Song Qualities

In Lab 6A, we used the audio features from 100 songs each from 100 artists to find out which audio features are predictive of explicit language.

Perhaps instead of focusing on explicit language, we want to study how audio feautes relate to song and artist styles and genres in general.

Our goal in this lab will be to perform **unsupervised learning** to find clusters of similar songs.

Question 0

Read and clean the data

In [138…
```
## YOUR CODE HERE
import pandas as pd
import numpy as np

df_songs = pd.read_csv("https://www.dropbox.com/s/hijzbof7nnche09/top_artists_
df_songs
```

Out[138]:

	Acousticness	Danceability	Duration	Energy	Explicit	Instrumentalness	Liveness	Loud
0	0.631000	0.399	213492	0.491	True	0.000000	0.1710	-8
1	0.099900	0.758	230484	0.888	True	0.033600	0.6610	-4
2	0.161000	0.529	195720	0.690	True	0.000000	0.4980	-
3	0.528000	0.766	188703	0.531	False	0.000665	0.1100	-10
4	0.042900	0.828	211643	0.482	False	0.000008	0.0807	-8
...	...	...	...	...	...	...	...	
1395	0.407000	0.526	273467	0.665	False	0.000271	0.1690	-4
1396	0.002650	0.652	225400	0.783	False	0.000002	0.2680	-6
1397	0.000266	0.368	293733	0.631	False	0.055900	0.0682	-1
1398	0.751000	0.536	35973	0.229	False	0.000000	0.6290	-24
1399	0.410000	0.545	278893	0.947	False	0.000705	0.0385	-7

1400 rows × 13 columns

In [139…
```
df_new = df_songs.drop('Explicit', axis="columns")
df_new
```

Out[139]:

	Acousticness	Danceability	Duration	Energy	Instrumentalness	Liveness	Loudness	Mo
0	0.631000	0.399	213492	0.491	0.000000	0.1710	-8.992	
1	0.099900	0.758	230484	0.888	0.033600	0.6610	-4.899	
2	0.161000	0.529	195720	0.690	0.000000	0.4980	-7.870	
3	0.528000	0.766	188703	0.531	0.000665	0.1100	-10.899	
4	0.042900	0.828	211643	0.482	0.000008	0.0807	-8.469	
...	...	...	...	...	...	...	...	
1395	0.407000	0.526	273467	0.665	0.000271	0.1690	-4.997	
1396	0.002650	0.652	225400	0.783	0.000002	0.2680	-6.706	
1397	0.000266	0.368	293733	0.631	0.055900	0.0682	-12.191	
1398	0.751000	0.536	35973	0.229	0.000000	0.6290	-24.760	
1399	0.410000	0.545	278893	0.947	0.000705	0.0385	-7.456	

1400 rows × 12 columns

Question 1

Fit 3-means clustering models to this dataset. Briefly interpret the meaning of each cluster based on its cluster centers. (For example, you might say, "The songs in cluster 1 all have high speechiness, and low instrumentalness, so they might be rap.)

In [140…

```
## YOUR CODE HERE
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import cross_val_score
```

In [141…

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().set_output(transform = "pandas")
df_scaled = scaler.fit_transform(df_new[["Acousticness", "Danceability", "Dura

df_scaled.head()
```

Out[141]:

	Acousticness	Danceability	Duration	Energy	Instrumentalness	Liveness	Loudness
0	1.750322	-0.899042	-0.237774	-1.045510	-0.355683	-0.311990	-0.713950
1	-0.328278	1.115998	-0.007811	0.873533	-0.187901	2.016812	0.450124
2	-0.089147	-0.169362	-0.478293	-0.083572	-0.355683	1.242129	-0.394847
3	1.347204	1.160901	-0.573259	-0.852155	-0.352362	-0.601902	-1.256312
4	-0.551363	1.508903	-0.262797	-1.089014	-0.355644	-0.741155	-0.565206

In [142…

```
centroids = df_scaled.sample(3, random_state=1234)
centroids.index = ["orange", "purple", "green"]

centroids
```

Out[142]:

	Acousticness	Danceability	Duration	Energy	Instrumentalness	Liveness	Loudn
orange	-0.155681	-0.792397	-0.390054	1.211903	-0.205378	0.657552	1.020
purple	1.237619	-1.421045	0.017944	-1.002005	1.826485	-0.045841	-0.954
green	-0.378374	1.419096	0.088495	-0.784481	-0.354180	-0.728323	-0.798

In [143…

```
from sklearn.metrics import pairwise_distances
dists = pairwise_distances(df_scaled, centroids)
dists
```

Out[143]:

```
array([[4.27124045, 2.60311187, 3.21560713],
       [3.76172869, 5.65461251, 4.05436072],
       [4.68848551, 6.9027578 , 6.09233665],
       ...,
       [3.91545425, 3.64567011, 3.04195504],
       [8.62684263, 7.44979065, 7.43928746],
       [3.9953989 , 4.47576479, 3.39911912]])
```

In [144…

```
closest_centroid = dists.argsort()[:,0]
closest_centroid
```

Out[144]:

