1.4 Assignment

Benjamin Melitz, Daniel Jang, Jimin Hong, Tanush Deka

Original Model

The original model used the kmeans library to make 5 clusters of spotify songs. The user inputs their favorite songs by id, and then the model finds songs in the same group to suggest to the user.

Output of the model after song ids are input:

Pros: Used unsupervised learning to make clusters

Cons: Only 5 clusters, Unformatted Output

```
                 track_id                            artists  \
1  4qPNDBW1i3p13qLCt0Ki3A                        Ben Woodward
3  6lfxq3CG4xtTiEg7opyCyx                        Kina Grannis
4  5vjLSffimiIP26QG5WcN2K                     Chord Overstreet
6  6Vc5wAMmXdKIAM7WUoEb7N  A Great Big World;Christina Aguilera
7  1EzrEOXmMH3G43AXT1y7pA                        Jason Mraz

                                  album_name  \
1                                Ghost (Acoustic)
3  Crazy Rich Asians (Original Motion Picture Sou...
4                                    Hold On
6                      Is There Anybody Out There?
7              We Sing. We Dance. We Steal Things.

                track_name  popularity  duration_ms  explicit  \
1          Ghost - Acoustic          55       149610     False
3  Can't Help Falling In Love          71       201933     False
4                   Hold On          82       198853     False
6              Say Something          74       229400     False
7                  I'm Yours          80       242946     False

   danceability  energy  key  ...  mode  speechiness  acousticness  \
1         0.420  0.1660    1  ...     1       0.0763         0.924
3         0.266  0.0596    0  ...     1       0.0363         0.905
4         0.618  0.4430    2  ...     1       0.0526         0.469
6         0.407  0.1470    2  ...     1       0.0355         0.857
7         0.703  0.4440   11  ...     1       0.0417         0.559

   instrumentalness  liveness  valence    tempo  time_signature  track_genre  \
1          0.000006    0.1010   0.2670   77.489               4     acoustic
3          0.000071    0.1320   0.1430  181.740               3     acoustic
4          0.000000    0.0829   0.1670  119.949               4     acoustic
6          0.000003    0.0913   0.0765  141.284               3     acoustic
7          0.000000    0.0973   0.7120  150.960               4     acoustic

  type
1    3
3    3
4    3
6    3
7    3

[5 rows x 21 columns]
```

Model Enhancements: Outlier Removal

The first enhancement was removing outliers from the dataset. This allowed for better representation of the data, and more practical clusters of songs. We used scipy.stats to generate z scores. Outliers were any entries where at least one column had a z score of more than 4 or less than -4 (~1.26% of entries). Without outliers, the model does not have to make small groups for a couple outlier songs and can be focus on the majority of songs

```
# checks if a z score for any column in a row is above 4
# .any ensures that only 1 column has to be above 4 (or below -4) to be an outlier, not every column
# with 3 standard deviations, 4.73% of data are outliers. This seemed too high, so I'm using 4 standard deviations
outliers = ((z_scores > 4) | (z_scores < -4)).any(axis=1)
# makes a dataframe of non-outliers
non_outliers = tracks_df[~outliers]
# prints % of data that are outliers
print(f'{100 - (len(non_outliers) / len(tracks) * 100)}% of data are outliers')
```

Model Enhancements: Other Models

The next enhancement was using other models, like DBSCAN and agglomerative clustering. The agglomerative clustering model took too long on the large dataset, so we used the Birch algorithm in sklearn.cluster instead.

Number of Cluster Comparisons

We tried the silhouette method, but it took too long. Here are results from the elbow test. The elbow test takes the number of clusters where the distortion point levels off. Distortion point measures distance from data points to center of cluster. We chose 15 clusters

5 clusters with value of 423432.3
10 clusters with value of 267103.8
15 clusters with value of 221535.1
20 clusters with value of 195752.4
25 clusters with value of 177833.0
30 clusters with value of 165197.3
35 clusters with value of 154795.6
40 clusters with value of 146217.4

Model Enhancements: More Clusters

As mentioned in the previous slide, we found 15 clusters to be ideal. We enhanced the model by making 15 clusters instead of 5, leading to more diverse clusters.

