

Spotify Dance Mode

<https://drive.google.com/file/d/1yDW17PDEdUEcW-IrxU39xnMuv3pPjRa1/view?usp=sharing>

Ishita Raval and Aleksander Rodriguez

University of North Texas

DTSC 4050 Statistical Methods for Data Science and Analysis

Dr. Yang Zhang

May 8th, 2025

Abstract:

Music has been a significant part of our lives. From leisure entertainment to being involved in it through activities like dancing. Our goal with Dance Mode is to introduce Spotify listeners to new music, personalized to them, just for dancing. With more than 100 million songs and more than 675 million active listeners, Spotify is tasked to create personalized mood playlists for its users. In our project, we decided to combine what we know about Data Science to narrow the our scope to the danceability factor. We take a songs features like loudness, tempo, and in what playlists they can be found in and make a predictive model out of them to determine if they have high danceability. By doing this we would like to improve on the overall Spotify user experience, whether they're there to simply listen or to dance.

Introduction & Motivation:

Music is something we listen to daily, whether that be while we get ready in the morning, driving in a car, working out or studying for an exam. We all listen to music, music holds so much value in our emotion, mood, and lives. This is why our group was pulled to do this project in attempting to create a dance mode. Our group are avid Spotify users who enjoy dancing to music but for us to dance to a song it must have certain features that make a song danceable, that is what we sought out to find.

The data we are using is a dataset consisting of descriptive song features that include information about the song such as the artist's name, album name and release date. It also includes audio features include key, valence, danceability and energy which are results of Sportify's audio analysis. This dataset will allow us to start building our data analysis around the danceability audio feature. Our group will use this dataset, but it will need to be cleaned, transformed, analyzed, and then lastly, made into a model. Before starting our project well also explore other projects and methodologies that can help us learn or improve our project/model. Our three contributions are to fully analyze our model, test our correlation to ensure they are correct, and make an efficient model to predict danceability in songs.

Existing Methodologies:

- **Spotify Recommendation System-** Spotify's recommendation system is centered around audio analysis, natural language processing and collaborative filtering. It uses a vast range of data about music to understand trends and relationships. Spotify uses a

large variety of data and uses natural language processing which we do not use. We want to focus on the songs specifically.

- **Music Recommendation System**- This project trained a machine learning model to accurately predict and recommend music to users through the following variables: acoustic-ness, danceability, energy, instrumental-ness, and more. This used similar data our put was an overall song recommendation system compared to our which is more of a song analyzer. They also used clustering for their model, and we used a light gradient machine model.
- **Explore Spotify**- This is an app that explores Spotify to give a user's top artists and tracks to in turn, create a playlist for them. This project was able to connect Spotify to their own app and used HTML and JavaScript. The project wanted to make an app for a playlist a user could enjoy. This could be something our model could be used for, and it gives an idea of how to create an app. Very insightful for our models/project's future.
- **Spotify Song Recommendation**- With the use of Spotify API, this project generated song recommendations based of off song data in a given playlist. They used Jupyter to create a neural network model. They created a database compared to us already having a dataset. We cleaned and analyzed our data compared to them focusing on building their database and creating a model.
- **Spotify Recommendation System**- This project aimed to recommend songs to help Spotify users discover new music and keep them engaged with the platform. The project consists of finding the best model to complete their goal. They ended up only building visuals and was not very informative for our project.

Data:

Our data was found on kaggle.com where the one created the dataset used an API from Spotify to scrape song data to make the dataset we used. It consists of 24 columns and 3146 rows. The dataset is arranged from Spotify's API using python scripts to extract songs and their associated audio and descriptive features. Below are the attributes in the dataset with a description of each.

