Spotify Music

Multi-class Genre Classification

Annamária Vass, Jannik Hösch

June 3, 2024

1 Introduction

In our machine learning project, we're diving into the world of predicting music genres using Spotify's song dataset. This choice isn't just random, it's based on our past experience dealing with clustering and grouping data. We're excited to explore this dataset because it's not only interesting but also everybody listens to music. Through this project, we aim to further our understanding of various machine learning algorithms and concepts taught in this course.

The dataset selected for this project is available on Kaggle [1]. It comprises various attributes of songs on Spotify like danceability, energy, key and loudness, and it's genre, making it a good choice for developing a genre classification model. The dataset contains 35877 samples of songs with 22 variables. Most of the variables are numerical (e.g. *danceability*, *loudness*, *tempo*, *speechiness*), some are categorical (e.g. *key*, *time_signature*) and one is binary (*mode*). The target value *genre* is categorical and has 15 unique values. A detailed description of the dataset is available online [2].

We are going to build a model that should be able to accurately determine the genre of a song based on its musical attributes. Our goal is to find the best performing model utilizing and optimizing all machine learning methods that were introduced during the course. It will be interesting to see if we can build a relationship between objective song attributes and the subjective perception of genres.

2 Related Work

Previous work on this dataset has been extensively documented on platforms such as Kaggle and Data Science-themed blogs. One notable project on Analytics Vidhya achieved an impressive accuracy of 88% in predicting genres by employing a neural network model. [3]

Another good example is a Kaggle notebook that explores different clustering methods to group the dataset's data points. This notebook demonstrates the application of clustering algorithms to uncover inherent patterns within the dataset.[4]

Additionally, several other notebooks have been released on Kaggle, experimenting with machine learning techniques like Random Forests, Support Vector Machines, and Gradient Boosting Machines. These works collectively provide a understanding of the potential and challenges associated with genre classification using the Spotify dataset and helped us developing our own best model for this task.[5]

[1] https://www.kaggle.com/datasets/mrmorj/dataset-of-songs-in-spotify

[2] https://docs.google.com/document/d/1LWl88F8wGY1WkkOSzzVzwSYij1yeMR8hFXllRpkMyrI

[3] https://www.analyticsvidhya.com/blog/2023/03/solving-spotify-multiclass-genre-classification-problem/

[4] https://www.kaggle.com/code/qilinchu/spotify-songs-clustering

[5] https://www.kaggle.com/datasets/mrmorj/dataset-of-songs-in-spotify/code

3 Data Exploration

One of the first steps in our project was to explore the composition and properties of our dataset. This helped us to better understand what models we would need and to interpret their underlying performance.

3.1 Target Variable

The very first step was to examine the target variable, which is basically the genre of the song. We can see that initially there are 15 different genres in our dataset, these are *Underground Rap, Dark Trap, Hiphop, Trance, Trap, Techhouse, Dnb, Psytrance, Techno, Hardstyle, RnB, Trap Metal, Rap, Emo*, and *Pop*. It is observable, that we have some similar genres, which later can be grouped, so we have fewer target variables, and it is easier to find a good model. For instance, *Rap* and *Underground Rap* can be grouped later.

If we take a look at the distribution of the data (Figure 1), it is obvious that some genres are underrepresented compared to others, for example, *Pop* and *Emo*. Later we will have to handle this problem, as imbalanced data can cause problems regarding the performance of some models.

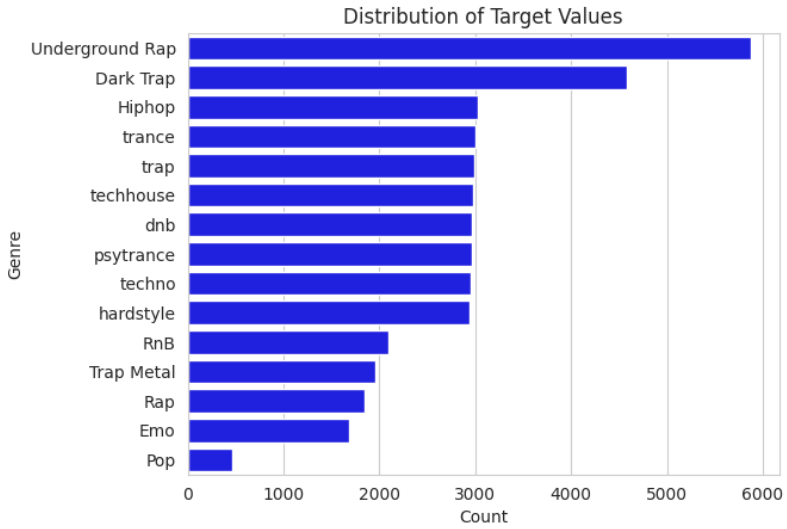


Figure 1: The distribution of the target variable. *Pop* is really underrepresented, while *Underground Rap* has the most data instances

3.2 Exploring the Numerical Features

The numerical features of our dataset are *danceability, energy, loudness, speechiness, acousticness, instrumentalness, liveness, valence, tempo*, and *duration_ms*. During the Exploratory Data Analysis we took a closer look at these variables.

First, we examined the distribution of each column, with the help of the histograms (Figure 2), and most of them are not Gaussian, as *loudness* is the only one. The *speechiness* feature shows a highly right-skewed distribution. Most values are clustered near 0.05. Similarly, *acousticness* also displays a right-skewed distribution, with most values near 0.0., and *instrumentalness* also peaks at 0. This indicates that a majority of songs do not contain features like speech, instrumentals, or acoustic elements. Speaking of the other features as well, *energy* and *liveness* are right-skewed, *danceability* and *tempo* have multimodal, and *valence* has uniform distribution. We observe that the scales of the features aren't uniform, so later we will have to make sure that all variables have similar scaling to improve our models performance.

