Music genre prediction

1 Dataset

Our group examined the *Prediction of music genre* dataset[1] by vicsuperman on the Kaggle platform. We were drawn to this dataset because of our shared interest in music and the wide array of features that the dataset offers. The dataset consists of 50,000 rows, each representing a song/track, and 18 columns. Below is a table of the relevant numerical columns, which we also refer to as musical attributes in this paper.

Column	**Description**
popularity	How popular a song is from 0.0 to 100.0
acousticness	Confidence measure of whether track is acoustic from 0.0 (low) to 1.0 (high)
danceability	How suitable a track is for dancing from 0.0 (not danceable) to 1.0
duration_ms	The duration of a track in milliseconds from 0 to $4.83*10^6$
energy	Measure of intensity and activity from 0.0 (low) to 1.0 (high)
instrumentalness	Absence of vocals; value over .5 suggests track is instrumental
liveness	Presence of audience in recording; value over .8 indicates live recording
loudness	Loudness of a track in decibels; typical range is -60 to 0 dB
speechiness	How much spoken word is in a track from 0 (no speech) to 1 (all speech); songs typically do not exceed 0.66
tempo	Tempo of track in beats per minute
valence	How positive a song is from 0.0 (sad, angry) to 1.0 (happy, upbeat)

This information is from the Spotify Web API Reference[2].

There is also information about each track's *music_genre*, which can take on one of the following values: Electronic, Anime, Jazz, Alternative, Country, Rap, Blues, Rock, Classical, Hip-Hop. There are 5,000 tracks for each genre.

In our exploratory analysis, we found the distribution of each attribute for all genres. We noticed that classical music tended to differ in *acousticness, danceability, energy, instrumentalness, loudness,* and *valence*. We also found that rap music and hip-hop music tend to have a higher *speechiness* than the other genres. Overall the distribution of each of the attributes differ throughout genres, so we cannot exclude any of these attributes in further analysis.

However, a correlation matrix for these attributes revealed that *energy* was highly correlated with *acousticness* and *loudness*, so we chose not to include this attribute in further analysis (Figure 1.1).

