=SpotifyColumns.GENRE,
        persistence=True,
        label_transformer=Genre.from_label,
    ),
    category_col=SpotifyColumns.GENRE,
)
```

Before diving into the model analysis, I first plotted the correlation matrix for the entire track dataset to get a high level overview of the correlations between all the track features.

I also created a radar (or spider) plot of the track features for each genre to see the distribution of values per genre. The radar plot shows that the **Metal** genre has a high energy and loudness and every other value is relatively low. For the **Country** genre, there is a higher danceability value and valence, but all the values are below 0.6, so there isn't as much of a weight on loudness and energy compared to the **Metal** tracks.

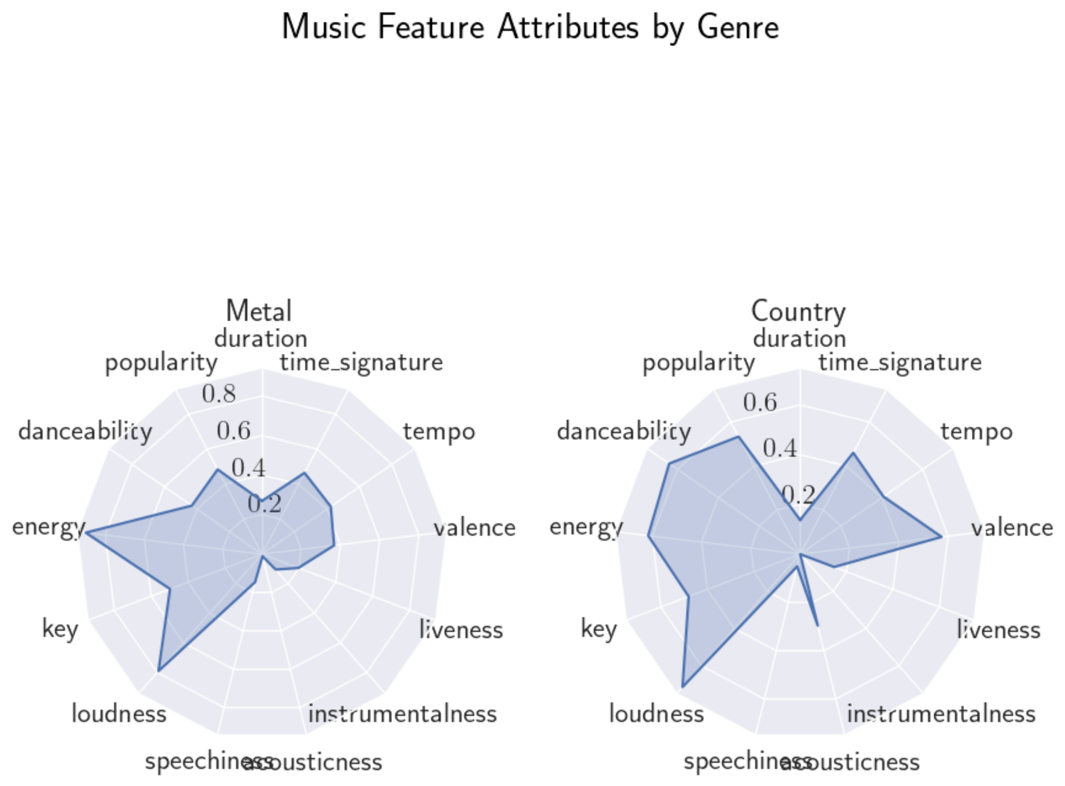
Cell 09

```python
# Plot a correlation matrix between the different music attributes.
correlation_matrix: DataFrame = analyzer.correlation_matrix()
```



Cell 10

```python
# Plot a radar plot of the music attributes per genre.
analyzer.radar_plot(
    title="Music Feature Attributes by Genre",
    label_transformer=Genre.from_label
)
```



I started off by training and testing a Linear SVM model and computing the accuracy and confusion matrix. The Linear SVM performed pretty well with a 93.78% accuracy.

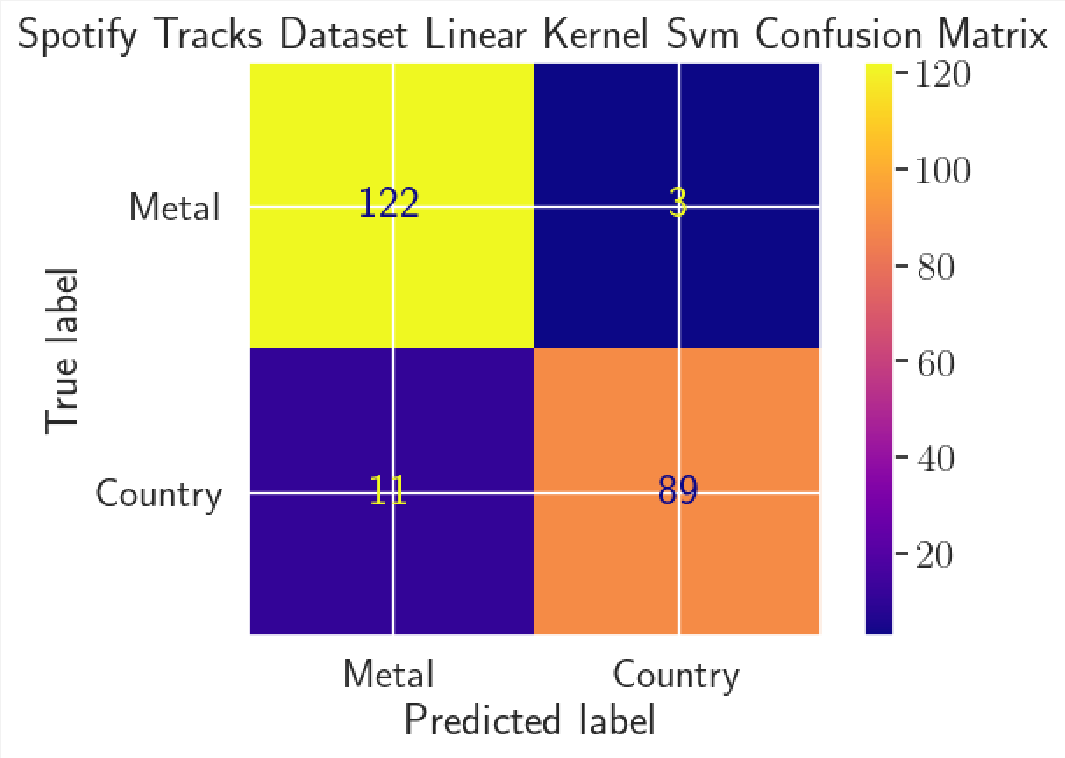
Cell 11