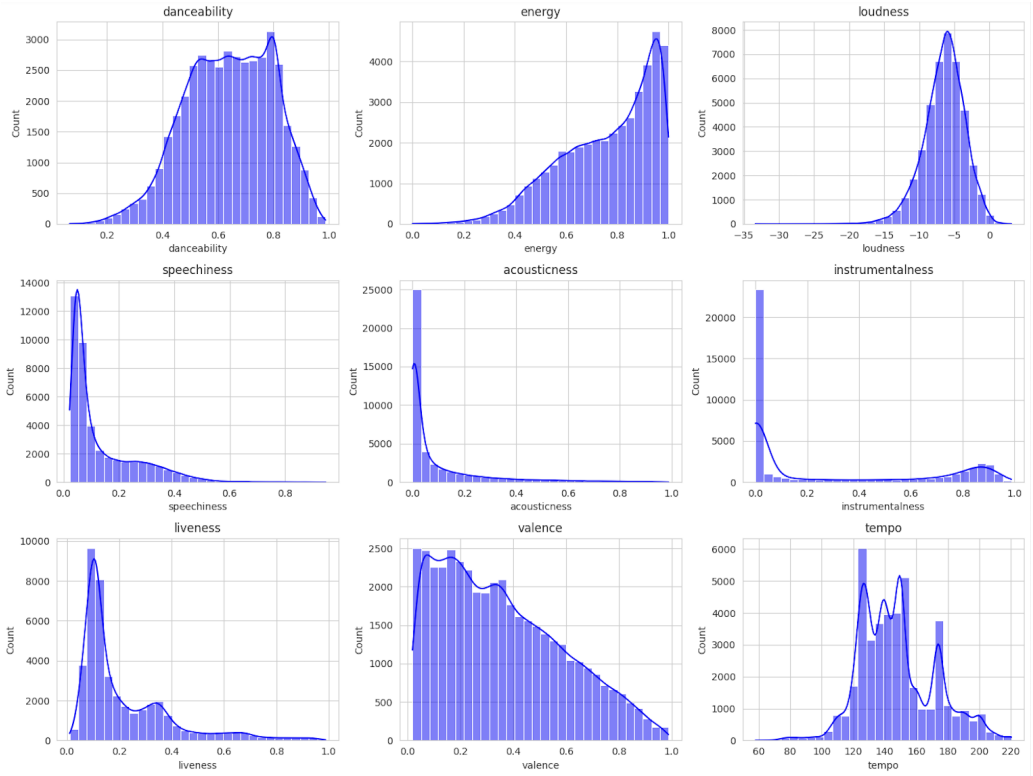


Figure 2: The distribution of numerical features. *acousticness, instrumentalness*, and *speechiness* mostly have values around 0.

Furthermore, for each feature, we can examine the average value related to each genre (Figure 3). *danceability*, *energy*, *loudness* and *liveness* levels are relatively uniform throughout the genres. *speechiness* varies significantly, with genres like *Dark Trap*, *Hip Hop*, and *Trap* showing higher values, indicating more spoken words. Other genres such as *Techno* and *Psytrance* have very low *speechiness*, but *instrumentalness* is notably higher, suggesting a predominance of instrumental tracks. Genres like *Hip Hop* and *Pop* have very low *instrumentalness*. These significant variations across genres suggest that features such as *speechiness*, *acousticness* and *instrumentalness* could be particularly important for genre classification models due to their distinctive patterns.

Additionally, we can explore the duration of the songs, which is encoded as a numerical feature in milliseconds (*duration_ms*). We can clearly see that the most popular song durations are between 3-4 minutes (Figure 4a). Transforming this numerical feature into duration categories can us help to better understand its influence on the genres (Figure 4b). In Figure 4c we can see the differences in song length between genres. For instance, *Dark Trap*, *Trap Metal* and *Underground Rap* have a high number of short songs, while *Dnb*, *Hardstyle*, *Psytrance*, *Techhouse*, *Techno*, *Trance* and *Trap* rarely have any. Especially, *Techno* and *Psytrance* mainly contain long songs, and the latter many extremely long songs. Because of this, we can assume that *duration_ms* might be an important feature for the models to base their predictions on.

3.3 Exploring the Categorical Features

Another important part of the EDA was the analysis of categorical variables, where we dealt with the *mode*, *key* and *time_signature* columns. Looking at the distribution of key types in different genres, we could not any specific relationships (Figure 5a). All genres contain songs with all key types, in somewhat equal portions, which might mean that this column does not carry much additional

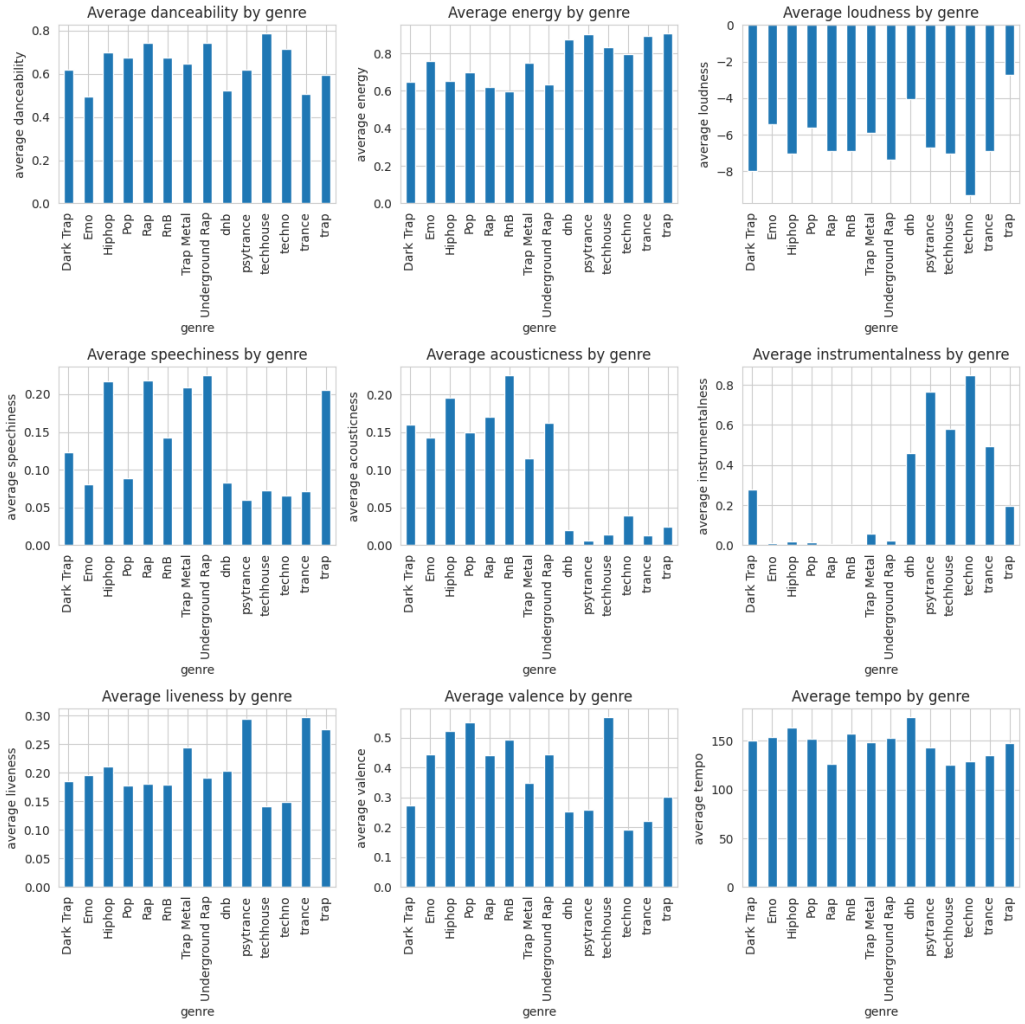
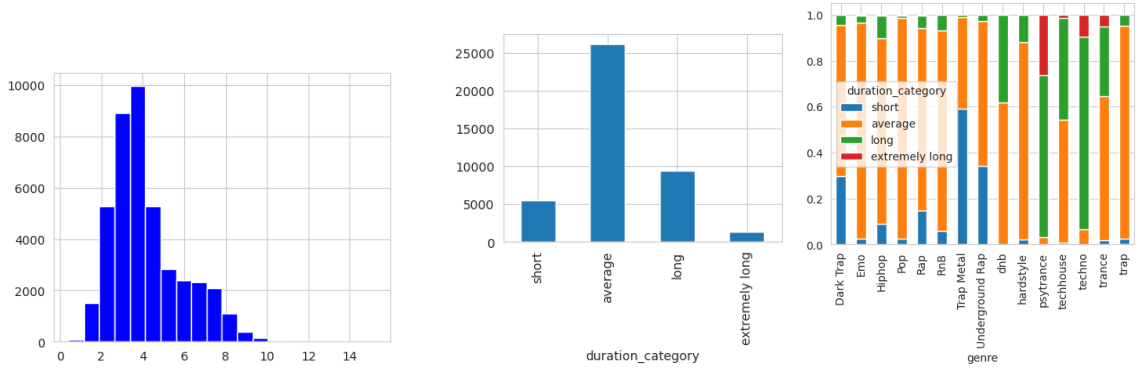