Model Enhancements: Data Standardization

When looking up how to implement DBSCAN, many sources used sklearn.preprocessing to standardize data. We implemented this with the following code:

```
# normalize data (helpful for all models)
scaler = StandardScaler()
tracks = scaler.fit_transform(tracks)
```

Model Enhancements: Multiple Input Songs

The next enhancement was the ability to input multiple song ids, and have the model recommend songs for every one of them. Originally, the model only recommended songs for the most common cluster, but we updated it so the model gives recommendations for every cluster of song that was input. For every cluster, the model prints the song names that the recommendation is for and the recommended songs

```
# This method gives recommendations for every cluster of songs

# cluster dict has key = cluster and value = set of song ids (no duplicates)
clusters = defaultdict(set)
for i, song in favorites.iterrows():
    clusters[song['type']].add(song['track_id'])

# iterate through every cluster (not every song)
for cluster in list(clusters.items()):
    suggestions = tracks[tracks['type'] == cluster[0]]
    # print out all the songs in that cluster, which are the songs that the recommendation in for
    print(f'suggestion for songs: ', end='')
    for song_id in cluster[1]:
        song_name = tracks_df[tracks_df['track_id'] == song_id]['track_name'].values[0]
        print(song_name, end=', ')
    print('')
    print(suggestions.head())
    print('\n\n\n')
```

Feature Distributions

Plots of the columns from the spotify dataset

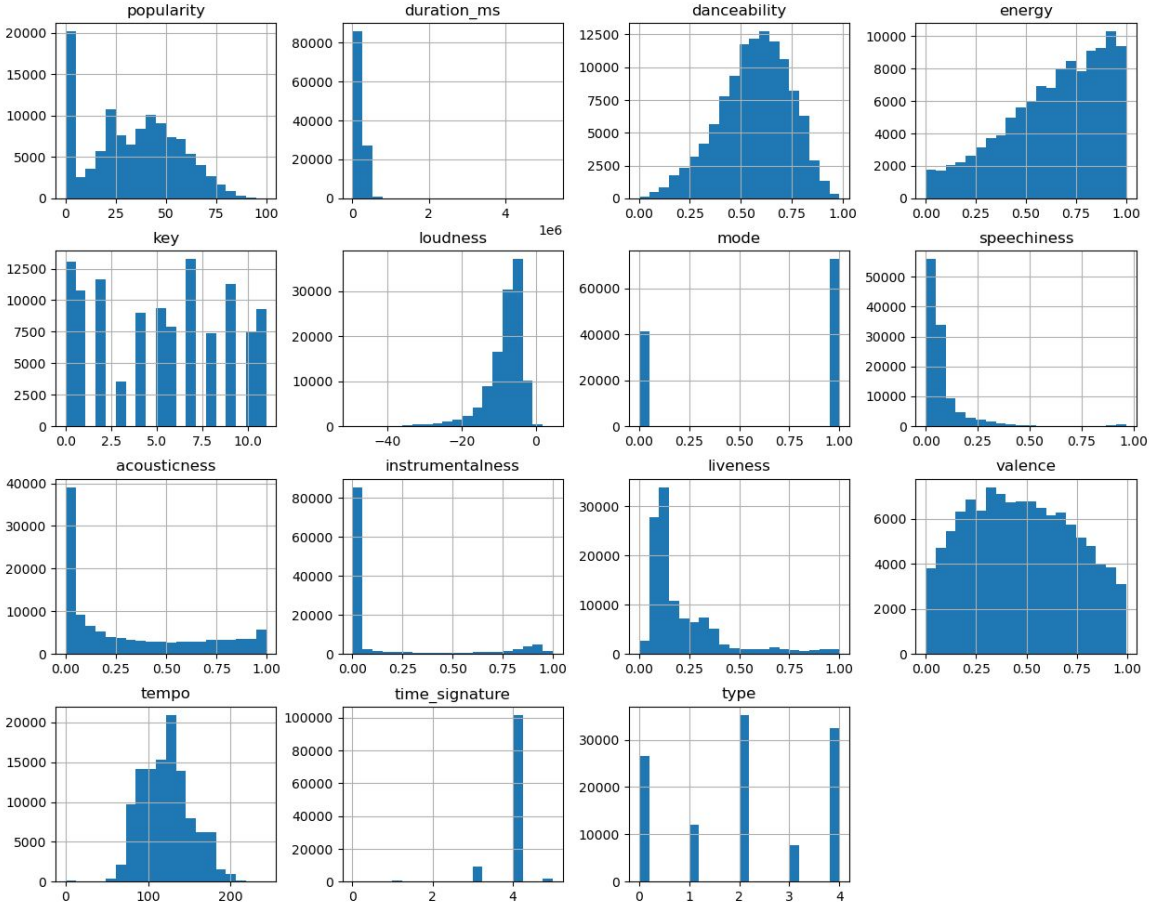
Takeaways:

Popularity is about a normal distribution, but with a spike at 0

Danceability and tempo appear to be in a normal distribution

Accousticness, Instrumentalness, and liveness all have peaks near 0, and low frequencies everywhere else

Energy and loudness generally increase in frequency as increases

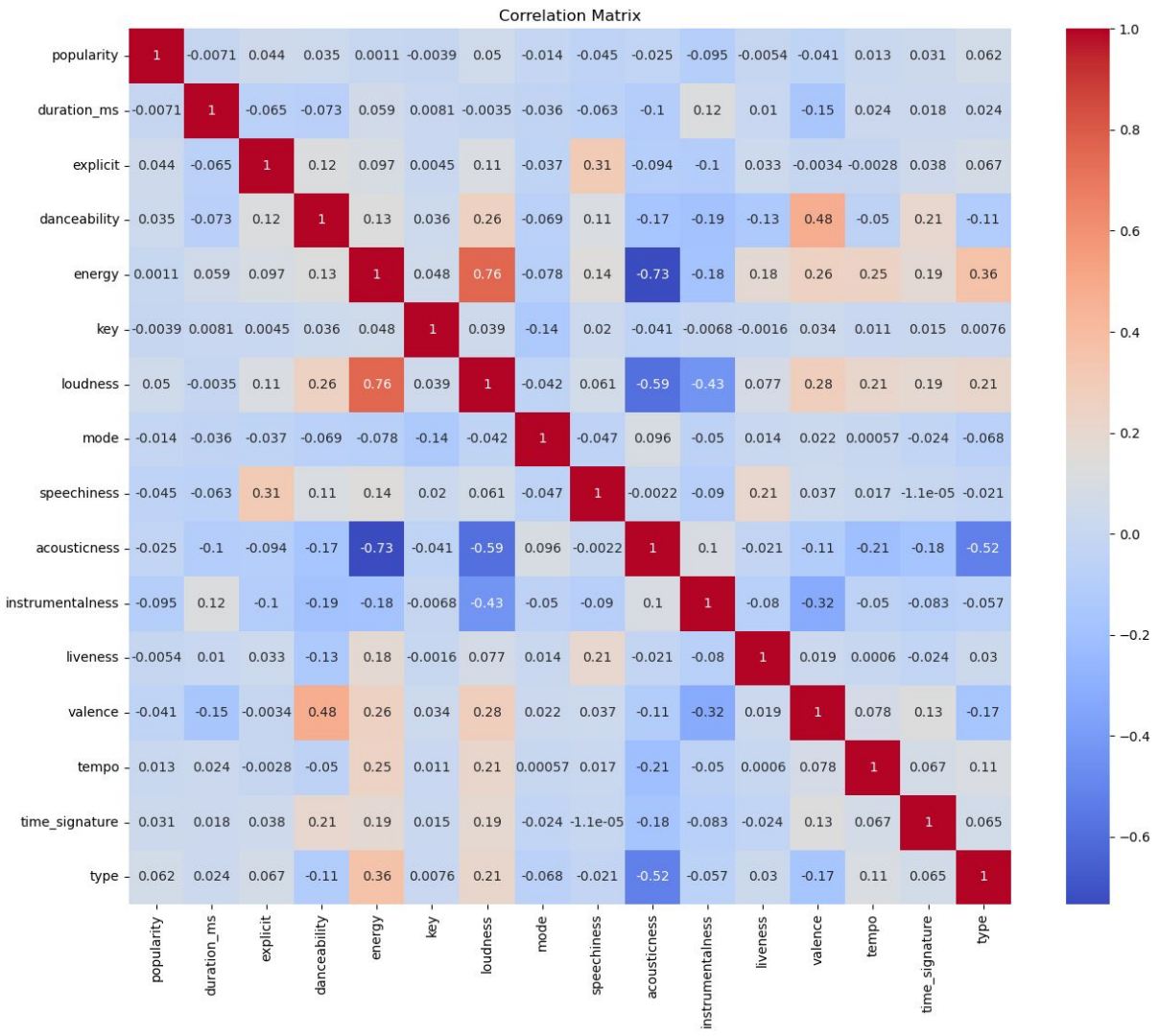


Correlation Matrix

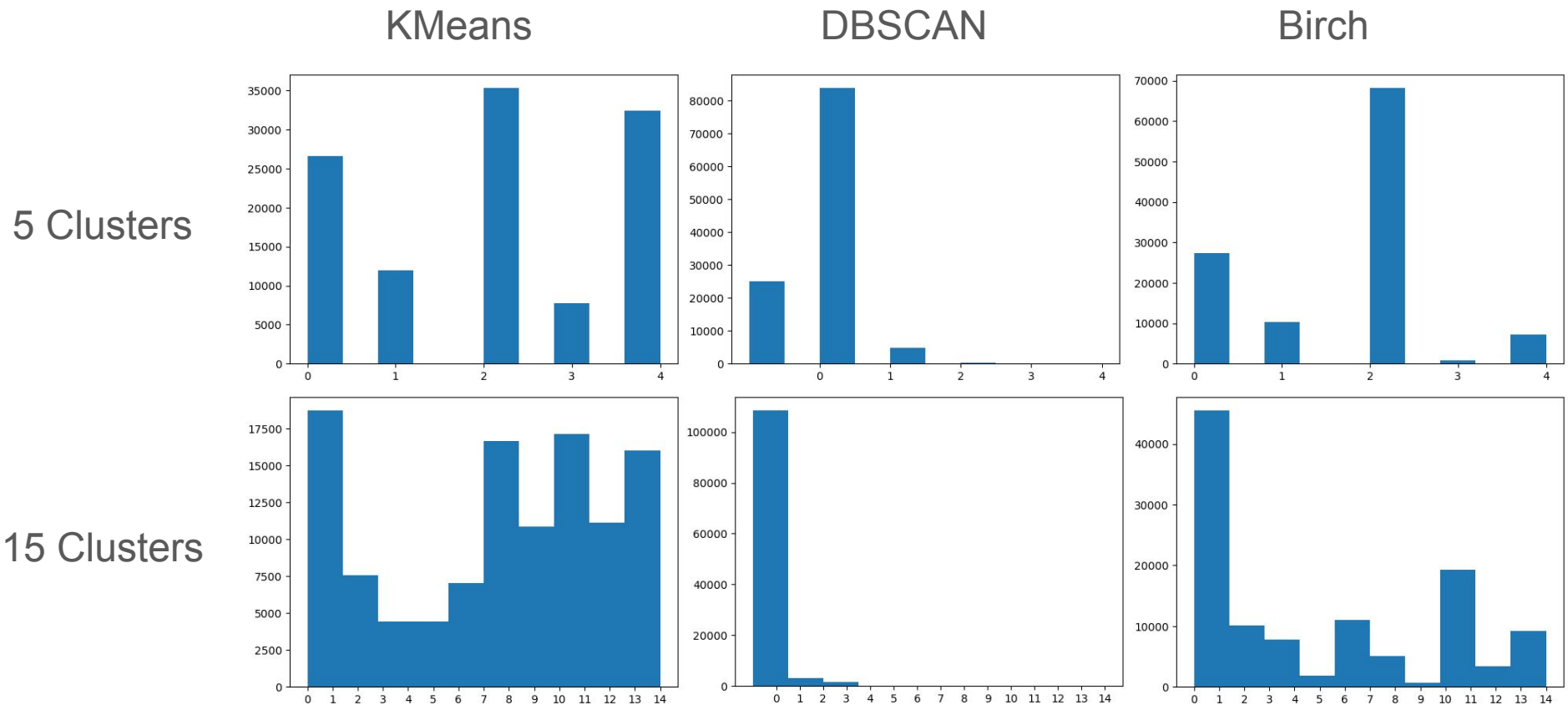
Correlation matrix of the columns from the spotify dataset

Biggest positive correlations:
Energy and Loudness
Valence and Danceability

Biggest negative correlations:
Energy and Accousticness
Loudness and Accousticness



Model Cluster Comparison



Model Choice

We chose to use KMeans, because it had the most even distribution and was faster than the other clustering algorithms

User Feedback

To collect feedback, first we ask the user if they want to submit feedback. If the user enters one of the following answers: y, yes, n, no (in any case) then it accepts the answer, otherwise it asks again. If the user enters y or yes, the program asks for feedback and adds the feedback to a ‘feedback.txt’ file. If no ‘feedback.txt’ file exists, one is created. Example interaction:

```
Any feedback? (y/n)
not sure
Invalid answer. Answer yes/no
Any feedback? (y/n)
Y
Enter feedback:
Amazing program, whoever made it is a genius
Thank you for your feedback!
```

The impact of collecting feedback means that the users of the program have a voice on how to fix/update the program, and we can make changes to the program to meet the needs of the users. It provides interaction between the user and the programmer/s

Future Enhancements

The program could take a playlist link and recommend songs from the link

The program could give a nice interface or GUI for the user

The program could choose genres of music to include/exclude in the recommendations

The program could ask the user if the recommendations were good or not, and learn from the feedback to include or exclude songs like the ones it recommended