```
array([1, 0, 0, ..., 2, 2, 2])
```

In [145…

```
df_scaled.index = centroids.index[closest_centroid]
df_scaled.head()
```

Out[145]:

	Acousticness	Danceability	Duration	Energy	Instrumentalness	Liveness	Loudn
purple	1.750322	-0.899042	-0.237774	-1.045510	-0.355683	-0.311990	-0.713
orange	-0.328278	1.115998	-0.007811	0.873533	-0.187901	2.016812	0.450
orange	-0.089147	-0.169362	-0.478293	-0.083572	-0.355683	1.242129	-0.394
green	1.347204	1.160901	-0.573259	-0.852155	-0.352362	-0.601902	-1.256
green	-0.551363	1.508903	-0.262797	-1.089014	-0.355644	-0.741155	-0.565

In [146…
```
centroids = df_scaled.groupby(df_scaled.index).mean()
centroids
```

Out[146]:

	Acousticness	Danceability	Duration	Energy	Instrumentalness	Liveness	Loudne
green	-0.044314	0.663617	0.077396	-0.284606	-0.244648	-0.328506	-0.116(
orange	-0.331110	-0.506047	-0.100181	0.693213	-0.137684	0.422594	0.536(
purple	1.194708	-0.887001	0.024639	-1.101348	1.335957	-0.096405	-1.237

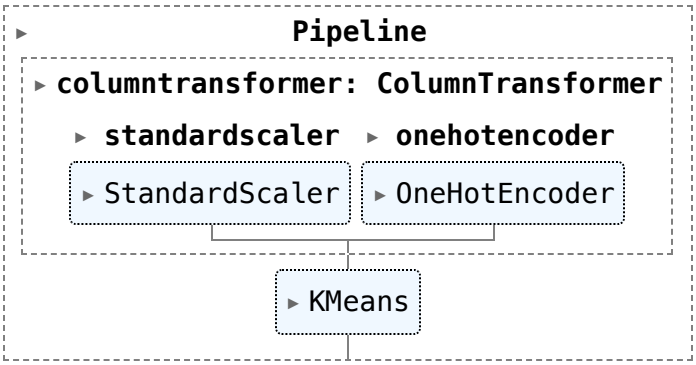
In [147…
```
for i in range(1,3):
  dists = pairwise_distances(df_scaled, centroids)
  df_scaled.index = centroids.index[closest_centroid]
  centroids = df_scaled.groupby(df_scaled.index).mean()
  print(centroids)
```

```
        Acousticness  Danceability  Duration    Energy  Instrumentalness  \
green       -0.331110     -0.506047 -0.100181  0.693213         -0.137684
orange       1.194708     -0.887001  0.024639 -1.101348          1.335957
purple      -0.044314      0.663617  0.077396 -0.284606         -0.244648

        Liveness  Loudness  Speechiness     Tempo   Valence
green   0.422594  0.536018     0.183526  0.569596  0.083923
orange -0.096405 -1.237578    -0.472215 -0.468234 -0.851144
purple -0.328506 -0.116017    -0.026652 -0.351583  0.159051
        Acousticness  Danceability  Duration    Energy  Instrumentalness  \
green       -0.331110     -0.506047 -0.100181  0.693213         -0.137684
orange       1.194708     -0.887001  0.024639 -1.101348          1.335957
purple      -0.044314      0.663617  0.077396 -0.284606         -0.244648

        Liveness  Loudness  Speechiness     Tempo   Valence
green   0.422594  0.536018     0.183526  0.569596  0.083923
orange -0.096405 -1.237578    -0.472215 -0.468234 -0.851144
purple -0.328506 -0.116017    -0.026652 -0.351583  0.159051
```

In [148…
```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

model = KMeans(n_clusters=3, random_state=1234)

ct = make_column_transformer(
    (StandardScaler(), ["Acousticness", "Danceability", "Duration", "Energy", "
    (OneHotEncoder(), ["Mode", "TimeSignature"]),
    remainder="drop"
)

pipeline = make_pipeline(
  ct,
  model)

pipeline.fit(df_new, df_songs['Explicit'])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: Future
Warning: The default value of `n_init` will change from 10 to 'auto' in 1.4. S
et the value of `n_init` explicitly to suppress the warning
```

Out[148]:



In [149…
```
centroids = model.cluster_centers_
clusters = model.labels_

clusters
```

Out[149]:
```
array([1, 0, 2, ..., 1, 1, 0], dtype=int32)
```

In [150…
```
centroids
```

Out[150]:
```
array([[-3.19760081e-01,  6.34790319e-01,  6.82804781e-02,
         1.41935431e-01, -1.83000397e-01, -3.45719507e-01,
         2.43547526e-01, -4.05502824e-02, -3.34831868e-01,
         3.42626960e-01,  3.66032211e-01,  6.33967789e-01,
        -6.28837260e-18,  1.46412884e-03,  2.92825769e-02,
         9.66325037e-01,  2.92825769e-03],
       [ 1.45630873e+00, -3.57798939e-01, -1.38967473e-01,
        -1.42334811e+00,  4.56005268e-01, -1.70066422e-01,
        -1.30518415e+00, -1.15241162e-01, -2.71764166e-01,
        -7.29579530e-01,  3.06273063e-01,  6.93726937e-01,
         7.38007380e-03,  2.95202952e-02,  8.11808118e-02,
         8.56088561e-01,  2.58302583e-02],
       [-3.95209709e-01, -7.54704654e-01, -2.01241737e-02,
         6.47500986e-01,  3.16556818e-03,  6.32767766e-01,
         4.20094043e-01,  1.32121520e-01,  6.77888463e-01,
        -8.13860118e-02,  3.16143498e-01,  6.83856502e-01,
        -4.33680869e-19,  2.24215247e-03,  3.81165919e-02,
         9.46188341e-01,  1.34529148e-02]])
```

In [151…
```
results = pd.DataFrame({
  "cluster": clusters,
  "Explicit": df_songs['Explicit']
})

