Spotify Per-Song Genre Classifier

Group 100 ML Project

Demi Tran, Yixin Wei, Farah Alkadri, Jacob Rogers, Brian Lee

Navigation

Go to

- Proposal
- Midterm Checkpoint
- [✓ Final Report]

Final Report

Intro

In recent decades, the rise of music streaming applications have made music much more accessible, and furthermore, the idea of genres have become a popular way for people to categorize their musical preferences. [1]. However, Spotify does not label its songs with their respective genres: instead, genres are attached to artists. This makes the music listening experience inconvenient, as artists are not necessarily bound to a single genre [2] and may have some songs that do not align with that person's taste. Studies show that genres can be defined primarily through the instrumental composition of the song track.[3]. With this in mind, we plan to use **Spotify's API** and their extensive track dataset to develop a model that can correctly identify song genre. To lead a supervised learning model, we plan to append **FMA's** data on song genre to check our model.

Problem Definition

Users spend a considerable amount of time sifting through songs from an artist to find songs that align with their specific genre preferences. By developing a model that tags each song with its corresponding genre based on a set of audio features, users can conveniently explore artists that interest them without having to worry about out of place songs.

Methods

Data Sourcing and Processing:

To be able to perform supervised learning, we needed a dataset that included a "genres" feature. Unfortunately, the Spotify API does not provide per track genre classification so we had to source each track's genre from the FMA dataset and append it to our Spotify dataset. By sampling a portion of the FMA dataset and querying Spotify, we were able to form a list of tracks that included its genre and corresponding audio features (defined by the Spotify API). The order of the features are as follows:

0: Track (String), 1: Artist (String), 2: Genre (Categorical), 3:danceability (Numerical) , 4: energy (Numerical), 5: key (Numerical), 6: loudness (Numerical), 7: mode (Numerical), 8: speechiness (Numerical), 9: acousticness (Numerical), 10: instrumental (Numerical), 11: liveliness (Numerical), 12: valance (Numerical), 13: tempo (Numerical), 14: type (String), 15: id (String), 16: uri (String), 17: track_href (String), 18: analysis_url (String), 19: duration_ms, 20: time_signature

From there, we dropped any non-numerical or categorical data to be able to run our data through our preprocessing algorithm and supervised machine learning model. It is important to note that our yield was quite low with only 10% of our sampled songs from the FMA dataset being able to successfully query through the Spotify API.

To implement our algorithms and models, we used Scikit-learn's library [6].

Preprocessing: PCA

For our first preprocessing method, we used Principal Component Analysis (PCA). We chose PCA as our pre-processing method because it allows us to reduce the number of features from the original dataset while still retaining as much of the original information as possible. A couple main reasons for feature reductions are that it allows us to decrease the time needed to train our model and also help avoid the possibility of running into overfitting issues where it could perform quite well on the training data but very poorly on the new testing data. Additionally, PCA would also allow us to see how correlated features are in relation to each other by finding the variance. By maximizing the variability and only collecting the features that are the most unique, we could ignore the features with lower variance (meaning they're more related) and only focus on the ones that are important. Specifically for our dataset, a lot of the features in the Spotify API are more niche such as liveliness and danceability which can be subjective. Since we did not have a great definition for some of the features, we were interested if some could be reduced or if we could find any correlation between them for contextualization purposes.

Machine Learning Algorithm 1: Naive Bayes

Since our focus for our model is categorization, we chose a machine learning model that supports such. Naive Bayes is a well known and utilized model for classification which supports smaller datasets well so we decided to use it as our first machine learning model. To observe how effective PCA was at reducing the dataset's features, we made 3 models:

1. Naive Bayes trained on every feature
2. Naive Bayes trained on the top 2 features from PCA
3. Naive Bayes trained on the top 3 features from PCA

Machine Learning Algorithm 2: K-Nearest Neighbors

To further explore classification methods suitable for our dataset, we implemented KNN algorithm as it is a non-parametric, instance based machine learning method which classifies data points based on the majority vote of their nearest neighbors in the feature space. KNN can capture complex relationships without the assumption of an underlying probability distribution.

