footer line

Machine Learning Project Report

CPEN 335 - Group 10

Idil Bil

Mehmet Berke Karadayi

Peter Na

*November 27, 2024*

footer line

# 

# 

**Table of Contents**

[1. Abstract 1](#_mvbgqhvlvc2j)

[2. Introduction 1](#_oqzc1tw0d4ko)

[3. Data 1](#_omc9hzkfrobk)

[Graph 1: Distribution of Music Genres in the Dataset 2](#_whb23tgakrbx)

[4. Model 2](#_h9sg91mq9h0t)

[4.1 Random Forest Classifier 2](#_bc7dedp6q286)

[4.2 Support Vector Classifier 2](#_mx0g28rdd2io)

[4.3 Comparison of the Classification Models 3](#_7kfvumowztad)

[Graph 2: RFC Test and Train Scores Line Graph 3](#_4ir1yxufijtx)

[Graph 3: SVC Test and Train Scores Line Graph 3](#_mwiasik9cdtf)

[5. Evaluation 3](#_ap1tzo1dlii)

[Table 1: Evaluation Metrics Comparison Table 4](#_leh1mt62um46)

[Table 2: Confusion Matrix of the Best Case (RFC with Random Hyperparameters) 4](#_h20fvw331wvu)

[6. Discussion 4](#_i2xbp4z1j5z9)

[References 5](#_louts360nr0m)

[Appendix 6](#_kfwmv1ipgqy7)

[A. RFC with Minimal Hyperparameters 6](#_4qkff5cyes52)

[B. RFC with Random Hyperparameters 7](#_eaapzh3xpkmg)

[C. RFC with GridSearchCV Hyperparameters 8](#_2i3507b8y4pe)

[D. RFC with BayesSearchCV Hyperparameters 9](#_7lzo4gtjcdoj)

[E. SVC with Minimal Hyperparameters 10](#_qeg04dwv4ars)

[F. SVC with Random Hyperparameters 11](#_5ykjgtce2xn7)

[G. SVC with GridSearchCV Hyperparameters 12](#_l27rhb8oncyp)

[H. SVC with BayesSearchCV Parameters 13](#_q8ao2w6k2hnp)

# Abstract

This project enhances genre accuracy for Spotify’s ‘daylist’ playlist using supervised models, specifically Random Forest Classifier (RFC) and Support Vector Classifier (SVC). Trained on song metadata the models were optimized for accuracy, with RFC achieving 70%, slightly above SVC's 68% due to its ensemble of decision trees. One-hot encoding for categorical features and random parameter tuning yielded the best results within time constraints.

# Introduction

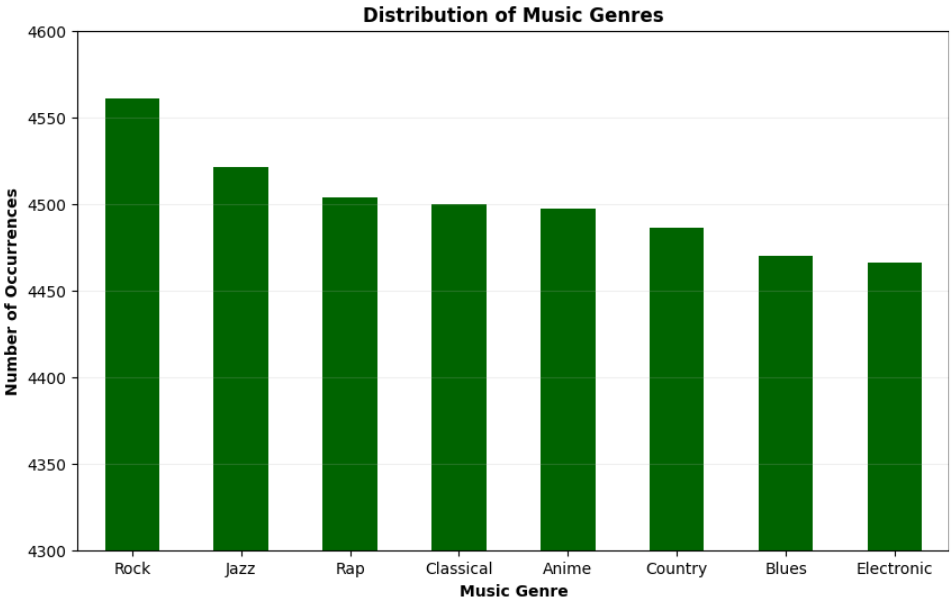
Spotify’s new AI-supported playlist, ‘daylist,’ generates a custom playlist based on the day and time the user opens it, but its genre descriptions are often inaccurate. Existing genre classifiers on Kaggle [1] and Hugging Face [2] use Transformers and Neural Networks respectively. Our solution employs two supervised machine learning models to classify genres accurately, ensuring the playlist description matches the songs.