results
```

Out[151]:

	cluster	Explicit
0	1	True
1	0	True
2	2	True
3	1	False
4	0	False
...	...	...
1395	2	False
1396	0	False
1397	1	False
1398	1	False
1399	0	False

1400 rows × 2 columns

In [152…
```
results.value_counts(["cluster", "Explicit"]).unstack()
```

Out[152]:

Explicit	False	True
cluster		
0	427	256
1	217	54
2	327	119

The centers for the features in the 3-means model are all very low in value, with many decimal values. The mode is especially high in Cluster 1, and the centers for Acousticness vary significantly between the 3 clusters. This is seen in that the centers for Clusters 0 and 2 are negative while the opposite is true for Cluster 1.

Question 2

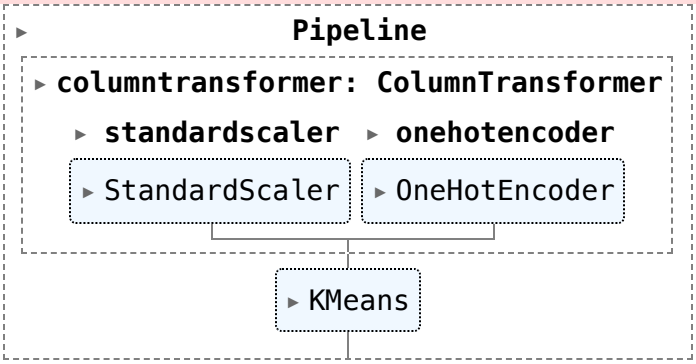
Repeat Question 1, but for a 5-means clustering model.

In [153…
```
## YOUR CODE HERE
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

model_5_means = KMeans(n_clusters=5, random_state=1234)

ct_5_means = make_column_transformer(
    (StandardScaler(), ["Acousticness", "Danceability", "Duration", "Energy", '
    (OneHotEncoder(), ["Mode", "TimeSignature"]),
    remainder="drop"
```

```
)

pipeline_5_means = make_pipeline(
  ct_5_means,
  model_5_means)

pipeline_5_means.fit(df_new, df_songs['Explicit'])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: Future
Warning: The default value of `n_init` will change from 10 to 'auto' in 1.4. S
et the value of `n_init` explicitly to suppress the warning
```

Out[153]:



In [154…

```
centroids_5_means = model_5_means.cluster_centers_
clusters_5_means = model_5_means.labels_

clusters_5_means
```

Out[154]:

```
array([3, 4, 4, ..., 2, 3, 0], dtype=int32)
```

In [155…

```
centroids_5_means
```

Out[155]:
```
array([[-2.83904042e-01,  7.13270615e-01,  8.17364766e-02,
         2.47846997e-02, -3.16809324e-01, -3.50472943e-01,
         1.81922037e-01,  1.33195539e-02, -4.36036753e-01,
         2.80829087e-01,  3.69863014e-01,  6.30136986e-01,
        -4.55364912e-18,  2.08166817e-17,  2.56849315e-02,
         9.70890411e-01,  3.42465753e-03],
       [-1.68707785e-01, -3.52553137e-01,  4.41167740e-01,
        -4.09167393e-02,  3.16941214e+00, -2.35028532e-01,
        -6.12845384e-01, -4.73472174e-01, -8.59403527e-02,
        -2.77907396e-01,  3.62637363e-01,  6.37362637e-01,
        -2.60208521e-18,  1.21430643e-17,  8.79120879e-02,
         9.01098901e-01,  1.09890110e-02],
       [-4.51246234e-01, -6.88340670e-01, -2.26756764e-01,
         6.52103254e-01, -1.90440306e-01, -1.53572130e-01,
         5.61814931e-01, -1.02471906e-01,  8.00376029e-01,
         8.64948060e-02,  2.80000000e-01,  7.20000000e-01,
         2.38524478e-18,  5.33333333e-03,  4.00000000e-02,
         9.46666667e-01,  8.00000000e-03],
       [ 1.88271792e+00, -3.61378866e-01, -3.78455619e-01,
        -1.55760884e+00,  3.55137907e-02, -1.78614447e-01,
        -1.34410655e+00,  6.29815042e-03, -2.86700879e-01,
        -6.67374991e-01,  2.87804878e-01,  7.12195122e-01,
         9.75609756e-03,  3.90243902e-02,  8.29268293e-02,
         8.34146341e-01,  3.41463415e-02],
       [-2.45430803e-01, -3.60395069e-01,  5.15426356e-01,
         4.41518976e-01, -2.70793597e-01,  2.20875383e+00,
         9.92255404e-02,  4.99608223e-01,  1.45508313e-01,
        -2.36815812e-01,  4.20689655e-01,  5.79310345e-01,
        -2.16840434e-18,  1.12757026e-17,  2.75862069e-02,
         9.58620690e-01,  1.37931034e-02]])
```

In [156…
```
results_5_means = pd.DataFrame({
  "clusters_5_means": clusters_5_means,
  "Explicit": df_songs['Explicit']
})