Using a dataset from Spotify, we applied KNN to classify the songs into genres based on their audio features. We focused on scaling the features into a uniform range and encoding genre labels into integers for compatibility with our ML models. For model training we split the data into 80-20 ratio, reserving 20% for testing to evaluate our model's performance. Then, we trained the KNN classifier on the scaled and labeled training data.

Machine Learning Algorithm 3: Random Forest

For the accuracy of our genre classification model, we used the Random Forest algorithm which constructs multiple decision trees during training and outputs the classification of the individual trees. Random Forest is suitable for our dataset as it can handle high dimensional data and can reduce overfitting. It manages the large numbers of features from datasets without significant reduction in the dimensionality which is crucial for us given the numerous audio features from the Spotify API.

Our implementation of the Random Forest classifier included 100 trees configuration and a maximum depth of to balance between learning the training data details and preventing overfitting. We trained the model on 80% of the preprocessed data where we normalized the features to ensure that they equally contribute to the model's decision making. We used the remaining 20% of the data to evaluate the model's performance.

In order to achieve a more optimal accuracy on the testing data, the hyperparameters of the algorithm (number of estimators and maximum tree depth) were tuned manually, while a random number that gave good results was chosen for our random seed (random state). First, the number of estimators was chosen to be relatively low for a quick training speed, but high enough to sufficiently reduce variance. Then, a few values of maximum tree depth were tested for accuracy and 15 was settled on since many more than that led to extreme overfitting, and fewer did not capture enough meaningful information about the training set.

For performance metrics, we tracked the general precision, accuracy, and f1 scores as they are common indicators of the performance of classification models.

Results & Discussion

Naive Bayes:

Table 1: Variance for each Feature

	Feature	Variance
0	energy	0.4801
1	loudness	0.4748
2	acousticness	-0.4292
3	valence	0.3926
4	danceability	0.2279
5	tempo	0.2182
6	instrumentalness	-0.2131
7	duration_ms	-0.1639
8	time_signature	0.0975
9	mode	0.0928
10	key	-0.0926
11	liveness	0.0833
12	speechiness	0.0170

Our group decided to utilize Principal Component Analysis (PCA) for our initial preprocessing to find the features that have the highest variance to represent our dataset and eliminate the features that are less relevant. Our dataset only contained 14 features so we did not find it necessary to adjust the confidence level of the algorithm and just ran it as is. We were able to reduce the total features from 13 to only 3 by only accepting the features with variance greater than 0.4. We chose the cutoff to be 0.4 because the rest of the features' variances were comparatively low. With PCA, reduced features that remain along with their variance are: loudness (0.474814), energy (0.480126), and acousticness (0.429154).

Table 2: Testing Results

	Model	Testing Score
0	Naive Bayes	0.4020
1	2D PCA Classification	0.4950
2	3D PCA Classification	0.4540

As described before, we implemented 3 versions of the General Naive Bayes (GNB) Model to adequately explore how effective our feature reduction process was. We decided to graph both the top 3 and top 2 features as described from our PCA algorithm as we were curious of any differences. After generating the output, we observed that they show the same trends. Both PCA models are very similar to each other. More importantly, they both were very similar to the Naive Bayes score. This means that our feature reduction was effective and properly captured our dataset without losing too much information. Additionally, the models with PCA applied generated scores are higher than Naive Bayes (table 2). We can most likely attribute this to the noise reduction occurring when reducing features.

That being said, our scores are still generally low. A good model's score is closer to 100 but we were only able to generate an output that scored in the mid 40s (table 2). This could be attributed to the low yield of our querying process.