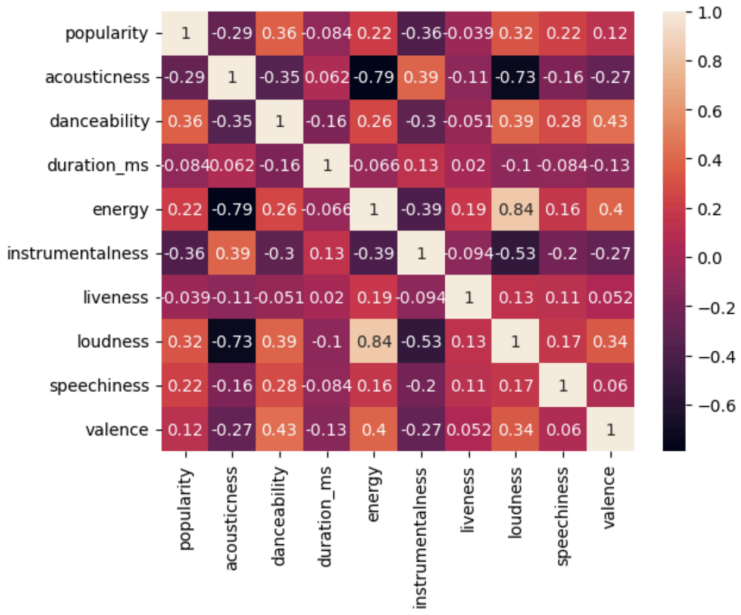
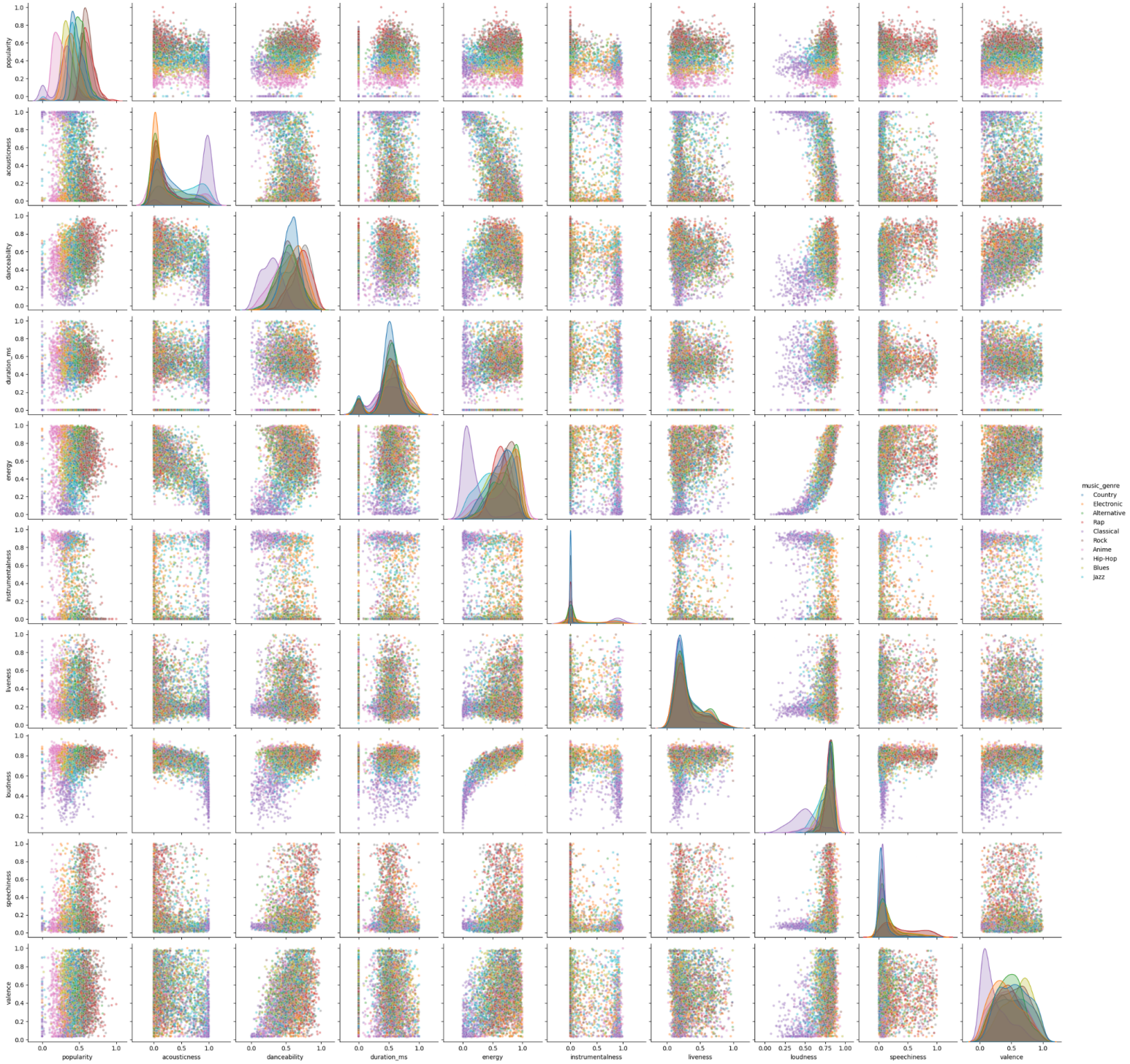


Figure 1.1

To further explore the dataset we used a pairplot. The diagonal represents the distribution of each feature [popularity, acousticness, danceability, etc] for different genres. The distribution shapes and overlaps show how well-separated the genres are based on each feature. The narrow peaks suggest that the range of values is small, the feature has a distinct value for a specific genre. Broad, overlapping peaks indicate that the feature does not differentiate well between genres. For example, `instrumentalness` likely has high value for genres like Classical and Jazz, but very low value for Rap and Hip-hop. `valence` has broad peaks for most of the features, so it might not be useful for separating the genres.

The scatter plots show the relationships between pairs of features, with points colored by genre. The clusters with distinct regions suggest the two features together can effectively distinguish the genres. Overlapping points indicate poor separation between genres, which from the graph we can see that most genres overlap with each other. However, classical is the most unique genre, as its features peaks and clusters are always distinguished from other genres.

Figure 1.2


popularity
acousticness
danceability
duration_ms
energy
instrumentalness
liveness
loudness
speechiness
valence
music_genre
Country
Electronic
Alternative
Rap
Classical
Rock
Anime
Hip-Hop
Blues
Jazz

2 Predictive Task

We had a few initial ideas of a predictive task: recommend similar artists or songs, predict the popularity of a song, or predict the genre of a song. The exploratory analysis led us to favor the third idea, genre prediction, as our dataset did not contain user-item interactions. To assess the validity of our models we split the dataset into a training and test set where we trained our models only using the training set and then tested with only the test set. It is important to note that before splitting we shuffled our dataset to ensure that each genre was represented in the training and testing sets. To determine how well our model performed we used an accuracy score which represents the ratio of number of songs labeled with the correct genre from the test set.