```python
# Train linear svm model, make predictions, and gather analytics.
linear_svm_analytics: ClassifierAnalytics = analyzer.analyze()

# Compute the accuracy.
print(
    f"Linear kernel SVM accuracy: "
    f"{linear_svm_analytics.confusion_matrix.accuracy.score}"
)
```

```
Linear kernel SVM accuracy: 93.78%
```

Cell 12

```python
# Compute the confusion matrix.
linear_svm_analytics.show_confusion_matrix()
```



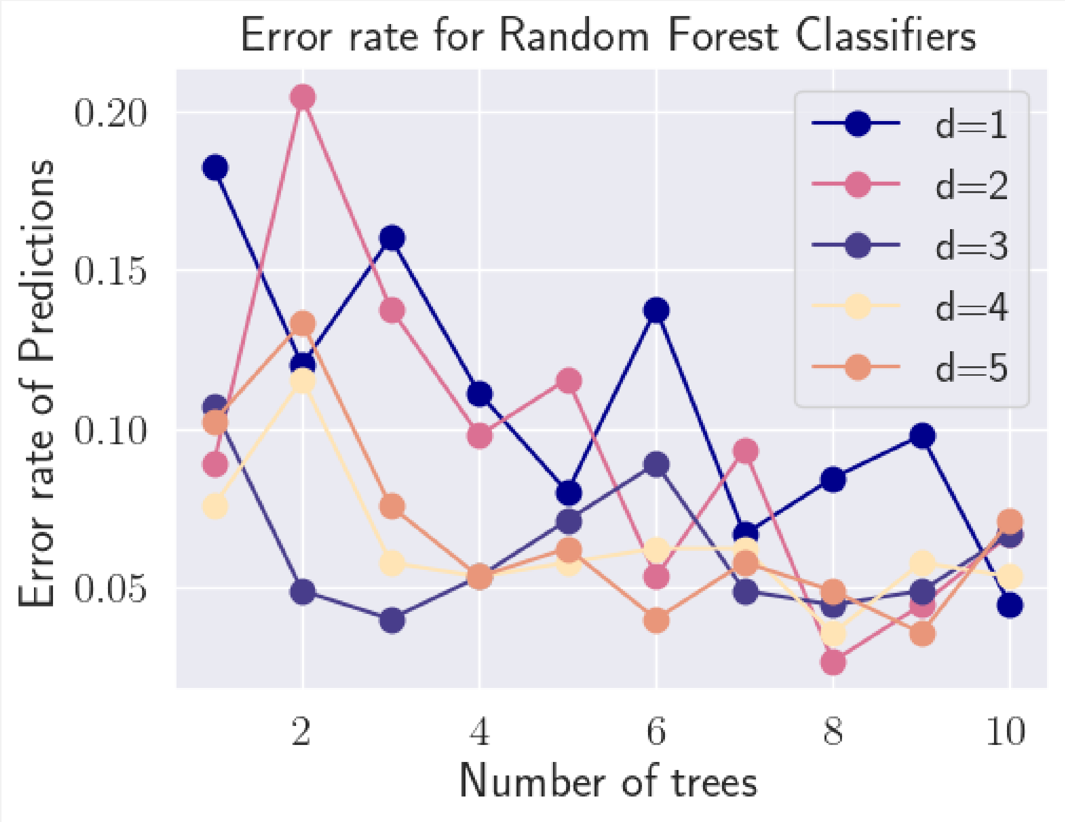
I then ran training and testing on multiple Random Forest classifiers to determine the best resulting hyperparameter values for *trees* and *max_depth*.

Cell 13

```python
# Train random forest model, make predictions, and gather analytics.
random_forest_analytics: RandomForestAnalyticsCollection = (
    RandomForestAnalyticsCollection()
)

random_forests: list[RandomForestModel] = [
    RandomForestModel(
        category_col=SpotifyColumns.GENRE,
        persistence=True,
        label_transformer=Genre.from_label,
        trees=n,
        max_depth=d,
    )
    for n in range(1, 11)
    for d in range(1, 6)
]
for random_forest in random_forests:
    analyzer.model = random_forest
    random_forest_analytics.append(analyzer.analyze())
```

Cell 14

```
# Error plot for all random forest classifiers.
random_forest_analytics.show_error_plot()
```



The random forest classifier with the best performance had high accuracy of 97.33%.

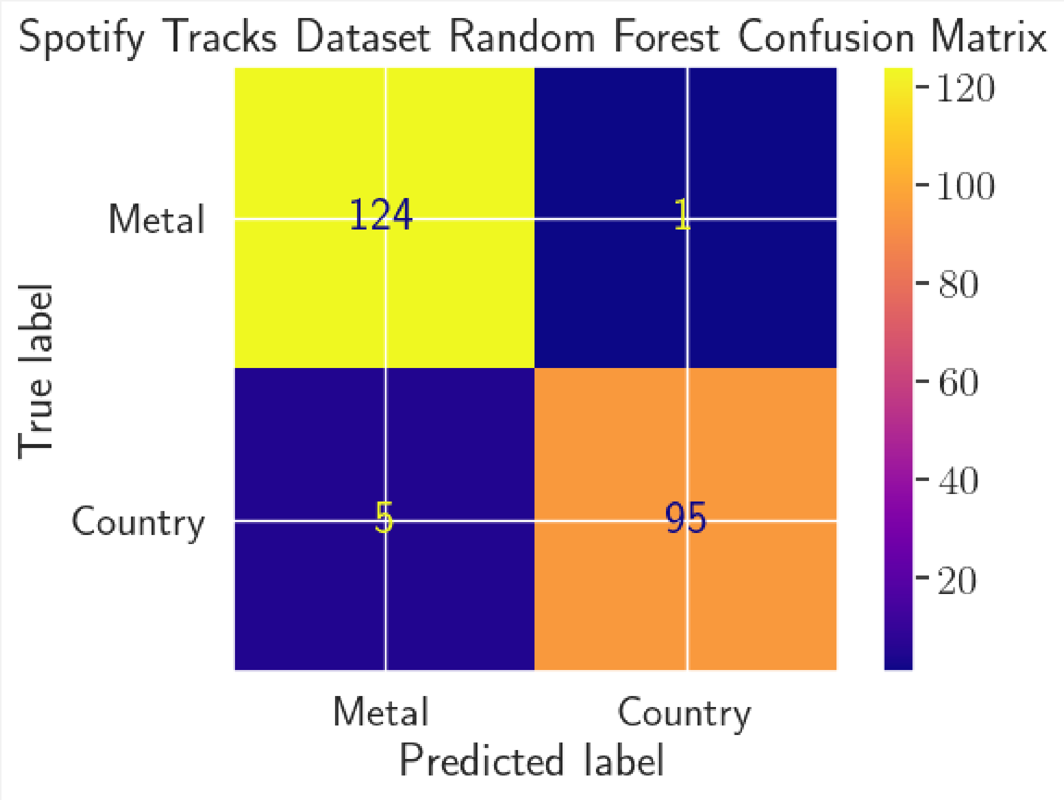
Cell 15