Figure 1A: 2D Visualization

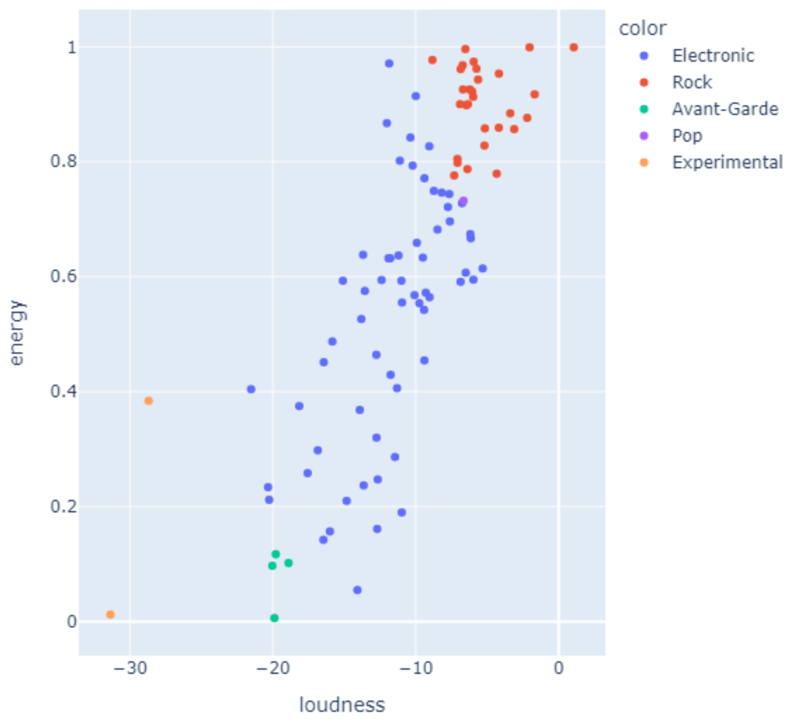
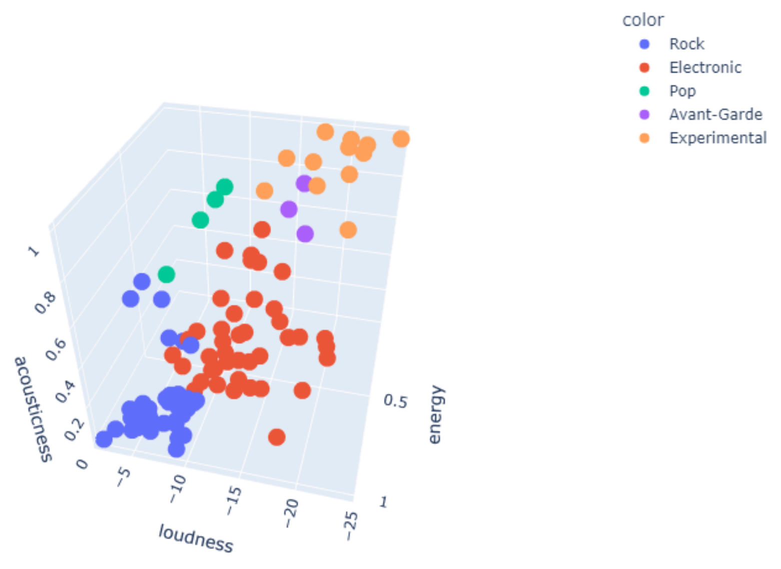


Figure 1B: 3D Visualization



Despite sampling around 4500 data points from the FMA set, we were only able to successfully query 10% of the samples. This led to our model generally being inaccurate. From both the 2D and 3D plots shown in figures 1A and 1B, with limited dataset, some of the genres such as “experimental” are less well-trained compared to genres like “electronic” where we have an adequate amount of data which have much better accuracy. This was most likely caused by our dataset containing an overabundance of “electronic” tracks so our model was most likely overfitted to “electronic”. However, from the graphs, we can still observe that our genre predictions are generally clustered together which means that our features had some sort of correlation to each genre.

Through General Naive Bayes Classification, we were able to get the precision, recall, and F1-Score metrics, as defined below, for two models focused on different subsets of features: the two (loudness & energy) and three (loudness, energy, & acousticness) largest features in terms of variance.

- **Precision:** measures how well a model correctly predicts positive outcomes
- **Recall:** measures how well a model can correctly identify positive instances
- **F1-Score:** measures the accuracy of a model based on its precision and recall scores
- **Support:** the number of actual occurrences in the dataset

Table 3: 2D Classification Report

	Genre	Precision	Recall	F1-Score	Support
0	Avant-Garde	0.2500	0.1200	0.1700	8
1	Electronic	0.5200	0.7300	0.6100	44
2	Experimental	0.5000	0.1100	0.1800	9
3	Pop	0.0000	0.0000	0.0000	15
4	Rock	0.4800	0.6700	0.5600	21

