Spotify Track Genre Classification

Universitat Politècnica de Catalunya
Facultad de Informática
Master Artificial Intelligence

Project Report
for "Machine Learning"

written by

Karl - Augustin Jahnel
karlaugustin.jahnel@estudiantat.upc.edu
Daniel Banov
daniel.banov@estudiantat.upc.edu

Feb 2024

Professor: Marta Arias Vicente marias@cs.upc.edu
Lab tutor: Ignasi Gómez Sebastià igomez@cs.upc.edu

Contents

1 Introduction

In this project, we aim to develop a machine learning model for the classification of music genres based on an extensive dataset of audio features. The primary goal is to accurately predict the genre of a given music track using a combination of audio attributes. The dataset includes a diverse collection of songs, each annotated with a specific genre label. Key features in the dataset encompass various aspects of the audio signal, such as danceability, energy, loudness, liveness, and speechiness, which collectively provide a rich representation of the musical content.

Our approach involves several steps, starting with data preprocessing to ensure the quality and consistency of the input features. Following this, we experiment with a range of classification algorithms, including traditional methods like Support Vector Machines and Random Forests, as well as more advanced approaches such as deep learning models.

The ultimate objective is to achieve high accuracy in genre classification, which has practical applications in enhancing music recommendation systems, automated playlist generation, and digital music categorization. By rigorously evaluating the performance of different models and fine-tuning hyperparameters, we strive to identify the optimal solution for this classification task. Our work not only contributes to the field of music information retrieval but also provides valuable insights into the distinct audio features that differentiate musical genres. Our code is published on github at `https://github.com/Barathaner/Spotify-Track-Genre-Classification.git` and our Report is accessible at `https://www.overleaf.com/read/ywzjnsvpbhws#585eae`. We used the Kaggle Dataset from `https://www.kaggle.com/datasets/mrmorj/dataset-of-songs-in-spotify`.

1.1 Data Set

The dataset used in this project is a comprehensive collection of music tracks from spotify, obtained by using the spotify api. Each song is annotated with several audio features such as:

- acousticness: A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic
- danceability: Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.
- duration_ms: The duration of the track in milliseconds.
- liveness: Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.
- time_signature: An estimated time signature. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure). The time signature ranges from 3 to 7 indicating time signatures of "3/4", to "7/4".

Those descriptions were taken from the Spotify API.
This rich set of features allows for a detailed analysis of the audio content, facilitating the development of accurate and reliable genre classification algorithms.

The dataset consists of over 40.000 different songs with 15 different genres, ranging from pop over hardstyle to Hiphop.

1.2 Aim

The aim of this project is to develop a machine learning model for the classification of music genres using a comprehensive dataset of audio features. This project serves a dual purpose: it provides us, as students, with a practical learning experience in applying machine learning techniques to a real-world problem, and it contributes to the field of music information retrieval. Through this project, we gain hands-on experience with data preprocessing, feature extraction, and the implementation of various classification algorithms, including both traditional methods and advanced deep learning models. Our ultimate goal is to achieve high accuracy in genre classification, which has practical applications in enhancing music recommendation systems, automated playlist generation, and digital music categorization. By rigorously evaluating and fine-tuning our models, we aim to identify the optimal solution for this task, thereby deepening our understanding of the distinct audio features that differentiate musical genres and improving the accuracy and efficiency of automated music classification systems.

2 Data Exploration Process

The dataset consists of 42.305 songs with 22 features.

To get a rough overview over the data we provided table 1 with key statistics of the numerical features. Other, string features are type, id, uri, track_href analysis_url, song_name, Unnamed: 0 and title as well as the feature to be predicted genre.

The genres that exist in the dataset are Dark Trap, Underground Rap, Trap Metal, Emo, Rap, RnB, Pop, Hiphop, techhouse, techno, trance, psytrance, trap, dnb and hardstyle. A graph showing the distribution of the genres is shown in figure 1

	danceability	energy	key	loudness	mode	speechiness	acousticness
mean	0.64	0.76	5.37	-6.47	0.55	0.14	0.10
std	0.16	0.18	3.67	2.94	0.50	0.13	0.17
min	0.07	0.00	0.00	-33.36	0.00	0.02	0.00
25%	0.52	0.63	1.00	-8.16	0.00	0.05	0.00
50%	0.65	0.80	6.00	-6.23	1.00	0.08	0.02
75%	0.77	0.92	9.00	-4.51	1.00	0.19	0.11
max	0.99	1.00	11.00	3.15	1.00	0.95	0.99

	instrumentalness	liveness	valence	tempo	e duration_ms	time_signature
mean	0.28	0.21	0.36	147.47	250865.85	3.97
std	0.37	0.18	0.23	23.84	102957.71	0.27
min	0.00	0.01	0.02	57.97	25600.00	1.00
25%	0.00	0.10	0.16	129.93	179840.00	4.00
50%	0.01	0.14	0.32	144.97	224760.00	4.00
75%	0.72	0.29	0.52	161.46	301133.00	4.00
max	0.99	0.99	0.99	220.29	913052.00	5.00

Table 1: Stats

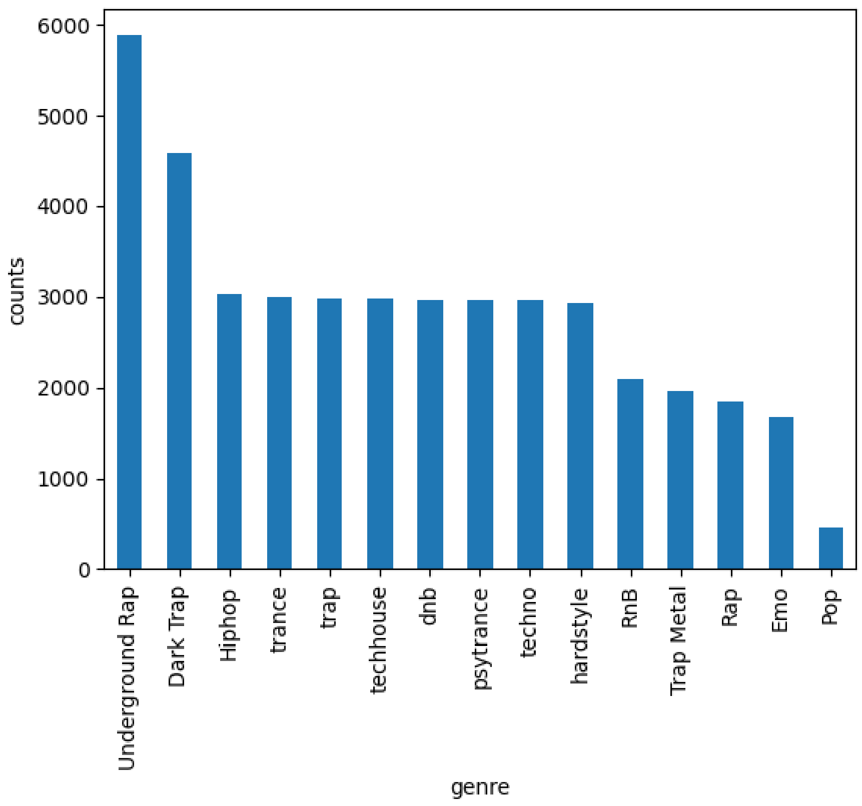


Figure 1: Stats

2.1 Pre-processing

In the first step of preprocessing we omitted the string features id, uri, track_href analysis_url, song_name, Unnamed: 0 and title. The features id, uri, track_href and analysis_url were not giving any information about the song and were only technical. The feature Unnamed: 0 was completely empty and the features title and song_name were too high dimensional to process are often not related to the genre. After looking at the correlation we found the highest correlation was between the features energy and loudness with 0.6 and between duration_ms and instrumentalness also with 0.6, but thought those coefficients are not high enough to directly omit one of the features.