results_5_means
```

Out[156]:

	clusters_5_means	Explicit
0	3	True
1	4	True
2	4	True
3	3	False
4	0	False
...	...	...
1395	2	False
1396	0	False
1397	2	False
1398	3	False
1399	0	False

1400 rows × 2 columns

In [157…
```
results_5_means.value_counts(["clusters_5_means", "Explicit"]).unstack()
```

Out[157]:

Explicit	False	True
clusters_5_means		
0	345	239
1	80	11
2	288	87
3	163	42
4	95	50

Based on the centroids (the centers of the clusters for my 5-means model), it appears that Cluster 0 has very similar scores across all of the features. However, the centers for Mode and Time Signature are especially low. These features are similarly low in Cluster 1, though the values of all of the other characteristics have shifted slightly, with the decimal place maintaining a similar value.

In Cluster 2, the center for Time Signature is significantly higher than those in the other clusters, meaning these are likely songs that have a faster ryhthm. The center for mode also increases significantly in Cluster 3, with a fast-paced time signature maintained. Cluster 4 demonstrates very similar characteristics to Cluster 2.

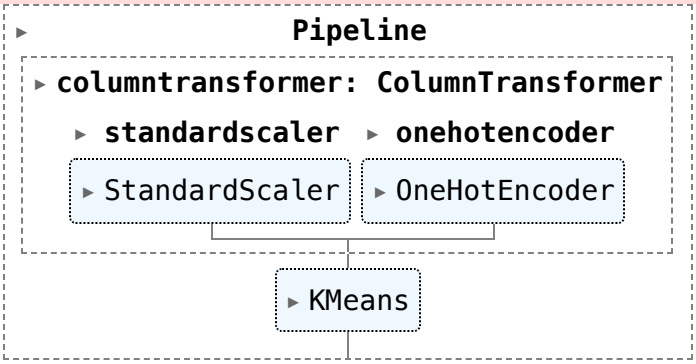
Question 3

Repeat Question 1, but for a 7-means clustering model.

In [158…
```
## YOUR CODE HERE
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

model_7_means = KMeans(n_clusters=7, random_state=1234)

ct_7_means = make_column_transformer(
    (StandardScaler(), ["Acousticness", "Danceability", "Duration", "Energy", '
    (OneHotEncoder(), ["Mode", "TimeSignature"]),
    remainder="drop"
)

pipeline_7_means = make_pipeline(
  ct_7_means,
  model_7_means)

pipeline_7_means.fit(df_new, df_songs['Explicit'])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: Future
Warning: The default value of `n_init` will change from 10 to 'auto' in 1.4. S
et the value of `n_init` explicitly to suppress the warning
```

Out[158]:



In [159…
```
centroids_7_means = model_7_means.cluster_centers_
clusters_7_means = model_7_means.labels_

clusters_7_means
```

Out[159]:
```
array([5, 1, 3, ..., 2, 5, 6], dtype=int32)
```

In [160…
```
centroids_7_means
```

Out[160]:
```
array([[ 1.40512023e-01, -4.97484340e-01,  2.42913870e-01,
        -3.39625163e-01,  3.29983294e+00, -2.00812716e-01,
        -1.01880974e+00, -4.50600256e-01, -1.39410345e-02,
        -3.79517702e-01,  3.64583333e-01,  6.35416667e-01,
        -1.73472348e-18,  1.04166667e-02,  1.25000000e-01,
         8.43750000e-01,  2.08333333e-02],
       [-2.46727115e-01, -5.91163039e-01,  2.58813805e-01,
         5.55035048e-01, -2.32567916e-01,  2.63030091e+00,
         1.37761651e-01,  9.04697357e-02,  2.77659378e-02,
        -2.11174792e-01,  3.88888889e-01,  6.11111111e-01,
        -1.95156391e-18,  7.80625564e-18,  3.70370370e-02,
         9.53703704e-01,  9.25925926e-03],
       [-2.27633825e-01, -1.15838013e-01,  4.68799165e-01,
        -4.53881729e-01, -2.55120567e-01, -3.23678202e-01,
        -1.99942597e-01, -4.66779002e-01, -1.95963839e-01,
        -7.77006135e-01,  2.94339623e-01,  7.05660377e-01,
         2.16840434e-18,  3.77358491e-03,  5.28301887e-02,
         9.24528302e-01,  1.88679245e-02],
       [ 1.42411754e-01,  7.93902459e-01,  1.47357627e-01,
        -1.35285291e-01, -3.33879765e-01,  8.48201072e-02,
        -1.36294577e-01,  1.89467175e+00, -2.00430647e-01,
        -4.38554997e-02,  4.43786982e-01,  5.56213018e-01,
         2.81892565e-18,  5.91715976e-03,  5.32544379e-02,
         9.34911243e-01,  5.91715976e-03],
       [-4.66691666e-01, -7.93919372e-01, -3.95026982e-01,
         8.56643559e-01, -1.81567795e-01, -4.20065835e-02,
         6.96521858e-01, -1.29331933e-02,  1.02201742e+00,
         2.08264842e-01,  2.46963563e-01,  7.53036437e-01,
         1.30104261e-18,  1.73472348e-17,  2.83400810e-02,
         9.67611336e-01,  4.04858300e-03],
       [ 2.11396012e+00, -3.31496488e-01, -3.43139999e-01,
        -1.61965225e+00, -1.63616760e-01, -2.88373688e-01,
        -1.22208260e+00, -3.43041461e-01, -3.42495726e-01,
        -5.65691475e-01,  2.75167785e-01,  7.24832215e-01,
         1.34228188e-02,  4.02684564e-02,  6.71140940e-02,
         8.52348993e-01,  2.68456376e-02],
       [-4.10640613e-01,  6.92958263e-01, -1.41276819e-01,
         3.97648440e-01, -2.68873919e-01, -3.82543843e-01,
         4.61736220e-01, -2.97017814e-01, -3.20392415e-01,
         8.34440973e-01,  3.87978142e-01,  6.12021858e-01,
        -6.50521303e-19,  2.73224044e-03,  8.19672131e-03,
         9.86338798e-01,  2.73224044e-03]])
```

In [161…
```
results_7_means = pd.DataFrame({
  "clusters_7_means": clusters_7_means,
  "Explicit": df_songs['Explicit']
})