Table 4: 3D Classification Report

	Genre	Precision	Recall	F1-Score	Support
0	Avant-Garde	0.3300	0.1200	0.1800	8
1	Electronic	0.7100	0.5500	0.6200	53
2	Experimental	0.1700	0.2500	0.2000	8
3	Pop	0.0000	0.0000	0.0000	12
4	Rock	0.3200	0.7500	0.4500	16

As shown by the figure above, the Electronic genre has the highest precision (0.52 for 2D classification and 0.71 for 3D classification) while Pop did the poorest due to the low sample data we have, leading to the value 0.0 being displayed for all metrics. Additionally, the Electronic genre had nearly the exact same F1-Scores - 0.61 and 0.62 - despite the precision and recall being seemingly flipped between the two classifications: precision and recall of 0.52 and 0.73 for 2D and 0.71 and 0.55 for 3D. Apart from those discrepancies, all of the metrics maintain the same proportions between the 2D and 3D models.

Our group also modeled all of the features (before feature reduction is applied) and compared the precision, recall, and F1-Score metrics with the two aforementioned 2D and 3D models returned by PCA. We found that the results we got from both are generally similar to each other which means that the subset of features remaining after feature reduction represents the original dataset very well. As shown below, we can see that the training and testing data scores are comparable to that of the 2D and 3D classifications as well:

Table 5: All Classification Report

	Genre	Precision	Recall	F1-Score	Support
0	Avant-Garde	0.0000	0.0000	0.0000	16
1	Electronic	0.4100	0.7800	0.5400	36
2	Experimental	0.3300	0.0700	0.1100	15

	Genre	Precision	Recall	F1-Score	Support
3	Pop	0.2500	0.3300	0.2900	9
4	Rock	0.5000	0.3300	0.4000	21

Compared to the previous figure with the post-PCA reduction features, we can see that with 2D, we obtained a test score of 0.4948, 0.4536 for 3D, and 0.4021 for all features. By conducting a percent difference calculation, we can see that we have a 20.67% difference for 2D and 12.04% difference for the 3D model. This means that our 3D model did better in representing all the features and we're only looking to improve the percent difference as we train more data.

In summary, we conducted both the pre-processing and algorithm implementation to predict the genre of a song based on different features from the Spotify API. The preprocessing is done using PCA to help us eliminate the less useful features and improve the run-time of our model. For our actual algorithm, we implemented the General Naive Bayes classifier using both 2 features, 3 features, and all features and made a comparison of each one to the original. Additionally, we also learned that because of the imbalance in our ground truth data size for the different genres, we did achieve a much higher precision, recall, and f1 score for the Electronic genre due to simply having more data. Meanwhile, the other genres are performing worse due to insufficient amount of training data. While our accuracy rate isn't what we're aiming to achieve as of now, we believe that with an optimized data retrieval program we can greatly improve not only the size of our dataset but also the accuracy because then we would have more sample data to train our model on.

Disclaimer:

After discussion of the Naive Bayes model, there was an overall consensus that the dataset would have to be expanded for more meaningful results. The code was expanded to do such (see /archived/spotify_song_classifier copy.ipynb and /archived/spotify_api copy.py in GitHub repo) but during this time, Spotify API stopped supporting their "Get Track's Audio Features" feature as of 11/27/2024 [8]. As this data was crucial in training our models, we were forced to use the original small dataset to train the rest of our machine learning models.

K-Nearest Neighbors:

Table 6A: KNN Classification Report

	Genre	Precision	Recall	F1-Score	Support
0	Avant-Garde	0.0000	0.0000	0.0000	10
1	Electronic	0.6100	0.8400	0.7100	45
2	Experimental	0.3300	0.1000	0.1500	10
3	Pop	0.5800	0.4400	0.5000	16
4	Rock	0.4700	0.4400	0.4500	16

Table 6B: KNN Confusion Matrix

	0	1	2	3	4
0	0	9	0	1	0
1	1	38	2	1	3
2	3	5	1	1	0
3	0	4	0	7	5
4	1	6	0	2	7

The resulting general KNN score or mean accuracy, fell at 0.546. A score closer to 1 represents high accuracy so this score is quite low. Considering that KNN relies on Euclidean distance, it is possible that our features were not varied enough to clearly define clusters. As seen back in Table 1, the variance for features were generally low which most likely affected how the distance for the KNN model were calculated and analyzed. We can also assume that this number could be improved if the dataset was more vast considering that most of the testing data points tested for the Electronic genre.