We continued by plotting the distributions and outliers to the features and found some that look normal distributed, see figure 2

Calculated how many outliers there are for different features with the quantile method and found 278, 532, 796, 45, 5149, 1361 outliers for duration_ms, tempo, loudness, danceability, acousticness, liveness respectively. Since we didn't want to omit examples without being able to explain why, we looked at the class distribution before and after omiting outliers and found that omiting the outliers of duration_ms made a very small relative impact in the class distribution but made the distribution of duration_ms way better.

What was also notable that for the time signature, there were very few examples that hadn't a signature of 4/4 th, indicated by the value 4. We looked at the class distribution after omiting examples with a different time signature and found out that the relative impact per class was about the same, so it would making the database smaller. this also told us that that feature probably doesn't tell a lot about the class so we didn't omit any example but removed the feature time_signature

Afterwards we used a MinMaxScaler for the features that weren't in the range of 0-1, which were key, loudness, tempo and duration_ms.

Also for prelimanry measure we added a one-hot encoding for the genres.

After all of this, we were left with the features danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo and duration_ms and had 42.027 datapoints

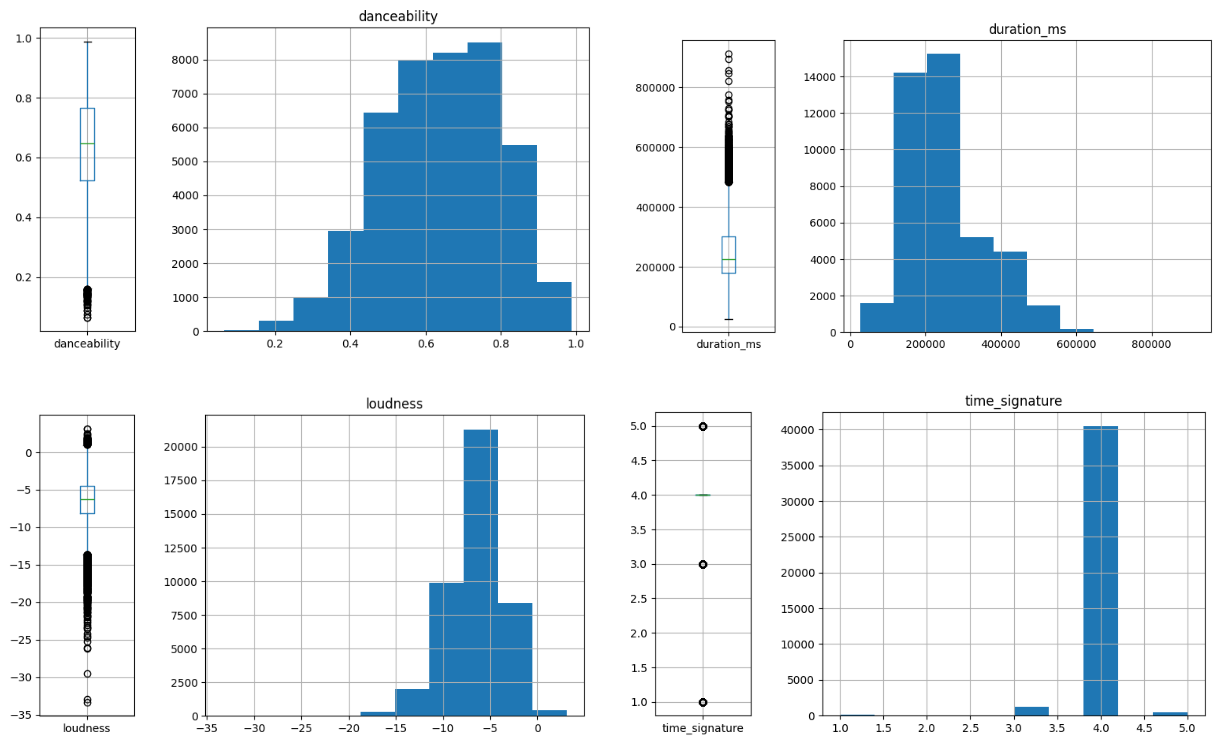


Figure 2: feature distributions

2.2 Feature Selection

One of the primary advantages of using Random Forest for feature selection is its intrinsic method of evaluating feature importance. This is achieved during the tree-building process where each split of a node is made based on a specific feature that maximizes the decrease in impurity (such as Gini impurity or entropy in classification tasks). After the forest is built, the importance of each feature can be measured by averaging the decrease in impurity across all trees where the feature is used. We did this and got this as our importance for our features:

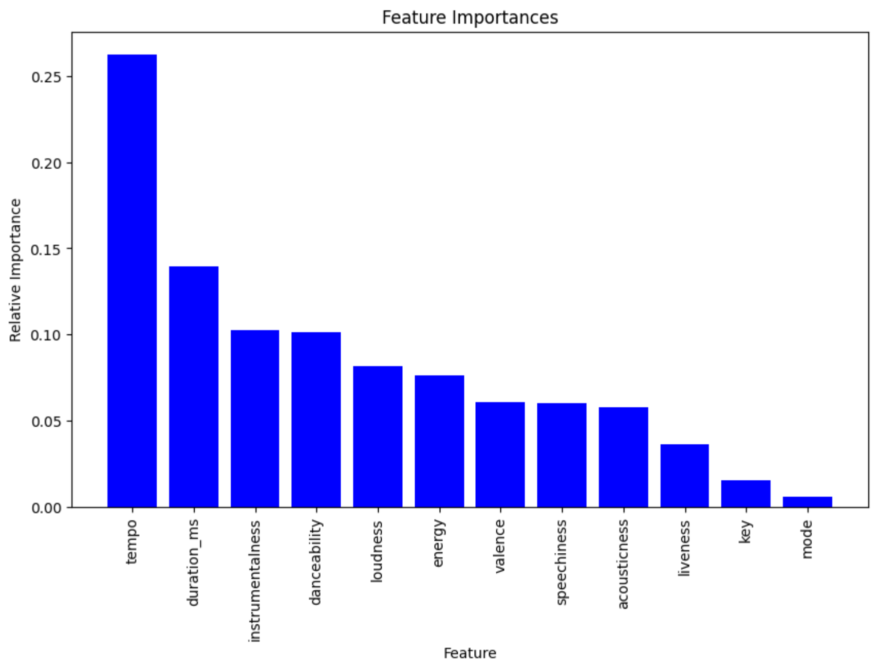


Figure 3: Feature importance from RF

As we can see feature key and mode are under 5% of importance so we decided to delete these features since it is a big reduction in complexity and a very small loss of importance.

