Spotify Recommendations

Algorithms

Demo

Discussion

Logistic Regression and *k*-Means: Spotify Recommendation

By Aiesha Ayub, Lulia Aklilu, Adden Hartl

Algorithms

Demo

Discussion

Spotify Recommendations

01

Algorithms

Algorithm Purpose

Predict

Important features based on a dataset and what songs the user likes.

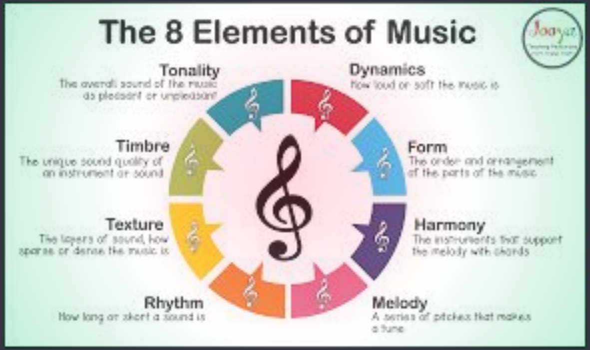
Cluster

Songs based on those features into similar groups.

Suggest

Additional songs based on their proximity to liked songs in the cluster.

Difficulty

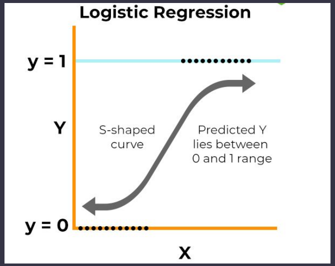


Songs are multifaceted, and often our enjoyment of them is not because they simply check a number of boxes.

We may like songs outside of our typical preferred genres, artists, and style (or vice versa)

Therefore, utilizing only a **supervised** algorithm would be prone to errors.

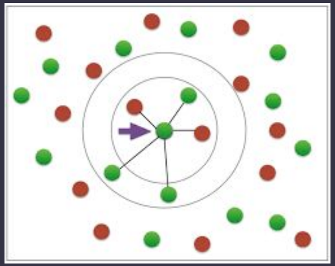
Algorithms Used



Logistic Regression

predict important features

A way to **predict** the probability of binary category placement (i.e. like/dislike) based on the previous algorithm.



k-means

cluster similar songs

An **unsupervised** learning algorithm which creates clusters based on similar characteristics

Spotify Recommendations

Algorithms

Demo

Discussion

02

Demonstration

Utilizing Python coding language

Dataset

The dataset used split songs based on a variety of categories: danceability, energy, track key, loudness, track mode, speechiness, acousticness, instrumentalness, liveness, tempo, duration, time signature and valence.

Variables were collected from Spotify's API documentation.

Each variable was on a different scale, requiring standardization.

The dataset had an accompanying variable "likeability" based on a binary scale (0 being disliked, 1 being liked).

Variable Examples

Instrumentalness

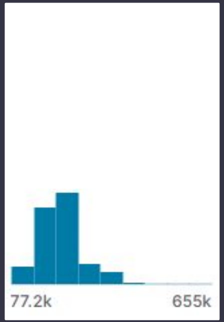
A metric for how much of a track is instrumental

Speechiness

A metric for how “wordy” a track is

Loudness

Loudness of the track in db (averaged within track)



Duration

The duration of the track in milliseconds (min 1:17 , max 10:54)

Logistic Regression

1. Import all necessary libraries and load the data into a DataFrame
2. Define the features (X) and target variable (y)
   - (1 for liked, 0 for disliked)
3. Split the data into training and testing sets (80% train, 20% test)
4. Standardize the data so all features have the same scale (μ = 0, σ = 1)
5. Initialize and Train the Logistic Model using the training data
6. Make Predictions and evaluate model (accuracy, model coefficients, confusion matrix)

92.30%

Accuracy

The model correctly predicted 92.30% of the test data

90%

Precision

90% of the songs predicted as liked were actually liked

95%

Recall

95% of the liked songs were correctly identified

Demonstration – Loading Data

```python
data = pd.read_csv('spotify_data.csv')

print(data.head())
```

```
   danceability  energy  key  loudness  mode  speechiness  acousticness  \
0         0.803  0.6240    7    -6.764     0       0.0477         0.451
1         0.762  0.7030   10    -7.951     0       0.3060         0.206
2         0.261  0.0149    1   -27.528     1       0.0419         0.992
3         0.722  0.7360    3    -6.994     0       0.0585         0.431
4         0.787  0.5720    1    -7.516     1       0.2220         0.145

   instrumentalness  liveness  valence    tempo  duration_ms  time_signature  \
0          0.000734    0.1000   0.6280   95.968       304524               4
1          0.000000    0.0912   0.5190  151.329       247178               4
2          0.897000    0.1020   0.0382   75.296       286987               4
3          0.000001    0.1230   0.5820   89.860       208920               4
4          0.000000    0.0753   0.6470  155.117       179413               4

   liked
0      0
1      1
2      0
3      1
4      1
```