2.1 Baseline Models

For the baseline model we chose to use random classification which randomly assigns a song a genre type. Since each genre is represented equally in the dataset, we did not have to weigh each genre since they all have an equal probability of occuring. The accuracy score for randomly assigning a genre to a song was 10%, which we will use as a baseline accuracy score for the models we use.

2.2 Feature Selection

To further analyse the correlation between features, we applied Principle Component Analysis (PCA) technique. Once the PCA matrix is calculated, we plot the data instances on the first 2 principal components (for 2D plot, Figure 2.1), and first 3 principal components (for 3D plot, Figure 2.2). Using the values of principal component vectors, we can plot arrows representing the directions of the original features on the PC0-PC1 plane.

The length of the arrows indicates how much each original feature contributes to the variance captured by the components. Features with short arrows might contribute less and could be candidates for exclusion or transformation. If two arrows are closely aligned, their features are highly correlated. This suggests that one of the two features might be sufficient for training, or the two features can be combined. Features pointing in opposite directions are negatively correlated. We want to select the features that are unique and less correlated with other features, which has its arrow perpendicular to other arrows.

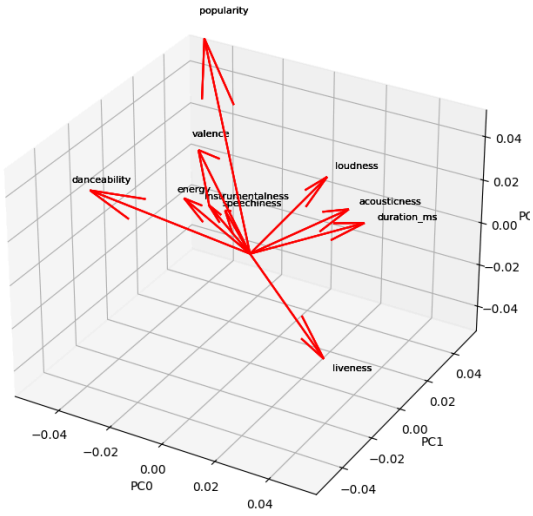


Figure 2.1

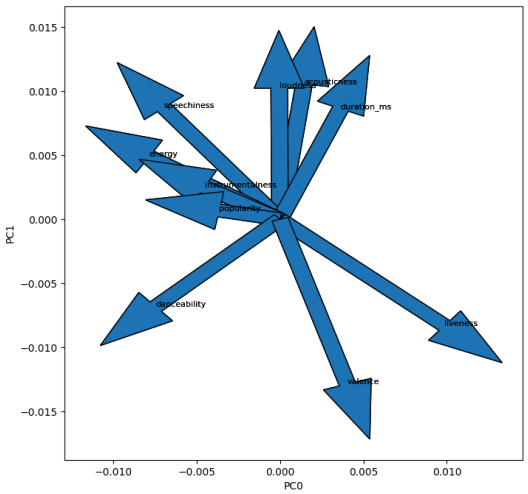


Figure 2.2

From the 2D and 3D plots, we see that the groups of relatively correlated features are [*'loudness', 'accousticness', 'duration_ms'*], [*'speechiness', 'energy', 'instrumentalness', 'popularity'*], [*'danceability'*], [*'valence'*], [*'liveness'*]. We could combine features within a group into a single feature (via averaging).

However, we have few reasons to keep all 10 features as training input. 10 is not a significantly large number in terms of dimensions. PCA primarily captures linear correlations; features that seem unimportant in the PCA plot might still have non-linear relationships with the target. Intuitively, the dataset includes these features because they are all important aspects when considering a music track instance. Using all features ensures no potentially useful information is excluded prematurely.

`*tempo*` could be a useful feature. However, it has 4985 NaN values, that is about 10% of the data. Since our dataset has only 50000 instances, it is better to not use `tempo` as a feature, rather than dropping 4985 instances with missing `tempo` values. We assume that *'danceability', 'energy',* and *'loudness'* already provided related information, excluding `tempo` might not hurt the predictive power of the model.