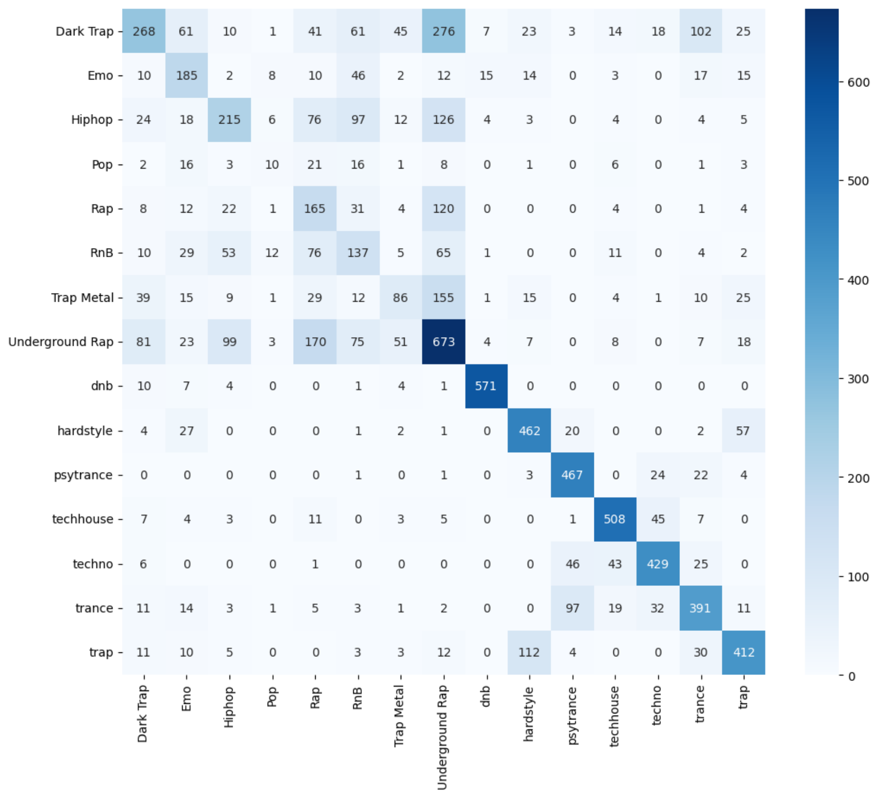
3 Modeling methods

In this section we explain the methods we chose, why we chose them and how we made their parameter tuning aswell as their ability to generalize. Our method is as follows. We picked the most popular classifier and explained their advantages and disadvantages for our specific spotify track genre classification. We then looked for the parameter that had the most influence on the F1-score result for the test data. Then we set up a 5 Fold Cross validation Grid search and let it run to get the best fit for the classifier on our preprocessed dataset. When it was possible we included helpful visualizations and explainations for the best configuration on the result and tried to interprate them in terms of computational cost, accuracy, interpretability, overfitting and scalability aswell as flexibility.

3.1 Bayes

Naive Bayes classifiers are popular due to their simplicity, efficiency, and proven track record in text categorization and spam filtering. These characteristics make them appealing for classification tasks where interpretability and computational efficiency are desired alongside reasonable accuracy.

We made a test/train split of 80/20 and trained a Gaussian Naive Bayes Classifier. For hyperparameters we used a 5-Fold cross validation for smoothing parameters from $10^{-9}, 10^{-8}, ..., 10^{-0}$ and with or without priors, and found with f1-weighting that using priors and a smoothing parameter of 10^{-5} gave the best results. We used f1-weighted scoring since the classes are very differently distributed. The weighted f1-score was 0.58. The confusion matrix can be found in figure 4 This indicates a reasonable balance between precision and recall across the diverse set of genres.



	Dark Trap	Emo	Hiphop	Pop	Rap	RnB	Trap Metal	Underground Rap	dnb	hardstyle	psytrance	techhouse	techno	trance	trap
Dark Trap	268	61	10	1	41	61	45	276	7	23	3	14	18	102	25
Emo	10	185	2	8	10	46	2	12	15	14	0	3	0	17	15
Hiphop	24	18	215	6	76	97	12	126	4	3	0	4	0	4	5
Pop	2	16	3	10	21	16	1	8	0	1	0	6	0	1	3
Rap	8	12	22	1	165	31	4	120	0	0	0	4	0	1	4
RnB	10	29	53	12	76	137	5	65	1	0	0	11	0	4	2
Trap Metal	39	15	9	1	29	12	86	155	1	15	0	4	1	10	25
Underground Rap	81	23	99	3	170	75	51	673	4	7	0	8	0	7	18
dnb	10	7	4	0	0	1	4	1	571	0	0	0	0	0	0
hardstyle	4	27	0	0	0	1	2	1	0	462	20	0	0	2	57
psytrance	0	0	0	0	0	1	0	1	0	3	467	0	24	22	4
techhouse	7	4	3	0	11	0	3	5	0	0	1	508	45	7	0
techno	6	0	0	0	1	0	0	0	0	0	46	43	429	25	0
trance	11	14	3	1	5	3	1	2	0	0	97	19	32	391	11
trap	11	10	5	0	0	3	3	12	0	112	4	0	0	30	412

Figure 4: Stats

The confusion matrix shows substantial correct classifications along the diagonal, but there are notable misclassifications, suggesting areas where the model may confuse one genre with another. This kind of insight is valuable for understanding the model's limitations and can guide further improvements in feature engineering or model selection. Naive Bayes also gives us insights about the average probability of each class:

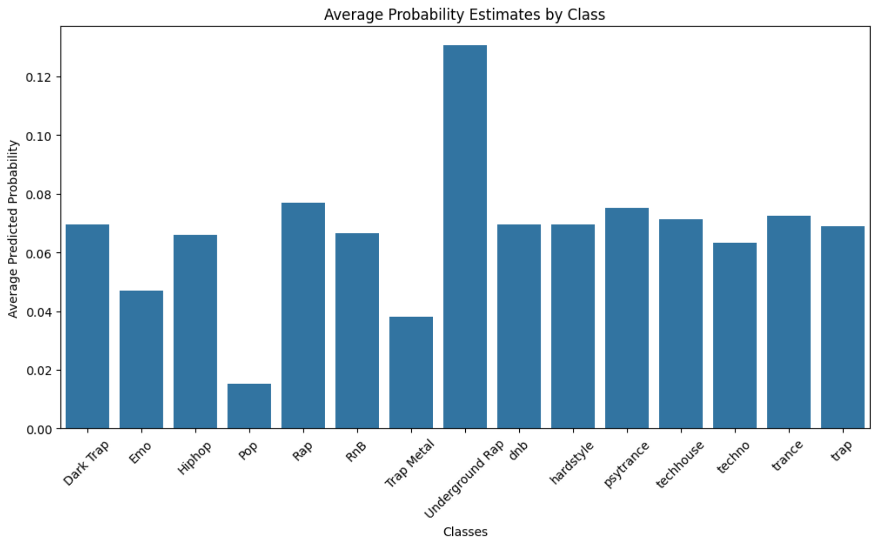


Figure 5: Average probability of NB

Gaussian Naive Bayes is inherently scalable and computationally efficient, making it suitable for larger datasets. However, the simplicity of the model may limit its effectiveness in capturing complex patterns necessary for high accuracy in multi-class settings like genre classification.