Attribute	Description
Energy	Measure of intensity and activity.
Tempo	Speed of a track. Measured in beats per minute (BPM).
Danceability	Score describing how suitable a track is for dancing
Loudness	Loudness of a track in decibels (dB)
Liveness	The likelihood of a track being performed live
Valence	Musical positivity of the song. High valence is happy. Low valence is sad or angry.
Speechiness	Measures the presence of spoken words.
Instrumentalness	Values closer to 1.0 suggest solely instrumental tracks.
Mode	Indicates the modality of the track.
Key	The musical key
Duration_ms	The length of the track in milliseconds.
Acousticness	A confidence measure of whether a track is acoustic (1) or not (0).
Track Name	The name of the track.
Track Artist	The artist(s) performing the track.
Track Album Name	The album in which the track is featured.
Track Album Release Date	The release date of the album contains the track.
Track ID	A unique identifier assigned to the track by Spotify.
Track Album ID	A unique identifier for the album.
Playlist Name	The name of the playlist where the track is included.
Playlist Genre	The main genre associated with the playlist (e.g., pop, rock, classical).
Playlist Subgenre	A more specific subgenre tied to the playlist (e.g., indie pop, punk rock).
Playlist ID	A unique identifier for the playlist.
Track Popularity	A score (0–100) which is calculated based on total number of streams in relation to other songs.

Table.1: Table explains all the attributes in the dataset. It's a data dictionary.

The only preprocessing I did was highlighting columns green when we were deciding to keep and remove columns in Excel. We mainly deleted any unique variables or and some text variables that would not hold any value to our model.

Problems:

Generally, music recommendation systems recommend users songs they will like, based off of their music history. Often, this means that users will be introduced to popular music in the category they prefer. Our goal with Dance Mode, is not only to recommend users to danceable songs but also to music that is new to them and is able to expand their listening range.

Cleaning and Transforming the Data

To begin our cleaning process, we decided to deal with missing values. Since there was only a small sum of missing values we decided to drop the rows with nulls. There was a total of 14 nulls so there was no need to keep them. Next, we ensured that the data was formatted correctly from .csv file to the data frame in python. All the attributes contained their respected data type so there was no need for use to make any data conversions. We then had to map each genre, subgenre, and playlist_name for our heatmap to work because the heatmap only takes numerical data. Finally we ended up dropping the track_album_release_date, id, and track_album_name because they provide no value to our model and aren't numerical data types.

Analyzing and Visualizing the Data:

Heatmap: To analyze our data after our cleaning and transforming process we used a heatmap to help decide which variables would be best for our model and which variables would be of no value or affect our model negatively. The heatmap told us speechiness, energy, valence, and loudness were strong positive correlation. It also taught us that duration_ms, instrumentalness, and acousticness are strong negative correlations. Both negative and positive correlations are what we want. A positive correlation consists of x and the y variable both going up and down at the same time. Negative correlation would be the y variable going up but the x going down and vice versa. In the image below is our heatmap, with our heatmap we want numbers closer 1 and -1 and far from 0.

