***Dataset Creation***

We began our pipeline by retrieving audio metadata from the balanced subset of *AudioSet*, specifically filtering for tracks associated within the range of electronic music genres. Our initial target labels included: *House music, Techno, Electronic music, Electronica, Trance music, Electronic dance music, Ambient music, Drum and bass, Dubstep, and Electro.* Using these labels, we compiled a metadata subset containing *YouTube* video IDs and precise 10-second time segments per track. For each entry, we downloaded the highest-quality audio using *yt-dlp [ref]* python library, then trimmed and standardized it to mono *16kHz WAV* format using *ffmpeg* [ref].

During post-processing, we mapped and parsed the associated labels, applying regex-based rules to identify genre annotations. Our aim was to prepare the dataset for *multiclass* classification (as opposed to multi-label), where each audio file must belong to exactly one class. To satisfy this, we performed label filtering, ultimately selecting a clean subset of three well-represented, mutually exclusive genres: *Techno, House music, and Trance music*. These genres were chosen based on their higher instance counts and their relatively distinct label distributions across the dataset. Only audio clips that uniquely matched one of these three target genres, without overlapping with any of the others, were retained. However, *Trance* class was omitted from the analysis because after manual audible verification, the tracks was not really representing this genre of music. The final dataset contained 68 tracks, 39 representing the *House* class while 29 the *Techno* class. Then the dataset was organized into corresponding subdirectories per genre, providing a clean, well-structured foundation for training and evaluating multiclass electronic music classification models. The final directory tree structure was:

ELECTRONIC\_MUSIC/

├── House/

│ ├── <audio\_file\_1>.wav

│ ├── <audio\_file\_2>.wav

│ └── ...

├── Techno/

│ ├── <audio\_file\_1>.wav

│ ├── <audio\_file\_2>.wav

│ └── ...

└── Trance/

├── <audio\_file\_1>.wav

├── <audio\_file\_2>.wav

└── ...

***Data Preparation and Preprocessing***

In the initial stages of our preprocessing pipeline, all music tracks were loaded at a *16 kHz* sampling rate and converted to *mono* to ensure consistency and compatibility across the dataset. Each 10-second audio clip was not segmented into smaller chunks to avoid data leakage and bias during training.

***Feature Extraction***

For all *short-time Fourier transform (STFT)-*based computations, we employed a frame length of *2048* samples and a hop length of *512* samples. From each track/input instance, we extracted a comprehensive set of features widely used in music information retrieval, including:

1. **Zero-Crossing Rate 🡪 (313, 1)**
2. **Spectral Centroid 🡪 (313, 1)**
3. **Onset Strength 🡪 (313, 1)**
4. **The 13 first Mel-frequency cepstral coefficients (MFCCs) 🡪 (313, 13)**
5. **Chroma Features 🡪 (313, 12)**
6. **Spectral Bandwidth 🡪 (313, 1)**
7. **Spectral Roll-off 🡪 (313, 1)**
8. **Root Mean Square (RMS) energy 🡪 (313, 1)**

Given each track is 10-second long, sampled at 16KHz, the framed shape will be:

However, many *Librosa* by default uses padding at the boundaries (typically center=True). This causes an additional:

After stacking the vector representation of each track, we form a 3D tensor of shape (instances, frames, features) 🡪 (68, 313, 31)

***Exploratory Data Analysis***

Following feature extraction, *exploratory data analysis (EDA)* was conducted to assess the statistical distribution and interrelationships among the extracted features. This included visualizing feature distributions using, boxplots (see Figure 1), computing pairwise correlations (see Figure 2), heatmap feature comparison per genre (see Figure 3), and plotting temporal feature trajectories across frames per genre (see ??). Based on the insights of each plot, we will determine later the feature to train the statistical models.

A graph of different colored bars

AI-generated content may be incorrect.

Figure 1. Boxplot Mean Features Distribution. Most useful features for class/genre discrimination are: 'MFCC\_0', 'Chroma\_0', 'Chroma\_1', 'Chroma\_2', 'Chroma\_3', 'Chroma\_4', 'Chroma\_5', 'Chroma\_6', 'Chroma\_7', 'Chroma\_8'. Those will later be used as selected features for performance comparison.

A diagram of a triangle

AI-generated content may be incorrect.

Figure 2. Heatmap correlation matrix between extracted features. Highly correlated features indicate co-linearity and can me omitted for the training process. In our case: 'ZCR', 'Centroid', 'Onset', 'MFCC\_4', 'MFCC\_5', 'MFCC\_6', 'MFCC\_7', MFCC\_8', 'MFCC\_9', 'MFCC\_10', 'MFCC\_11', 'MFCC\_12', 'Chroma\_9', 'Chroma\_10', 'Chroma\_11', 'Bandwidth', 'Rolloff', 'RMS'.

A colorful lines with white text

AI-generated content may be incorrect.