3.2 Random Forests

In our exploration of ensemble methods for Spotify track genre classification, we employed the Random Forest classifier due to its robustness and accuracy in handling complex data. It was discovered by Breiman (2001). The optimization process involved an extensive grid search over 432 combinations, taking approximately 10 hours. The optimal settings were found to be {'bootstrap': True, 'max_depth': 30, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators': 200}.

This configuration emphasizes the model's balance between exploration of the feature space and prevention of overfitting, with a depth that allows for detailed feature interactions without being overly complex. The result was a weighted F1-score of 0.65, indicating effective classification performance across the diverse set of genres.

To further get insight into how the features affect the classes we could build a Partial Dependency Plot for some class and some features. Here is an example for Dark Trap class and two features.As "danceability" increases from 0.4 to around 0.7, the partial dependence decreases slightly, suggesting that higher danceability slightly reduces the probability of a track being classified as "Dark Trap." Beyond 0.7, the plot shows a more pronounced decrease in partial dependence, indicating that as danceability significantly increases, the likelihood of a track being "Dark Trap" markedly decreases.

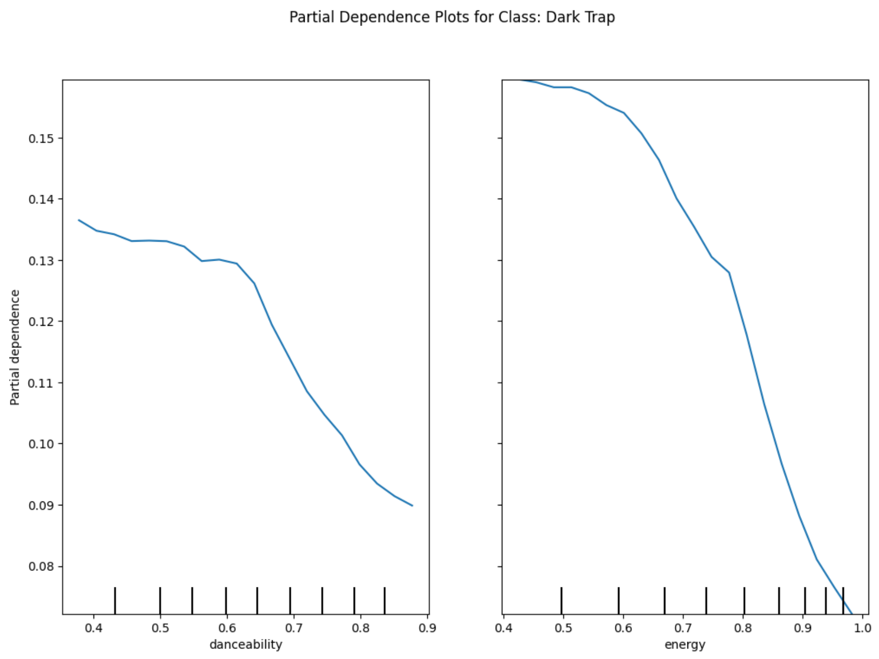


Figure 6: Partial Dependency Plot for Dark Trap with RF

The corresponding confusion matrix, depicted in Figure 7, provides a visual evaluation of the model's performance, highlighting strengths and areas for potential improvement across different genres.

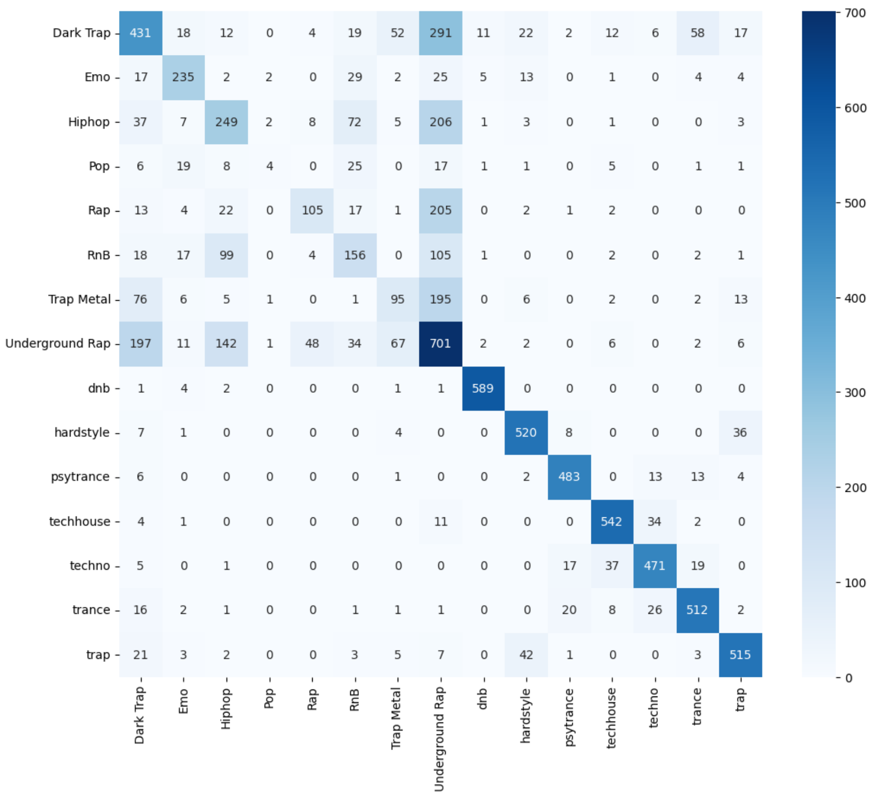


Figure 7: Stats

Overall, the Random Forest model has proven to be highly effective for this classification task, offering a substantial improvement over simpler models due to its ability to model complex patterns and relationships within the data.

3.3 KNN

In our analysis of classification methods for Spotify track genre classification, we utilized the K-Nearest Neighbors (KNN) classifier, valued for its simplicity and effectiveness. The optimization included a parameter grid search assessing different numbers of neighbors, distance metrics, and weighting schemes. This process was relatively quick, taking about 2 minutes, and identified the optimal parameters as {'metric': 'manhattan', 'n_neighbors': 11, 'weights': 'distance'}.

These settings highlight the importance of the distance metric in KNN's performance, with 'manhattan' proving effective for our high-dimensional feature space by emphasizing differences in individual dimensions. The use of 'distance' weights means that nearer neighbors influence the prediction more than those farther away, which is particularly useful for music genres where closer data points likely share more similarities.

The performance, with a weighted F1-score of 0.55, indicates a moderate level of accuracy and balance between precision and recall, considering the difficulty of the task and the diversity of genres.

The confusion matrix, presented in Figure 8, illustrates the classifier's performance across the genres, showing both successes and areas for potential enhancement.