```
# Get the random forest with the lowest error rate.
best_random_forest: RandomForestAnalytics = random_forest_analytics.lowest_error
print(
    f"Best combination of N and d is: "
    f"N={best_random_forest.trees}, "
    f"d={best_random_forest.max_depth}"
)

print(
    f"The accuracy for best random forest is: "
    f"{best_random_forest.confusion_matrix.accuracy.score}"
)
```

```
Best combination of N and d is: N=8, d=2
The accuracy for best random forest is: 97.33%
```

Cell 16

```python
# Compute the confusion matrix.
best_random_forest.show_confusion_matrix()
```



Cell 17

```python
# Train logistic regression model, make predictions, and gather analytics.
analyzer.model = LogisticRegressionModel(
    category_col=SpotifyColumns.GENRE,
    persistence=True,
    label_transformer=Genre.from_label,
)
logistic_regression_analytics: ClassifierAnalytics = analyzer.analyze()

# Compute the accuracy.
print(
    f"Logistic regression classifier accuracy: "
    f"{logistic_regression_analytics.confusion_matrix.accuracy.score}"
)
```

```
Logistic regression classifier accuracy: 70.67%
```

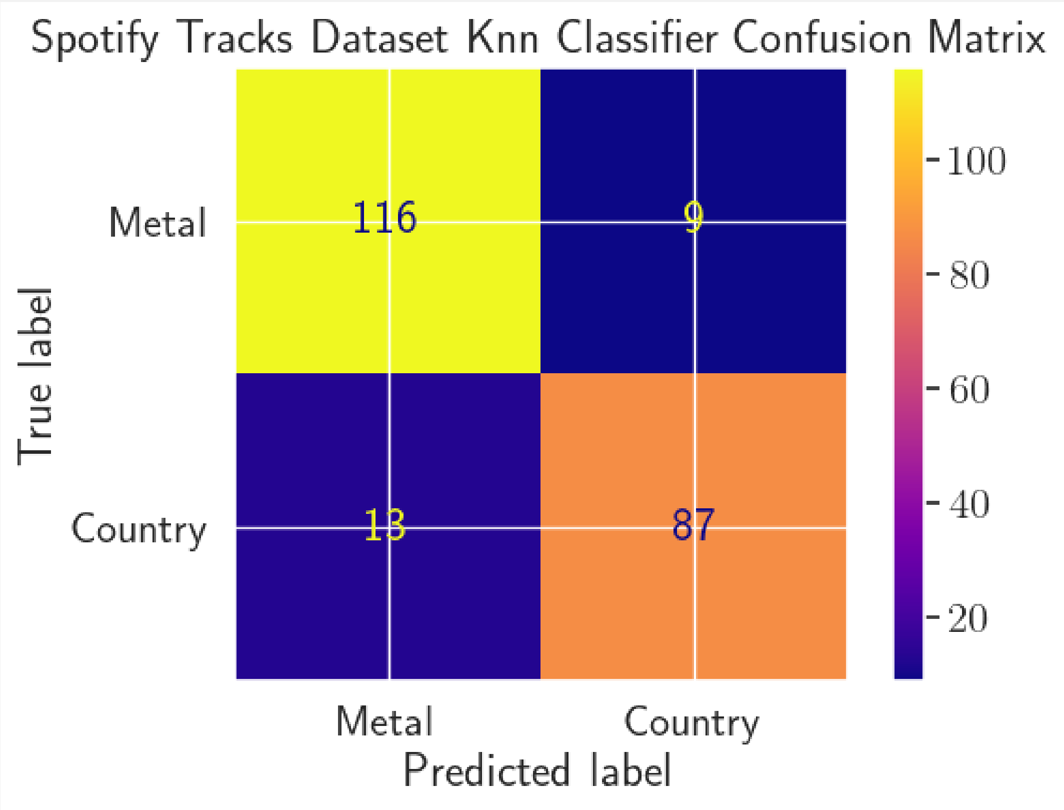
CELL 18

```python
# Train kNN model, make predictions, and gather analytics.
analyzer.model = KNNModel(
    category_col=SpotifyColumns.GENRE,
    persistence=True,
    label_transformer=Genre.from_label,
)
knn_analytics: ClassifierAnalytics = analyzer.analyze()

# Compute the accuracy.
print(
    f"kNN classifier accuracy: "
    f"{knn_analytics.confusion_matrix.accuracy.score}"
)
```