Figure 3: The average values of the numerical features



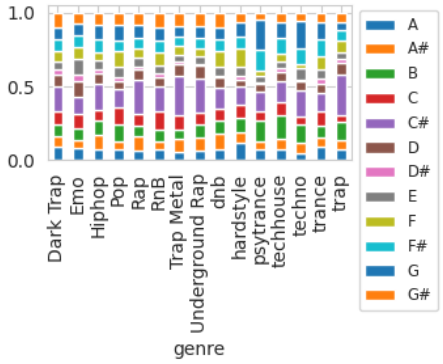
(a) Distribution of song durations in minutes.

(b) Number of songs per duration category.

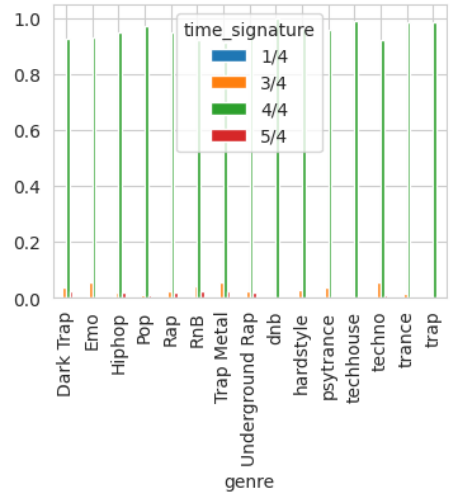
(c) Duration of songs per genre.

Figure 4: Data exploration on song durations (*duration_ms*).

information. Similarly, we cannot draw any relationship between specific time signatures and genres, as it is dominantly 4/4 for all the genres, while the other values are highly underrepresented (Figure 5b). Later we explain, that we ended up dropping these categorical columns, as it did not effect the performance of our best performing models, or they even got slightly better.



(a) Distribution of key types per genre.



(b) Distribution of time signatures per genre.

Figure 5: Data exploration on categorical variables.

3.4 Outlier Detection

We also experimented with detecting outliers, trying the IQR method and the Local Outlier Factor. The latter is a method used in anomaly detection that identifies anomalies based on the local density deviation of a given data point with respect to its neighbors. However, it did not improve the performance of the models when we ran them on the data set without outliers, so we decided to keep them to have sufficient data.

4 Preprocessing

4.1 Feature Selection

First, we selected relevant features by dropping columns that were deemed irrelevant for the classification task, because they do not contain relevant information for the classification task. These columns included identifiers and URLs such as 'type', 'id', 'uri', 'track_href', 'analysis_url', 'song_name', 'Unnamed: 0', and 'title'. During working on our projects, while analyzing feature importances of our best models, we noticed that some features have a very small influence on the predictions (Figure 6). To evaluate, if we can delete these features we examined several scenarios, considering that all of them are categorical and can be used with one-hot encoding.

For example, in case of the Random Forest F1-scores for the scenarios:

- With keeping the features *time_signature*, *key* and *mode*, without one-hot encoding: 0.851
- With keeping the features *time_signature*, *key* and *mode*, with one-hot encoding: 0.841
- With keeping only the feature *mode*, dropping the others, without one-hot encoding: 0.848

- With dropping all the columns: 0.852

So our final decision was to drop those columns, and it improved the performances and the time needed to train the models.

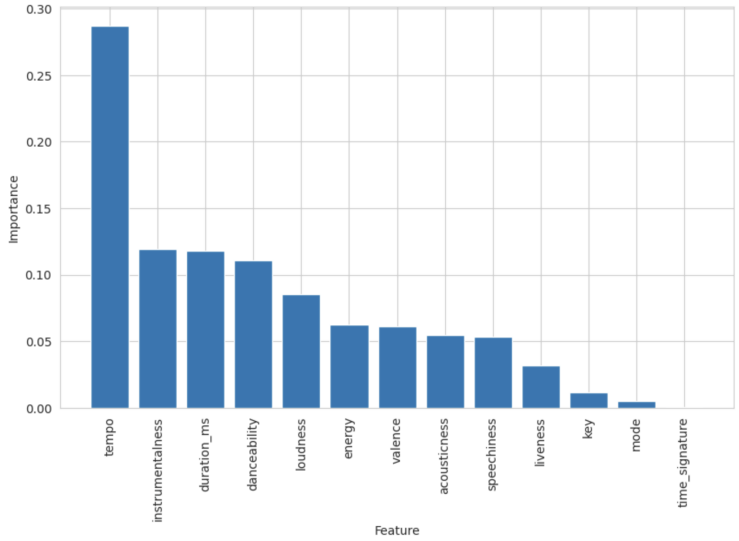


Figure 6: Feature importances of the best Random Forest model. *tempo* has a big influence on predictions, while the features *time_signature*, *key* and *mode* seem fairly irrelevant.