Demonstration – Logistic Regression

```
X_train, X_test, y_train, y_test =
train_test_split(df.drop('liked',
axis=1),df['liked'],test_size = 0.2, random_state= 42)

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

model = LogisticRegression(random_state=42)
model.fit(X_train_scaled,y_train)

y_pred = model.predict(X_test_scaled)

coefficients = model.coef_[0]
feature_importance = pd.DataFrame({'Feature':
X.columns, 'Coefficient': coefficients})
print(feature_importance)
```

	Feature	Coefficient
0	danceability	0.979264
1	energy	-0.323337
2	key	0.130298
3	loudness	1.764453
4	mode	-0.403396
5	speechiness	1.535564
6	acousticness	-0.062750
7	instrumentalness	-1.389849
8	liveness	0.097142
9	valence	-0.097727
10	tempo	0.591655
11	duration_ms	-1.630988
12	time_signature	0.357594

k-means Clustering

1. **Import all necessary libraries and load the data into a DataFrame**

2. **Determine two features for clustering**
   → duration and loudness because they have the most feature importance according to the logistic regression

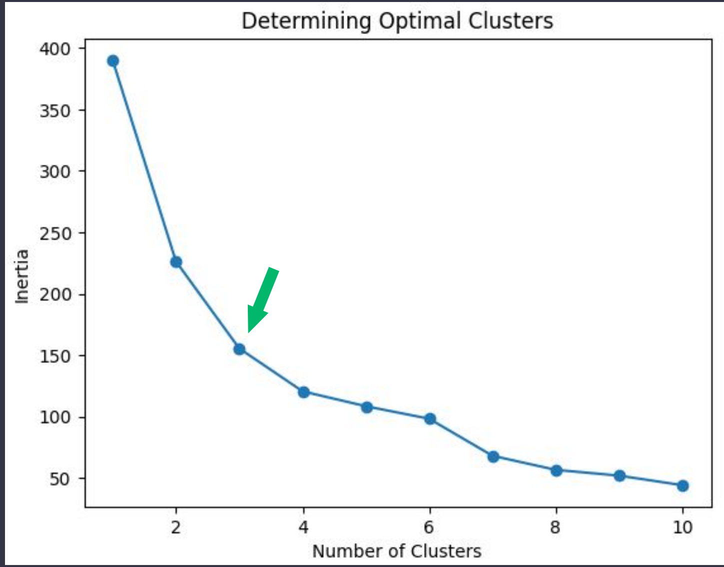
3. **Standardize the data so all features have the same scale**
   → specifically a $\mu = 0$, $\sigma = 1$

4. **Determine the optimal number of clusters (via an elbow curve)**
   → measures how well the *k*-means algorithm groups the data for different numbers of clusters (*k*)

5. **Cluster the data into the optimal number of clusters utilizing k-means**
   → Each song gets a cluster label, indicating which group it belongs to
   → Allows us to identify patterns, like songs with similar duration and loudness being grouped together

Demonstration – Elbow Curve

```
features = data[['duration_ms', 'loudness']]

scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k,
    random_state=42)
    kmeans.fit(scaled_features)
    inertia.append(kmeans.inertia_)

plt.plot(range(1, 11), inertia, marker='o')
plt.title('Determining Optimal Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.show()
```



Demonstration – Cluster Assignment

```
n_clusters = 3

kmeans = KMeans(n_clusters=n_clusters,
random_state=42)
clusters = kmeans.fit_predict(scaled_features)
data['cluster'] = clusters

print(data.head())

data.to_csv('clustered_songs.csv', index=False )
```