# Data

We selected a music metadata set [3] with various song features, focusing on how different musical attributes relate to genres rather than genre connections with lyrics or audio samples. Here’s a list of all included features:

* *instance\_id:* Unique ID for each song
* *artist\_name:*Name of the artist who owns the song
* *track\_name:* Name of the song
* *popularity (0 to 99):* How popular the song is
* *acousticness (0 to 1):* Likelihood that the song is acoustic
* *danceability (0.06 to 0.99):* How suitable a song is for dancing
* *duration\_ms (0 to 4 minutes):* Duration of the song in milliseconds
* *energy (0 to 1):* Perceived activity in the music, with higher values for faster songs
* *instrumentalness (0 to 1):* Probability that a song has no vocals
* *liveness (0.01 to 1):* Measures the presence of a live audience sound
* *loudness (-47 to 3.74):* Overall volume level of the song, measured in decibels (dB)
* *mode:* Tonality of the track, indicating whether it's major (0) or minor (1)
* *speechiness (0.02 to 0.94):* Detects the presence of spoken words
* *tempo (35 to 221):* Speed of the music, measured in beats per minute (BPM)
* *obtained \_date:* Date on which the data was collected from a source
* *valence (0 to 0.99):* Mood of the music, with higher values for more positive tracks

For our Exploratory Data Analysis (EDA), we first removed columns irrelevant to genre classification: instance\_id, track\_name, artist\_name, and obtained\_date. We then dropped rows with unknown features marked by a '?'. Next, we visualized the distribution of data across genres using a bar graph. The results showed a well-balanced dataset, so we determined that no weight adjustments were necessary.



### Graph 1: Distribution of Music Genres in the Dataset

# Model

We chose two models to classify music genres using Supervised Learning: Random Forest Classifier and Support Vector Classifier. This allowed us to compare their performance and evaluate which model offers better results for music genre classification.

## 4.1 Random Forest Classifier

RFC, a model that combines outputs from multiple decision trees for classification [4], was chosen for its effectiveness with high-dimensional datasets like ours. Each tree is trained on a subset of data, with randomness introduced at each split to reduce overfitting, resulting in more accurate and robust predictions. The following hyperparameters were tuned focusing on the optimal balance between model performance and training time:

* *n\_estimators = 1000:* Sets the forest to 1000 trees, stabilizing model output and increasing test score.
* *max\_features = ‘sqrt’:* Limits features for each split to the square root of the total, reducing tree correlation and improving diversity.
* *max\_depth = 20:* Caps tree depth to 20, balancing complexity and preventing overfitting.
* *min\_samples\_split = 20:* Requires at least 20 samples to split a node, reducing noise impact by ensuring data sufficiency for splits.
* *min\_sample\_lead = 5:*Maximum number of samples in each leaf node.
* *random\_state = 42:* Random seed to ensure reproducibility.