results_7_means
```

Out[161]:

	clusters_7_means	Explicit
0	5	True
1	1	True
2	3	True
3	3	False
4	3	False
...	...	...
1395	2	False
1396	6	False
1397	2	False
1398	5	False
1399	6	False

1400 rows × 2 columns

In [162…
```
results_7_means.value_counts(["clusters_7_means", "Explicit"]).unstack()
```

Out[162]:

Explicit	False	True
clusters_7_means		
0	83	13
1	84	24
2	202	63
3	40	129
4	180	67
5	124	25
6	258	108

The centers for each of the features in the 7-means model also demonstrate very low values in each of the clusters. The center for Time Signature is especially high in Cluster 0, meaning that similar to Cluster 2 in the 5-means model, these songs are likely faster in pace. The center for mode in Cluster 6 is also very high in comparison to the rest of the clusters in this model.

Question 4

Here you will find a dataset that is indentical to `df_songs`, except that it also includes the artist and song title for each song.

(a) For your 5-means clustering, add the *cluster assignments* as a column to this dataset.

(b) Make a visualization of the *cluster enrichment* of the artists; that is, how many of their songs were assigned to each cluster.

Do you think that the clusters correspond to particular genres or styles?

In [163…
```
## YOUR CODE HERE
df_artists = pd.read_csv("/content/top_artists_spotify.csv")
df_artists
```

Out[163]:

	Name	Album	Artist	Acousticness	Danceability	Duration	Energy	Explici
0	Brand New	So Far Gone	drake	0.631000	0.399	213492	0.491	Tru
1	Little Bit (feat. Lykke Li)	So Far Gone	drake	0.099900	0.758	230484	0.888	Tru
2	8 Out Of 10	Scorpion	drake	0.161000	0.529	195720	0.690	Tru
3	Jersey	What A Time To Be Alive	drake	0.528000	0.766	188703	0.531	Fals
4	Live From The Gutter	What A Time To Be Alive	drake	0.042900	0.828	211643	0.482	Fals
...	...	...	...	...	...	...	...	..
1395	Stuck In A Moment You Can't Get Out Of	All That You Can't Leave Behind	u2	0.407000	0.526	273467	0.665	Fals
1396	You're The Best Thing About Me	Songs Of Experience (Deluxe Edition)	u2	0.002650	0.652	225400	0.783	Fals
1397	The Unforgettable Fire	The Unforgettable Fire	u2	0.000266	0.368	293733	0.631	Fals
1398	Freedom For My People - Excerpt	Rattle And Hum	u2	0.751000	0.536	35973	0.229	Fals
1399	Sunday Bloody Sunday - Remastered 2008	War	u2	0.410000	0.545	278893	0.947	Fals

1400 rows × 16 columns

In [164…
```
df_artists["Cluster Assignments"] = results_5_means['clusters_5_means']
df_artists
```

Out[164]:

	Name	Album	Artist	Acousticness	Danceability	Duration	Energy	Explici
0	Brand New	So Far Gone	drake	0.631000	0.399	213492	0.491	Tru
1	Little Bit (feat. Lykke Li)	So Far Gone	drake	0.099900	0.758	230484	0.888	Tru
2	8 Out Of 10	Scorpion	drake	0.161000	0.529	195720	0.690	Tru
3	Jersey	What A Time To Be Alive	drake	0.528000	0.766	188703	0.531	Fals