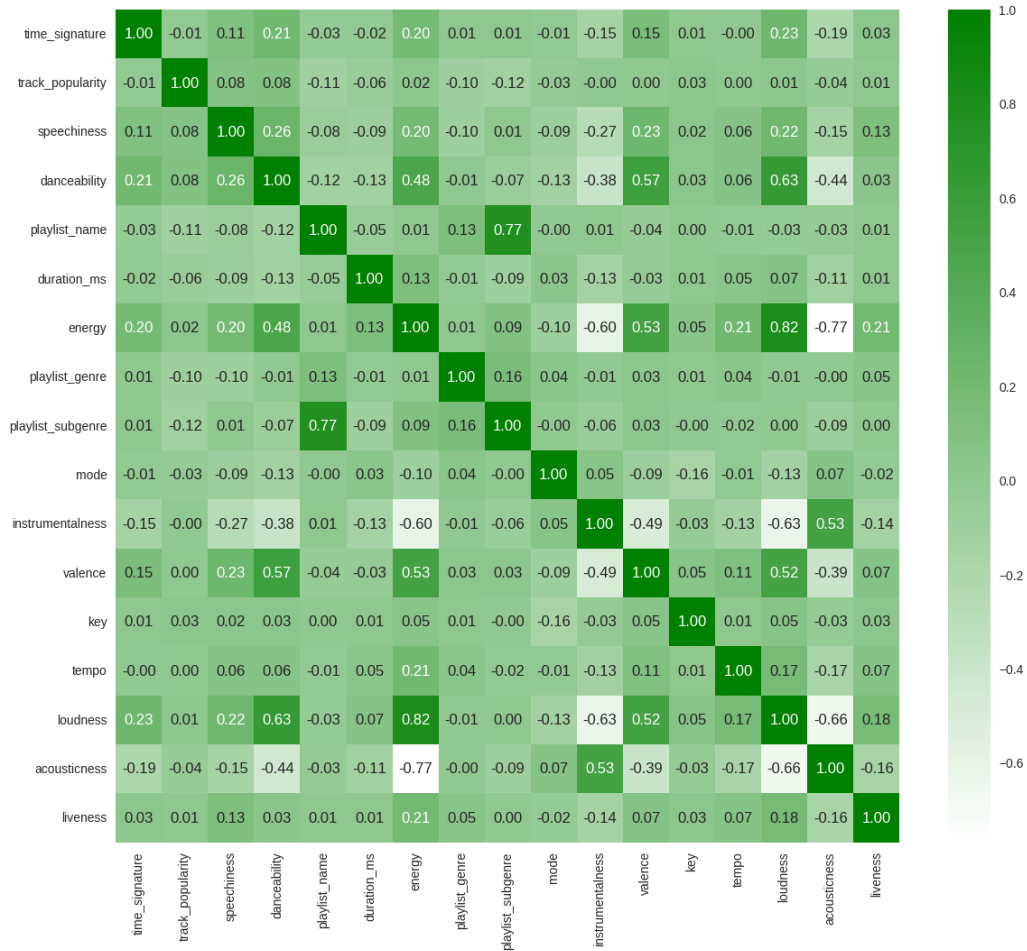


Image.1: Heatmap showing how the variables are correlated with each other.

Compare Models: Our creation of the model began by using the Pycaret library in Python. From there we utilized the `setup()` and `compare_models()` functions to make danceability the target variable and find a model that would best be able to statistically depict and present our data. The top three models were the Light Gradient Boosting Machine (LightGBM) model, Extra Trees Regressor Model, and the Gradient Boosting Regressor. This is shown below in the image that contains statistics of each model.

```
compare_models(exclude=['catboost'], n_select = 10)
```