```
kNN classifier accuracy: 90.22%
```

CELL 19

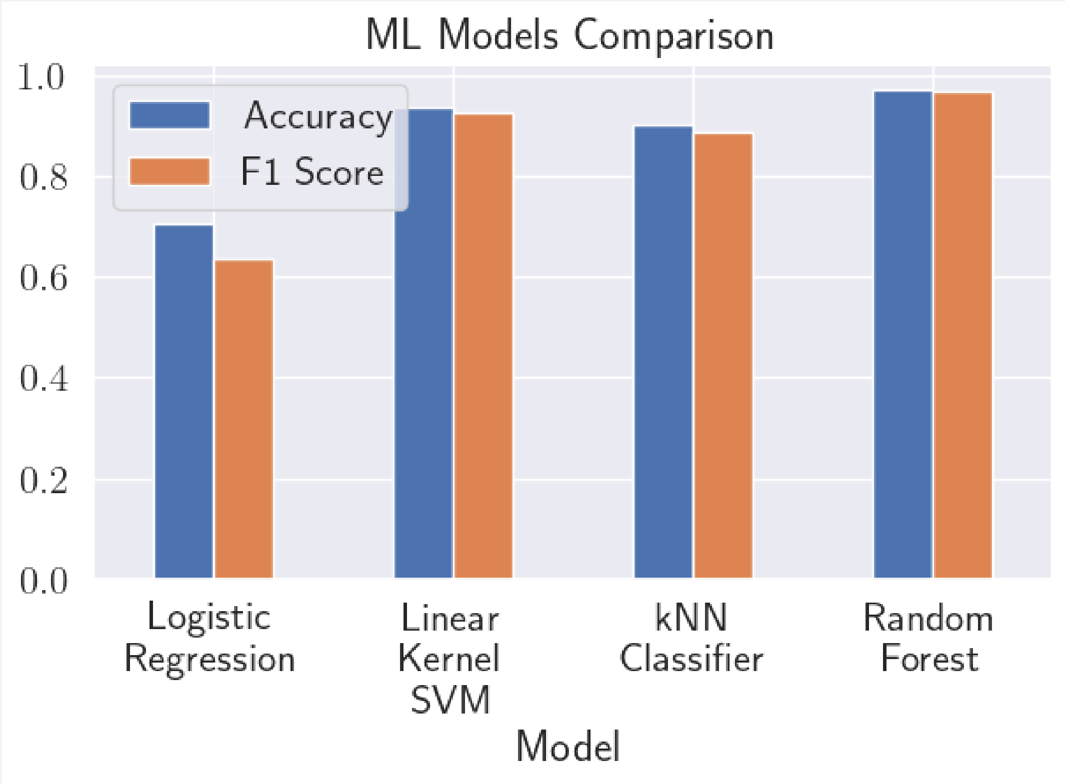
```python
# Compute the confusion matrix.
knn_analytics.show_confusion_matrix()
```



Comparing the four machine learning models resulted with the random forest classifier being the best performer for both *accuracy* and *f1 score*.

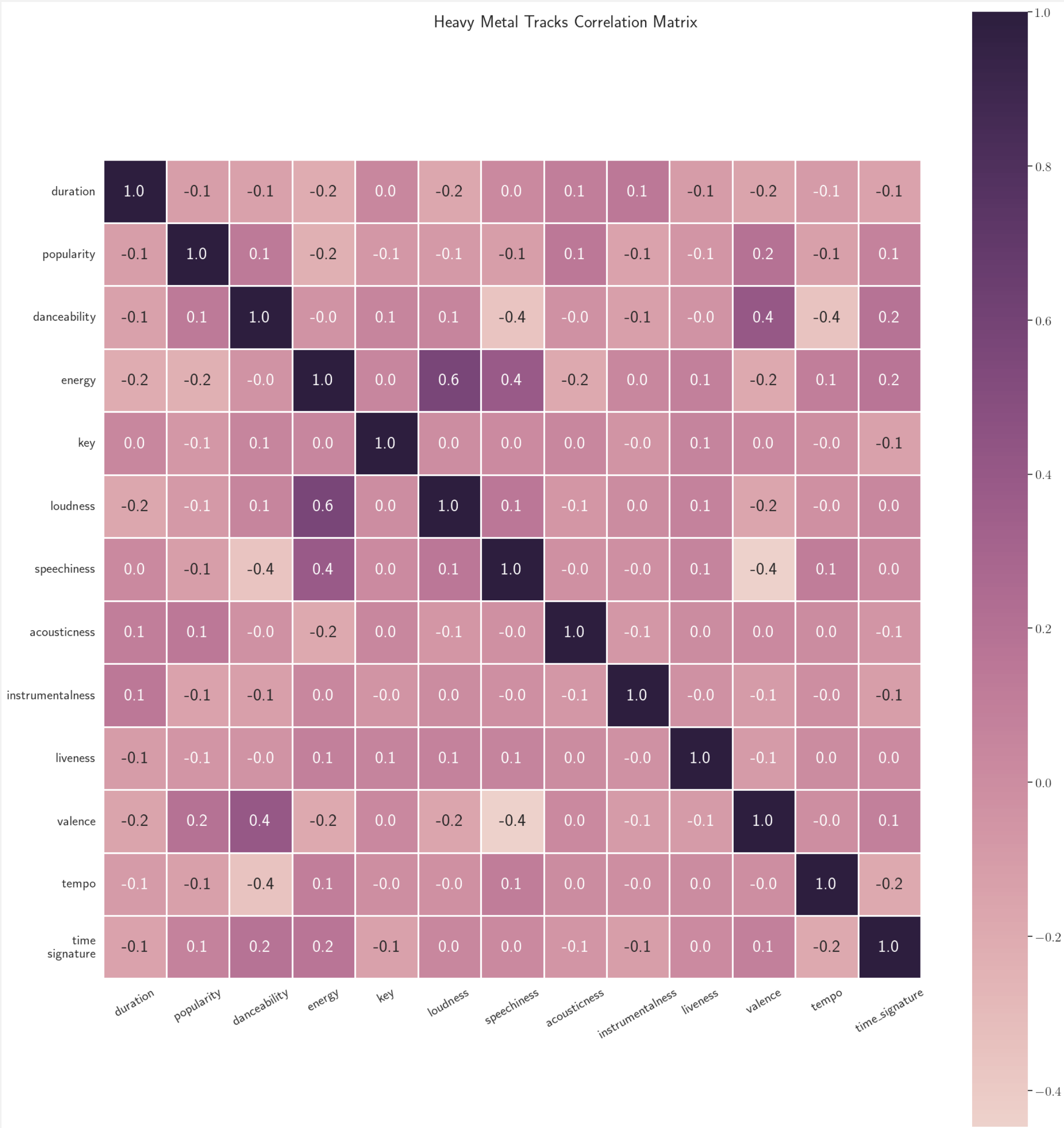
Cell 20