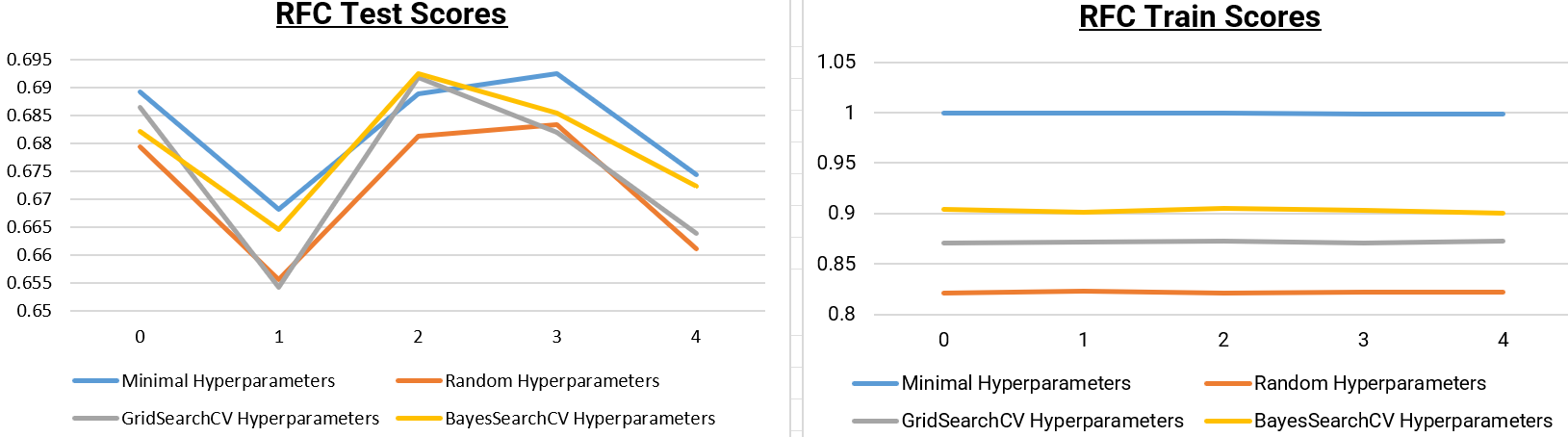
## 4.2 Support Vector Classifier

SVC classifies data by finding the hyperplane that maximizes the margin between classes, making it effective for high-dimensional datasets, especially when classes aren’t linearly separable. Using an RBF kernel, SVC maps our multi-feature dataset into a higher dimension, enhancing class separation and reducing overfitting from noise. The following hyperparameters were tuned focusing on maximizing accuracy without overfitting and balancing flexibility:

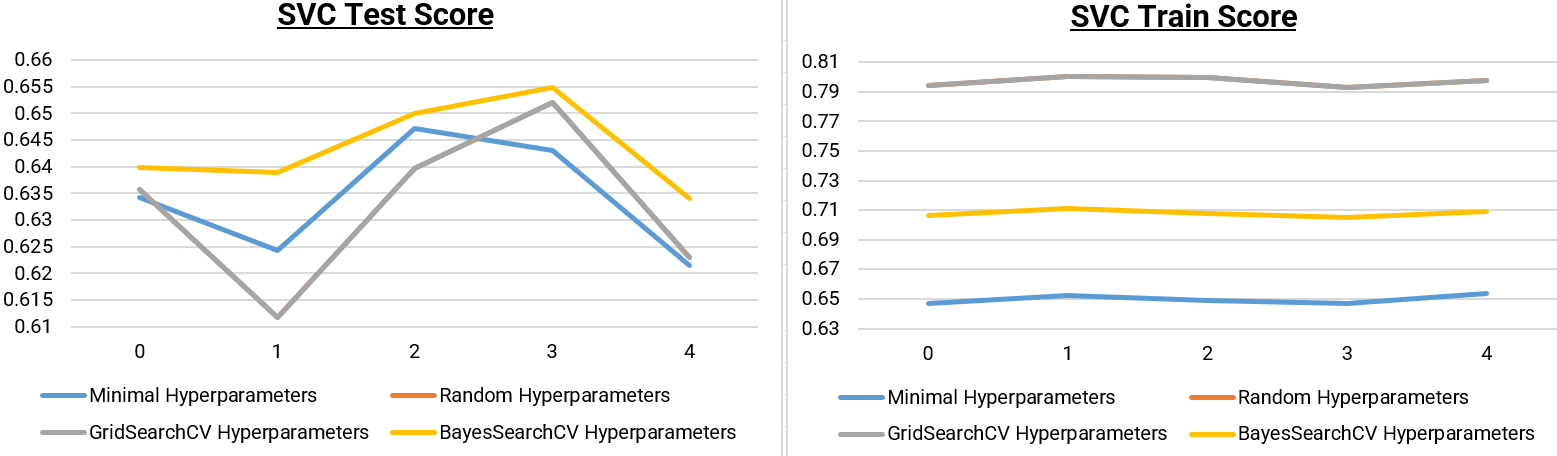
* *kernel = 'rbf':* RBF kernel selected for strong performance with non-linear data.
* *C = 5.0:* Regularization parameter balancing margin and classification error; moderate value to avoid overfitting.
* *gamma = 'scale':* Adjusts influence of data points on the boundary, accounting for feature variance.
* *max\_iter = -1:* Unlimited iterations allowed for convergence if resources permit.
* *random\_state = 42:* Sets a random seed to ensure reproducibility.