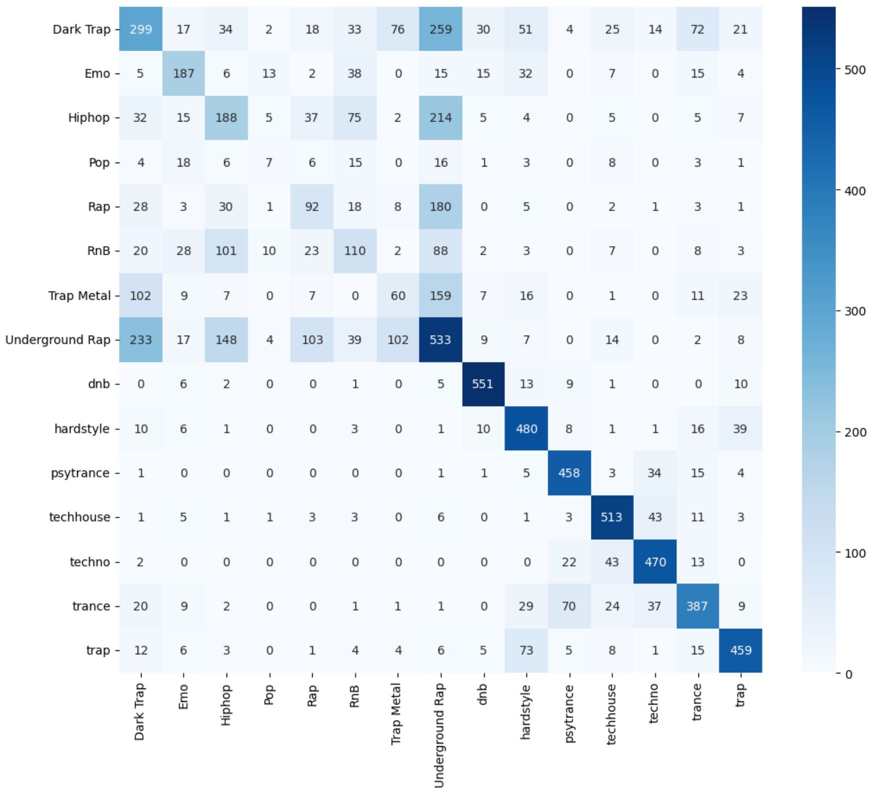


Figure 8: Stats

The KNN classifier demonstrates considerable utility for genre classification, providing a baseline for comparison with more complex models and offering insights into the nature of our dataset's feature relationships.

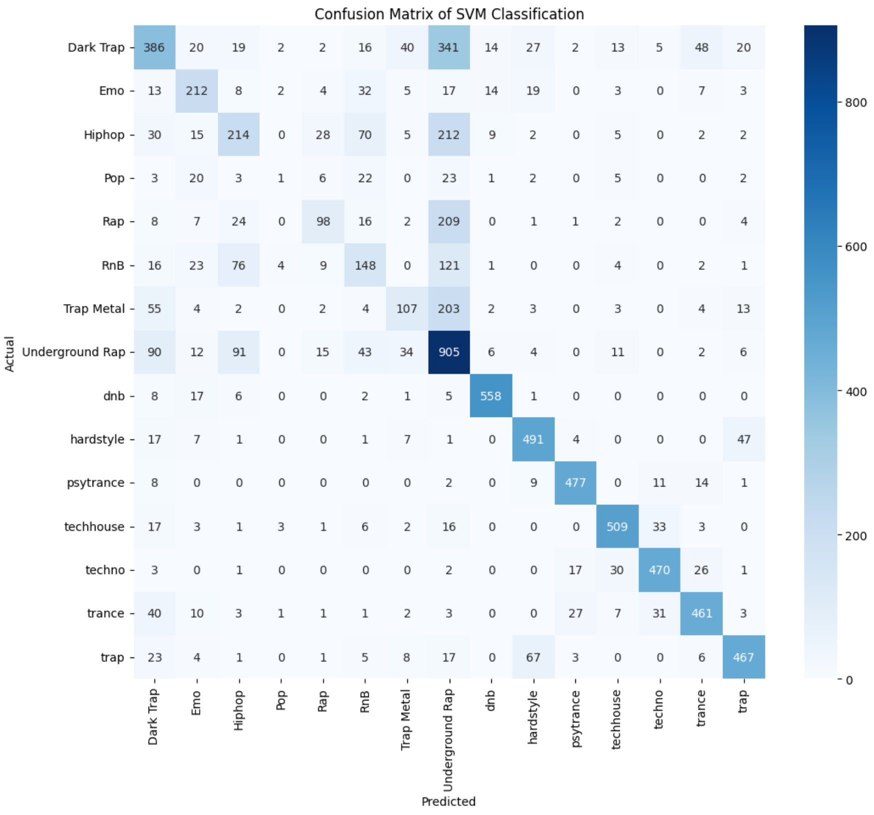
3.4 Support Vector Machines

As another candidate we chose SVMs. SVMS solve classification tasks by constructing hyperplanes that seperate the data points optimally. The hyperplane is such that it maximizes the margin between the nearest data points of the different classes. For the SVMs the input data needs to be scaled. We already did this in the preprocessing. SVMs should fit very well to music genre classification for the following reasons: SVMs can handle high dimensional data well without falling for the curse of dimensionality. We have a lot of features and therefore alot dimensions. SVMs are also quiet robust to overfitting and we also can try out different kernels to capture more complex relationships, since genres are probably not linearly seperable since music genres can blend in one another.

We tried grid search on a train (80% of data) and test split (20% of data) on the following grid of hyperparameters for 5-Fold cross validation. This results in 180 fits. It took 90min to run the grid search:

- C (Regularization parameter):0.1, 1, 10
- Gamma (Kernel Coefficient): 0.001, 0.01, 0.1, 1
- Kernel: rbf, poly, sigmoid

We decided for this grid since we want to test out different margins as regularization, different kernel coefficient since we want to test out how far the influence of a single training example reaches so basically the smoothness of the kernel and we wanted to test out some non linear kernels. The best hyperparameter configuration is : 'C': 10, 'gamma': 1, 'kernel': 'poly' with an F1-Score of 0.653. The result seems very promising. This is the confusion matrix. We see compared to the other ones, that we have alot less misclassifications.



Actual \ Predicted	Dark Trap	Emo	Hiphop	Pop	Rap	RnB	Trap Metal	Underground Rap	dnb	hardstyle	psytrance	techhouse	techno	trance	trap
Dark Trap	386	20	19	2	2	16	40	341	14	27	2	13	5	48	20
Emo	13	212	8	2	4	32	5	17	14	19	0	3	0	7	3
Hiphop	30	15	214	0	28	70	5	212	9	2	0	5	0	2	2
Pop	3	20	3	1	6	22	0	23	1	2	0	5	0	0	2
Rap	8	7	24	0	98	16	2	209	0	1	1	2	0	0	4
RnB	16	23	76	4	9	148	0	121	1	0	0	4	0	2	1
Trap Metal	55	4	2	0	2	4	107	203	2	3	0	3	0	4	13
Underground Rap	90	12	91	0	15	43	34	905	6	4	0	11	0	2	6
dnb	8	17	6	0	0	2	1	5	558	1	0	0	0	0	0
hardstyle	17	7	1	0	0	1	7	1	0	491	4	0	0	0	47
psytrance	8	0	0	0	0	0	0	2	0	9	477	0	11	14	1
techhouse	17	3	1	3	1	6	2	16	0	0	0	509	33	3	0
techno	3	0	1	0	0	0	0	2	0	0	17	30	470	26	1
trance	40	10	3	1	1	1	2	3	0	0	27	7	31	461	3
trap	23	4	1	0	1	5	8	17	0	67	3	0	0	6	467

Figure 9: Confusion Matrix for SVM

3.5 Logistic Regression

Logistic regression models are highly interpretable, as each coefficient in the model can directly show the effect of increasing or decreasing a particular feature on the likelihood of a track belonging to a certain genre. It can provide probabilities that indicate how likely a track is to belong to each genre. This could give valuable insights of a track. If the relationship between features and the genre is approximately linear, logistic regression can perform exceptionally well with much less computational cost than more complex models. For 150 fits with 2000 maximum iterations until convergence it took around 7minutes. This gives a good first baseline in terms of first insight for a ML Problem. Since genre classification assumes a non linear seperation LR will probably struggle to model it. LR is also inherently binary, this is why in Pthon it uses One versus Rest for this Multi classification problem.