4.2 Data Transformation

We used Min-Max Scaling to normalize the numerical columns ensuring that all features are on a similar scale. This normalization is important for many machine learning algorithms, particularly those that rely on distance measurements, such as KNN and SVM. One-Hot Encoding was applied to the categorical columns. This technique converts categorical variables into a binary matrix representation, which is necessary for algorithms that require numerical input. However, as mentioned, we ended up not using the categorical variables in the final version of the project. These transformations were chosen to enhance the performance and accuracy of the models by ensuring that the data is in an optimal format for analysis and modeling.

4.3 Genre Selection

As mentioned in the data exploration, some classes like *Pop* were underrepresented, while others like Rap occure more often than the average. Furthermore, the big number of target values (15) together with some of the target values being very similar (e.g. *Rap* and *Underground Rap*) we decided to adjust the genre classes. First, we dropped the genre *Pop*, because the small number of samples did not allow for accurate prediction and there is no other similar genre, which *Pop* could be added to (Figure 8). The genres genres *Rap, Underground Rap, Hiphop, Trap Metal* and *Dark Trap* were put together in a class *Rap*. Training our models with the initial genre distribution, we noticed that these genres were often confused (Figure 7). Combining them makes the prediction easier, while maintaining meaningfulness of the predictions because of the genres similarity.

4.4 Oversampling

The result of combining those genres was a significant class imbalance, with the new genre *Rap* being overrepresented in comparison the the other genres. Considering our dataset contains sufficient

predicted	Dark Trap	Emo	Hiphop	Pop	Rap	RnB	Trap Metal	Underground Rap	dnb	hardstyle	psytrance	techhouse	techno	trance	trap
target															
Dark Trap	370	18	25	5	19	37	109	184	12	29	1	10	9	69	20
Emo	7	246	4	15	3	32	5	7	5	18	1	2	1	4	2
Hiphop	31	17	262	6	31	94	18	117	3	1	0	6	0	3	5
Pop	3	18	6	31	3	17	1	8	0	1	0	6	0	2	1
Rap	13	2	31	2	151	47	5	113	0	0	0	0	0	1	2
RnB	14	18	87	20	30	178	4	57	1	1	0	4	0	1	1
Trap Metal	63	5	9	2	7	7	143	103	1	8	0	3	0	5	13
Underground Rap	121	11	202	13	102	64	159	477	6	1	0	4	0	3	10
dnb	0	6	3	1	0	1	0	0	596	0	0	0	0	0	0
hardstyle	9	5	0	0	0	0	3	0	0	556	12	0	0	1	41
psytrance	3	0	0	0	0	0	0	0	0	2	549	0	8	24	8
techhouse	7	2	0	6	3	0	1	2	0	0	0	511	30	5	1
techno	5	0	0	0	0	0	0	0	0	0	11	31	490	22	0
trance	17	5	0	0	1	1	2	0	0	0	20	7	31	544	4
trap	17	4	1	1	2	3	4	3	0	41	2	0	0	13	498

Figure 7: Confusion matrix on the validation set of the best Random Forest model trained on train set. Genres related to Rap and Hip Hop are often confused.

	precision	recall	f1-score	support
Dark Trap	0.54	0.40	0.46	917
Emo	0.69	0.70	0.70	352
Hiphop	0.41	0.44	0.43	594
Pop	0.30	0.31	0.30	97
Rap	0.43	0.43	0.43	367
RnB	0.37	0.42	0.39	416
Trap Metal	0.32	0.40	0.36	369
Underground Rap	0.45	0.40	0.42	1173
dnb	0.95	0.98	0.96	607
hardstyle	0.85	0.89	0.87	627
psytrance	0.92	0.92	0.92	594
techhouse	0.88	0.89	0.88	568
techno	0.85	0.87	0.86	559
trance	0.79	0.86	0.82	632
trap	0.82	0.84	0.83	589
accuracy			0.66	8461
macro avg	0.64	0.65	0.64	8461
weighted avg	0.66	0.66	0.66	8461

Figure 8: Classification report on the validation set of the best Random Forest model trained on train set. F1-score of genres related to Rap and Hip Hop are low. Particularly the genre Pop is not predicted well.

samples we decided to drop 85% of the sample of the *Rap* genre, resulting in more balanced classes. In order to achieve perfect class balance, we finally applied SMOTE (Synthetic Minority Oversampling Technique). SMOTE generates synthetic samples for the minority classes by interpolating between existing samples, effectively balancing the class distribution.

4.5 Validation Protocol

For this project, we are using a simple train-val-test split (60-20-20). Because we have a large amount of data we are not restricted to create a validation set, which is only used to compute confusion matrices and the classification reports. For estimating the generalization error of each model the train set is used with cross-validation and finally the generalization error of our best model is measured with the test set.

5 Modeling

We employed several modeling methods introduced in the course to build classifiers and fine-tuned them by adjusting hyperparameters to achieve optimal performance. We evaluated the models using cross-validation and measuring accuracy, precision, recall and F1-score. For some methods we also analyzed feature importance and confusion matrices, which allowed us to continuously understand and improve the models. The overall performance of the models was ranked based on their macro F1-score, providing an assessment of each model's effectiveness. The final list of the performance scores can be seen at the end of this section. For the models or procedures that include some kind of randomness, we made sure that the results can be reproduced, by assigning a random state.

5.1 Methods

5.1.1 Linear Discriminant Analysis (LDA)

LDA is a classification method that finds a linear combination of features that best separates two or more classes. The highest performance was observed for the *Dnb* genre with an F1-score of 0.78, while the *Emo* genre showed the lowest performance with an F1-score of 0.44. One notable observation is the varying performance of LDA across different genres, as the *Psytrance* genre performed relatively well, achieving a precision of 0.69 and a recall of 0.81 (Table 3a) This suggests that limited to linear decision boundaries the LDA model is not able to predict our complex dataset well.