Danceability

	danceability	energy	key	loudness	mode	speechiness	acousticness	\
0	0.803	0.6240	7	-6.764	0	0.0477	0.451	
1	0.762	0.7030	10	-7.951	0	0.3060	0.206	
2	0.261	0.0149	1	-27.528	1	0.0419	0.992	
3	0.722	0.7360	3	-6.994	0	0.0585	0.431	
4	0.787	0.5720	1	-7.516	1	0.2220	0.145	

	instrumentalness	liveness	valence	tempo	duration_ms	time_signature	\
0	0.000734	0.1000	0.6280	95.968	304524	4	
1	0.000000	0.0912	0.5190	151.329	247178	4	
2	0.897000	0.1020	0.0382	75.296	286987	4	
3	0.000001	0.1230	0.5820	89.860	208920	4	
4	0.000000	0.0753	0.6470	155.117	179413	4	

	liked	cluster
0	0	0
1	1	2
2	0	1
3	1	0
4	1	0

Algorithms

Demo

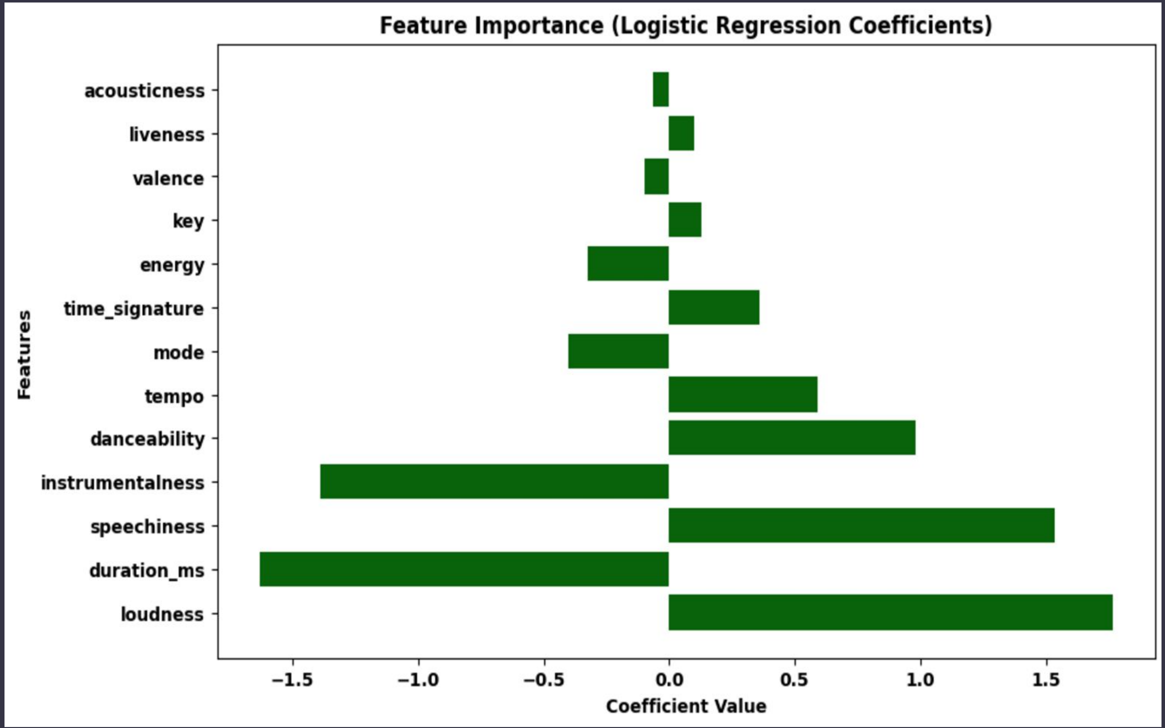
Discussion

Spotify Recommendations

03

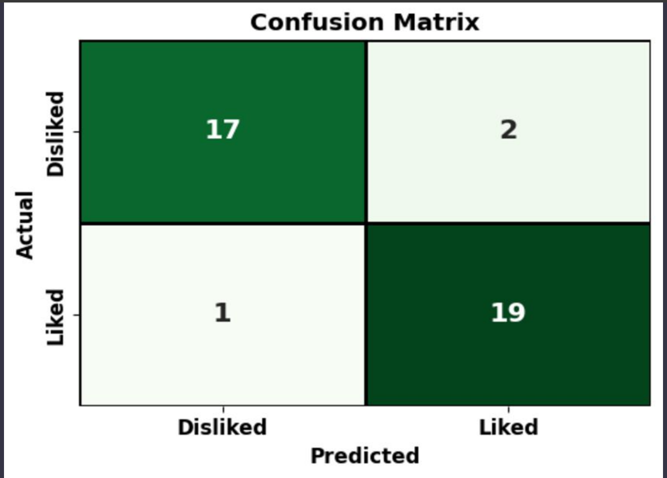
Discussion

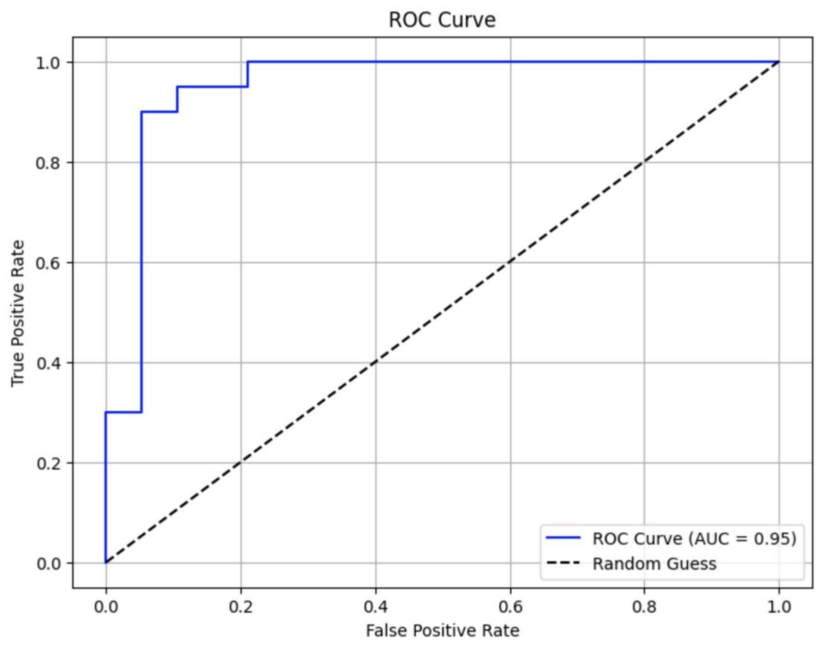
Results – Logistic Regression



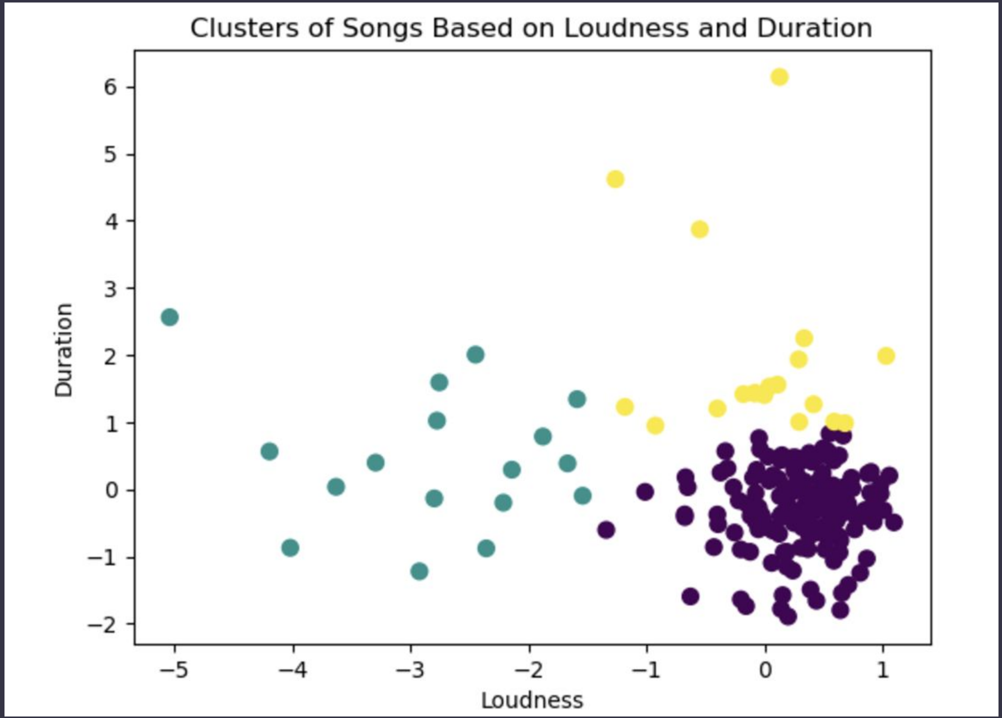
Results – Logistic Regression

```
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.89      0.92        19
           1       0.90      0.95      0.93        20

    accuracy                           0.92        39
   macro avg       0.92      0.92      0.92        39
weighted avg       0.92      0.92      0.92        39
```

