Spotify Song Recommender Project

Group 4: Cagan Abney, Jason Cisneros, Joshua Hale and Raheem Yusuff

Inspiration

The inspiration of this project was to take a personal look at how the recommender systems in our everyday lives work. When you are watching Netflix, how do they decide what we should watch next? When you buy something online, how do they decide what we also might like?

Specifically, for this project, if not using a playlist, how does Spotify decide what song I might want to listen to next? As we all love listening to music and are always looking for that new song we can't stop playing or get out of our head, how can we make finding this song easier? During this project we were able to see how machine learning answers some of these questions we have.

Datasets and Data Engineering

We chose a dataset that consisted of Spotify tracks and their respective features. We got this dataset from Kaggle (link found in Works Cited section). This data was originally collected from the Spotify API. Our dataset consisted of the Spotify unique track identifier, the track name, artists, the album name the song belongs to, the song's genre, and how popular each song is. The popularity is calculated by algorithm and is based, in the most part, on the total number of plays the track has had and how recent those plays are. Generally speaking, songs that are being played a lot now will have a higher popularity than songs that were played a lot in the past. In addition to the details of each track, the dataset also consisted of features for each song. These features included track duration, explicitness, danceability, energy, key, loudness, "speechiness", acousticness, and more.

For cleaning this dataset we only had to drop one track that had a null value for a feature. In addition, we also removed all duplicate track IDs. Lastly, we had to drop the column "Unnamed: 0" which had no insightful information on each song.

In order to create insightful Tableau dashboards we also decided to use another Spotify track playlist consisting of the most streamed songs in 2023. This dataset was also from Kaggle. Similarly, this dataset consisted of the song details and features such as track duration, explicitness, danceability, energy, key, loudness, "speechiness", acousticness, and more. In addition, this dataset also consisted of how many Spotify user playlists the song was in, but also other users of other music streaming services such as Apple Music and Deezer. Unlike our previous dataset, it also consisted of the date when the song was released.

Feature Engineering

Before inputting our dataset into a machine learning model we had to do some feature engineering to decide what features the models would train on to make predictions. For our data, we separated the song details such as the track ID, track name, artists, and album name from the rest of the columns of our data set. For our features in the model we decided to use the following columns: 'popularity', 'duration_ms', 'explicit', 'danceability', 'energy', 'key', 'loudness', 'mode', 'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo', 'time_signature'. As ‘explicit’ was a boolean data type, we had to transform this into a float data type in order to scale the features as the ranges did vary for each. Scaling features allow for all the numeric features to be considered as the same with the same magnitude.

After scaling our numeric features, we decided to bring the track genre in as a feature as well. Using the genre of a song may help the model to consider songs in the same genre when giving the user recommendations. As track genre is categorical data, we had to encode this feature which creates a column for each unique value of track genre and gives each song a boolean (0 or 1) value to label which genre the track belongs to. We then combined the track genre encoded data frame with our scaled feature data frame to create our features that the model would use.

Machine Learning

Our objective in this project was to embed a machine learning model into a UI to make predictions on a user input. We decided to create a song recommender product. This application would predict what songs from Spotify a user may like based upon a song the other user already likes.

For this prediction, we decided to use the K-Nearest Neighbors (KNN) model. The KNN model is a supervised classification machine learning model which uses proximity to make classifications or predictions about the grouping of an individual data point. The model takes the features inputted and calculates the distance between the query point and the other data points. This distance can be calculated using several distance metrics such as Euclidean, Manhattan, Minkowski, or Hamming. We used the Euclidean Distance and were able to get “okay” song recommendations. Euclidean distance measures a straight line between the query point and the other point being measured. This is usually the most commonly used distance metric used because of its simplicity, however, with simplicity comes less predictability. The K value chosen by us determines how many nearest neighbors the model will look for.

Advantages of K-Nearest Neighbors (KNN)

- Simple to Implement: KNN is easy to understand and implement, making it a good starting point for beginners in machine learning.
- Non-Parametric: KNN is a non-parametric algorithm, meaning it does not make any assumptions about the underlying data distribution.
- No Training Phase: KNN is instance-based and does not require a training phase. The model is simply memorizing the training instances.
- Interpretability: The decision-making process of KNN is transparent and easy to interpret. It's based on the majority vote of the nearest neighbors.
- Robust to Noisy Data: KNN can perform well even with noisy or incomplete data.

Cosine Similarity

Unlike Euclidean Distance which measures the straight-line distance between two points in Euclidean space. Cosine similarity is a metric used to measure how similar two vectors are in a multi-dimensional space. It calculates the cosine of the angle between two non-zero vectors and provides a measure of the similarities between them.

For example, we have three people buy items at a store:

Customer 1 buys: 1 egg, 1 bag of tortilla, 1 can of salsa.

Customer 2 buys: 100 eggs, 100 bags of tortilla, 100 cans of salsas

Customer 3 buys: 1 dish soap, 1 battery, 1 can of cat food.