## 4.3 Comparison of the Classification Models

The best configurations for RFC were GridSearchCV and Random Hyperparameters, achieving test scores of ~0.68-0.69 and similar train scores to reduce overfitting. For SVC, the Random Hyperparameters setup performed best, with a test score of ~0.65 and consistent train scores. Overall, RFC outperformed SVC, as its ensemble approach more effectively captures complex data patterns, yielding higher and more stable test scores.



### Graph 2: RFC Test and Train Scores Line Graph



### Graph 3: SVC Test and Train Scores Line Graph

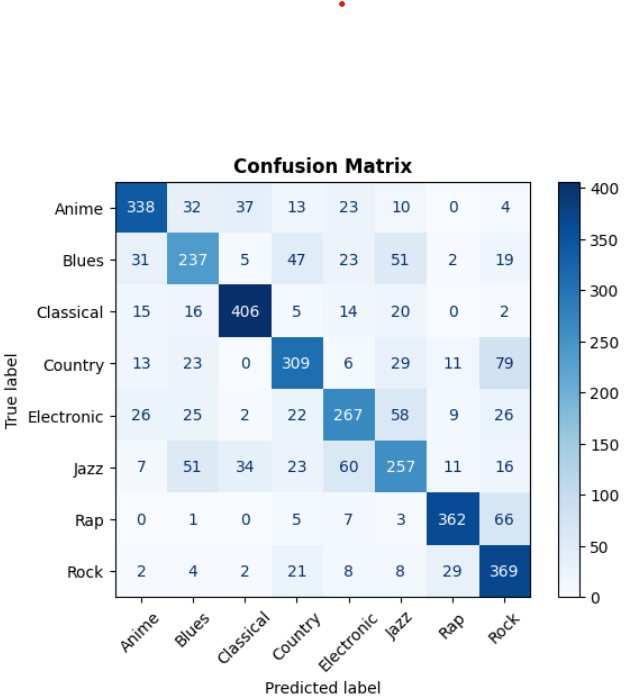
# Evaluation

To evaluate our models, we have chosen the list of metrics below:

* **Precision:** Measures the reliability of predictions for each genre. In our case precision is more important than recall because minimizing incorrect genre labels is prioritized over missing some correctly labeled songs in a playlist. Our best precision score was 0.71.
* **F1-score:** Balances precision and recall, making it ideal for evaluating performance on less frequent genres. A high score indicates effective genre identification with minimal errors. Our best F1-score was 0.71.
* **Accuracy:** Reflects the overall correctness of the model but may be misleading with imbalanced classes, so it’s considered alongside precision and F1-score. With RFC, we consistently achieved over 70% accuracy, with a peak of 70.7%.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Evaluation Metrics** | *RFC with Minimal Parameters* | *RFC with Random Parameters* | *RFC with GridSearchCV Parameters* | *RFC with BayesSearchCV Parameters* | *SVC with Minimal Parameters* | *SVC with Random Parameters* | *SVC with GridSearchCV Parameters* | *SVC with BayesSearchCV Parameters* |
| **Accuracy** | 70.1% | 70.7% | 70.2% | 70.5% | 65.4% | 68.0% | 68.0% | 67.9% |
| **W. Avg. Precision** | 0.70 | 0.71 | 0.70 | 0.71 | 0.65 | 0.68 | 0.68 | 0.68 |
| **W. Avg. F1-Score** | 0.70 | 0.71 | 0.70 | 0.70 | 0.65 | 0.68 | 0.68 | 0.68 |