When looking at the same performance metrics we used for Naive Bayes (Table 6A), it is clear that there was not enough “support” or testing data points to get any usable metrics for certain genres (Avant-Garde and Experimental) because our training was mostly on Electronic music. The confusion matrix (Table 6B) explains this as it showed that no “Avant-Garde” or “Experimental” songs were able to be correctly identified during testing. With the remaining three genres, all performance metrics were generally low besides Electronic music’s precision and f1 (0.62 precision, 0.87 recall, and 0.72 f1). Considering that recall and f1 mostly refer to identifying true positives, it shows that our model was good at determining true positives but bad at identifying negatives. This is likely due to our dataset mostly consisting of Electronic music which overfitted the model to it and in turn, affected the accuracy for the other two genres.

Random Forest:

Table 7: Random Forest Classification Report

	Genre	Precision	Recall	F1-Score	Support
0	Avant-Garde	0.5000	0.1000	0.1700	10
1	Electronic	0.6100	0.8400	0.7100	45
2	Experimental	0.6700	0.4000	0.5000	10
3	Pop	0.3800	0.3800	0.3800	16
4	Rock	0.5500	0.3800	0.4400	16

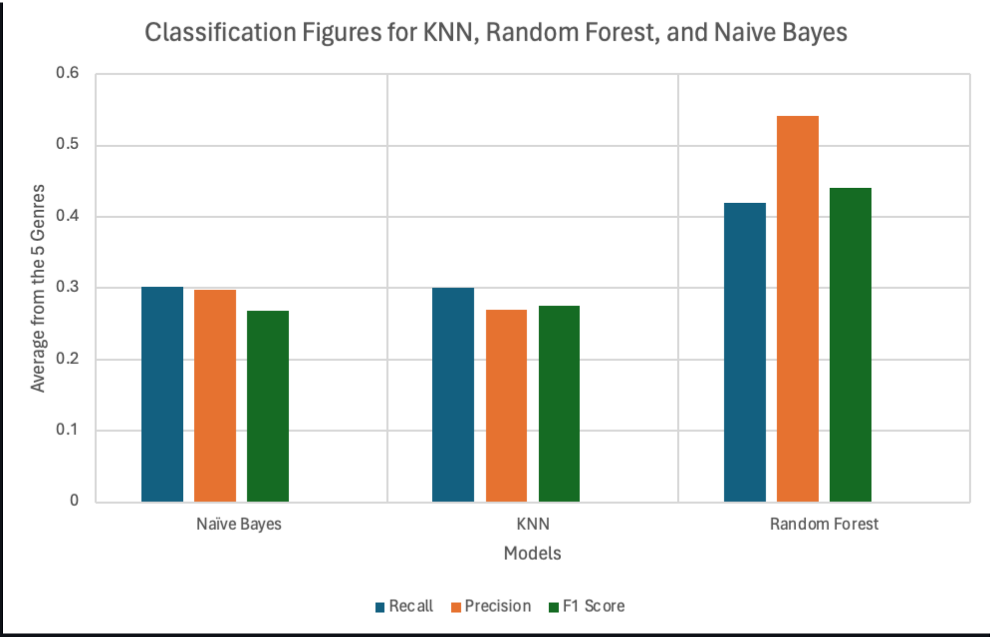
The average accuracy score of the testing data for our Random Forest implementation was also quite low at 0.567. This is most likely due to the same reasons as KNN as mentioned previously. The performance metrics for Electronic, Pop, and Rock tracks also generally showed similar results as what was discussed in KNN. Random Forest’s threshold system does not rely on feature variability so we were able to get metrics for Avant-Garde and Rock, something we were not able to consistently get for the other models. Interestingly, Avant-Garde, Experimental, and Rock music’s precision was higher than its recall and f1 scores (Table 7), opposite of what we observed in KNN. This indicates that the model did not output too many false positives and instead had trouble with identifying positives, resulting in many false negatives.