```python
# Gather analytics for all classifier models.
classifier_analytics: AnalyticsCollection = AnalyticsCollection(
    [
        logistic_regression_analytics,
        linear_svm_analytics,
        knn_analytics,
        best_random_forest,
    ]
)

classifier_analytics.plot()
```



Cell 21

```python
# Plot a correlation matrix for heavy metal tracks.
metal_correlation_matrix = ModelAnalyzer.create_correlation_matrix(
    dataset=heavy_metal_tracks.drop(SpotifyColumns.GENRE, axis=1),
    title="Heavy Metal Tracks Correlation Matrix"
)
```



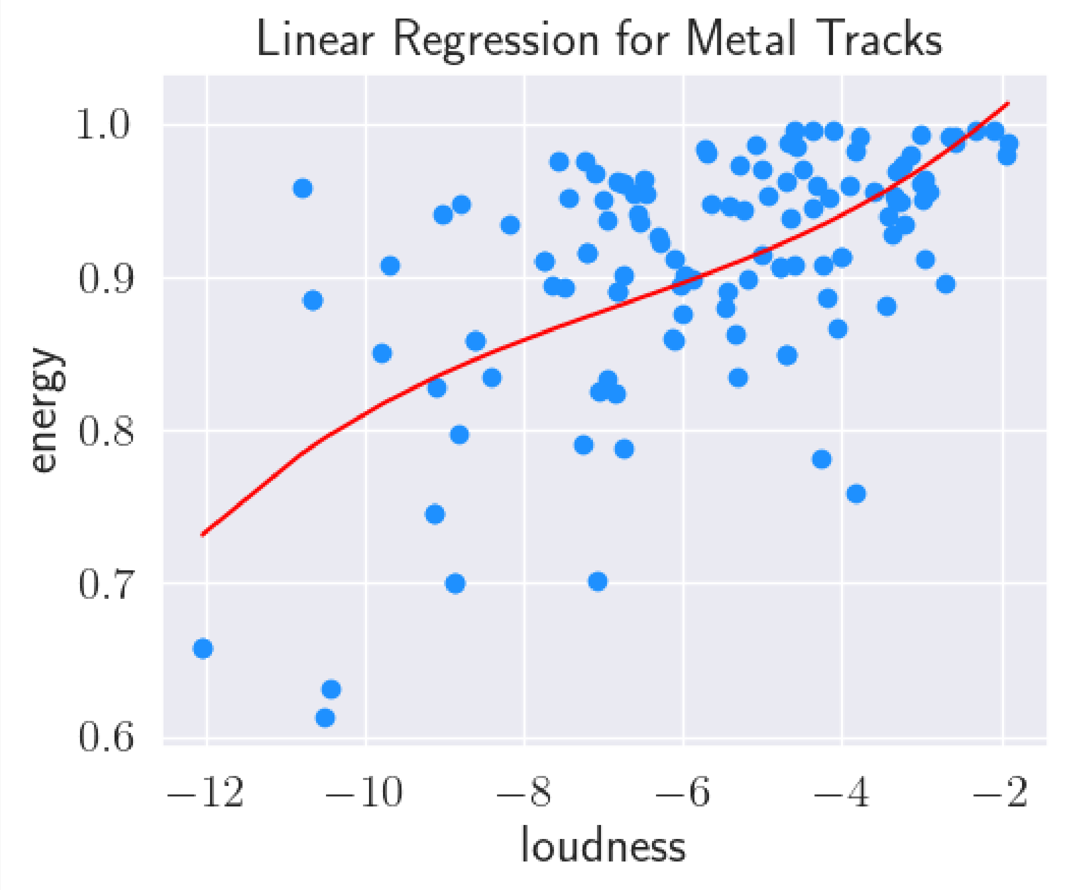
Cell 22

```python
# Examine the heavy metal tracks' correlation matrix.
metal_correlations = examine_correlation_matrix(metal_correlation_matrix)