### Table 1: Evaluation Metrics Comparison Table



### Table 2: Confusion Matrix of the Best Case (RFC with Random Hyperparameters)

# Discussion

Our first challenge was converting string-type features into numerical values. Initially, we used the Label Encoder function, which impacted the weights of the "Key" and "Mode" columns by assigning integer values to the strings, resulting in ~57% accuracy with RFC. Switching to One-Hot Encoding, which created separate columns for each unique string and encoding them as 0s and 1s, improved accuracy to ~67% with RFC.

The second challenge was tuning hyperparameters to improve accuracy. We began by using minimal parameters to avoid overfitting, gradually increasing them to reach ~70% accuracy with RFC and ~68% with SVC. We then explored GridSearchCV and BayesSearchCV, but they took up to four hours to run before disconnecting on Google Colab, so we had to limit their runtime to two hours, not being able to use them to their full potential. Despite these efforts, neither search algorithm outperformed our initial random parameters.

# References

**[1]**

dima806, “Music genre classification wav2vec2-base-960h,” *Kaggle.com*, Aug. 15, 2023. <https://www.kaggle.com/code/dima806/music-genre-classification-wav2vec2-base-960h/notebook#Load-facebook/wav2vec2-base-960h-model> (accessed Nov. 11, 2024).

**‌[2]**

“ruben09/music\_genre\_classification at main,” *Huggingface.co*, Oct. 09, 2024. <https://huggingface.co/ruben09/music_genre_classification/tree/main> (accessed Nov. 11, 2024).

**‌[3]**

vicsuperman, “Prediction of music genre,” *Kaggle.com*, 2021. <https://www.kaggle.com/datasets/vicsuperman/prediction-of-music-genre/data> (accessed Nov. 11, 2024).

**[4]**

IBM, “Random Forest,” *Ibm.com*, Oct. 20, 2021. <https://www.ibm.com/topics/random-forest> (accessed Nov. 11, 2024).

**‌[5]**

IBM, “Support Vector Machine,” *Ibm.com*, Dec. 12, 2023. <https://www.ibm.com/topics/support-vector-machine> (accessed Nov. 11, 2024).

‌

# 

# Appendix

## RFC with Minimal Hyperparameters

n\_estimators=1000, # Number of decision trees in the random forest

random\_state=42, # Set a seed to ensure reproducibility

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **fit\_time** | **score\_time** | **test\_score** | **train\_score** |
| *0* | 17.9295 | 0.403038 | 0.689105 | 0.999479 |
| *1* | 18.5817 | 0.513586 | 0.668056 | 0.999479 |
| *2* | 17.6812 | 0.413723 | 0.688889 | 0.999653 |
| *3* | 18.3633 | 0.53648 | 0.692361 | 0.999132 |
| *4* | 17.6192 | 0.397905 | 0.674306 | 0.999132 |

**Accuracy = 0.7014718133851708 ~ 70.1%**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| *Anime* | 0.8 | 0.77 | 0.78 | 457 |
| *Blues* | 0.6 | 0.57 | 0.58 | 415 |
| *Classical* | 0.84 | 0.85 | 0.85 | 478 |
| *Country* | 0.68 | 0.65 | 0.67 | 470 |
| *Electronic* | 0.65 | 0.61 | 0.63 | 435 |
| *Jazz* | 0.58 | 0.56 | 0.57 | 459 |
| *Rap* | 0.83 | 0.8 | 0.82 | 444 |
| *Rock* | 0.63 | 0.78 | 0.7 | 443 |
|  |  |  |  |  |
| **accuracy** |  |  | 0.7 | 3601 |
| **macro avg** | 0.7 | 0.7 | 0.7 | 3601 |
| **weighted avg** | 0.7 | 0.7 | 0.7 | 3601 |