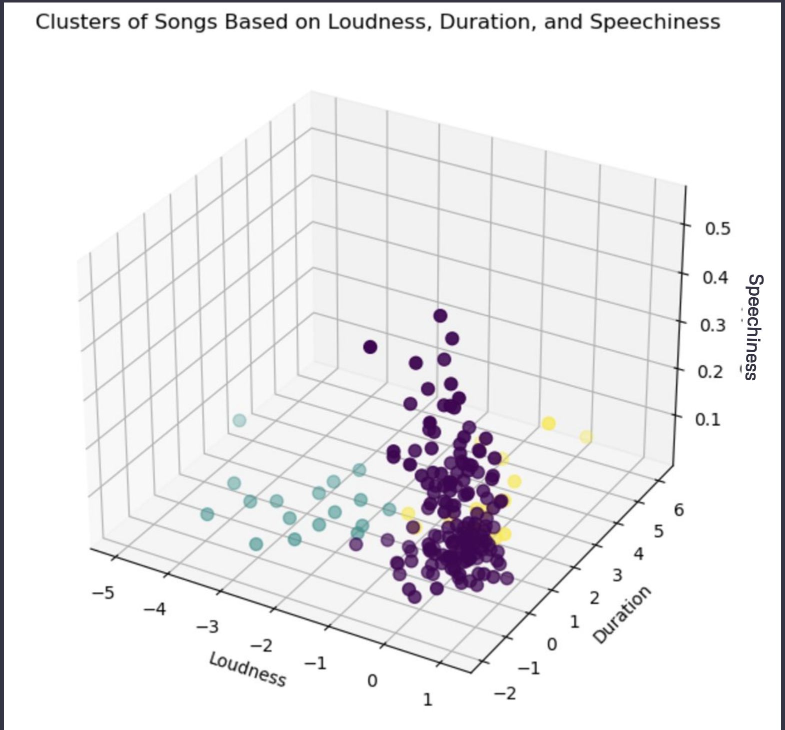




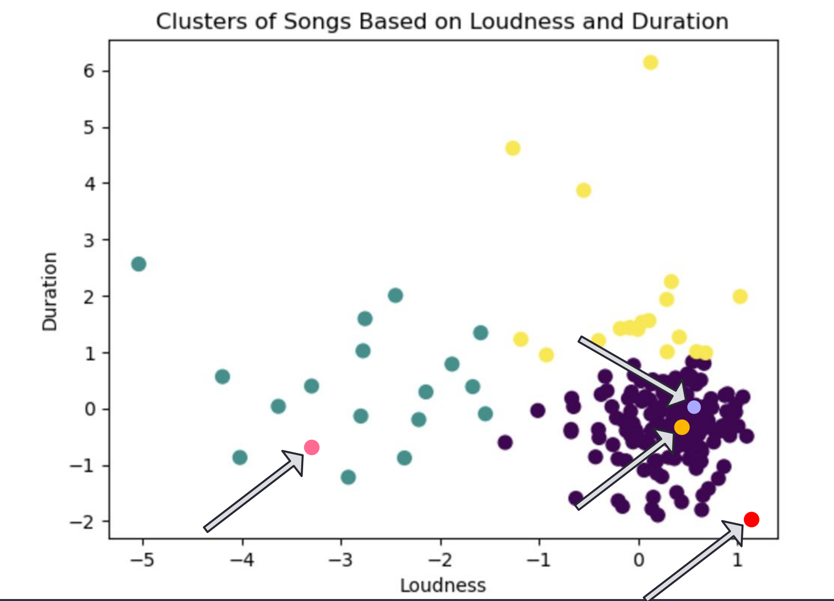
Results – *k*-Means



Results – *k*-Means



In Action



Centroid for purple cluster (cluster 1).

Three new songs the user hasn't listened to before (song 1, song 2, and song 3).

If the *most recent song* the user listened to was in cluster 1, the algorithm should recommend song 2, then song 1, and maybe not song 3.

Limitations

Dataset Genres

The dataset was mainly from the user's most listened to (French/American rap, rock, electro, metal, classical, and Disco genres).

Dataset sample size

The dataset was limited to 195 songs.

Features chosen

Only two - three features were selected of the 14 total.

Future Work

Expand dataset and variables

Collect both more samples from a wider range of genres and additional variables such as popularity/trendiness and release date.

Change testing dataset

For Logistic Regression, change the split percentages of testing data.

Explore alternate features

Iterations of the program can be ran to determine optimal features to utilize for clustering.

Algorithms

Demo

Discussion

Spotify Recommendations

04

Questions?