5.1.2 Quadratic Discriminant Analysis (QDA)

QDA is similar to LDA but allows for a quadratic decision boundary, making it more flexible for classifying datasets with non-linear relationships. We are using GridSearchCV to find the best regularization parameter for QDA. We receive 0 as the best regularization value. The highest performance was observed for the *Dnb* genre with an f1-score of 0.97, while the *Rap* genre showed the lowest performance with an F1-score of 0.45 (Table 3b). As a result, QDA is performing far better than the LDA model, which is reasonable given that the classes have different covariance structures.

5.1.3 K-Nearest Neighbors (KNN)

KNN is a simple, instance-based learning algorithm that classifies a data point based on the majority class of its nearest neighbors. Again, we are using GridSearchCV to find the best number of neighbours, which is 1. Notably, genres like *Hardstyle, Psytrance*, and *Techhouse* demonstrated relatively consistent precision and recall scores, indicating robust classification performance across these genres (Table 3c). The KNN model seems like a good choice for a baseline model, as it reaches a performance similar to the QDA model. The rather low dimensionality of the data and moderate-sized dataset helps the model to predict accurately. Weaknesses might be that the classes are not well separated.

5.1.4 Naive Bayes (GaussianNB)

Gaussian Naive Bayes is a probabilistic classifier based on Bayes' theorem, assuming independence between features and a Gaussian distribution for continuous variables. Naive Bayes does not allow for extensive hyperparameter tuning. Our model does not reach the performance of our best models, which may be the case due to most features being non gaussian. It achieved an overall accuracy of 73%. The highest performance was observed for the *Dnb* genre with an F1-score of 0.97, while the *Rap* genre showed the lowest performance with an F1-score of 0.36 (Table 3d).

5.1.5 Logistic Regression

Logistic Regression is a linear model used for binary classification that estimates the probability of a class label using the logistic function. We are using GridSearchCV to find the best value for the regularization parameter C, which is 1000. Sharing similar limitations with LDA, Logistic Regression struggles to separate the non linear relationships between classes, leading to a rather weak performance. Similar to LDA, we observe that the duration of the songs has a big influence on the predictions, while other features like key seem obsolete (Table 3e).

5.1.6 Decision Tree

A Decision Tree is a non-linear model that splits the data into subsets based on feature values, creating a tree-like structure to make predictions. With GridSearchCV we tune the hyperparameters *max_dephts*, *min_samples_split*, *min_samples_leaf* and *max_features*. In comparison to the previous models, decision trees perform very well on the non linear genre classification data. We can further improve this using random forests reducing the variance of the model (Table 3f).

5.1.7 Random Forest

Random Forest is an ensemble method that builds multiple decision trees and combines their outputs to improve accuracy and control overfitting. To find the best model we tune the hyperparameters *ntrees*, *max_depth*, *min_samples_split*, *min_samples_leaf* and *class_weight* using RandomizedSearchCV to reduce the time effort. Random Forest exhibited high precision and recall across most genres, particularly excelling in genres like *Dnb, Hardstyle*, and *Psytrance* (Table 4a).

5.1.8 Extra Trees

Extra Trees, or Extremely Randomized Trees, is similar to Random Forest but uses a more randomized approach to split nodes, reducing variance in the model. We are using the same hyperparameter tuning as with Random Forests (Table 4b).

5.1.9 Support Vector Machine (SVM)

SVM is a powerful classifier that finds the hyperplane maximizing the margin between classes in a high-dimensional space. We used RandomizedSearchCV for fine-tuning the SVC model (SVM for classification problems).

The SVC classifier with optimized parameters ('kernel': 'rbf', 'gamma': 1, 'C': 1) achieved an overall accuracy of 78% on the multiclass classification problem. It demonstrated strong performance across various genres, with particularly high precision and recall for the *Dnb* and *Psytrance* classes, indicating effective differentiation between these categories. The model showed the weakest performance on the *Emo genre*, with a precision of 0.50 and a recall of 0.72. This indicates that while the model identified a majority of *Emo* instances correctly, it also had a high rate of false positives, suggesting challenges in accurately distinguishing Emo from other genres (Table 4c).

5.1.10 XGBoost

XGBoost is a gradient boosting framework that builds an ensemble of trees sequentially, optimizing performance and speed for large-scale classification tasks.

Using the XGB model with the best parameters identified through randomized search CV ('subsample': 0.8, 'n_estimators': 400, 'min_child_weight': 4, 'max_depth': 5, 'learning_rate': 0.05), the model achieved an overall accuracy of 86%. The highest performance was observed for the *DnB* genre with an F1-score of 0.99, while the *Rap* genre showed the lowest performance with an F1-score of 0.69. Notably, the *Emo* genre also performed relatively well, achieving an F1-score of 0.76 (Table 4d).

5.1.11 Neural Networks (MLPClassifier)

Multi-Layer Perceptron (MLP) is a type of neural network that uses multiple layers of nodes to learn non-linear mappings from inputs to outputs.

Using an MLP neural network with the best parameters identified through randomized search CV ('hidden_layer_sizes': [128], 'alpha': 0.0001, 'activation': 'relu'), the model achieved an overall accuracy of 82%. The highest performance was observed for the *DnB* genre with a precision of 0.98 and recall of 0.95, while the *RnB* genre showed the lowest performance, with an F1-score of 0.65 (Table 4e). Multiple feature combinations were explored to optimize model performance. We also tried using PCA to reduce dimensionality, as seen in the labs, to see if it improves the results, but it ended up worsening them.

5.1.12 Ensemble Methods (Voting, Stacking, Bagging)

Ensemble methods combine the predictions of multiple models to improve overall performance to enhance robustness and accuracy. We use three different methods: voting, stacking and bagging.

In voting, multiple models make predictions independently, and the final prediction is determined by a majority vote (hard voting) or averaging of individual predictions (soft voting), which can be weighted by the models individual performance (best voting). We combined our four models in the voting classifiers, based on diversity of the types of models and their results: XGBoost Classifier, Extra Trees, MLP and SVM.

Stacking involves training a meta-model on the predictions of multiple base models. We used XGBoost Classifier and Extra Trees as base models and trained a GradientBoost Classifier on their predictions.

Using the stacking classifier, the model achieved an overall accuracy of 85%. The highest performance was observed for the *DnB* genre with an F1-score of 0.98, while the *Rap* genre showed the lowest performance with an F1-score of 0.68 (Table 4f).