## RFC with Random Hyperparameters

n\_estimators=1000, # Number of decision trees in the random forest

max\_features='sqrt', # Number of features considered for splitting at each node

max\_depth= 20, # Maximum depth of each decision tree

min\_samples\_split=20, # Minimum number of samples required to split an internal node

min\_samples\_leaf= 5, # Minimum number of samples required to be at a leaf node

random\_state=42, # Set a random seed to ensure reproducibility

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **fit\_time** | **score\_time** | **test\_score** | **train\_score** |
| *0* | 13.544948 | 0.254406 | 0.679389 | 0.821007 |
| *1* | 13.583202 | 0.259867 | 0.655556 | 0.822947 |
| *2* | 13.487768 | 0.258764 | 0.68125 | 0.821038 |
| *3* | 13.975006 | 0.291484 | 0.683333 | 0.822253 |
| *4* | 13.464609 | 0.330981 | 0.661111 | 0.822253 |

**Accuracy = 0.7067481255206887 ~ 70.7%**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| *Anime* | 0.78 | 0.74 | 0.76 | 457 |
| *Blues* | 0.61 | 0.57 | 0.59 | 415 |
| *Classical* | 0.84 | 0.85 | 0.84 | 478 |
| *Country* | 0.69 | 0.66 | 0.68 | 470 |
| *Electronic* | 0.65 | 0.61 | 0.63 | 435 |
| *Jazz* | 0.59 | 0.56 | 0.57 | 459 |
| *Rap* | 0.85 | 0.82 | 0.83 | 444 |
| *Rock* | 0.64 | 0.83 | 0.72 | 443 |
|  |  |  |  |  |
| **accuracy** |  |  | 0.71 | 3601 |
| **macro avg** | 0.71 | 0.7 | 0.7 | 3601 |
| **weighted avg** | 0.71 | 0.71 | 0.71 | 3601 |

## RFC with GridSearchCV Hyperparameters

n\_estimators=500, # Number of decision trees in the random forest

max\_features='sqrt', # Number of features considered for splitting at each node

max\_depth= 20, # Maximum depth of each decision tree

min\_samples\_split=10, # Minimum number of samples required to split an internal node

min\_samples\_leaf= 5, # Minimum number of samples required to be at a leaf node

random\_state=42, # Set a random seed to ensure reproducibility

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **fit\_time** | **score\_time** | **test\_score** | **train\_score** |
| *0* | 8.798651 | 0.137202 | 0.686329 | 0.871007 |
| *1* | 6.871165 | 0.139467 | 0.654167 | 0.872071 |
| *2* | 8.400421 | 0.138711 | 0.691667 | 0.872244 |
| *3* | 7.104589 | 0.190786 | 0.681944 | 0.870682 |
| *4* | 8.043493 | 0.141147 | 0.663889 | 0.872939 |

**Accuracy = 0.7020272146625938 ~ 70.2%**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| *Anime* | 0.79 | 0.76 | 0.77 | 457 |
| *Blues* | 0.61 | 0.57 | 0.59 | 415 |
| *Classical* | 0.84 | 0.84 | 0.84 | 478 |
| *Country* | 0.68 | 0.65 | 0.67 | 470 |
| *Electronic* | 0.64 | 0.61 | 0.63 | 435 |
| *Jazz* | 0.58 | 0.55 | 0.57 | 459 |
| *Rap* | 0.84 | 0.81 | 0.83 | 444 |
| *Rock* | 0.63 | 0.81 | 0.71 | 443 |
|  |  |  |  |  |
| **accuracy** |  |  | 0.7 | 3601 |
| **macro avg** | 0.7 | 0.7 | 0.7 | 3601 |
| **weighted avg** | 0.7 | 0.7 | 0.7 | 3601 |

## RFC with BayesSearchCV Hyperparameters

n\_estimators=500, # Number of decision trees in the random forest

max\_features='log2', # Number of features considered for splitting at each node

max\_depth= 18, # Maximum depth of each decision tree

min\_samples\_split=4, # Minimum number of samples required to split an internal node

min\_samples\_leaf= 4, # Minimum number of samples required to be at a leaf node

random\_state=42, # Set a random seed to ensure reproducibility

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **fit\_time** | **score\_time** | **test\_score** | **train\_score** |
| *0* | 5.434311 | 0.154588 | 0.682165 | 0.904167 |
| *1* | 6.31635 | 0.167892 | 0.664583 | 0.901232 |
| *2* | 5.891382 | 0.178977 | 0.692361 | 0.905225 |
| *3* | 5.554878 | 0.120275 | 0.685417 | 0.902968 |
| *4* | 5.648211 | 0.150435 | 0.672222 | 0.900712 |