4	Live From The Gutter	What A Time To Be Alive	drake	0.042900	0.828	211643	0.482	Fals
...	...	...	...	...	...	...	...	..
1395	Stuck In A Moment You Can't Get Out Of	All That You Can't Leave Behind	u2	0.407000	0.526	273467	0.665	Fals
1396	You're The Best Thing About Me	Songs Of Experience (Deluxe Edition)	u2	0.002650	0.652	225400	0.783	Fals
1397	The Unforgettable Fire	The Unforgettable Fire	u2	0.000266	0.368	293733	0.631	Fals
1398	Freedom For My People - Excerpt	Rattle And Hum	u2	0.751000	0.536	35973	0.229	Fals
1399	Sunday Bloody Sunday - Remastered 2008	War	u2	0.410000	0.545	278893	0.947	Fals

1400 rows × 17 columns

In [165…
```
clust_props = df_artists["Cluster Assignments"].value_counts(normalize=True)
clust_props
```

Out[165]:
```
Cluster Assignments
0    0.417143
2    0.267857
3    0.146429
4    0.103571
1    0.065000
Name: proportion, dtype: float64
```

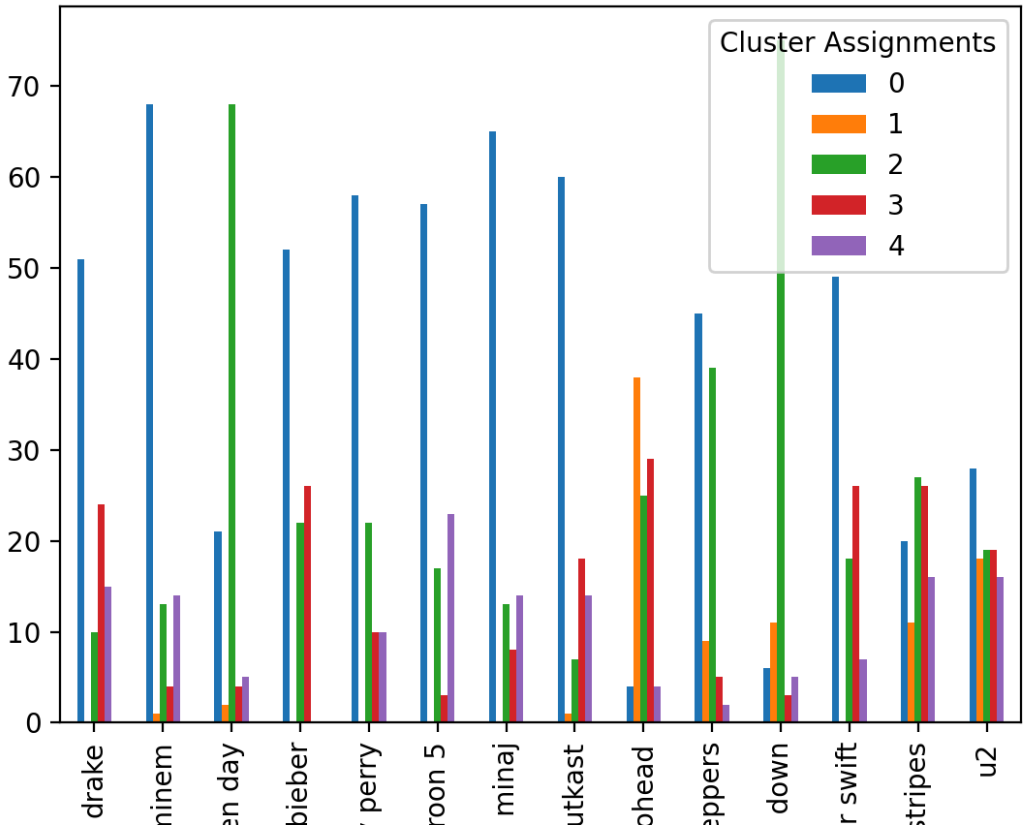
In [166…
```
import matplotlib as plt
df_results = pd.DataFrame(df_artists.value_counts(["Artist", "Cluster Assignme
df_results
```

Out[166]:

Cluster Assignments	0	1	2	3	4
Artist					
drake	51.0	0.0	10.0	24.0	15.0
eminem	68.0	1.0	13.0	4.0	14.0
green day	21.0	2.0	68.0	4.0	5.0
justin bieber	52.0	0.0	22.0	26.0	0.0
katy perry	58.0	0.0	22.0	10.0	10.0
maroon 5	57.0	0.0	17.0	3.0	23.0
nicki minaj	65.0	0.0	13.0	8.0	14.0
outkast	60.0	1.0	7.0	18.0	14.0
radiohead	4.0	38.0	25.0	29.0	4.0
red hot chili peppers	45.0	9.0	39.0	5.0	2.0
system of a down	6.0	11.0	75.0	3.0	5.0
taylor swift	49.0	0.0	18.0	26.0	7.0
the white stripes	20.0	11.0	27.0	26.0	16.0
u2	28.0	18.0	19.0	19.0	16.0

In [167…
```
df_results.plot.bar()
```

Out[167]:
```
<Axes: xlabel='Artist'>
```



Based on the above visualiztion, it appears that Cluster 0 is the most widely prominent cluster among these artists. This likely means that the cluster contains very common danceability, tempo, time signature, duration, and loudness scores that are applicable to a wide variety of genres and artists. Green Day and System of a Down had by far the most songs in Cluster 2, meaning that they are both likely Indie artists. Radiohead had by far the most songs in Cluster 1 while also having many songs in Cluster 3, which they shared with Drake, Justin Beiber, Taylor Swift and the White Stripes. This most likely means that Cluster 3 contains faster songs that are more characteristic of pop sounds. Lastly, Drake, Eminem, Nicky Minaj, Maroon 5, the White Stripes, and u2 all have high concentrations of songs in Cluster 4. These artists all have very distinct styles, with some having a faster rap sound and others having more Indie features. This likely means that a feature such as speechiness, liveness, duration, energy, or another that does not have a direct impact on the rythm of the song.