Bagging involves training multiple instances of the same base model on different subsets of the training data and averaging their predictions. As base model we used XGBoost, and the result of the bagging was exactly the same as the base models'. This is likely not a coincidence, as XGBoost already incorporates several ensemble techniques, such as boosting and bagging, into its algorithm. So when we apply a bagging ensemble on top of XGBoost, the additional benefit might be minimal or nonexistent, especially if the model parameters and dataset characteristics are such that XGBoost alone is already capturing the full potential of the data.

5.2 Results

After training and evaluating all our models, finally, we can examine the results (Table 1). Starting from the top of our table we see that the XGBoost Classifier and the Bagging Classifier reach the best F1-score of 0.857. With a slightly better precision we choose the XGBoost Classifier as our final model, as the Bagging Classifier also uses this model as base. Following, we have the Stacking Classifier with a F1-score of 0.853, which combines our best models, however not reaching the XGB models performance. Our best Voting Classifier could performed the worst out of our self created ensemble methods with a F1-score of 0.845). However, experimenting more with our base models could yield different results. From our tree based models the Random Forest achieves the best results with a F1-score of 0.849 outperforming Extra Trees (0.845) and a simple Decision Tree (0.774). This result is not surprising, as the Random Forest is a composition of Decision Trees enabling it to make better predictions. Surprisingly the MLP Classifier did not achieve a competitive F1-score (0.787) making it a unsuitable choice for this dataset considering its high computational cost. With more computational resources we might be able to find better parameter combinations, which allow the neural network to perform better, however our shallow machine learning methods work pretty good on this dataset, making a computational expensive neural network obsolete. The remaining models SVM (0.779), QDA (0.759), KNN (0.744), GNB (0.72), Logistic Regression (0.711) and LDA (0.651)

could not perform very well on the dataset. Despite being fast their inaccurate predictions make them a bad choice for this classification problem.

Considering these results, we can definitely say that grouping together some genres helped to get better results. Before the grouping our best F1-score was around 0.66, and now our worst model has about the same performance.

Model	Accuracy	Precision	Recall	F1-Score
XGBClassifier	0.862	0.855	0.861	0.857
BaggingClassifier	0.862	0.854	0.861	0.857
StackingClassifier	0.859	0.851	0.857	0.853
RandomForestClassifier	0.855	0.847	0.854	0.849
VotingClassifier	0.850	0.843	0.849	0.845
ExtraTreesClassifier	0.851	0.842	0.848	0.844
MLPClassifier	0.794	0.784	0.793	0.787
SVC	0.786	0.780	0.784	0.779
DecisionTreeClassifier	0.781	0.771	0.781	0.774
QuadraticDiscriminantAnalysis	0.769	0.770	0.768	0.759
KNeighborsClassifier	0.752	0.742	0.750	0.744
GaussianNB	0.735	0.737	0.735	0.720
LogisticRegression	0.722	0.710	0.717	0.711
LinearDiscriminantAnalysis	0.659	0.655	0.655	0.651

Table 1: Final list of modelling methods and their performance metrics.

5.3 Final Model

We chose the XGBoost Classifier as our final model. To estimate its generalization performance we evaluated its performance on the unseen test set.

With a macro F1-score of 0.85 our final model performs very well on the test data (Table 2). We can therefore say that our resampling methods and model selection worked well in terms of selecting a good model and preventing overfitting. This is underscored by the confusion matrix, which shows similar results as on the validation set (Figure 9). While our final model still struggles with some genres that are not easily differentiable like *Emo*, *Rap*, *RnB*, our model was able to classify all songs from the *DnB* genre correctly reaching a recall of 1.0.

Class	Precision	Recall	F1-Score	Support
Emo	0.70	0.77	0.73	324
Rap	0.74	0.69	0.71	529
RnB	0.67	0.64	0.66	446
dnb	0.98	1.00	0.99	596
hardstyle	0.90	0.93	0.92	616
psytrance	0.92	0.94	0.93	589
techhouse	0.91	0.92	0.91	589
techno	0.87	0.87	0.87	555
trance	0.87	0.88	0.88	564
trap	0.90	0.87	0.89	627
Accuracy			0.86	5435
Macro avg	0.85	0.85	0.85	5435
Weighted avg	0.86	0.86	0.86	5435

Table 2: Classification Report for the final XGB model on the test set

predicted target	Emo	Rap	RnB	dnb	hardstyle	psytrance	techhouse	techno	trance	trap
Emo	249	11	38	2	13	0	2	0	5	4
Rap	23	364	92	6	5	2	6	1	16	14
RnB	58	87	287	4	1	0	2	0	5	2
dnb	0	0	0	596	0	0	0	0	0	0
hardstyle	7	1	1	0	572	5	0	0	0	30
psytrance	0	1	1	0	3	551	0	9	17	7
techhouse	2	3	3	0	0	0	539	39	3	0
techno	0	2	1	0	0	15	36	483	18	0
trance	7	6	2	0	0	21	8	22	495	3
trap	11	17	3	0	40	2	0	0	8	546

Figure 9: Confusion matrix on the test set of the XGBoost Classifier, which we chose as our final best model.

The ROC curve (receiver operating characteristic curve) visualizing the performance of a classification model at all classification thresholds, further shows the differences in performance of the different genres (Figure 10). While the class *DnB* reaches a perfect ROC curve the areas under genres like *Emo*, *Rap* and *RnB* are slightly smaller, indicating a worse performance.

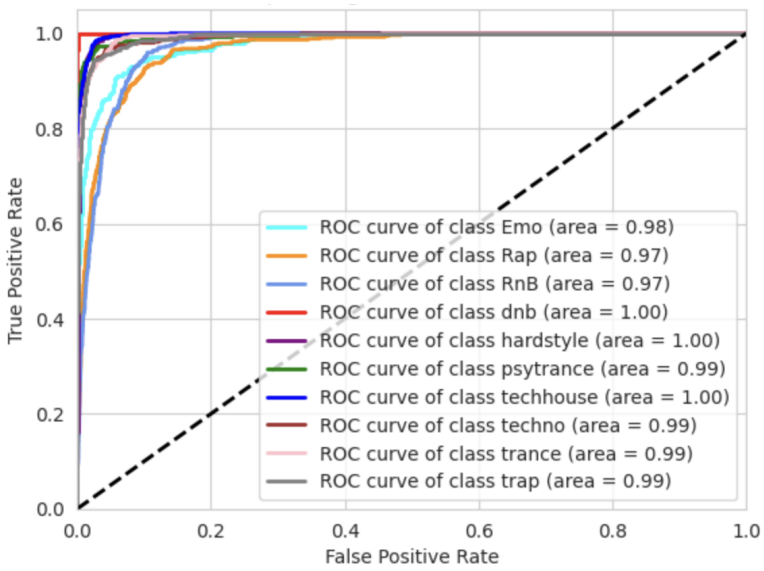


Figure 10: ROC curves for each target value of the final XGBoost Classifier on the test set.