**Accuracy = 0.7050819216884199 ~ 70.5%**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| *Anime* | 0.8 | 0.75 | 0.77 | 457 |
| *Blues* | 0.61 | 0.57 | 0.59 | 415 |
| *Classical* | 0.84 | 0.85 | 0.84 | 478 |
| *Country* | 0.69 | 0.66 | 0.68 | 470 |
| *Electronic* | 0.65 | 0.62 | 0.63 | 435 |
| *Jazz* | 0.58 | 0.56 | 0.57 | 459 |
| *Rap* | 0.84 | 0.8 | 0.82 | 444 |
| *Rock* | 0.63 | 0.82 | 0.71 | 443 |
|  |  |  |  |  |
| **accuracy** |  |  | 0.71 | 3601 |
| **macro avg** | 0.71 | 0.7 | 0.7 | 3601 |
| **weighted avg** | 0.71 | 0.71 | 0.7 | 3601 |

## SVC with Minimal Hyperparameters

kernel='linear', # Use a linear kernel for linear separation between classes

random\_state=42 # Set a random seed to ensure reproducibility

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **fit\_time** | **score\_time** | **test\_score** | **train\_score** |
| *0* | 3.216594 | 0.34528 | 0.634282 | 0.646875 |
| *1* | 1.310938 | 0.219468 | 0.624306 | 0.65197 |
| *2* | 1.596697 | 0.363236 | 0.647222 | 0.649193 |
| *3* | 1.785099 | 0.222131 | 0.643056 | 0.646763 |
| *4* | 1.286128 | 0.22146 | 0.621528 | 0.653532 |

**Accuracy: 0.654262704804221 ~ 65.4%**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| *Anime* | 0.68 | 0.66 | 0.67 | 457 |
| *Blues* | 0.5 | 0.5 | 0.5 | 415 |
| *Classical* | 0.8 | 0.81 | 0.8 | 478 |
| *Country* | 0.58 | 0.61 | 0.6 | 470 |
| *Electronic* | 0.61 | 0.61 | 0.61 | 435 |
| *Jazz* | 0.55 | 0.47 | 0.51 | 459 |
| *Rap* | 0.84 | 0.79 | 0.81 | 444 |
| *Rock* | 0.65 | 0.78 | 0.71 | 443 |
|  |  |  |  |  |
| **accuracy** |  |  | 0.65 | 3601 |
| **macro avg** | 0.65 | 0.65 | 0.65 | 3601 |
| **weighted avg** | 0.65 | 0.65 | 0.65 | 3601 |

## 

## SVC with Random Hyperparameters

kernel='rbf', # Use an rbf kernel for non-linear separation between classes

C=5.0, # Regularization parameter

gamma='scale', # Control the influence of individual data points on the decision boundary

max\_iter=-1, # Set the maximum number of iterations (-1 = no limit, let it converge)

random\_state=42 # Set a random state for reproducibility

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **fit\_time** | **score\_time** | **test\_score** | **train\_score** |
| *0* | 1.271409 | 0.472161 | 0.63567 | 0.794271 |
| *1* | 1.255899 | 0.811025 | 0.611806 | 0.800729 |
| *2* | 1.264189 | 0.49519 | 0.639583 | 0.799861 |
| *3* | 1.236747 | 0.473022 | 0.652083 | 0.792744 |
| *4* | 1.240925 | 0.489833 | 0.622917 | 0.797605 |