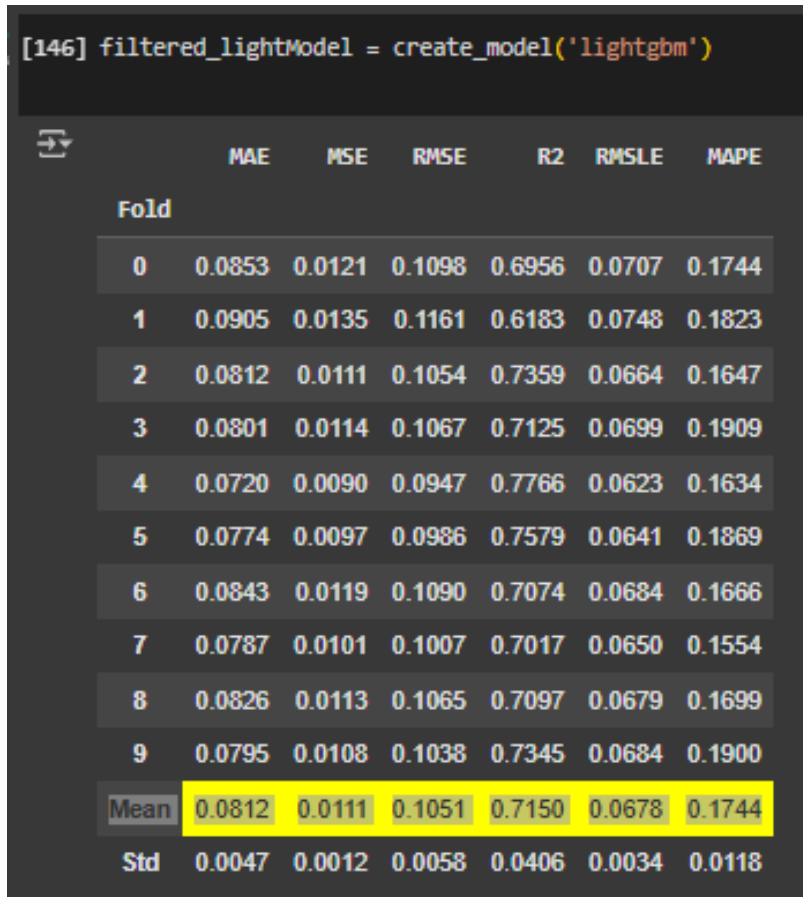
	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
lightgbm	Light Gradient Boosting Machine	0.0812	0.0111	0.1051	0.7150	0.0678	0.1744	0.3950
gbr	Gradient Boosting Regressor	0.0849	0.0119	0.1088	0.6950	0.0703	0.1858	0.7320
et	Extra Trees Regressor	0.0855	0.0120	0.1096	0.6909	0.0703	0.1833	1.0240
rf	Random Forest Regressor	0.0857	0.0122	0.1101	0.6879	0.0709	0.1853	2.1110
xgboost	Extreme Gradient Boosting	0.0857	0.0127	0.1126	0.6739	0.0724	0.1846	0.3360
ada	AdaBoost Regressor	0.1039	0.0159	0.1261	0.5914	0.0815	0.2427	0.3160
lr	Linear Regression	0.1091	0.0184	0.1354	0.5281	0.0875	0.2534	0.0260
ridge	Ridge Regression	0.1091	0.0184	0.1354	0.5281	0.0875	0.2535	0.0240
br	Bayesian Ridge	0.1091	0.0184	0.1354	0.5280	0.0875	0.2537	0.0270
lar	Least Angle Regression	0.1091	0.0184	0.1356	0.5270	0.0875	0.2532	0.0310
dt	Decision Tree Regressor	0.1189	0.0248	0.1574	0.3592	0.1004	0.2438	0.0500
en	Elastic Net	0.1336	0.0267	0.1634	0.3146	0.1087	0.3693	0.0250
lasso	Lasso Regression	0.1530	0.0385	0.1961	0.0161	0.1316	0.4714	0.0230
llar	Lasso Least Angle Regression	0.1530	0.0385	0.1961	0.0161	0.1316	0.4714	0.0230
omp	Orthogonal Matching Pursuit	0.1533	0.0387	0.1966	0.0101	0.1320	0.4728	0.0240
dummy	Dummy Regressor	0.1554	0.0393	0.1980	-0.0039	0.1328	0.4781	0.0210
knn	K Neighbors Regressor	0.1665	0.0456	0.2134	-0.1678	0.1419	0.4945	0.0320
huber	Huber Regressor	0.1801	0.0493	0.2215	-0.2642	0.1442	0.4654	0.0420
par	Passive Aggressive Regressor	0.3153	0.2006	0.4420	-4.1075	0.2444	0.8487	0.0250

Image.2: The image illustrates the `compare_models()` the helped us choose our model for the project.