According to Euclidean Distance since customer 1 item numbers correspond to the same as customer 3. The model will state that customers 1 and 3 are like each other although they bought completely different items at the store. Cosine similarity on the other hand will be calculated since the items are the same and both customers are similar. It doesn’t consider that customer 1 bought 1 egg and customer 2 bought 100 eggs.

Advantages of Cosine Similarity

- Scale Invariance: this model is not affected by the magnitude of vectors, making it suitable for high-dimensional data where the scale of features may vary.
- Efficiency: It is computationally efficient, especially for sparse data, making it suitable for large datasets.
- Dimensionality Reduction: Useful in text mining and document clustering tasks where the number of dimensions (words) is large, as it helps identify similar documents efficiently.
- Robust to Outliers: Cosine similarity is robust to outliers since it focuses on the direction of vectors rather than their magnitude.
- Interpretability: The similarity score ranges from -1 to 1, making it easy to interpret. A score of 1 indicates identical vectors, 0 indicates no similarity, and -1 indicates diametrically opposite vectors.
- Widely Used: It is popular in information retrieval systems, recommendation systems, and clustering algorithms due to its effectiveness in measuring similarity between documents, items, or users.

Tableau

For this project we created 5 tableau worksheets and combined them into 2 dashboards. The first dashboard, called “Playlist” contains visualizations that show what Artists are found in the most Spotify user playlists and what songs of the artists are found the most. In addition, you can see the comparison of the top artists in Spotify vs Apple. From the visualizations you can see it is similar for the two platforms as Taylor Swift and Bad Bunny are the top 2 for both. Lastly, on this dashboard you will find a bar chart of songs’ danceability combined with a line chart of song’s energy throughout the 2000s. As expected, the higher energy of music the more danceable it likely is to be. This dataset does contain more recent songs, so there is bias in the results, but from the visualization, it can be concluded the early 2000s’ songs had more energy and were more likely to be danced to.

The next dashboard was the “Genre Dashboard". On this dashboard we have 2 visualizations. One being a bubble cart of the most popular genres. Based on the size of the bubbles and the color key, it can be concluded that k-pop, chill and anime are the most popular genres. Once again, this dataset is mostly 2023 released songs, so this is what’s popular right now. On the second visualization, you can see the genre's “Instrumentalness” vs their “Acousticness”. This list allows you to be able to see what are the general features of a song in a genre. By using data visualizations, you are able to find useful insights and tell a story in an interesting way.

Conclusions

In conclusion, recommender systems in our everyday life likely use a supervised classification machine learning from features of the product the algorithm is trying to recommend. For this project we used k-Nearest Neighbors, but it would be interesting to see what companies like Spotify, Apple Music, Netflix, and Amazon recommend songs, movies, and products based on users past experiences. There are endless amounts of opportunities to optimize machine learning models to achieve your expected outcome. This is the fun of machine learning because there is no limit for what you can achieve!

Limitations/Bias

Some limitations included our newly introduced machine learning knowledge. With more experience we would be able to use more complex models or better optimization of our feature engineering. Our recommender is a very simple model, and to get closer to modern day recommenders we would definitely have to include more features and probably a different model. Another limitation was the dataset, the data set of course could not obtain every song in Spotify due to size limitations, so the user can not just input any song. Another limitation was the amount of time to complete this application. As this was a two week project we were not able to incorporate some more UI customization that would provide a better user experience.

The bias in our dataset included the songs that are in the dataset, as this is a sample of songs on Spotify there is a bias to the songs that are available. There may be some older or less known songs that did not get included.

Future Work

For future work, we could attempt to use other classification machine learning models if KNN truly is the best for a recommendation system. Also, within the KNN model, it would be useful to try other distance metrics to see how different our recommendations would be.

Of course with more time we could always better the user experience of the application by adding more user customization such as to provide all the songs on the same album as the user inputted song or allowing for minor errors in user inputs. Also, allowing the user to decide what metric of the KNN model they want to use themselves and allow for them to see how each makes a difference in what is recommended.

If possible, embedding a preview of the songs or the whole song, so the user can hear the songs recommended in the application would be ideal.

Works Cited

Algolia. “Cosine Similarity: What Is It and How Does It Enable Effective and Profitable Recommendations?” Algolia Blog, www.algolia.com/blog/ai/cosine-similarity-what-is-it-and-how-does-it-enable-effective-and-profitable-recommendations/.

IBM. “K-Nearest Neighbors (KNN).” IBM, www.ibm.com/topics/knn.
Kaggle. “Spotify Tracks Dataset.” Kaggle, www.kaggle.com/datasets/maharshipandya/-spotify-tracks-dataset/data.

Kaggle. “Top Spotify Songs 2023.” Kaggle, www.kaggle.com/datasets/nelgiriyewithana/top-spotify-songs-2023.

Scikit-learn. “sklearn.neighbors.NearestNeighbors.” Scikit-learn Documentation, scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html.