Overall Comparison:

Table 8: Average Accuracy Scores Across All Models

	Model	Testing Score
0	All-Feats Naive Bayes	0.4020
1	K-Nearest Neighbors	0.5460
2	Random Forest	0.5670

Figure 2: Overall Performance Metrics Chart



The Random Forest classifier emerged as the most effective model outperforming Naive Bayes and KNN (Figure 2). Accuracy scores for Random Forest were approximately 56.7% compared to 50.5% for KNN and 40.2% for Naive Bayes (Table 8). This high performance can be explained by the ability of Random Forest to handle data variance and feature dependencies.

For consistency, we used Naive Bayes with all features, aligning it with KNN and Random Forest, which also utilized all features. Naive Bayes recorded a precision and recall of 0.00 for the Avant-Garde and Experimental genre indicating failure to correctly classify any instances of this genre. The reason relies on the fact that Naive Bayes assumes that all features are independent; however, in our dataset musical attributes can interact in complex ways. On the other hand, KNN relies on Euclidean distance to identify the nearest neighbors. Due to the high similarity between genres such as Electronic and Experimental, overlapping occurred between data points which made it challenging for KNN to distinguish between genres. This led to zero precision and recall scores for some categories.

Random Forest was able to address these issues by utilizing an ensemble of decision trees that split data based on threshold values to achieve pure branches. This is less reliant on data variance and does not assume feature independence. Thus, Random Forest was able to achieve non zero precision and recall for all of the five different genres that we have used. This indicates that Random Forest is more effective at handling more complex and less distinct genres than Naive Bayes and KNN.

It is important to note that if we had more data points, Naive Bayes and KNN might have resulted in better models. A larger dataset could help these models capture the underlying patterns more effectively, potentially improving their performance in classifying the genres.

Next Steps:

The main limitation during this entire exploration was our dataset. Finding a new sizeable dataset that contains tracks, its genres, and audio content would most benefit this exploration. With that being said, if Random Forest remains our best machine learning model, it would be good to continue tuning it with the new dataset.

Our exploration proves that there are some contributing factors that help categorize genres. With a more robust dataset and proper training, a genre classification system can be implemented into Spotify or other platforms so that listeners are able to have an easier time navigating tracks and discovering new music.

References

[1] J.-J. Aucouturier and F. Pachet, “Representing Musical Genre: A State of the Art,” Journal of New Music Research, vol. 32, no. 1, pp. 83–93, Mar. 2003, doi: https://doi.org/10.1076/jnmr.32.1.83.16801.

[2] V. Muchitsch and A. Werner, “The Mediation of Genre, Identity, and Difference in Contemporary (Popular) Music Streaming,” Twentieth-Century Music, pp. 1–27, Feb. 2024, doi: https://doi.org/10.1017/S1478572223000270.

[3] S. T. Mace, C. L. Wagoner, D. J. Teachout, and D. A. Hodges, “Genre identification of very brief musical excerpts,” Psychology of Music, vol. 40, no. 1, pp. 112–128, Feb. 2011, doi: https://doi.org/10.1177/0305735610391347.

[4] Spotify, “Web API | Spotify for Developers,” developer.spotify.com. https://developer.spotify.com/documentation/web-api.

[5] mdeff, “GitHub - mdeff/fma: FMA: A Dataset For Music Analysis,” GitHub, 2016. https://github.com/mdeff/fma.git (accessed Oct. 04, 2024).

[6] Scikit-learn, “scikit-learn: machine learning in Python,” Scikit-learn.org, 2019. https://scikit-learn.org/stable/.

[7] Spotify, “Web API | Spotify for Developers,” developer.spotify.com. https://developer.spotify.com/documentation/web-api.

[8] Spotify. “Introducing Some Changes to Our Web API.” developer.spotify.com, 27 Nov. 2024, developer.spotify.com/blog/2024-11-27-changes-to-the-web-api

Gantt Chart Link

Click Here For Gantt Link

Juypter Notebook GitHub Link

Click Here For GitHub Link

Contribution Table

	Name	Contribution
0	Demi Tran	Data processing, PCA, ML, Jupyter Documentation, Report
1	Yixin Wei	Data processing, PCA, KNN, Jupyter Documentation,
2	Brian Lee	Data processing, Debugging, PCA, ML, Jupyter Documentation
3	Farah Alkadri	Debugging, Report, Video Presentation
4	Jacob Rogers	Random Forest, Report

Click Here For Presentation Link

Click Here For Youtube Link