Figure 3. Class-wise Feature Means (Standardized). Features between classes with high difference are good candidates. In our case: 'Chroma\_7', 'Chroma\_8', 'MFCC\_0', 'Chroma\_6', 'Chroma\_0', 'Chroma\_10', 'MFCC\_5', 'MFCC\_7', 'Chroma\_1', 'Chroma\_5', 'Chroma\_11', 'MFCC

***AI Modeling and Experimental Setups***

To better understand the separability of the classes, Principal Component Analysis (PCA) were applied for dimensionality reduction, to different 2D representations of the 3D feature tensor, evaluating the effects of collapsing across time or flattening the two axes of frames and features, resulting into a vector of (frames x features). Below, we provide these PCA plots:

|  |  |
| --- | --- |
| A graph with yellow and purple dots  AI-generated content may be incorrect. | A graph with many dots  AI-generated content may be incorrect. |

Figure 4. bla bla..

|  |  |
| --- | --- |
| A graph with yellow and purple dots  AI-generated content may be incorrect. | A graph of a graph showing different colored dots  AI-generated content may be incorrect. |

Figure 5. bla bla

As we can observe, the data points of each class are not easily separable, however, at the first PCA component we can observe up to 95% of the explained variance.

For evaluation, the dataset was split into training and testing sets using stratified sampling with an 80/20 ratio and a fixed random seed for reproducibility. We applied three data normalization techniques, *MinMax* scaling, *z-score* *standardization* (StandardScaler), and *Robust* scaling (due to the outliers observed in Figure 1). Each scaling method was evaluated in conjunction with four dimensionality reduction setups: no dimensionality reduction (baseline – all extracted features included), PCA with 2 components, PCA with 3 components, and PCA retaining 95% of the explained variance. For the classification, two classifiers were tested: a Support Vector Machine (SVM) with an RBF kernel and a Random Forest (RF) classifier. For both models, hyperparameters were optimized using a grid search with 5-fold cross-validation, executed only on the training partition. The SVM grid included regularization parameter C = {0.1, 1, 10} and kernel coefficient gamma = {‘scale’, ‘auto’}, while the Random Forest grid tested n\_estimators = {100, 200} and max\_depth = {None, 10, 20}.

After running initial experiments with different combinations of scalers, dimensionality reduction techniques (like PCA), and classifiers (e.g., SVM, Random Forest), we selected the best-performing configuration based on evaluation metrics such as weighted F1-score and accuracy. Once this baseline was established, we focused on improving the input feature set through systematic analysis. Below we summarize the results of the experiments:

| **Scaler** | **PCA Method** | **SVM (Collapsed)** | **SVM (Flattened)** | **RF (Collapsed)** | **RF (Flattened)** |
| --- | --- | --- | --- | --- | --- |
| MinMax | no\_pca | 0.714 / 0.714 | 0.782 / 0.786 | ***0.857 / 0.857*** | 0.552 / 0.571 |
| MinMax | pca\_2d | ***0.927 / 0.929*** | 0.416 / 0.571 | 0.503 / 0.500 | 0.782 / 0.786 |
| MinMax | pca\_3d | 0.782 / 0.786 | 0.416 / 0.571 | 0.787 / 0.786 | 0.714 / 0.714 |
| MinMax | pca\_95var | 0.857 / 0.857 | ***0.787 / 0.786*** | 0.492 / 0.500 | 0.558 / 0.643 |
| Standard | no\_pca | 0.416 / 0.571 | 0.782 / 0.786 | 0.857 / 0.857 | 0.552 / 0.571 |
| Standard | pca\_2d | 0.702 / 0.714 | 0.558 / 0.643 | 0.702 / 0.714 | 0.714 / 0.714 |
| Standard | pca\_3d | 0.552 / 0.571 | 0.416 / 0.571 | 0.782 / 0.786 | ***0.857 / 0.857*** |
| Standard | pca\_95var | 0.702 / 0.714 | 0.714 / 0.714 | 0.702 / 0.714 | 0.671 / 0.714 |
| Robust | no\_pca | 0.702 / 0.714 | 0.612 / 0.643 | ***0.857 / 0.857*** | 0.552 / 0.571 |
| Robust | pca\_2d | ***0.927 / 0.929*** | 0.456 / 0.500 | 0.787 / 0.786 | 0.403 / 0.429 |
| Robust | pca\_3d | 0.552 / 0.571 | 0.416 / 0.571 | 0.787 / 0.786 | 0.571 / 0.571 |
| Robust | pca\_95var | 0.702 / 0.714 | 0.226 / 0.357 | 0.571 / 0.571 | 0.612 / 0.643 |

Table 1. Experiment results for all configurations with all extracted features as input.