3 Model

Our most effective model was achieved with XGBoostClassifier; it was the highest in terms of accuracy at 57%. To address scalability and optimization, we employ min-max scaling and grid-searching techniques, respectively. In particular, we utilized GridSearchCV with 5-fold cross-validation to find the best hyperparameters for *learning_rate* and *subsample*, finding that the former is most optimal at 0.1 and the latter at 0.7. We also tuned *gamma* to 1, *max_depth* to 4, and *min_child_weight* to 3 in order to prevent overfitting. These parameters ensure that the

model is simple and conservative, and thus yields high accuracy without over- or under-fitting. Below we describe our trial-and-error processes.

The first model that we began with was logistic regression since it is a simple model that finds the relationships between data features and can be used to predict more than two classes, which in our case is genre types. To begin with this model we mapped each genre to a numerical value and then split our data into a training and testing set. The multiclass logistic regression had an accuracy score of 51%. To try to improve upon this model we dropped different features such as popularity, liveness, and duration_ms however this model performed worse than including all features. The issue with using this model is that genre types may not be linearly separable as the patterns between features are very complex.

To attempt to improve upon logistic regression, we considered using logistic regression for each genre separately. When only predicting one genre using logistic regression, the accuracy score for each genre was very high with the classical genre having the highest of 96% and rock having the lowest of 88%. However, the individual models had high accuracies due to it predicting 'no' most of the time. To fix this issue of mostly predicting 'no' we used class_weight = 'balanced'. The individual models still had fairly high accuracies, with the highest being 92% for classical and lowest being 59% for country. Then we built a model to predict each genre through running the individual models after another until a genre is predicted for a song, which had an accuracy score of 44%. This model did worse than the multiclass logistic regression as each logistic regression model is trained independently, however the order of the models affects the genre classification which introduces bias and errors depending on the sequence. Thus, overall the best logistic regression we found was multiclass logistic regression with an accuracy score of 51%.

After this, we experimented with a few other models, using GridSearchCV to optimize for accuracy. We tried k-Nearest Neighbors using the parameters, {'*leaf_size*': 20, *'metric'*: 'minkowski', '*n_neighbors*': 10, '*p*': 1, *'weights'*: 'distance'}, yet our accuracy was as low as 21%. This accuracy score was really low due to the scalability issue: some features had larger ranges than others which can lead to underperforming due to distorted distance calculations performed. To address the scalability issue, we normalized the features by applying MinMax scaling, and our accuracy increased to 46%. Additionally, we tested a Decision Tree model and found the optimal parameters to be {*'criterion'*: 'gini', '*max_depth*': 10, '*min_samples_split*': 10}. The accuracy for this model was 51.6%. Ultimately, XGBoost ended up working best, as described in the first paragraph. Notably, we attempted to employ the SVC model but it did not work—it was running for nearly an hour before we decided not to continue further with it.

Model	Baseline	Logistic	Multiclass	Decision Tree	XGB	KNN
Accuracy	10%	44%	51%	51.6%	57%	46%

4 Related Literature

4.1 Our Dataset

The dataset for our study is from Kaggle user vicsuperman and uses audio features provided by the Spotify API. These features include *acousticness, danceability, duration_ms, energy, instrumentalness, key, liveness, loudness, mode, speechiness, tempo,* and *valence*. The Spotify API also provides *time_signature* information, though this is not included in our dataset. The title of this dataset is *Prediction of music genre*, so naturally the majority of its uses have to do with genre prediction. Others have used it for song recommendation or classifying a specific genre (e.g. electronic).

4.2 Similar Studies

The studies below all used datasets that incorporate the Spotify API and thus share the aforementioned numerical characteristics.

4.2.1 "Music Genre Classification: A Machine Learning Exercise", published by Medium[3]

Juan Francisco Leonhardt's dataset consists of 125 genres and 114,000 tracks. Similar to our approach, he created a graph depicting the distribution of each musical attribute (listed above), per genre, and he constructed a correlation matrix, finding that correlations between the numerical features were not high.

To prepare the data, Leonhardt dropped null and duplicate values, as well as features that are not relevant for his model, such as *track_name*. He also scaled the numerical values since these variables spanned such different ranges. In his exploratory analysis, he found that the distribution of the musical attributes were similar across many genres, which would likely make genre prediction difficult. To mitigate this issue, he constructed a dendrogram by averaging the values for each music metric across all genres, grouping each genre into a 'parent' genre. Prediction models were constructed with the 56 new genres rather than the 114 originals.