**Accuracy = 0.6800888642043876 ~ 68.0%**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| *Anime* | 0.72 | 0.72 | 0.72 | 457 |
| *Blues* | 0.57 | 0.58 | 0.57 | 415 |
| *Classical* | 0.85 | 0.83 | 0.84 | 478 |
| *Country* | 0.61 | 0.61 | 0.61 | 470 |
| *Electronic* | 0.63 | 0.6 | 0.62 | 435 |
| *Jazz* | 0.59 | 0.52 | 0.55 | 459 |
| *Rap* | 0.83 | 0.81 | 0.82 | 444 |
| *Rock* | 0.63 | 0.75 | 0.69 | 443 |
|  |  |  |  |  |
| **accuracy** |  |  | 0.68 | 3601 |
| **macro avg** | 0.68 | 0.68 | 0.68 | 3601 |
| **weighted avg** | 0.68 | 0.68 | 0.68 | 3601 |

## SVC with GridSearchCV Hyperparameters

kernel='rbf', # Use a linear kernel for linear separation between classes

C=5.0, # Regularization parameter

gamma='scale', # Control the influence of individual data points on the decision boundary

max\_iter=-1, # Set the maximum number of iterations (-1 = no limit, let it converge)

random\_state=42 # Set a random state for reproducibility

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **fit\_time** | **score\_time** | **test\_score** | **train\_score** |
| *0* | 1.408214 | 0.501198 | 0.63567 | 0.794271 |
| *1* | 1.414702 | 0.820254 | 0.611806 | 0.800729 |
| *2* | 1.482958 | 0.473751 | 0.639583 | 0.799861 |
| *3* | 1.336097 | 0.489697 | 0.652083 | 0.792744 |
| *4* | 1.311151 | 0.497307 | 0.622917 | 0.797605 |

**Accuracy = 0.6800888642043876 ~ 68.0%**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| *Anime* | 0.72 | 0.72 | 0.72 | 457 |
| *Blues* | 0.57 | 0.58 | 0.57 | 415 |
| *Classical* | 0.85 | 0.83 | 0.84 | 478 |
| *Country* | 0.61 | 0.61 | 0.61 | 470 |
| *Electronic* | 0.63 | 0.6 | 0.62 | 435 |
| *Jazz* | 0.59 | 0.52 | 0.55 | 459 |
| *Rap* | 0.83 | 0.81 | 0.82 | 444 |
| *Rock* | 0.63 | 0.75 | 0.69 | 443 |
|  |  |  |  |  |
| **accuracy** |  |  | 0.68 | 3601 |
| **macro avg** | 0.68 | 0.68 | 0.68 | 3601 |
| **weighted avg** | 0.68 | 0.68 | 0.68 | 3601 |

## SVC with BayesSearchCV Parameters

kernel='rbf', # Use a linear kernel for linear separation between classes

C=10.0, # Regularization parameter

gamma='0.01', # Control the influence of individual data points on the decision boundary

max\_iter=-1, # Set the maximum number of iterations (-1 = no limit, let it converge)

random\_state=42 # Set a random state for reproducibility

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **fit\_time** | **score\_time** | **test\_score** | **train\_score** |
| *0* | 0.839808 | 0.33744 | 0.639833 | 0.706597 |
| *1* | 0.867333 | 0.337835 | 0.638889 | 0.711161 |
| *2* | 0.852713 | 0.343906 | 0.65 | 0.70769 |
| *3* | 0.839532 | 0.343611 | 0.654861 | 0.704739 |
| *4* | 1.300707 | 0.330031 | 0.634028 | 0.709078 |

**Accuracy = 0.6792557622882532 ~ 67.9%**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| *Anime* | 0.73 | 0.71 | 0.72 | 457 |
| *Blues* | 0.57 | 0.58 | 0.57 | 415 |
| *Classical* | 0.83 | 0.82 | 0.82 | 478 |
| *Country* | 0.63 | 0.65 | 0.64 | 470 |
| *Electronic* | 0.6 | 0.61 | 0.61 | 435 |
| *Jazz* | 0.57 | 0.47 | 0.51 | 459 |
| *Rap* | 0.84 | 0.8 | 0.82 | 444 |
| *Rock* | 0.64 | 0.78 | 0.71 | 443 |
|  |  |  |  |  |
| *accuracy* |  |  | 0.68 | 3601 |
| *macro avg* | 0.68 | 0.68 | 0.68 | 3601 |
| *weighted avg* | 0.68 | 0.68 | 0.68 | 3601 |