As a final step, it is worth looking at the feature importance of the model, as interesting conclusions can be drawn about the model's decisions. (Figure 9) The most important features were *tempo, duration_ms* and *danceability*. The least significant was *liveness*, however it still contributed a lot to the final prediction. Now we don't have any features with low importance anymore, as we deleted them in the preprocessing part, as explained earlier.

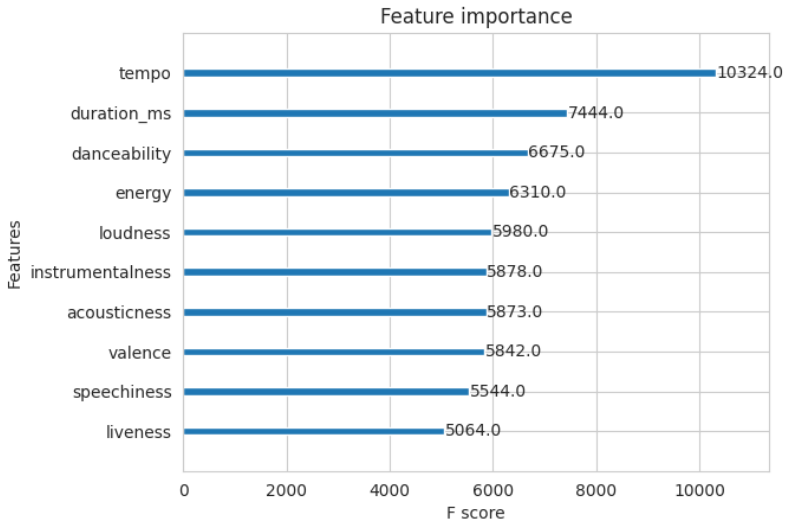


Figure 11: Feature importance for the final XGBoost Classifier on the test set.

Overall, we consider our task in predicting the genre classes from song features as successful as we could reach an overall accuracy of 86%.

6 Conclusion

In this project, we set out to predict music genres using Spotify's song dataset. Through extensive exploration and experimentation with different machine learning algorithms, we aimed to explore the relationship between objective musical characteristics and subjective genre classifications.

Our analysis revealed that ensemble methods, particularly the XGBoost Classifier, outperformed other machine learning models in accurately predicting music genres. The XGBoost model demonstrated robustness and high accuracy across multiple genres. It showed that we don't always need the complexity and computational demands of neural networks, as a simpler solution can be adequate for the specific dataset we are working on. We found that specific musical attributes played significant roles in genre classification. Features such as *tempo*, *duration_ms*, and *danceability* emerged as influential factors, contributing to the final model's predictive performance. We were not surprised that the length of the songs was an important aspect of the classification, as we have seen before that some genres are very much characterised by this feature, e.g. *Psytrance* songs are on average longer than others. Tempo on the other hand, didn't seem as influential, as it turned out to be. (Figure 9) Our experiments with data preprocessing techniques, including resampling methods, outlier detection and feature selection, highlighted the importance of data quality and balance in model training. By addressing class imbalances and optimizing feature sets, we were able to enhance model performance and mitigate overfitting. While the overall performance of our final XGBoost Model was pretty good with a F1-score of 0.85 on the test dataset, it still struggled to get good results on the genres *Emo*, *Rap* and *RnB*. In general all models had a problem to some degree with the differentiation of these three, and usually the performance was worse in these genres, while other genres like *DnB* were predicted more accurately. Furthermore, during our work we really experienced how parameter tuning is crucial for getting the best performance out of our models. Neural networks were particularly sensitive to parameter choices like learning rate and hidden layer sizes. Poorly tuned neural networks often performed bad. In contrast, models like XGBoost and Random Forests were more stable across different parameter settings. This highlights the importance of careful parameter tuning, especially for complex models like neural networks, to ensure they perform their best on the selected dataset.

Future work on this dataset could try to improve the models performance even further, especially when using the initial set of genres. Our simplification to group genres together improved the performance, but also made the classification problem way easier. With more computational resources available we could also try to create a neural network model that is able to outperform our shallow modelling methods. While being computational expensive this could generate an even better classifier like we observed in a project mentioned in related work [6].

[6] https://www.analyticsvidhya.com/blog/2023/03/solving-spotify-multiclass-genre-classification-problem/

Appendix

	Precision	Recall	F1-score	Support
Emo	0.39	0.50	0.44	316
Rap	0.65	0.50	0.57	532
RnB	0.55	0.54	0.55	406
dnb	0.83	0.74	0.78	584
hardstyle	0.50	0.60	0.55	596
psytrance	0.69	0.81	0.74	603
techhouse	0.78	0.75	0.76	601
techno	0.71	0.81	0.75	578
trance	0.64	0.50	0.56	603
trap	0.71	0.65	0.68	616
Accuracy			0.65	5435
Macro avg	0.65	0.64	0.64	5435
Weighted avg	0.66	0.65	0.65	5435

(a) Classification Report for the LDA Model.

	Precision	Recall	F1-score	Support
Emo	0.57	0.70	0.63	316
Rap	0.67	0.34	0.45	532
RnB	0.48	0.83	0.61	406
dnb	0.99	0.95	0.97	584
hardstyle	0.81	0.79	0.80	596
psytrance	0.80	0.90	0.85	603
techhouse	0.87	0.86	0.86	601
techno	0.79	0.81	0.80	578
trance	0.79	0.73	0.76	603
trap	0.84	0.70	0.76	616
Accuracy			0.77	5435
Macro avg	0.76	0.76	0.75	5435
Weighted avg	0.78	0.77	0.76	5435

(b) Classification Report for the QDA Model.

	Precision	Recall	F1-score	Support
Emo	0.56	0.65	0.60	316
Rap	0.58	0.47	0.52	532
RnB	0.50	0.52	0.51	406
dnb	0.94	0.92	0.93	584
hardstyle	0.76	0.78	0.77	596
psytrance	0.82	0.84	0.83	603
techhouse	0.80	0.80	0.80	601
techno	0.73	0.79	0.76	578
trance	0.72	0.71	0.71	603
trap	0.77	0.76	0.76	616
Accuracy			0.74	5435
Macro avg	0.72	0.72	0.72	5435
Weighted avg	0.74	0.74	0.74	5435

(c) Classification Report for the KNN Model.