# Get the highest correlated features.
mhc_features = list(metal_correlations.head(1).to_dict().items())[0]

# Train linear regression model, make predictions, and gather analytics.
analyzer.dataset = heavy_metal_tracks
analyzer.model = LinearRegressionModel(
    category_col=SpotifyColumns.GENRE,
    predictor_col=SpotifyColumns.from_column(mhc_features[0][0]),
    response_col=SpotifyColumns.from_column(mhc_features[0][1]),
    persistence=True,
    label_transformer=Genre.from_label,
    degree=3,
)
linear_regression_analytics: LinearModelAnalytics = analyzer.analyze()
```

For the **Metal** tracks, the highest correlating features are *energy* and *loudness* and the plot of the linear regression fitted model shows that the higher energy values correspond to higher loudness (in decibels) values.

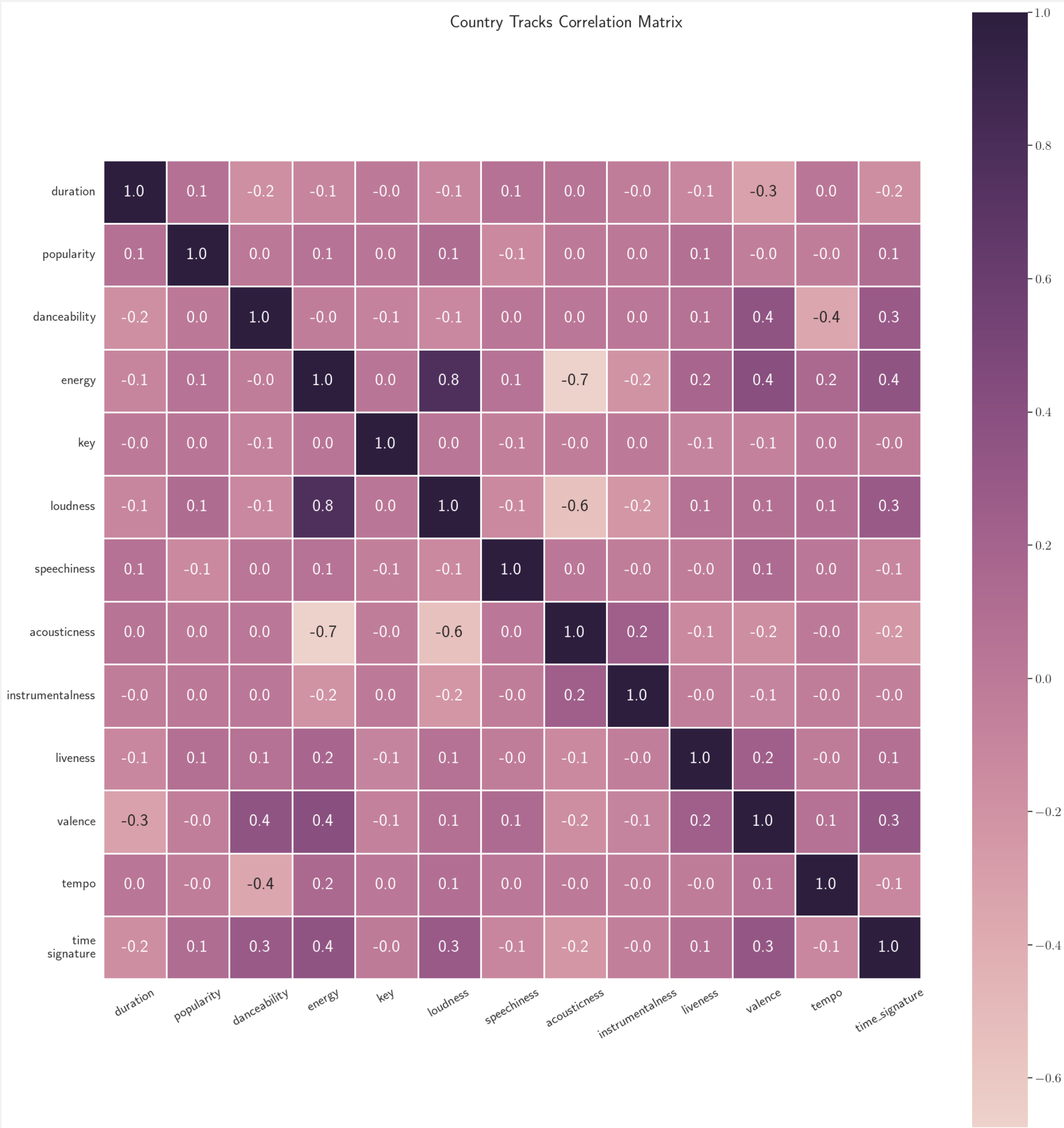
CELL 23

```python
linear_regression_analytics.plot(title="Linear Regression for Metal Tracks")
```



Cell 24

```python
# Plot a correlation matrix for heavy metal tracks.
country_correlation_matrix = ModelAnalyzer.create_correlation_matrix(
    dataset=country_tracks.drop(SpotifyColumns.GENRE, axis=1),
    title="Country Tracks Correlation Matrix"
)
```

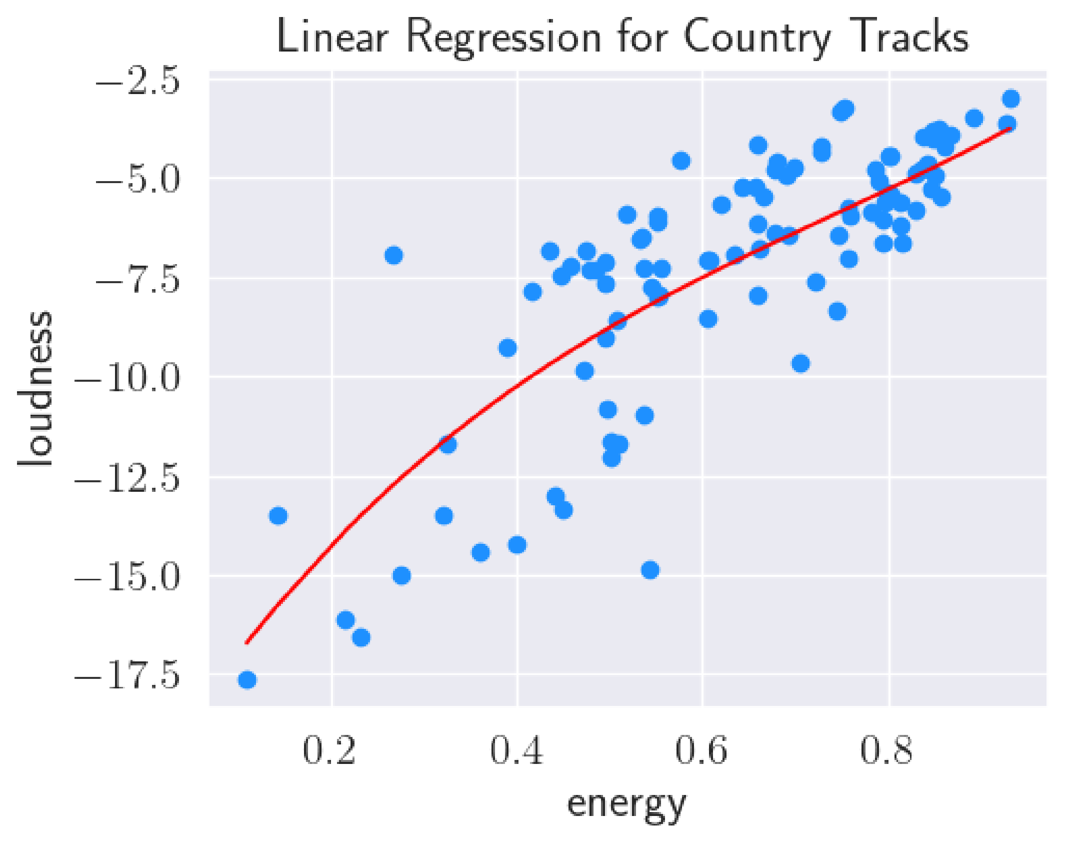


Cell 25