Question 5

What cluster (from your 5-means model) do you predict Twenty One Pilots would best fit in? Based on this, what other artists might you recommend to someone who enjoys Twenty One Pilots?

(Optional: Would you make different recommendations based on which Twenty One Pilots album they most prefer?)

data link

In [168…
```
## YOUR CODE HERE
df_21 = pd.read_csv("/content/twenty-one-pilots.csv")
df_21 = df_21.drop("Name", axis="columns")
df_21 = df_21.drop("Album", axis="columns")
df_21 = df_21.drop("Artist", axis="columns")
df_21
```

Out[168]:

	X1	Acousticness	Danceability	Duration	Energy	Explicit	Instrumentalness	Liveness	Lo
0	0	0.00891	0.446	308388	0.760	False	0.000108	0.2320	
1	1	0.24100	0.608	263733	0.776	False	0.000000	0.0713	
2	2	0.11000	0.690	239015	0.678	False	0.000000	0.2460	
3	3	0.44500	0.680	163880	0.693	False	0.000000	0.2080	
4	4	0.09800	0.766	267720	0.519	False	0.000000	0.0855	
…	…	…	…	…	…	…	…	…	
65	65	0.75200	0.746	86844	0.121	False	0.000000	0.8420	
66	66	0.74600	0.624	63467	0.205	False	0.000000	0.4360	
67	67	0.65200	0.677	204575	0.201	False	0.000000	0.1270	
68	68	0.58400	0.683	64525	0.182	False	0.000000	0.2620	
69	69	0.69200	0.731	124184	0.232	False	0.000000	0.3210	

70 rows × 14 columns

In [169…
```
df_21_dropped = df_21.drop("Explicit", axis="columns")
df_21_dropped
```

Out[169]:

	X1	Acousticness	Danceability	Duration	Energy	Instrumentalness	Liveness	Loudness
0	0	0.00891	0.446	308388	0.760	0.000108	0.2320	-3.608
1	1	0.24100	0.608	263733	0.776	0.000000	0.0713	-3.635
2	2	0.11000	0.690	239015	0.678	0.000000	0.2460	-6.520
3	3	0.44500	0.680	163880	0.693	0.000000	0.2080	-7.234
4	4	0.09800	0.766	267720	0.519	0.000000	0.0855	-7.355
...	...	...	...	...	...	...	...	...
65	65	0.75200	0.746	86844	0.121	0.000000	0.8420	-26.296
66	66	0.74600	0.624	63467	0.205	0.000000	0.4360	-27.435
67	67	0.65200	0.677	204575	0.201	0.000000	0.1270	-27.235
68	68	0.58400	0.683	64525	0.182	0.000000	0.2620	-29.028
69	69	0.69200	0.731	124184	0.232	0.000000	0.3210	-27.719

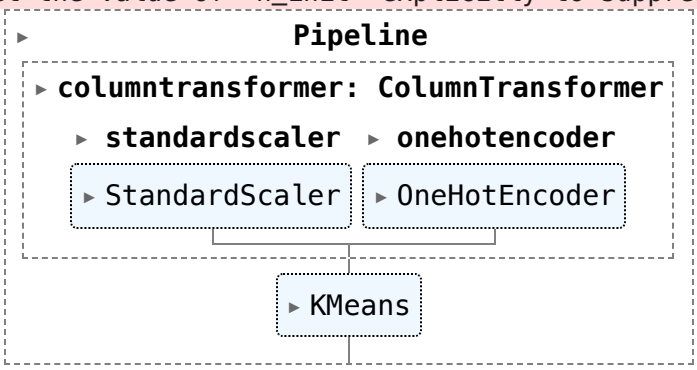
70 rows × 13 columns

In [170…

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

model_21_pilots = KMeans(n_clusters=5, random_state=1234)

ct_21_pilots = make_column_transformer(
    (StandardScaler(), ["Acousticness", "Danceability", "Duration", "Energy", '
    (OneHotEncoder(), ["Mode", "TimeSignature"]),
    remainder="drop"
)

pipeline_21_pilots = make_pipeline(
  ct_21_pilots,
  model_21_pilots)

pipeline_21_pilots.fit(df_new, df_songs['Explicit'])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: Future
Warning: The default value of `n_init` will change from 10 to 'auto' in 1.4. S
et the value of `n_init` explicitly to suppress the warning
```