Experiments and Results:

Once given the comparison between different models, we decided to move forward using the LightGBM model. This model captured the variation of the target an estimated 71.5% of the time. Its Root Mean Squared Error (RMSE) was ~ 0.1051 and the train time was around .3950 seconds, proving to be a highly efficient model for our data set. Upon plotting the data from the LightGBM model, we observed a Train R^2 of 0.935 and a Test R^2 0.714. This tells us that the model and our dataset are highly compatible and that the model can perform well with its given data. With this model we are also provided with a histogram on the far right side. This histogram is a depiction of the model's residuals. The histogram is seen to be centered around 0 and is symmetrical in shape, signifying small errors and a relatively equal ratio between positive and negative errors.

```
[146] filtered_lightModel = create_model('lightgbm')
```



	MAE	MSE	RMSE	R2	RMSLE	MAPE
Fold						
0	0.0853	0.0121	0.1098	0.6956	0.0707	0.1744
1	0.0905	0.0135	0.1161	0.6183	0.0748	0.1823
2	0.0812	0.0111	0.1054	0.7359	0.0664	0.1647
3	0.0801	0.0114	0.1067	0.7125	0.0699	0.1909
4	0.0720	0.0090	0.0947	0.7766	0.0623	0.1634
5	0.0774	0.0097	0.0986	0.7579	0.0641	0.1869
6	0.0843	0.0119	0.1090	0.7074	0.0684	0.1666
7	0.0787	0.0101	0.1007	0.7017	0.0650	0.1554
8	0.0826	0.0113	0.1065	0.7097	0.0679	0.1699
9	0.0795	0.0108	0.1038	0.7345	0.0684	0.1900
Mean	0.0812	0.0111	0.1051	0.7150	0.0678	0.1744
Std	0.0047	0.0012	0.0058	0.0406	0.0034	0.0118

Image.3: Show our creating of the model with a ten fold with each folds stats.

In Image.4 we created a plot to give us a deeper understanding of the performance of our model. The plot shows that the Train R^2 at a .932 meaning the dataset fits the model well and then the Test R^2 at a .714 which means that our model can perform well on new data. The histogram on the far right of the image depicts the distributions of the residuals. They are centered around 0 which allows us to conclude that errors are minimal. The is histogram is symmetrical meaning equal positive and negative errors. Overall, the model is well structured and balanced.

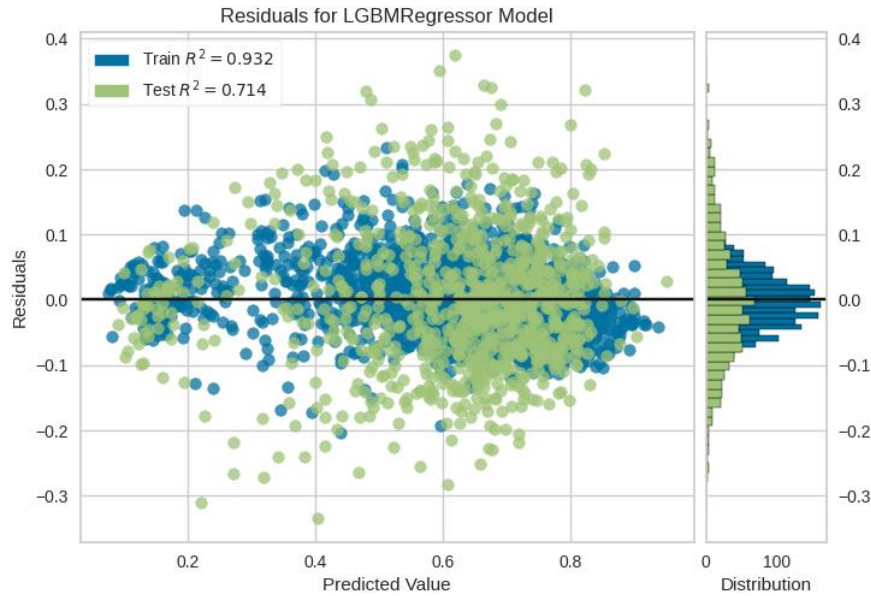


Image.4: Plot Illustrating the performance of the model.