Leonhardt implemented various approaches: Neural Networks, XGBoost, KNN Classifier, and an ensemble model (k-Nearest Neighbors, Logistic Regression, Decision Tree, and Support Vector Machine). He measured overall accuracy as well as top-3 categorical accuracy, and the latter accuracy was higher across all models. The XGBoost model was the most accurate, overall and top-3 categorical. The ensemble method had the poorest overall accuracy, primarily because the logistic regression and SVM models had poor results individually. Emo and techno were consistently the worst-predicted genres, while grindcore, sleep, study, and comedy were among the best-predicted. The latter group likely had more distinct attributes, making prediction easier.

4.2.2 "Music Genre Prediction Using Audio Features", published on Medium[4]

Navoda Senavirathne uses a dataset with 10,901 tracks and 10 major music genres, which is more similar to our dataset. While *key* and *mode* features in our dataset were categorical (e.g. A, B, etc. or major/minor), the *key* and *mode* features in Senavirathne's dataset were numerical. The author uses mutual information to determine the most important features, and since *key, mode,* and *liveness* have the lowest levels of mutual information, they are excluded from further observations. From a correlation matrix, Senavirathne found that *energy* had a strong positive correlation with *loudness* and a strong negative correlation with *acousticness*, which aligned with our exploratory analysis. To avoid multicollinearity, *energy* is also dropped from the dataset. The author also standardizes the dataset.

Senavirathne implements Logistic Regression, SVC, k-Nearest Neighbors, Decision Tree, Random Forest, Gradient Boosting, and XGB Classifier models; XGBClassifier ended up with the highest training and cross-validation accuracy. However, the author notes that the training accuracy (.99) being so much higher than the cross-validation accuracy (.64) shows that the XGB classifier was overfitting. Even after tuning randomness and complexity parameters, the cross-validation accuracy did not significantly improve.

4.3 Conclusions

The conclusions from the literature review was that XGBoost outperformed the other proposed models which included Neural Networks, KNN Classifier, an ensemble model, Logistic Regression, SVC, Decision Tree, Random Forest, and Gradient Boosting. The conclusions that XGBoost was the most accurate model to predict the genre of a song aligned with our results. Although we did not build all of the proposed models, our XGBoost model outperformed Logistic Regression, Decision Tree, and KNN Classifier by a significant amount.

5 Results and Conclusion

Overall the XGB model outperformed the other models we tested. The parameters of the final model as well as our rationale are laid out below:

Parameter	**Value**	**Reasoning**
learning_rate	0.1	A smaller step/learning rate leads to more accurate results.
subsample	0.7	Smaller values result in less complex models and thus prevents overfitting.
max_depth	4	The higher the value, the more complex the model and the more likely it is to overfit. Thus, 4 works well in maintaining high accuracy but not overfitting or underfitting.

gamma	1	The higher the value, the more conservative the algorithm is, the less likely it is to overfit.
min_child_weight	3	The minimum number of instances in a child node. A higher value indicates a simpler model and is a good way to prevent overfitting.

This information is from the XGBoost Documentation[5].

It is worth noting that we used grid-searching to find the optimal values for *learning_rate* and *subsample*, but we still provided ranges for the search with the above reasoning in mind.

XGB did the best out of all of the models that we tried since it avoids overfitting and can handle feature interactions well. The multiclass logistic regression models failed due to the data not being linearly separable and the patterns between features being very complex. Additionally, KNN struggled with the high dimensional data and amount of features. The decision tree did not perform as well as XGB since it struggles with complex interactions between features.

6 References

[1] Prediction of Music Genre. Kaggle Dataset, 2021. https://www.kaggle.com/datasets/vicsuperman/prediction-of-music-genre

[2] Spotify for Developers: Web API Reference. Spotify. Retrieved December 2024 from https://developer.spotify.com/documentation/web-api/reference/get-audio-features.

[3] Leonhardt, Juan Francisco. "Music Genre Classification: A Machine Learning Exercise." Medium, 10 May. 2024, https://medium.com/@juanfraleonhardt/music-genre-classification-a-machine-learning-exercise-9c83108fd2bb

[4] Senavirathne, Navoda. "Music Genre Prediction Using Audio Features." Medium, 12 Dec. 2022, medium.com/@navodas88/music-genre-prediction-using-audio-features-96dea4e71ad3.

[5] XGBoost Parameters. ReadTheDocs. Retrieved December 2024 from https://xgboost.readthedocs.io/en/stable/parameter.html.