Our procedure to find the best fitting models is as follows: Train Test split in 80% Train and 20% Test. 5fold Cross Validation for the Grid search for the following Hyperparameter Grid. We used Maximum iterations of 2000 since some converged around 1500:

- C (Regularisation Parameter): 0.01, 0.1, 1, 10, 100
- penalty(Lasso or Ridge to reduce overfitting): 'none', 'l2','l1'
- solver(algorithm to optimize this problem):'sag', 'lbfgs'

The best model for our problem was with this configuration: 'C': 0.01, 'maxiter': 2000, 'penalty': 'none', 'solver': 'sag' These hyperparameter settings suggest a strategy that seeks to optimize the logistic regression model by allowing more iterations for a potentially complex decision boundary in a large dataset, while opting not to limit the model's flexibility with regularization. This approach aims to capture the nuances in the data as effectively as possible, which is essential in accurately classifying the diverse and complex categories of music genres. It achieves an Accuracy: 0.5768.

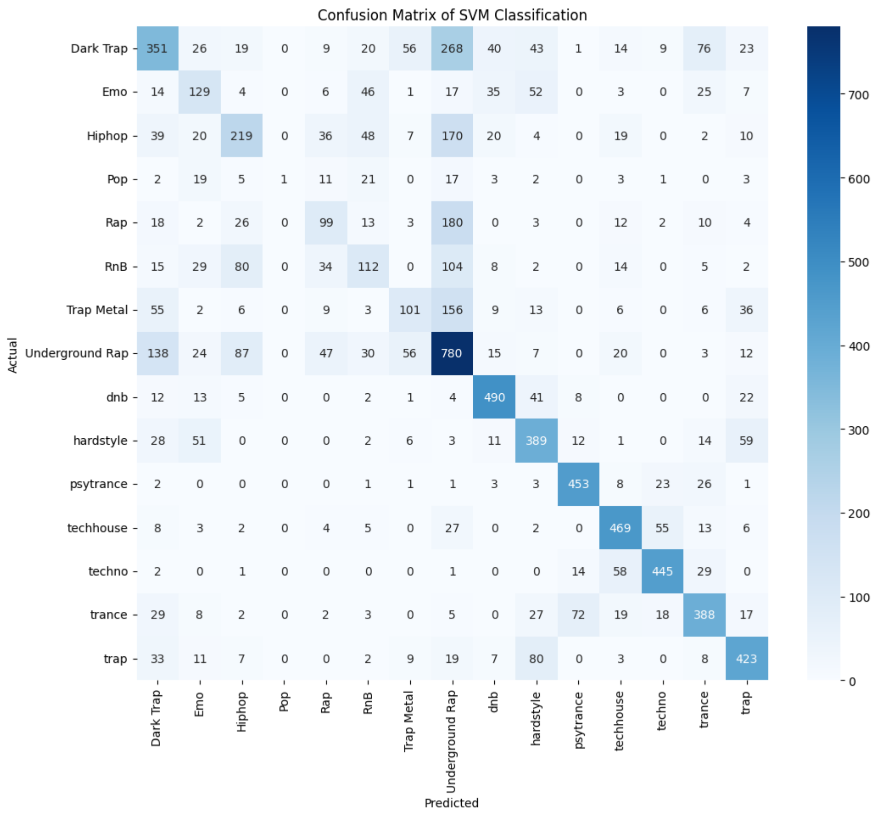


Figure 10: Confusion Matrix for LR

We can see that it achieves comparable results to the other classifier. In the following we can see the probability estimates and the coefficients for the first class. This gives us insights about how the features affect the decision of the classes. for example we see that tempo, speechiness and acousticness are positive correlated features for hip hop while duration is negatively correleated to hip hop. In the Probability estimates we can conclude, that there is a high density for 0.0 probability. this means the model is very sure that alot of tracks do not belong to hip hop. Then it falls of in a normal distribution manner. since the peak is in the small probabilities it says us, that the model knows really good what is not a hip hop track or that hip hop is underrepresented.