```
# Examine the country tracks' correlation matrix.
country_correlations = examine_correlation_matrix(country_correlation_matrix)

# Get the highest correlated features.
chc_features = list(country_correlations.head(1).to_dict().items())[0]

# Train linear regression model, make predictions, and gather analytics.
analyzer.dataset = country_tracks
analyzer.model = LinearRegressionModel(
    category_col=SpotifyColumns.GENRE,
    predictor_col=SpotifyColumns.from_column(chc_features[0][0]),
    response_col=SpotifyColumns.from_column(chc_features[0][1]),
    persistence=True,
    label_transformer=Genre.from_label,
    degree=3,
)
linear_regression_analytics: LinearModelAnalytics = analyzer.analyze()
```

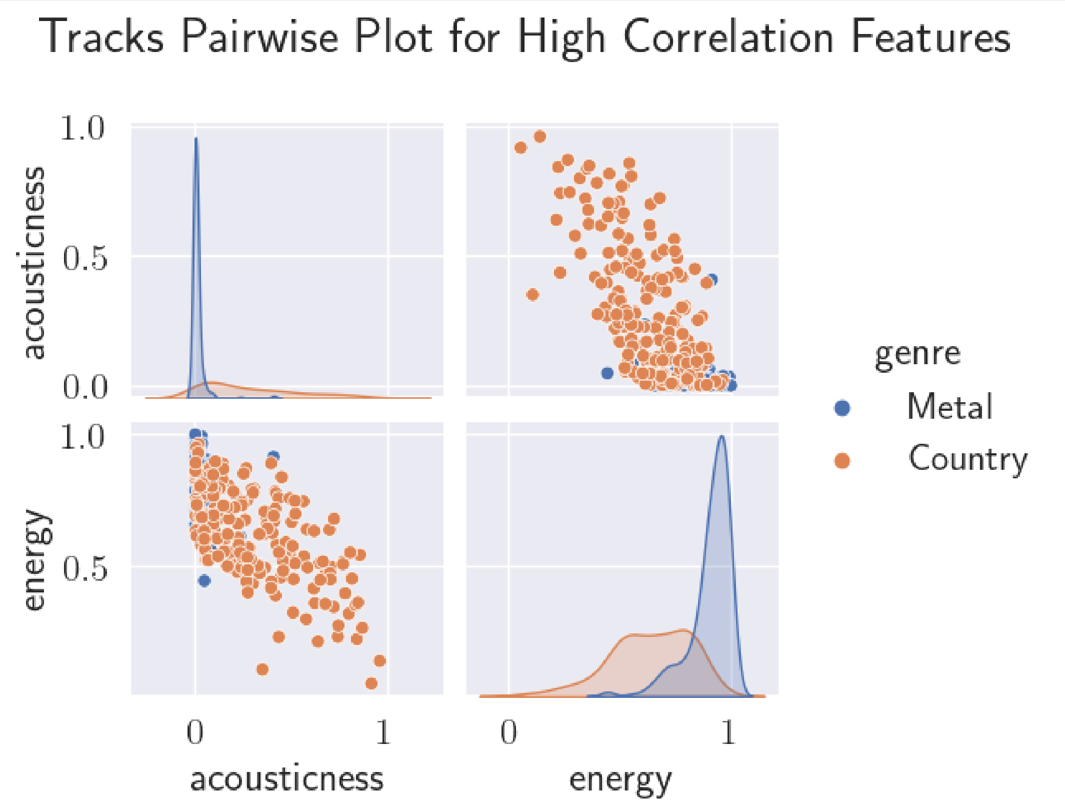
Cell 26

```
linear_regression_analytics.plot(title="Linear Regression for Country Tracks")
```



Cell 27

```python
# Examine the tracks' correlation matrix.
track_correlations = examine_correlation_matrix(correlation_matrix)

# Get the highest correlated features.
hc_features = list(track_correlations.head(1).to_dict().items())[0]

# Train linear regression model, make predictions, and gather analytics.
hc_predictor_col = SpotifyColumns.from_column(hc_features[0][0])
hc_response_col = SpotifyColumns.from_column(hc_features[0][1])

# Create pairwise plot for tracks with high correlation features.
ModelAnalyzer.create_pairwise_plot(
    dataset=tracks_dataset[
        [hc_predictor_col, hc_response_col, SpotifyColumns.GENRE]
    ],
    category_col=SpotifyColumns.GENRE,
    title="Tracks Pairwise Plot for High Correlation Features",
    label_transformer=Genre.from_label_to_title
)
```



Overall, it seems that the **Metal** tracks have higher *loudness* and *energy* and lower *danceability* values, so it seems that are the features that distinguish the values compared to the **Country** tracks.

Cell 28