Three Contributions:

As stated before, our three contributions are to test correlations between variables to ensure they are strong, make an efficient model to predict danceability in songs, and to provide a full analysis of our model. Our goal from the start was to make the data clean and useable to test for correlations. This was completed through data cleaning, and mapping variables to allow us to use them within a heatmap. The heatmap was able to provide us with the information we were looking for comparing models. We also ran a correlation and Anova test to ensure that our heatmap was providing correct numbers. The test concluded that the heatmap was producing correct numbers. The heatmap and other tests allowed us to make informed decisions when dropping and keeping columns for our model, which in turn made our model stronger and more efficient.

Through our first contribution, it allowed our second contribution to be accomplished. We used the Pycaret library in python to construct our model. The Pycaret library made creating our model simple as we told it our target value was to predict danceability. Then we used the `compare_models()` to help us determine the best model for our project which was ultimately the Light Gradient Boosting Machine Model. The stats for this model were appealing, depicting efficiency and being a reliable model.

Our last contribution is to provide a full analysis of our model which is completed by comparing the model in a ten fold and using a plot to illustrate its performance. In the ten fold of the model, the R-Squared value was at a .715, a Root Mean Squared Error (RMSE) of .1051, and a Time to Train of .3950 seconds. For a simple model, these numbers are strong, and I believe with some improvement over time they could be better. Our plot in Image.4 was also valuable to complete our last contribution. The Test R^2 is at a .714 which means it is good at predicting new data. The histogram gives us a summary of the distribution of the residuals.

Long Term:

This system has been designed to be flexible to any user's needs. The model created for this project could be in the background for a feature tailored to every user on the app according to the genres they like to listen to. Additionally, it can be effortlessly implemented, allowing anyone to interchange danceability to any other desired variable to predict.

Once this model has passed its refining stage it can be introduced to Spotify as an additional premium feature where users can access danceable, high-energy songs at anytime, anywhere. The ultimate goal behind the creation of Dance Mode is to add another feature to the Spotify app, enhancing the user experience to the max.

Conclusion:

Dance mode is an original innovation driven by the motive of enhancing user experiences and music exploration. With premium subscriptions, Spotify users are able to traverse music libraries with endless skips and streaming capabilities. Through the use of Python and Machine Learning models, we strive to deliver fresh and original music to users. Furthermore, our goal is to encourage exploration and cater to a user's dancing mood by being able to provide perfect music for them.

References:

- Abdelrhmanelruby. “Abdelrhmanelruby/Spotify-Recommendation-System: Spotify Recommendation System Using Spotify Million Playlist Dataset.” *GitHub*, github.com/abdelrhmanelruby/Spotify-Recommendation-System. Accessed 26 Feb. 2025.
- BobbyWilt. “Bobbywilt/Spotify_song_recommender: This Project Leverages Spotify’s API and Provided User Playlists to Create and Tune a Neural Network Model That Generates Song Recommendations Based off of Song Data in Provided Playlists.” *GitHub*, github.com/BobbyWilt/Spotify_Song_Recommender. Accessed 26 Feb. 2025.
- Prasad, Sathish. “SATHISHPRASAD/Music-Recommendation-System: This Repository Contains the Implementation of a Music Recommendation System Using the Spotify Dataset from Kaggle. the System Is Built with Machine Learning Techniques to Suggest Songs to Users Based on Their Listening History and Preferences.” *GitHub*, 2023, github.com/sathishprasad/Music-Recommendation-System.
- Torabi, Nima. “The Inner Workings of Spotify’s AI-Powered Music Recommendations: How Spotify Shapes Your Playlist.” *Medium*, *Beyond the Build*, 26 Apr. 2024, medium.com/beyond-the-build/the-inner-workings-of-spotifys-ai-powered-music-recommendations-how-spotify-shapes-your-playlist-a10a9148ee8d.
- “Understanding Recommendations on Spotify.” *Safety & Privacy Center*, www.spotify.com/us/safetyandprivacy/understanding-recommendations. Accessed 26 Feb. 2025.