In the aforementioned experimental setup, the configuration the yielded the best results was the one trained with the SVM (c=10, gamma=”scale”), using input the 2D PCA data, normalized by the MinMax (and Robust) scaler, resulting in an average F1-score = 0.927, and Accuracy = 0.929. Below we also present the confusion matrix results:

A blue squares with white text

AI-generated content may be incorrect.

Figure 6. Confusion matrix results of the best performing configuration at the baseline training.

***Feature Selection***

Based on the previous visualizations and analyses, we selected features that consistently demonstrated good separation between genres and low redundancy. Features with significant overlap or high correlation were excluded. Using the insights from each feature plot (as noted in the figure captions), we chose the most relevant features and retrained our AI pipeline accordingly. Each feature selection method was treated as a separate experiment.

* ***Heatmap-based selection included:*** Chroma\_0, Chroma\_1, Chroma\_5, Chroma\_6, Chroma\_7, Chroma\_8, Chroma\_10, Chroma\_11, MFCC\_0, MFCC\_1, MFCC\_5, and MFCC\_7
* ***Correlation-based filtering excluded:*** ZCR, Centroid, Onset, Bandwidth, Rolloff, RMS, Chroma\_9, Chroma\_10, Chroma\_11, and MFCCs from MFCC\_4 to MFCC\_12
* ***Boxplot-based selection included:*** MFCC\_0, Chroma\_0, Chroma\_1, Chroma\_2, Chroma\_3, Chroma\_4, Chroma\_5, Chroma\_6, Chroma\_7, and Chroma\_8

Since the SVM classifier consistently outperformed Random Forest in this setup, we report only the SVM results for these experiments. The performance metrics for each experiment are presented below.

| **Scaler** | **PCA Type** | **Feature Selection** | **F1-score** | **Accuracy** |
| --- | --- | --- | --- | --- |
| MinMax | no\_pca | Heatmap | 0.857 | 0.857 |
|  | no\_pca | Boxplots | 0.782 | 0.786 |
|  | no\_pca | Correlation Matrix | 0.857 | 0.857 |
|  | pca\_2d | Heatmap | 0.787 | 0.786 |
|  | pca\_2d | Boxplots | 0.851 | 0.857 |
|  | pca\_2d | Correlation Matrix | ***0.927*** | ***0.929*** |
|  | pca\_3d | Heatmap | 0.787 | 0.786 |
|  | pca\_3d | Boxplots | 0.782 | 0.786 |
|  | pca\_3d | Correlation Matrix | 0.714 | 0.714 |
|  | pca\_95var | Heatmap | 0.857 | 0.857 |
|  | pca\_95var | Boxplots | 0.857 | 0.857 |
|  | pca\_95var | Correlation Matrix | 0.857 | 0.857 |
| Standard | no\_pca | Heatmap | 0.702 | 0.714 |
|  | no\_pca | Boxplots | 0.702 | 0.714 |
|  | no\_pca | Correlation Matrix | 0.714 | 0.714 |
|  | pca\_2d | Heatmap | ***0.857*** | ***0.857*** |
|  | pca\_2d | Boxplots | 0.416 | 0.571 |
|  | pca\_2d | Correlation Matrix | 0.416 | 0.571 |
|  | pca\_3d | Heatmap | 0.787 | 0.786 |
|  | pca\_3d | Boxplots | 0.782 | 0.786 |
|  | pca\_3d | Correlation Matrix | 0.714 | 0.714 |
|  | pca\_95var | Heatmap | 0.782 | 0.786 |
|  | pca\_95var | Boxplots | 0.702 | 0.714 |
|  | pca\_95var | Correlation Matrix | 0.782 | 0.786 |
| Robust | no\_pca | Heatmap | 0.767 | 0.786 |
|  | no\_pca | Boxplots | 0.702 | 0.714 |
|  | no\_pca | Correlation Matrix | 0.782 | 0.786 |
|  | pca\_2d | Heatmap | ***0.851*** | ***0.857*** |
|  | pca\_2d | Boxplots | 0.416 | 0.571 |
|  | pca\_2d | Correlation Matrix | 0.416 | 0.571 |
|  | pca\_3d | Heatmap | 0.857 | 0.857 |
|  | pca\_3d | Boxplots | 0.782 | 0.786 |
|  | pca\_3d | Correlation Matrix | 0.767 | 0.786 |
|  | pca\_95var | Heatmap | 0.702 | 0.714 |
|  | pca\_95var | Boxplots | 0.702 | 0.714 |
|  | pca\_95var | Correlation Matrix | 0.782 | 0.786 |

Table 2. bla bla..

Based on the results after the feature selection strategies, we can observe that the AI Pipeline configuration that yielded the best results is the same as before (MinMax Scaler + PCA-2D), with the same performance metrics results based on the correlation matrix exclusion strategy. However, even though the performance remained the same, we achieved this by retaining half of the initial input features. This is still a big benefit since in a larger scale, we would save up computation cost and model size.