Out[170]:



In [171…
```
centroids_21 = model_21_pilots.cluster_centers_
clusters_21 = model_21_pilots.labels_

clusters_21
```

Out[171]:
```
array([3, 4, 4, ..., 2, 3, 0], dtype=int32)
```

In [172…
```
results_21 = pd.DataFrame({
  "clusters_21": clusters_21,
  # "Explicit": df_21['Explicit']
})

results_21
```

Out[172]:

	clusters_21
0	3
1	4
2	4
3	3
4	0
...	...
1395	2
1396	0
1397	2
1398	3
1399	0

1400 rows × 1 columns

In [173…
```
results_21["clusters_21"].value_counts()
```

Out[173]:
```
clusters_21
0    584
2    375
3    205
4    145
1     91
Name: count, dtype: int64
```

In [174…
```
df_21["Cluster Assignments"] = results_21['clusters_21']
df_21
```

Out[174]:

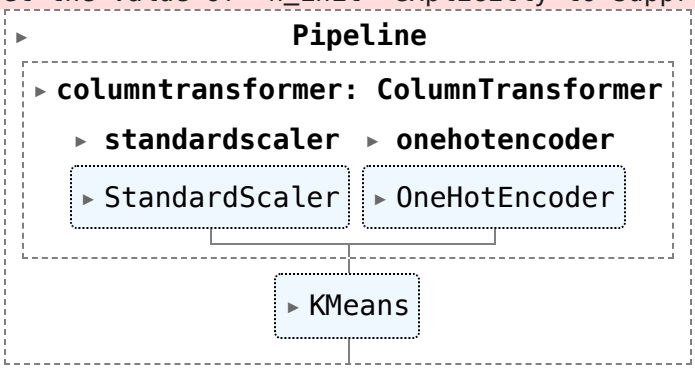
	X1	Acousticness	Danceability	Duration	Energy	Explicit	Instrumentalness	Liveness	Lo
0	0	0.00891	0.446	308388	0.760	False	0.000108	0.2320	
1	1	0.24100	0.608	263733	0.776	False	0.000000	0.0713	
2	2	0.11000	0.690	239015	0.678	False	0.000000	0.2460	
3	3	0.44500	0.680	163880	0.693	False	0.000000	0.2080	
4	4	0.09800	0.766	267720	0.519	False	0.000000	0.0855	
...	...	...	...	...	...	...	...	...	
65	65	0.75200	0.746	86844	0.121	False	0.000000	0.8420	
66	66	0.74600	0.624	63467	0.205	False	0.000000	0.4360	
67	67	0.65200	0.677	204575	0.201	False	0.000000	0.1270	
68	68	0.58400	0.683	64525	0.182	False	0.000000	0.2620	
69	69	0.69200	0.731	124184	0.232	False	0.000000	0.3210	

70 rows × 15 columns

In [ ]:
```
pipeline_21_pilots.fit(df_21_dropped, df_21['Cluster Assignments'])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
```

Out[ ]:



In [ ]:
```
pipeline_21_pilots.predict(df_21)
```

Out[ ]:
```
array([0, 3, 0, 3, 3, 0, 3, 0, 3, 3, 0, 2, 3, 3, 3, 3, 0, 3, 0, 0, 3, 3,
       0, 0, 3, 3, 0, 0, 3, 2, 4, 4, 0, 3, 0, 3, 2, 0, 0, 0, 0, 0, 3, 2,
       2, 3, 0, 0, 2, 2, 0, 2, 2, 2, 0, 2, 2, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1], dtype=int32)
```

In [ ]:
```
pd.DataFrame(pipeline_21_pilots.predict(df_21)).value_counts()
```

Out[ ]:
```
0    24
3    20
1    12
2    12
4     2
Name: count, dtype: int64
```

-YOUR INTERPRETATION HERE-

Based on the predicted clusters of the Twenty One Pilots songs, I would recommend artists such as Radiohead, Justin Beiber, the White Stripes, and Taylor Swift to a Twenty One Pilots fan. This is beacause all of these artists have many songs in clusters 0 and 3, which are the 2 most commonly predicted clusters for the Twenty One Pilots songs. Thus, it is most likely that fans of Twenty One Pilots will enjoy music from artists who also have many songs in Clusters 0 and 3.

Submission Instructions

- Restart this notebook and run the cells from beginning to end.
  - Go to Runtime > Restart and Run All.

In [ ]:
```
# @markdown Run this cell to download this notebook as a webpage, `_NOTEBOOK.h

import google, json, nbformat

# Get the current notebook and write it to _NOTEBOOK.ipynb
raw_notebook = google.colab._message.blocking_request("get_ipynb",
                                                      timeout_sec=30)["ipynb"]
with open("_NOTEBOOK.ipynb", "w", encoding="utf-8") as ipynb_file:
  ipynb_file.write(json.dumps(raw_notebook))

# Use nbconvert to convert .ipynb to .html.
!jupyter nbconvert --to html --log-level WARN _NOTEBOOK.ipynb

# Download the .html file.
google.colab.files.download("_NOTEBOOK.html")
```

- Open `_NOTEBOOK.html` in your browser, and save it as a PDF.
  - Go to File > Print > Save as PDF.
- Double check that all of your code and output is visible in the saved PDF.
- Upload the PDF to Gradescope.
  - Please be sure to select the correct pages corresponding to each question.