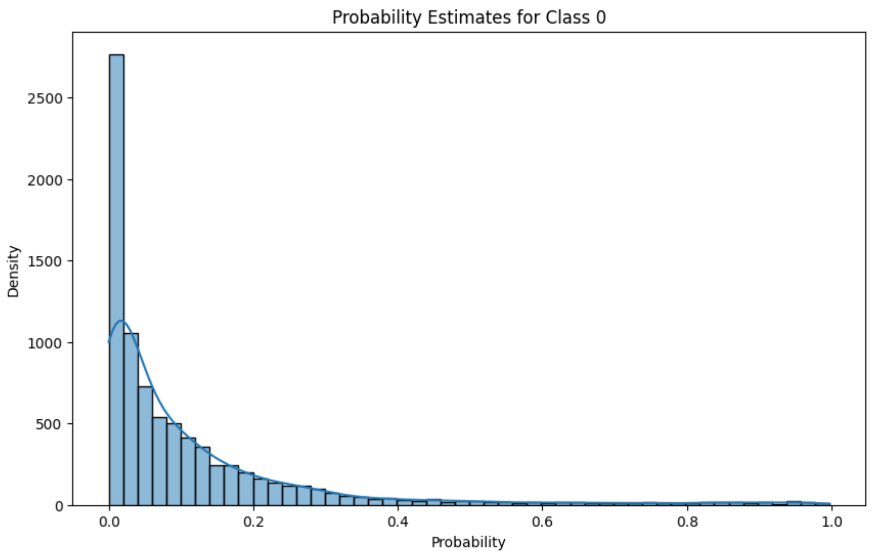


Figure 11: Probability estimates for class HipHop in LR

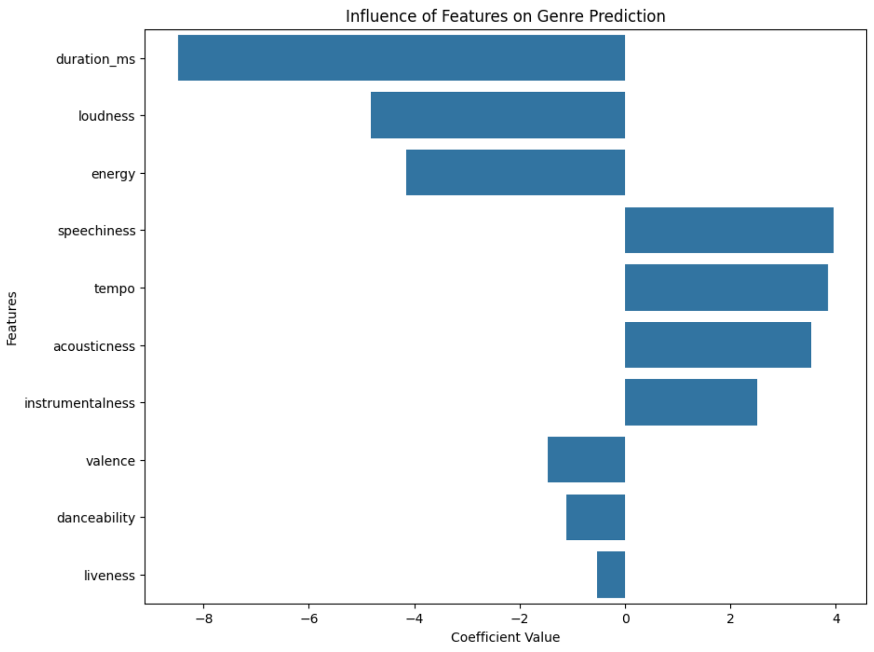


Figure 12: feature influence for HipHop for LR

In conclusion Logistic regression is very good as a first baseline to get a quick overview of your data and how the features affect the classes and a good estimate of accuracy.

3.6 Multi Layer Perceptron - Neural Networks

Multi-Layer Perceptrons (MLPs) are adept at capturing complex, non-linear relationships within datasets, crucial for handling intricate musical features such as tempo, rhythm, instrumentation, and harmony. Unlike simpler models that assume feature independence, MLPs effectively learn interactions between features, making them invaluable for music genre classification where interactions like pitch and tempo often define genres. Their high customizability allows significant flexibility in modeling, though this also introduces challenges like overfitting and high computational demands.

Our initial model configuration was an MLP with a single hidden layer:

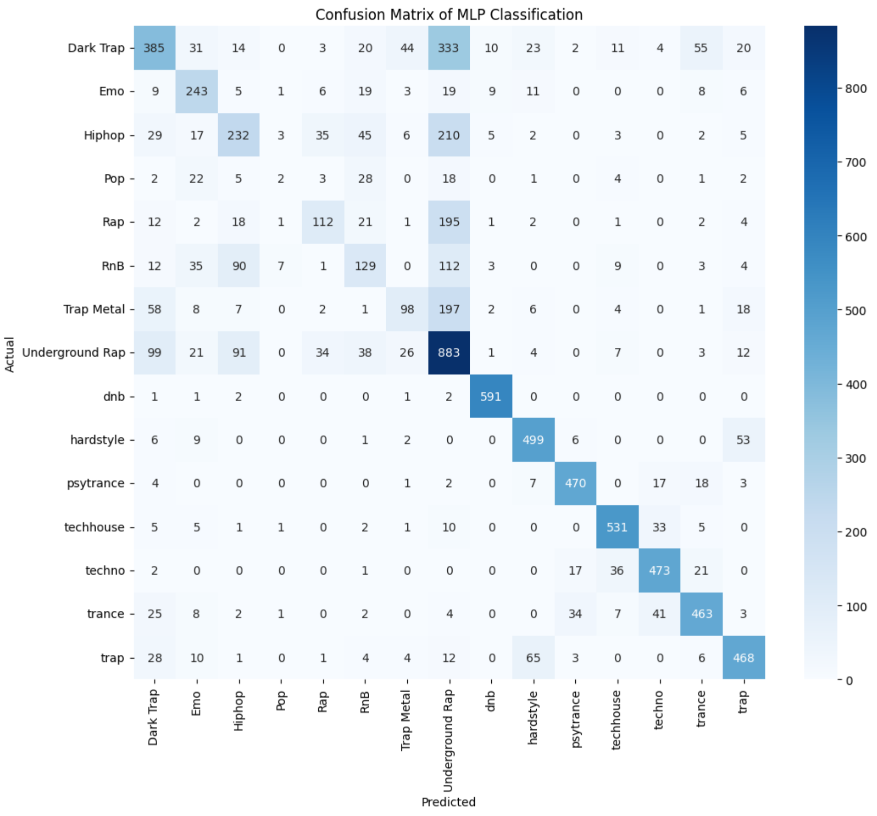
```
MLPClassifier(hidden_layer_sizes=(100,), max_iter=500, activation='relu', solver='ada
```

This baseline model, despite its simplicity, achieved an F1-score of 0.6743 but was slow to train, highlighting the computational expenses associated with MLPs.

To explore MLP capabilities further, we experimented with various architectures, adjusting layer complexity and size to gauge their impact on performance:

- **Hourglass Architecture** (`hidden_layer_sizes=(20,40,20), max_iter=500`): Designed to compress and expand feature representation, enhancing the model's ability to generalize. It achieved an F1-score of 0.6584 in 1 minute and 33 seconds.
- **Deep Architecture** (`hidden_layer_sizes=(128, 64, 32)`): Intended to capture deeper and more nuanced interactions, reaching an F1-score of 0.6743 in 12 minutes.
- **Reduced Deep Architecture** (`hidden_layer_sizes=(64, 32, 16)`): A scaled-down version to investigate potential overfitting, securing an F1-score of 0.664 in 8 minutes.
- **Wide Architecture** (`hidden_layer_sizes=(64, 64, 64)`): Focused on parallel learning pathways, suitable for datasets with diverse features, achieving the highest F1-score of 0.6748 in about 10 minutes.

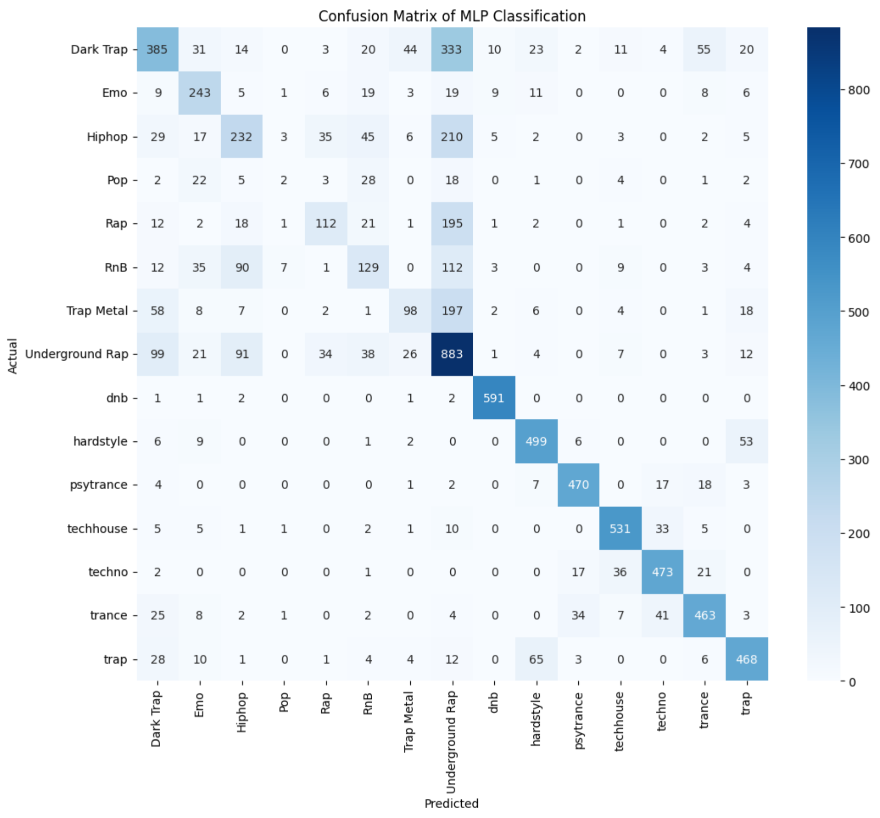
The performance of the `hidden_layer_sizes=(64, 64, 64)` architecture is visualized in the confusion matrix below, illustrating its classification performance across various music genres:



Actual \ Predicted	Dark Trap	Emo	Hiphop	Pop	Rap	RnB	Trap Metal	Underground Rap	dnb	hardstyle	psytrance	techhouse	techno	trance	trap
Dark Trap	385	31	14	0	3	20	44	333	10	23	2	11	4	55	20
Emo	9	243	5	1	6	19	3	19	9	11	0	0	0	8	6
Hiphop	29	17	232	3	35	45	6	210	5	2	0	3	0	2	5
Pop	2	22	5	2	3	28	0	18	0	1	0	4	0	1	2
Rap	12	2	18	1	112	21	1	195	1	2	0	1	0	2	4
RnB	12	35	90	7	1	129	0	112	3	0	0	9	0	3	4
Trap Metal	58	8	7	0	2	1	98	197	2	6	0	4	0	1	18
Underground Rap	99	21	91	0	34	38	26	883	1	4	0	7	0	3	12
dnb	1	1	2	0	0	0	1	2	591	0	0	0	0	0	0
hardstyle	6	9	0	0	0	1	2	0	0	499	6	0	0	0	53
psytrance	4	0	0	0	0	0	1	2	0	7	470	0	17	18	3
techhouse	5	5	1	1	0	2	1	10	0	0	0	531	33	5	0
techno	2	0	0	0	0	1	0	0	0	0	17	36	473	21	0
trance	25	8	2	1	0	2	0	4	0	0	34	7	41	463	3
trap	28	10	1	0	1	4	4	12	0	65	3	0	0	6	468

Figure 13: Confusion matrix for the MLP with architecture (64, 64, 64).

This systematic exploration and testing of various MLP architectures have provided valuable insights into the trade-offs between model complexity, training time, and classification performance, guiding further refinements in our approach to genre classification using deep learning techniques.



	Dark Trap	Emo	Hiphop	Pop	Rap	RnB	Trap Metal	Underground Rap	dnb	hardstyle	psytrance	techhouse	techno	trance	trap
Dark Trap	385	31	14	0	3	20	44	333	10	23	2	11	4	55	20
Emo	9	243	5	1	6	19	3	19	9	11	0	0	0	8	6
Hiphop	29	17	232	3	35	45	6	210	5	2	0	3	0	2	5
Pop	2	22	5	2	3	28	0	18	0	1	0	4	0	1	2
Rap	12	2	18	1	112	21	1	195	1	2	0	1	0	2	4
RnB	12	35	90	7	1	129	0	112	3	0	0	9	0	3	4
Trap Metal	58	8	7	0	2	1	98	197	2	6	0	4	0	1	18
Underground Rap	99	21	91	0	34	38	26	883	1	4	0	7	0	3	12
dnb	1	1	2	0	0	0	1	2	591	0	0	0	0	0	0
hardstyle	6	9	0	0	0	1	2	0	0	499	6	0	0	0	53
psytrance	4	0	0	0	0	0	1	2	0	7	470	0	17	18	3
techhouse	5	5	1	1	0	2	1	10	0	0	0	531	33	5	0
techno	2	0	0	0	0	1	0	0	0	0	17	36	473	21	0
trance	25	8	2	1	0	2	0	4	0	0	34	7	41	463	3
trap	28	10	1	0	1	4	4	12	0	65	3	0	0	6	468

Figure 14: Confusion Matrix for MLP

So in general we can conclude for MLP that in terms of computational cost it is the worst. But in terms of accuracy and flexibility it is the best.

4 Final model chosen and Scientific Conclusions

In this section we propose our ranking of models based on their use case. As Conclusion from our results. We give a short explaination of the ranking. We divided it into these main points: F1-Score (1st has the highest)as the main focus or Interpretability(1st gives the most insights) or computational cost(1. place is the most efficient.less is better). We went from best to worst.

1. MLP 0.67
2. SVM 0.653
3. RF 0.65
4. NB 0.58
5. LR 0.577
6. KNN 0.55

Figure 15: F1-Score Ranking

The MLP model tops our list in F1-Score, indicating the best overall performance among the classifiers for our dataset, particularly in balancing precision and recall effectively.

1. NB
2. LR
3. RF
4. SVM
5. KNN
6. MLP

Figure 16: Interpretability Ranking

Naive Bayes ranks the highest in terms of interpretability, offering straightforward probabilistic insights, which are easier to understand and explain than those from more complex models like MLPs and SVMs.

1. NB
2. LR
3. KNN
4. RF
5. SVM
6. MLP

Figure 17: Computational Cost

In terms of computational cost, Naive Bayes is also the most efficient, requiring less computational resources due to its simpler probabilistic calculations, followed by logistic regression which also tends to be computationally less demanding than models like MLP or SVM that require intensive training procedures.

These rankings help us determine the most appropriate model based on specific needs: whether the priority is performance, interpretability, or efficiency. While MLPs offer the best performance, they do so at the cost of higher computational demand and lower interpretability, making Naive Bayes a preferable option when ease of understanding and efficiency are paramount. But personally we would suggest using Logistic regression as first baseline, since you can generate very helpful insights to the features and their effect on every class.

5 Personal Conclusions

This project helped us to apply a wide variety of machine learning algorithms to a real world problem and get a feeling on when to use what algorithm and its advantages and disadvantages. We were surprised of the range of accuracy for hyperparameter tuning and the variety of visualizations that scikit learn offers and the statistics that are already implemented in the libraries. We also got a feeling of the complexity and computaional cost of every model and we feel quiet comfortable in applying them to the next problem.

6 Possible extensions and known limitation

We know that overall the F1-Score is not useful for a real world application since there are still too many missclassifications. the best we could achieve was 0.67 with MLPs so we suggest to take a deeper look into this topic if the accruacy is the main point of the application. There would be different architectures of the hidden layer to try. We suggest applying deep learning architectures on this problem. Since it is a classification problem where features interact with each other we would suggest taking a Convolutional Neural Network as Baseline and go from simple to more complex while adding more layers and increasing dimensions. We also think that it would be better to identify very similiar genres and combine them to a general genre since there are alot of msclassifications. For this there would be a threshold feature selection and CLustering Analysis needed. Other extensions and suggestions would be Using AutoML pipelines from google to identify the best ML model for this problem. Another source of knowledge could be the kaggle competition for this topic and therefore to study the best code examples and look for improvements that coudl be combined in this codebase. Also to organize the code it would be a good idea to implement it as a kind of AutoML Framework or library in the future.

References

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.