```
summary_table: str = create_latex_table(
    classifier_analytics.summary_table,
    label="tab:summary_datatable",
    caption="Classifier Summary"
)
Latex(summary_table)
```

Table 1: Classifier Summary

Model	TP	FP	TN	FN	Accuracy	TPR	TNR	F1 Score
Logistic Regression	58	24	101	42	0.71	0.58	0.81	0.64
Linear Kernel SVM	89	3	122	11	0.94	0.89	0.98	0.93
kNN Classifier	87	9	116	13	0.90	0.87	0.93	0.89
Random Forest	95	1	124	5	0.97	0.95	0.99	0.97

References

[1] Audio Tag classification: Instrument – Datasets — musicinformationretrieval.wordpress.com. **https://musicinformationretrieval.wordpress.com/2017/01/24/audio-tag-classification-instrument-datasets/**. [Accessed 26-Apr-2023].

[2] AudioSet — research.google.com. **https://research.google.com/audioset/**. [Accessed 26-Apr-2023].

[3] Beginner's Guide to Audio Data — kaggle.com. **https://www.kaggle.com/code/fizzbuzz/beginner-s-guide-to-audio-data**. [Accessed 26-Apr-2023].

[4] Dataset of songs in Spotify — kaggle.com. **https://www.kaggle.com/datasets/mrmorj/dataset-of-songs-in-spotify**. [Accessed 26-Apr-2023].

[5] Datasets - Spotify Research — research.atspotify.com. **https://research.atspotify.com/datasets/**. [Accessed 26-Apr-2023].

[6] Datasets ML Glossary documentation — ml-cheatsheet.readthedocs.io. **https://ml-cheatsheet.readthedocs.io/en/latest/datasets.html**. [Accessed 26-Apr-2023].

[7] FMA: A Dataset For Music Analysis — arxiv.org. **https://arxiv.org/abs/1612.01840**. [Accessed 26-Apr-2023].

[8] FMA: A Dataset For Music Analysis — zenodo.org. **https://zenodo.org/record/1066119**. [Accessed 26-Apr-2023].

[9] Freesound Audio Tagging 2019 — kaggle.com. **https://www.kaggle.com/c/freesound-audio-tagging-2019**. [Accessed 26-Apr-2023].

[10] GitHub - mdeff/fma: FMA: A Dataset For Music Analysis — github.com. **https://github.com/mdeff/fma**. [Accessed 26-Apr-2023].

[11] GTZAN Dataset - Music Genre Classification — kaggle.com. **https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification**. [Accessed 26-Apr-2023].

[12] How a Histogram Works to Display Data — investopedia.com. **https://www.investopedia.com/terms/h/histogram.asp**. [Accessed 26-Apr-2023].

[13] Machine Learning Glossary ML Glossary documentation — ml-cheatsheet.readthedocs.io. **https://ml-cheatsheet.readthedocs.io/en/latest/index.html**. [Accessed 26-Apr-2023].

[14] Million Song Dataset — millionsongdataset.com. **http://millionsongdataset.com/**. [Accessed 26-Apr-2023].

[15] Music Genre Classification USING KNN(Begineers) — kaggle.com. **https://www.kaggle.com/code/rxsraghavagrawal/music-genre-classification-using-knn-begineers/notebook**. [Accessed 26-Apr-2023].

[16] Radar chart (aka spider or star chart) Matplotlib 3.7.1 documentation — matplotlib.org. **https://matplotlib.org/stable/gallery/specialty_plots/radar_chart.html**. [Accessed 26-Apr-2023].

[17] Song Popularity Prediction - Spotify — kaggle.com. **https://www.kaggle.com/code/satyanarayanam/song-popularity-prediction-spotify**. [Accessed 26-Apr-2023].

[18] The Audio Set Ontology aims to provide a comprehensive set of categories to describe sound events. — github.com. **https://github.com/audioset/ontology**. [Accessed 26-Apr-2023].

[19] This repository collects information about different data sets for Music Emotion Recognition. — github.com. **https://github.com/juansgomez87/datasets_emotion**. [Accessed 26-Apr-2023].

[20] UCI Machine Learning Repository: Geographical Original of Music Data Set — archive.ics.uci.edu. **http://archive.ics.uci.edu/ml/datasets/geographical+original+of+music#**. [Accessed 26-Apr-2023].

[21] Using Spotify's audio features — kaggle.com. **https://www.kaggle.com/code/siropo/using-spotify-s-audio-features**. [Accessed 27-Apr-2023].

[22] 🎹 Spotify Tracks Dataset — kaggle.com. **https://www.kaggle.com/datasets/maharshipandya/-spotify-tracks-dataset**. [Accessed 26-Apr-2023].

[23] Nir Barazida. Open-Source Audio Datasets for ML — towardsdatascience.com. **https://towardsdatascience.com/40-open-source-audio-datasets-for-ml-59dc39d48f06**. [Accessed 26-Apr-2023].

[24] Anne Bode. Spotify API and Audio Features — towardsdatascience.com.

https://towardsdatascience.com/spotify-api-audio-features-5d8bcbd780b2. [Accessed 26-Apr-2023].

[25] Cd. Spotify Genre Classification Algorithm — towardsdatascience.com. **https://towardsdatascience.com/spotify-genre-classification-algorithm-88051db23d42**. [Accessed 26-Apr-2023].

[26] James Chryssanthacopoulos. Advanced Music Analytics using Machine Learning — towardsdatascience.com. **https://towardsdatascience.com/advanced-music-analytics-using-machine-learning-f344e4795bbc**. [Accessed 26-Apr-2023].

[27] Aakash Khandelwal. Part 5 - Plotting Using Seaborn - Radar — aakashkh.github.io. **https://aakashkh.github.io/python/visualisation/2019/08/26/Plotting-Seaborn-Radar.html**. [Accessed 26-Apr-2023].

[28] Parul Pandey. Music Genre Classification with Python — towardsdatascience.com. **https://towardsdatascience.com/music-genre-classification-with-python-c714d032f0d8**. [Accessed 27-Apr-2023].

[29] Sidharth Pandita. Music Genre Classification using Random Forest — medium.com. **https://medium.com/hackerdawn/music-genre-classification-using-random-forest-219fc2446666**. [Accessed 26-Apr-2023].

[30] Dario Radečić. How to Make Stunning Radar Charts with Python Implemented in Matplotlib and Plotly — towardsdatascience.com. **https://towardsdatascience.com/how-to-make-stunning-radar-charts-with-python-implemented-in-matplotlib-and-plotly-91e218** [Accessed 26-Apr-2023].

[31] Soumyaa Rawat. Music Genre Classification Using Machine Learning | Analytics Steps — analyticssteps.com. **https://www.analyticssteps.com/blogs/music-genre-classification-using-machine-learning**. [Accessed 26-Apr-2023].

[32] Maximilian Stäbler. Music Genre Prediction by Extracted Audio-Features + Comparison To the Spotify-API — pub.towardsai.net. **https://pub.towardsai.net/music-genre-prediction-by-extracted-audio-features-and-comparison-to-the-spotify-api-ebf8223** [Accessed 27-Apr-2023].

[33] Nathan Thomas. Using k-nearest neighbours to predict the genre of Spotify tracks — towardsdatascience.com. **https://towardsdatascience.com/using-k-nearest-neighbours-to-predict-the-genre-of-spotify-tracks-796bbbad619f**. [Accessed 26-Apr-2023].

[34] Pedro Henrique Gomes Venturott. Predicting Music Genres Using Waveform Features — towardsdatascience.com. **https://towardsdatascience.com/predicting-music-genres-using-waveform-features-5080e788eb64**. [Accessed 27-Apr-2023].