	Precision	Recall	F1-score	Support
Emo	0.53	0.61	0.57	316
Rap	0.62	0.25	0.36	532
RnB	0.45	0.84	0.59	406
dnb	0.98	0.95	0.97	584
hardstyle	0.77	0.76	0.77	596
psytrance	0.75	0.86	0.80	603
techhouse	0.87	0.84	0.86	601
techno	0.74	0.79	0.76	578
trance	0.74	0.66	0.70	603
trap	0.81	0.67	0.73	616
Accuracy			0.73	5435
Macro avg	0.73	0.72	0.71	5435
Weighted avg	0.75	0.73	0.72	5435

(d) Classification Report for the GNB Model.

	Precision	Recall	F1-score	Support
Emo	0.37	0.51	0.43	316
Rap	0.64	0.54	0.59	532
RnB	0.55	0.56	0.56	406
dnb	0.87	0.88	0.87	584
hardstyle	0.73	0.62	0.67	596
psytrance	0.83	0.85	0.84	603
techhouse	0.78	0.81	0.79	601
techno	0.78	0.83	0.80	578
trance	0.70	0.70	0.70	603
trap	0.77	0.73	0.75	616
Accuracy			0.72	5435
Macro avg	0.70	0.70	0.70	5435
Weighted avg	0.72	0.72	0.72	5435

(e) Classification Report for the LogReg Model.

	Precision	Recall	F1-score	Support
Emo	0.58	0.68	0.63	316
Rap	0.57	0.54	0.56	532
RnB	0.52	0.57	0.54	406
dnb	0.96	0.95	0.95	584
hardstyle	0.84	0.84	0.84	596
psytrance	0.85	0.88	0.86	603
techhouse	0.86	0.87	0.86	601
techno	0.79	0.80	0.79	578
trance	0.85	0.77	0.81	603
trap	0.82	0.77	0.79	616
Accuracy			0.78	5435
Macro avg	0.76	0.77	0.76	5435
Weighted avg	0.78	0.78	0.78	5435

(f) Classification Report for the Decision Tree Model.

Table 3: Classification Reports Part 1.

	Precision	Recall	F1-score	Support
Emo	0.69	0.77	0.73	316
Rap	0.71	0.65	0.68	532
RnB	0.64	0.66	0.65	406
dnb	0.98	0.98	0.98	584
hardstyle	0.90	0.90	0.90	596
psytrance	0.90	0.91	0.90	603
techhouse	0.91	0.92	0.91	601
techno	0.84	0.86	0.85	578
trance	0.86	0.86	0.86	603
trap	0.89	0.84	0.86	616
Accuracy			0.85	5435
Macro avg	0.83	0.83	0.83	5435
Weighted avg	0.85	0.85	0.85	5435

(a) Classification Report for the Random Forest Model.

	Precision	Recall	F1-score	Support
Emo	0.72	0.76	0.74	316
Rap	0.71	0.64	0.67	532
RnB	0.64	0.67	0.65	406
dnb	0.97	0.98	0.97	584
hardstyle	0.89	0.91	0.90	596
psytrance	0.91	0.91	0.91	603
techhouse	0.88	0.90	0.89	601
techno	0.82	0.87	0.84	578
trance	0.83	0.85	0.84	603
trap	0.91	0.82	0.86	616
Accuracy			0.84	5435
Macro avg	0.83	0.83	0.83	5435
Weighted avg	0.84	0.84	0.84	5435

(b) Classification Report for the Extra Tree Model.

	Precision	Recall	F1-score	Support
Emo	0.50	0.72	0.59	316
Rap	0.70	0.62	0.66	532
RnB	0.60	0.64	0.62	406
dnb	0.97	0.90	0.93	584
hardstyle	0.83	0.78	0.81	596
psytrance	0.86	0.89	0.87	603
techhouse	0.83	0.84	0.84	601
techno	0.80	0.83	0.82	578
trance	0.77	0.78	0.78	603
trap	0.87	0.74	0.80	616
Accuracy			0.78	5435
Macro avg	0.77	0.77	0.77	5435
Weighted avg	0.79	0.78	0.79	5435

(c) Classification Report for the SVM Model.

	Precision	Recall	F1-score	Support
Emo	0.76	0.77	0.76	316
Rap	0.73	0.65	0.69	532
RnB	0.63	0.70	0.66	406
dnb	0.98	0.99	0.99	584
hardstyle	0.91	0.93	0.92	596
psytrance	0.91	0.91	0.91	603
techhouse	0.90	0.93	0.91	601
techno	0.84	0.86	0.85	578
trance	0.87	0.86	0.87	603
trap	0.90	0.87	0.88	616
Accuracy			0.86	5435
Macro avg	0.84	0.85	0.84	5435
Weighted avg	0.86	0.86	0.86	5435

(d) Classification Report for the XGB Model.

	Precision	Recall	F1-score	Support
Emo	0.68	0.72	0.70	316
Rap	0.71	0.64	0.67	532
RnB	0.63	0.67	0.65	406
dnb	0.98	0.95	0.96	584
hardstyle	0.83	0.91	0.87	596
psytrance	0.88	0.89	0.89	603
techhouse	0.87	0.86	0.87	601
techno	0.80	0.87	0.83	578
trance	0.82	0.83	0.82	603
trap	0.89	0.77	0.83	616
Accuracy			0.82	5435
Macro avg	0.81	0.81	0.81	5435
Weighted avg	0.82	0.82	0.82	5435

(e) Classification Report for the MLP Model.

	Precision	Recall	F1-score	Support
Emo	0.73	0.79	0.76	316
Rap	0.70	0.67	0.68	532
RnB	0.64	0.64	0.64	406
dnb	0.98	0.98	0.98	584
hardstyle	0.91	0.93	0.92	596
psytrance	0.90	0.90	0.90	603
techhouse	0.90	0.91	0.90	601
techno	0.84	0.85	0.84	578
trance	0.88	0.87	0.87	603
trap	0.90	0.87	0.88	616
Accuracy			0.85	5435
Macro avg	0.84	0.84	0.84	5435
Weighted avg	0.85	0.85	0.85	5435

(f) Classification Report for the Stacking Model.

Table 4: Classification Reports Part 2.